Вероятностная робототехника

Sebastian Thrun Wolfram Burgard Dieter Fox

Содержание

Предисловие

Эта книга была написана с целью дать подробные первоначальные сведения о развивающейся области вероятностной робототехники. Вероятностная робототехника, в основном, описывает восприятие и управление роботов. Она основана на статистических методах представления информации и принятия решений и, таким образом, в ней учитывается неопределённость, возникающая в большинстве современных робототехнических систем. За последние годы вероятностные методы стали одной из ведущих парадигм разработки алгоритмов в робототехнике. В этой работе некоторые основные методы представлены впервые.

В книге большое внимание уделено алгоритмам и все они основаны на едином математическом аппарате: теореме Байеса, и его временном обобщении, известном как байесовские фильтры. Этот единый комплекс подходов и образует фундамент вероятностных алгоритмов.

При написании книги мы старались, по мере возможности, углубляться в технические детали. В каждой главе описаны один или несколько основных алгоритмов. Для каждого алгоритма представлены четыре вещи: (1) пример реализации в псевдокоде; (2) полный математический вывод из базовых соотношений до явного выражения алгоритма; (3) эмпирических результатов в той степени, в которой они соотносятся с тематикой книги; и (4) детального обсуждения сильных и слабых сторон каждого алгоритма с практической точки зрения. Разработка всего этого материала для многочисленных алгоритмов было нелёгким делом. Время от времени результат может оказаться несколько сложным для понимания читателя, хотя всегда можно просто пропустить математические выкладки! Мы надеемся, что внимательный читатель приобретёт значительно более глубокое понимание, чем поверхностный взгляд на предмет без привлечения математики.

Эта книга появилась на свет в результате более чем десяти лет исследований, выполненных нами, авторами, нашими студентами и множеством наших коллег по отрасли. Мы начали писать её в 1999, надеясь, что удастся закончить в течение нескольких месяцев. Прошло пять лет и от первичного черновика почти ничего не сохранилось. В ходе работы над книгой мы поняли о информации и теории принятия решений значительно больше, чем когда-либо рассчитывали. Мы счастливы сообщить, что большая часть того, что мы узнали, представлена в книге.

Эта монография написана для студентов, исследователей и специалистов, решающих практические задачи в робототехнике. Мы уверены, что каждый, кто строит роботов, должен разрабатывать и программы для них, поэтому представленный материал будет близок любому специалисту в робототехнике. Он также будет интересен специалистам по прикладной статистике и тем, кому необходимо работать с реальными данными датчиков без использования их в робототехнике. Чтобы удовлетворить потребности читателей с различным уровнем технической подготовки, мы старались сделать книгу как можно более полной. Некоторое понимание линейной алгебры и общей теории вероятности будет полезным, но мы также включили пример для иллюстрации базовых законов теории вероятности и старались избегать использования в тексте чересчур сложных математических выкладок.

Эта книга была написана для использования в качестве учебника. Каждая глава дополнена рядом упражнений и предполагает выполнение небольших проектов. При использовании в классе материал каждой главы следует разбивать на две лекции. Главы можно пропускать или менять местами

достаточно произвольно, например, мы сами начинаем обучение прямо с середины книги, с Главы 7. Мы рекомендуем дополнять чтение книги практическими наглядными экспериментами, как указано в упражнениях в конце каждой главы. Самая важная часть робототехники состоит в возможности делать всё своими руками!

Несмотря на все наши усилия, мы уверены, что в книге остались технические ошибки. Многие из них были исправлены в третьем издании. Мы продолжаем публиковать исправления на веб-сайте книги, где также можно найти дополнительные материалы, относящиеся к ней:

www.probabilistic-robotics.org

Мы надеемся, что Вам понравится книга!

Sebastian Thrun
Wolfram Burgard
Dieter Fox

Часть I
Основы

1 Введение

1.1 Значение неопределённости в робототехнике

Робототехника – это область знаний, изучающая манипулирование предметами физического мира и его восприятие с помощью устройств, управляемых компьютером. Примеры успешного применения робототехнических систем включают автономные платформы для исследований других планет, промышленные манипуляторы на производственных линиях, автомашины, которые могут двигаться без участия водителя и медицинские роботы для помощи хирургам при выполнении операций. Робототехнические системы действуют в физическом мире, воспринимают информацию об окружающем с помощью датчиков и воздействуют на предметы с помощью физических сил.

Хотя робототехника, в большей степени, все ещё проходит период становления, идея манипулирования объектами, используя «умные» устройства, имеет невероятный потенциал, способный изменить жизнь всего человечества. Разве не станет лучше, если все наши автомобили приобретут возможность безопасно передвигаться самостоятельно, сделав дорожно-транспортные происшествия лишь достоянием истории? И если роботы, вместо людей, будут очищать зоны радиоактивных катастроф, таких, как Чернобыль? И только представьте, что будет, если наши жилища будут населены умными помощниками, которые позаботятся обо всем, что связано с обслуживанием и ремонтом дома?

Чтобы выполнять такие задачи, роботам необходимо суметь приспособиться к невероятно высокой степени неопределённости, которая существует в физическом мире. С точки зрения робота, имеется целый ряд факторов, вносящих вклад в увеличение неопределённости.

Во-первых, и прежде всего, по определению непредсказуема *окружающая среда*, в которой действует робот. Хотя степень неопределённости в хорошо организованных средах, таких, как сборочные конвейеры, достаточно мала, условия на шоссе и в частных домах очень динамично, и, во многом, непредсказуемым образом изменяются. Степень неопределённости особенно высока для роботов, действующих рядом с людьми.

Датчики ограничены измеряемой характеристикой и целым рядом дополнительных факторов. Например, рабочее расстояние и разрешение датчика подвержены ограничениям его конструкции и физических законов, видеокамеры неспособны различать предметы сквозь стены, и имеют ограниченное пространственное разрешение. Датчики также подвержены воздействию шумов, которые не только искажают измерения непредсказуемым образом, но и ограничивают количество получаемой при измерении информации. Наконец, датчики могут просто выходить из строя, и обнаружить сбойный датчик очень сложно.

В *приводах робота* применяются двигатели, которые, по крайней мере, отчасти, непредсказуемы. Неопределённость вызывается такими эффектами, как шумы цепей управления, естественный износ и люфты, а также отказы механических составляющих. Некоторые приводы, наподобие используемых на промышленных манипуляторах, достаточно точные и надёжные, но дешёвые модели для мобильных роботов бывают крайне капризны.

Некоторая неопределённость вносится и программным обеспечением робота. Все *внутренние модели* окружающего мира носят приблизительный характер и являются лишь абстракциями реального мира. Таким образом, они лишь частично имитируют соответствующие физические процессы, происходящие внутри робота и окружающей его среды. Ошибки модели являются источником неопределённости, которая часто игнорируется в робототехнике, даже несмотря на то, что большинство моделей, в том числе для технологически совершенных роботов, и так достаточно приближённые.

Неопределённость вносится и в процессе *алгоритмического приближения*. Роботы – системы реального времени, и это накладывает ограничения на количество одномоментно выполняемых вычислений. Множество популярных алгоритмов приблизительны именно в силу того, что позволяют получить результат за необходимое время, жертвуя точностью.

Уровень неопределённости зависит от области применения. В некоторых областях использования робототехники, например, в поточном производстве, люди могут заранее спроектировать систему таким образом, чтобы неопределённость стала несущественным фактором. И, напротив, роботы, выполняющие задачи внутри жилых помещений или на других планетах, имеют дело со значительной неопределённостью. Такие роботы вынуждены действовать даже в условиях, когда ни датчики, ни внутренние имитационные модели неспособны предоставить достаточного количества информации для принятия решений с абсолютной уверенностью. Поскольку робототехника сегодня стремится выйти в открытый мир, проблема неопределённости стала серьёзным препятствием на пути создания эффективных систем. Управление степенью неопределённости, возможно, является самым важным шагом для создания надёжных робототехнических систем, способных действовать в реальном мире.

Этому и посвящена книга.

1.2 Вероятностная робототехника

В этой книге подробно описывается *вероятностная робототехника*. Это довольно новый подход, который учитывает неопределённость в системе восприятия и действия робота. Ключевой идеей вероятностной робототехники является явное представление неопределённости, путём активного использования вычислительных методов теории вероятности. Другими словами, вместо того, чтобы полагаться на единственную «лучшую догадку» относительно происходящего, вероятностные алгоритмы отображают данные во всём пространстве решений, используя вероятностные распределения. Таким образом, с их помощью возможно математически осмысленным способом отобразить как неоднозначность ситуации, так и степень уверенности. Остаточную часть неопределённости возможно учесть на этапе принятия управляющих решений. Более того, вероятностные роботы способны даже предпринимать активные действия по уменьшению неопределённости, когда это является наилучшим выбором. В силу этого, вероятностные алгоритмы достаточно надёжно работают в условиях неопределённости и часто демонстрируют лучший результат во многих прикладных задачах по сравнению с альтернативными методами.

Проиллюстрируем вероятностный подход в робототехнике с помощью двух ярких примеров: одного из области системы восприятия робота, и второго – из области планирования и управления.

Первым примером является *определение местоположения мобильного*

ОПРЕДЕЛЕНИЕ МЕСТОПОЛОЖЕНИЯ МОБИЛЬНОГО РОБОТА

робота.

Это задача оценки координат робота по отношению к внешнему набору ориентиров. У робота имеется карта окружающей среды, но, чтобы определить своё местоположение на карте, необходимо воспользоваться данными с датчиков. Эта ситуация показана на Рис 1.1. Известно, что в окружающей среде имеются три неразличимых между собой двери. Задачей робота является определение своего местоположения, используя лишь восприятие и движение.

Эта конкретная проблема обнаружения местоположения известна как *задача глобальной локализации* робота. В задаче глобальной локализации робот помещён в заранее неизвестное место известной среды и ему необходимо определить своё местоположение «с нуля». Вероятностная парадигма отражает *оценку* роботом функции плотности вероятности в пространстве всех местоположений для конкретного момента времени. Эта ситуация показана на схеме (a) Рис. 1.1. На схеме показано равномерное распределение по всем местоположениям. Теперь предположим, что робот выполнил первое измерение показаний датчиков и обнаружил, что находится около двери. В вероятностных методах эти данные используются для обновления оценки. «Апостериорная» оценка показана на схеме (b) Рис. 1.1. Как видим, значение вероятности было увеличено для местоположений в окрестностях дверей и снижено – около стен. Стоит обратить внимание на наличие в распределении вероятности пиков около каждой из неразличимых между собой дверей. Это ни в коей мере не означает, что робот *знает*, где он. Вместо этого, были выявлены три конкретные гипотезы, каждая из которых, в одинаковой степени подтверждается показаниями датчика. Нужно отметить, что робот помечает ненулевой вероятностью и места *не* напротив дверей. Это естественный результат учёта изначальной неопределённости восприятия, поскольку всегда имеется малая, но отличная от нуля вероятность, что робот ошибся, интерпретировав данные датчиков, как факт наличия двери. Способность всегда принимать во внимание маловероятные гипотезы крайне важна для обеспечения надёжности.

Теперь допустим, что робот переместился. На схеме (c) Рис. 1.1 показан эффект воздействия движения на оценку роботом своего местоположения. Оценка была смещена в направлении движения. Кроме того, ширина пиков увеличилась, отражая учёт дополнительной неопределённости, вызванной движением робота. На схеме (d) Рис. 1.1 показано оценочное распределение после обнаружения роботом ещё одной двери. Это следующее наблюдение заставило алгоритм разместить основную долю вероятности около одной из дверей, и робот, в данный момент, довольно уверенно оценивает своё местоположение. Наконец, на нижней схеме (e) показана оценка для случая, когда робот передвигается ещё дальше по коридору.

Этот пример иллюстрирует многие аспекты вероятностной парадигмы. С точки зрения вероятности, проблему восприятия робота можно представить как задачу оценки состояния. Приведённый в примере локализации алгоритм известен как *байесовский фильтр* и служит для нахождения апостериорной оценки в пространстве местоположений робота. Отображение информации представляет собой функцию плотности вероятности. Обновление функции отражает влияние дополнительной информации, полученной в результате измерений датчиков или же её потерю в силу влияния процессов окружающего мира, увеличивающих неопределённость.

БАЙЕСОВСКИЙ ФИЛЬТР

Второй пример позволяет познакомиться с реалиями планирования и управления действиями робота. Как только что было сказано, с помощью вероятностных алгоритмов возможно вычислить степень неопределённости

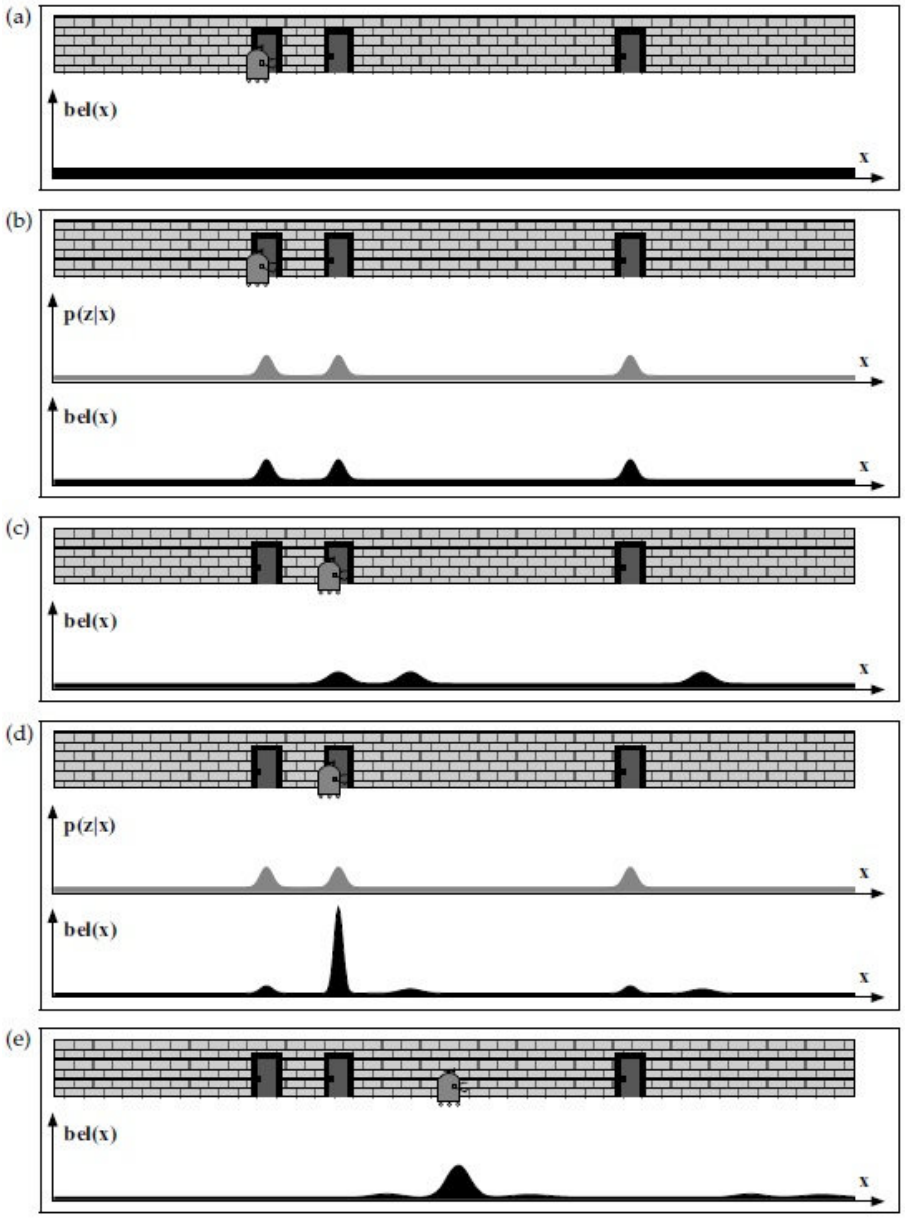


Рис. 1.1 Общий принцип *марковской локализации*: Мобильный робот в задаче глобальной локализации. Способы марковской локализации будут описаны в Главах 7 и 8.

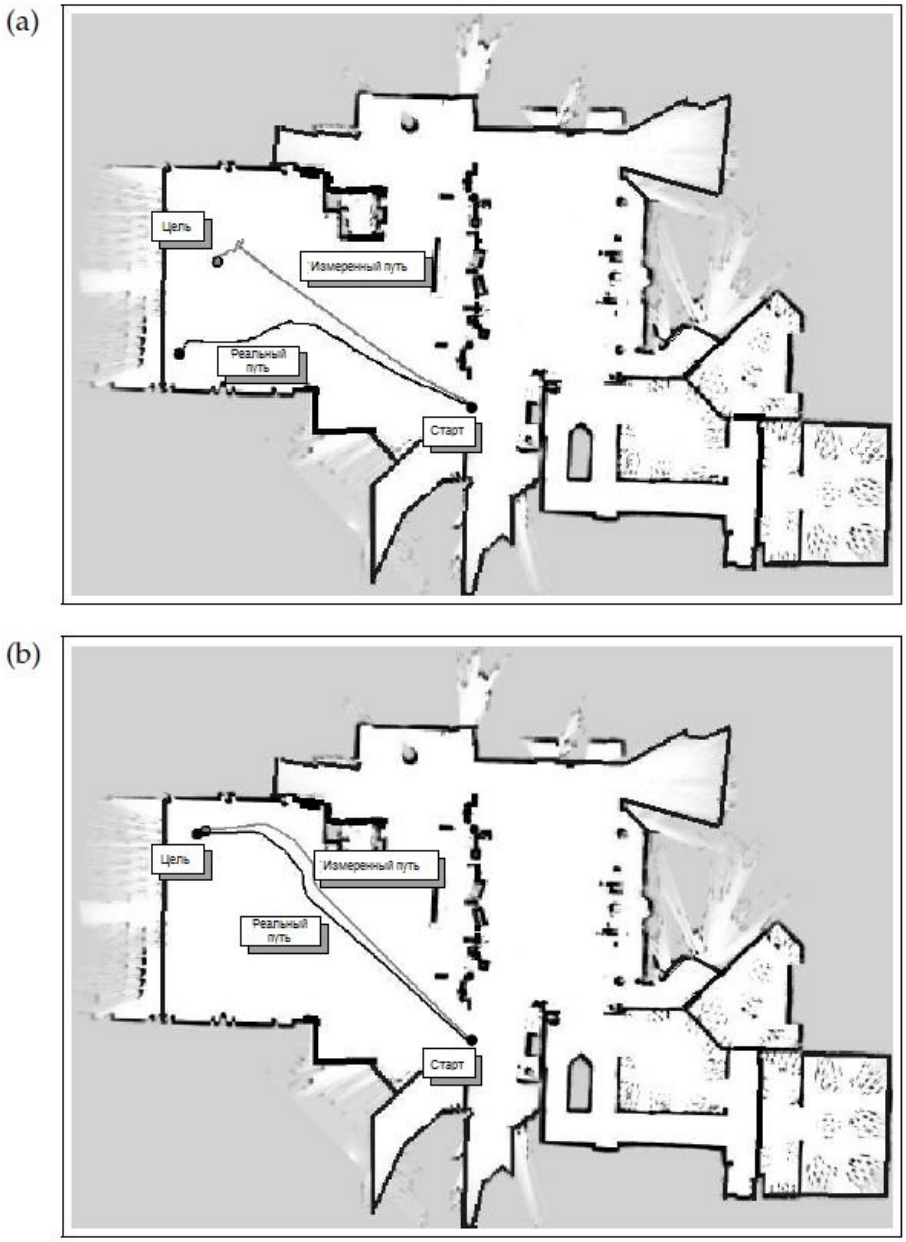


Рис 1.2 Верхний рисунок: робот, который передвигается в открытом, безориентирном пространстве, может потерять возможность отслеживать своё местоположение. Нижний рисунок: Этого можно избежать, если оставаться около известных препятствий. Данные схемы иллюстрируют работу алгоритма, который называется *прибрежная навигация*, и будет обсуждаться в Главе 16. Рисунки являются собственностью Николаса Роя из MIT.

для робота в конкретный момент. Кроме этого, возможно учесть и будущую неопределённость, приняв ее во внимание при определении необходимых управляющих действий. Один из таких алгоритмов называется *прибрежная навигация* и показан на Рис. 1.2. На рисунке изображена плоская карта реального здания. На верхней схеме показано сравнение реальной и расчётной траекторий. Наблюдаемое отклонение является результатом неопределённости для робота, которая только что обсуждалась. Представляет интерес факт того, что не все траектории подвержены одинаковой неопределённости. Траектория на Рис. 1.2a проходит по относительно открытому пространству, где почти нет ориентиров, которые робот может использовать, чтобы уточнить своё местоположение. На Рис. 1.2b показан альтернативный путь. В этом случае траектория сначала приближается к явно различимому с помощью датчиков углу, а затем проходит вблизи стены, что позволяет постоянно уточнять местоположения. Неудивительно, что неопределённость во втором случае существенно меньше, а это значит, что шансы действительно достичь расчётной целевой точки в конце маршрута заметно выше.

ПРИБРЕЖНАЯ НАВИГАЦИЯ

С помощью этой ситуации был проиллюстрирован лишь один из множества способов правильного учёта эффектов неопределённости в управлении роботом. В приведённом примере необходимость учёта возможной высокой неопределённости по одной из траекторий заставила робота предпочесть другой, более длинный путь, который позволил её уменьшить. Новый путь оказывается лучше в том смысле, что робот имеет гораздо более высокие шансы действительно оказаться у цели при достижении расчётной точки. Фактически, второй путь представляет иллюстрацию метода активного сбора информации. Робот, используя вероятностные соображения, определяет наилучший выбор действий таким образом, чтобы иметь возможность получать данные при продвижении к цели. Вероятностные способы планирования позволяют учесть неопределённость и запланировать сбор необходимой информации, а вероятностные способы управления – в полной мере использовать результаты такого планирования.

1.3 Выводы

Вероятностная робототехника органично объединяет в одно целое модели и данные датчиков, в то же время, обходя ограничения обоих. Эти принципы являются не просто прерогативой низкоуровневого управления, но проходят через все уровни программного обеспечения робота, от самых базовых до абстракций высшего уровня.

В отличие от традиционных подходов программирования в робототехнике — таких, как планирование движений на основе имитационной модели или реакции согласно шаблонам поведения, вероятностные методы обычно более надёжны в условиях наличия ограничений датчиков и моделей. Это позволяет гораздо легче, по сравнению с более старыми парадигмами, масштабировать их для сложных сред реального окружающего мира, где неопределённость ещё более выражена. Де-факто, некоторые вероятностные алгоритмы являются единственным существующим рабочим решением для сложных проблем оценки ситуации в робототехнике, таких как проблема локализации, обсуждаемая чуть выше, или задача построения точных карт очень большого окружающего пространства.

По сравнению с традиционными подходами робототехники, полагающимися на модели, вероятностные алгоритмы менее требовательны к алгоритмам моделирования, что освобождает программиста от нелёгкой задачи

улучшения их качества. Вдобавок, вероятностные алгоритмы ещё и не так чувствительны к точности датчиков робота, по сравнению с реактивными методами, для которых текущие показания датчиков являются единственным источником управляющего сигнала. С вероятностной точки зрения, *проблема обучения робота* – всего лишь проблема оценки состояния в долгосрочной перспективе. Таким образом, вероятностные алгоритмы предоставляют чёткую методологию для многих аспектов обучения роботов.

Как всегда, наличие преимуществ имеет цену. Два наиболее часто упоминаемых ограничения вероятностных алгоритмов — *вычислительная сложность* и использование *приближенных вычислений*. Вероятностные алгоритмы изначально не так эффективны, как аналоги, не использующие вероятности. Это происходит потому, что вместо единственной аналитической оценки рассматриваются все плотности вероятности. Необходимость приближения возникает в силу непрерывной природы большинства реальных сред, что значительно затрудняет получение точных апостериорных распределений. Правда, в некоторых редких случаях неопределённость удаётся довольно точно аппроксимировать, используя компактную параметрическую модель (например, нормальные распределения). Но чаще случается, что такие аппроксимации слишком неточны для практического использовать и необходимы более сложные представления.

Последние разработки в области аппаратного обеспечения позволили получить доступ к беспрецедентно высоким вычислительным мощностям по очень низкой цене, что положительно повлияло и на область вероятностной робототехники. Вдобавок, современные исследования позволили значительно повысить эффективность вероятностных алгоритмов для целого ряда сложных задач робототехники, многие из которых будут подробно рассмотрены в этой книге. Тем не менее, вопрос вычислительной сложности остаётся, и мы ещё не раз подробно на нем остановимся, освещая сильные и слабые стороны конкретных вероятностных решений.

1.4 Содержание

Книга состоит из четырёх основных частей.

• В Главах со второй по четвертую раскрываются общие математические принципы, лежащие в основе всех описанных в книге алгоритмов, а также некоторые ключевые алгоритмы. Материал этих глав составляет математическую основу всей книги.

• В Главах 5 и 6 представлены вероятностные модели, используемые для мобильных роботов. Во многом, содержание этих главах представляет лишь вероятностное обобщение классических моделей робототехники. Эти алгоритмы образует робототехническую основу для изложения последующего материала.

• Проблема локализации мобильного робота обсуждается в Главах 7 и 8. Здесь базовые оценочные алгоритмы объединяются с вероятностными моделями, обсуждаемыми в предыдущих двух главах.

• Главы с 9 по 13 посвящены значительно более обширной проблеме составления карт местности для роботов. Как и ранее, они основываются на алгоритмах, описанных в начальных главах, дополненных новыми подхо-

дами, чтобы приспособить их к невероятной сложности задачи.

• Вопросам планирования и управления с помощью вероятностных методов посвящены Главы с 14 по 17. Приведённые в начале раздела основные методы расширяются до практических алгоритмов вероятностного управления роботом. Завершающая Глава 17 посвящена обсуждению вероятностной точки зрения на проблему исследования с помощью роботов.

Книгу лучше всего читать в приведённом порядке, от начала до конца, хотя мы и постарались каждую отдельную главу сделать самостоятельной и полной. Многочисленные разделы под названием *"Математический вывод..."* при первом прочтении книги можно смело пропускать без риска утратить целостность изложения и общее понимание материала.

1.5 Обучение вероятностной робототехнике

При использовании в качестве учебного пособия, мы *не* рекомендуем преподавать главы в том порядке, в котором они приводятся в книге — если только студенты не отличаются необычайно сильной тягой к абстрактным математическим концепциям. Многочастичные фильтры легче объяснить, чем нормальные распределения, но студентам часто больше нравятся вопросы локализации мобильных роботов, нежели абстрактные алгоритмы фильтров. В наших курсах обучения мы обычно начинаем с Главы 2, а затем переходим прямо к Главам 7 и 8. Излагая вопросы локализации, по мере необходимости, мы возвращаемся к материалу в Главах с 3 по 7. Мы также рано преподаём материал Главы 14, чтобы как можно быстрее посвятить студентов в вопросы планирования и управления в рамках учебного курса.

В целях преподавания Вы можете свободно использовать слайды и анимацию с веб-сайта книги

www.probabilistic-robotics.org

для иллюстрации различных алгоритмов. Также Вы можете послать нам, авторам, ссылки на веб-сайты Ваших учебных курсов и любой материал, который может быть полезен другим в процессе обучения Вероятностной Робототехнике.

Лучше всего изучать материал этой книги, имея возможность реализовать всё на практике. Нет ничего лучше для обучения робототехнике, чем программирование настоящего робота. Никто и ничто не может выявить трудности и подводные камни лучше, чем сама Природа!

1.6 Библиографические примечания

Робототехника, по мере развития, прошла долгий путь через целую серию принципов создания программного обеспечения. Первая парадигма сформировалась в середине 1970-х годов и известна как *модельная парадигма*. Она началась с ряда исследований, продемонстрировавших трудности управления робототехническим манипулятором с большим количеством степеней свободы в непрерывных пространствах, в частности, с работы Рейфа (Reif, 1979). Позже она была в значительной степени развита в целом ряде публикаций, в частности, в аналитической работе Шварца (Schwartz

et al., 1987) об оценке сложности движений робота. Канни (Canny, 1987) предложил первый полностью экспоненциальный алгоритм планирования движения, а Латомб (Latombe, 1991) опубликовал основополагающий текст о планировании движения на основе модели (многие другие ключевые достижения будут обсуждаться в Главе 14). В этих ранних работах проблема неопределённости, по большей части, игнорировалась, несмотря на уже начавшееся широкое использование рандомизации в качестве техники решения сложных проблем планирования движения (Кавраки с соавторами (Kavraki et al., 1996)). Напротив, неотъемлемым условием было наличие полных и точных моделей робота, окружающей среды, а также полная определённость робототехнической системы. Модель при этом подразумевалась достаточно точной для обработки остаточной неопределённости с помощью одного лишь низкоуровневого контроллера движений. Большинство методов планирования движения просто создавало единичную эталонную траекторию для управления манипулятором, хотя методы *потенциальных полей* (Хатиб (Khatib, 1986) и *навигационных функций* (Кодишек (Koditschek, 1987) и предоставляли механизмы реагирования на непредвиденные факторы. Реализации этих ранних методов, если таковые вообще имелись, были ограничены искусственными средами, где любые проявления неопределённости могли быть устранены механическими методами или же с достаточной точностью измерены.

Ситуация радикально изменилась в середине 1980-х, когда все исследовательское сообщество робототехники обратило пристальное внимание на проблему недостатка обратной связи от датчиков. С большой уверенностью для области *поведенческой робототехники* была отвергнута идея необходимости наличия любой внутренней модели. Вместо этого было введён принцип взаимодействия *ситуационного агента* с физической средой (Кэлблинг и Розеншайн (Kaelbling и Rosenschein) 1991), что придало движениям робота комплексный характер (феномен, часто называемый *непредсказуемым поведением* (Стилз (Steels, 1991)). Теперь уже определяющую роль играло сенсорное обнаружение, и внутренние модели были отвергнуты (Брукс (Brooks) 1990).

Энтузиазм на поприще поведенческой парадигмы подогревался ранними успехами, когда удалось далеко превзойти возможности традиционных алгоритмов планирования движения на основе моделей. Одним из первых успешных образцов стал “Чингиз” ("Genghis"), шестиногий робот, разработанный Бруксом (Brooks, 1986). Относительно простой алгоритм на основе конечного автомата оказался способен управлять передвижением робота даже по пересечённой местности. Ключ к успеху данного метода состоял в наличии сенсорного восприятия: управление было полностью определено характеристиками взаимодействия с окружающей средой, воспринимаемой с помощью датчиков робота. На основе некоторых ранних работ был создан достаточно сложный робот, использующий комплексную обратную связь от окружающей среды (Коннелл (Connell, 1990)). Позже эта подход стал коммерчески успешным в виде проекта робота-пылесоса (IRobots Inc., 2004), программное обеспечение которого также было построено на основе поведенческого подхода.

В силу малого количества внутренних моделей и особого внимания к простым механизмам управления, большинство робототехнических систем были ограничены относительно несложными задачами, в которых текущих данных с датчиков оказывалось достаточно для выбора верного варианта управления. Чтобы обойти это ограничение, в более современных работы используются архитектуры *гибридного управления* (Аркин (Arkin, 1998)),

когда низкоуровневое управление обеспечивается поведенческими методами, а планировщик на основе модели координирует действия робота на высоком, абстрактном уровне. Такие гибридные архитектуры получили широкое распространение в сегодняшней робототехнике. Они хорошо согласуются с фундаментальной работой по трехуровневым архитектурам, написанной Гэтом (Gat, 1998), начало которой было положено «Роботом Шейки» Нилссона ("Shakey the Robot Nilsson 1984).

С середины 1990-х вероятностная робототехника пережила этап развития , хотя ее базовые принципы можно проследить вплоть до изобретения калмановского фильтра (Kalman, 1960). Во многом, она занимает промежуточное место между модельными и поведенческими методами. В вероятностной робототехнике имеются модели, но считается, что они неполны и недостаточны для осуществления управления. Принимаются во внимание и показания датчиков, но и они не считаются достаточными. Действие управления может быть определено путём комбинирования обеих составляющих – модели и данных измерений датчиков. Для интеграции моделей и измерений датчиков в одно целое используются математические методы статистики.

Многие из ключевых достижений в области вероятностной робототехники будут обсуждаться в будущих главах. Некоторые из фундаментальных открытий в этой области включают изобретение Смитом и Чизманом (Smith и Cheeseman, 1986) методов калмановской фильтрации для решения проблем восприятия в большом количестве измерений, открытие карт сеток занятости (Элфис (Elfes), 1987, Моравиц (Moravec), 1988), и вторичное введение в обиход Кэлблингом с соавторами (Kaelbling et al., 1998) методов планирования при неполной информации наблюдений. В последнее десятилетие наблюдалось просто прорывное развитие новых методов: широкую популярность приобрели многочастичные фильтры (Деллаэрт с соавторами (Dellaert et al.), 1999), были разработаны новые методологии программирования на основе байесовских методов обработки информации (Трун (Thrun), 2000, Лебелтел с соавторами (Lebeltel et al.) 2004, Парк с соавторами (Park et al.), 2005). Это развитие происходило рука об руку с созданием физических робототехнических систем под управлением вероятностных алгоритмов, таких, как промышленные механизмы для перемещения грузов, описанные в работе Дюрран-Уайта (Durrant-Whyte, 1996), развлекательных роботов для музеев (Баргард (Burgard et al.), 1999, Трун (Thrun et al.), 2000, Зигварт (Siegwart et al.), 2003) и роботов медицинского назначения, приспособленных для ухода за больными (Пино (Pineau et al.), 2003). Программный пакет управления мобильным роботом с открытым исходным кодом и широким использованием вероятностных методов был представлен в работах Монтемерло с соавторами (Montemerlo et al., 2003a).

Отрасль коммерческой робототехники также подошла к поворотной точке. В ежегодном Мировом обзоре робототехники (World Robotics Survey), опубликованном Европейской комиссией ООН и *Международной федерацией робототехники* в 2004 году был отмечен годовой прирост мирового рынка робототехники на 19%. Ещё более примечателен факт изменения структуры рынка, указывающий на переход от промышленного использования к сервисным роботам и потребительским продуктам.

2 Рекурсивная оценка состояния

2.1 Введение

Ключевая идея вероятностной робототехники состоит в оценке состояния на основе информации от датчиков. Эта задача сводится к оценке численных показателей, которые напрямую измерить невозможно, но можно каким-то образом воспринять их из свойств внешнего мира. В большинстве приложений робототехники довольно просто определить, что же нужно делать, если известны *точные* значения параметров. Например, перемещать мобильного робота, если известны точные местоположения как самого робота, так и всех ближайших препятствий, довольно легко. К сожалению, напрямую измерить эти значения невозможно и для получения нужной информации робот вынужден полагаться на измерения датчиков, способные передать только частичную информацию, при этом часто повреждённую воздействием помех. Оценка состояний производится для того, чтобы попытаться восстановить состояние переменной на основе имеющихся данных. Вероятностные алгоритмы оценки состояния вычисляют распределения оценок по всем возможным состояниям. Пример вероятностной оценки состояний уже приводился во вводной части книги – это проблема локализации мобильного робота.

Целью этой главы является представление базовой системы понятий и математических инструментов для оценки состояния на основе данных датчиков.

• В подразделе 2.2 вводятся основные концепции теории вероятности, использованные в книге.

• В подразделе 2.3 описывается формальная модель взаимодействия робота с окружающей средой и вводится ключевая терминология, которая используется в книге.

• В подразделе 2.4 впервые описываются *байесовские фильтры*, рекурсивный алгоритм оценки состояния, образующий первооснову для практически каждого представленного метода.

• В подразделе 2.5 обсуждаются трудности представлений и вычислений, которые возникают при использовании байесовских фильтров.

2.2 Основные концепции теории вероятности

Этот подраздел знакомит читателя с основными обозначениями и фактами теории вероятности, которые используются в книге. В вероятностной робототехнике все величины, такие как измерения датчиков, управляющие воздействия, состояние робота и окружающей среды, смоделированы с помощью случайных переменных.

СЛУЧАЙНЫЕ ПЕРЕМЕННЫЕ

Случайные переменные способны принимать несколько значений в соответствии с определёнными правилами теории вероятности. Вероятностное заключение – это процесс вычисления таких правил для случайных величин, зависящих от других случайных величин и наблюдений.

Обозначим через X случайную переменную, а x – конкретное значение, которое может принимать X. Стандартным примером случайной переменной является исход *броска монеты*, где X может принимать значения *«орла»* или *«решки»*. Если пространство всех значений, которые способна принимать X, дискретно, как в случае броска монеты, то можно записать

(2.1)
$$p(X = x)$$

чтобы обозначить вероятность того, что случайная переменная X принимает значение x. Например, для «честной» монеты эта величина характеризуется как $p(X = \text{«орёл»}) = p(X = \text{«решка»}) = 1/2$. Сумма всех дискретных вероятностей равна единице, отсюда

(2.2)
$$\sum_x p(X = x) = 1$$

Вероятности также всегда неотрицательны, поэтому $p(X = x) \geq 0$.

Для упрощения записи обычно будем опускать точное указание случайной переменной и использовать сокращение $p(x)$ вместо того, чтобы писать $p(X = x)$.

Большинство методов в этой книге имеет дело с оценкой и процессом принятия решений в непрерывных пространствах. Непрерывные пространства характеризуются тем, что случайные переменные способны принимать значения из непрерывного диапазона. Если иное не указано специально, будем считать, что все непрерывные случайные переменные имеют *функции плотности вероятности* (ФПВ - PDF). Обычно функция плотности представляет собой одномерное *нормальное распределение* с математическим ожиданием μ и среднеквадратичным отклонением σ^2. ФПВ нормального распределения выражена следующей *функций Гаусса*:

ПЛОТНОСТЬ ВЕРОЯТНОСТИ

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

ГАУССОВСКАЯ ФУНКЦИЯ

(2.3)
$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}$$

Нормальные распределения играют важную роль в книге. Часто мы будем сокращать их до $N(x; \mu, \sigma^2)$, что означает случайную переменную, её математическое ожидание и среднеквадратичное отклонение.

Для нормального распределения (2.3) x считается скалярным значением. Однако, часто x будет представлен в виде многомерного вектора. Нормальные распределения векторов называются *многомерными*.

МНОГОМЕРНОЕ РАСПРЕДЕЛЕНИЕ

Многомерные нормальные распределения характеризуются функциями распределения вероятности следующего вида:

(2.4)
$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

Здесь μ - многомерный вектор, а Σ - *неотрицательно определённая симметричная* матрица, называемая *ковариационной* Верхний индекс T означает транспонирование вектора. Аргумент экспоненты в функции квадратичен по x, с параметрами квадратичной функции μ и Σ.

КОВАРИАЦИОННАЯ МАТРИЦА

Следует обратить внимание, что Выражение (2.4) является строгим обобщением Выражения (2.3). Оба уравнения равны, если x - скалярное значение, а $\Sigma = \sigma^2$.

Уравнения (2.3) и (2.4) – это примеры функций распределения вероятности. Точно так же, как сумма дискретного распределения вероятности равна 1, интеграл ФПВ всегда равен единице:

(2.5)
$$\int p(x)dx = 1$$

Однако, в отличие от дискретной вероятности, значения ФПВ не ограничены сверху единицей. В ходе изложения в этой книге понятия *вероятности, плотности вероятности* и *функции плотности вероятности* будет использоваться как взаимозаменяемые. По умолчанию будем считать, что все непрерывные случайные переменные измеримы, а также, что для всех непрерывных распределений существуют плотности вероятности. *Совместное распределение* двух случайных переменных X и Y выражается в виде

СОВМЕСТНОЕ РАСПРЕДЕЛЕНИЕ

(2.6)
$$p(x, y) = p(X = x; Y = y)$$

Это выражение описывает вероятность события, когда случайная переменная X принимает значение x, и одновременно, переменная Y принимает значение y. Если X и Y *независимы*, то получится, что

НЕЗАВИСИМОСТЬ

(2.7)
$$p(x, y) = p(x)p(y)$$

Часто случайные переменные уже содержат определённую информацию о других случайных переменных. Допустим, уже известно, что значение Y равно y, и хотелось бы узнать вероятность того, как значение x переменной X связано с этим фактом. Такая вероятность будет равна

(2.8)
$$p(x|y) = p(X = x|Y = y)$$

УСЛОВНАЯ ВЕРОЯТНОСТЬ

и называться *условной вероятностью*. При $p(y) > 0$ условная вероятность определяется как

(2.9)
$$p(x|y) = \frac{p(x, y)}{p(y)}$$

Если X и Y независимы, то получится, что

(2.10)
$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x)$$

Другими словами, при независимости X и Y, информация о значении Y никак не поможет узнать значение X. Независимость и ее обобщение, известное как условная независимость играют очень важную роль в этой книге.

Интересный факт, который является следствием определения условной вероятности и аксиом мер вероятности, часто называют *Теоремой о полной вероятности*:

ТЕОРЕМА О ПОЛНОЙ ВЕРОЯТНОСТИ

(2.11)
$$p(x) = \sum_y p(x|y)p(y) \qquad \text{(для дискретного случая)}$$

(2.12)
$$p(x) = \int p(x|y)p(y)dy \qquad \text{(для непрерывного случая)}$$

Если $p(x|y)$ или $p(y)$ равны нулю, можно и произведение $p(x|y)\ p(y)$ принять равным нулю, вне зависимости от оставшегося множества.

ТЕОРЕМА БАЙЕСА

Также очень важна *теорема Байеса*, которая определяет отношение вероятности $p(x|y)$ к его «противоположности» $p(y|x)$. Для теоремы требуется, чтобы $p(y) > 0$:

(2.13)

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\Sigma_{x'} p(y|x')p(x')} \qquad \text{(дискретный случай)}$$

(2.14)

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx'} \qquad \text{(непрерывный случай)}$$

Теорема Байеса играет определяющую роль в вероятностной робототехнике (как и вообще в вероятностных выводах). Если x - это количество, которое бы нам хотелось узнать на основании y, то вероятность $p(x)$ будет называться *априорным распределением вероятности*, а y будет называться *данными* (например, измерений датчиков). В распределении $p(x)$ суммируются все данные, имеющиеся об X, без учёта y. Вероятность $p(x|y)$ называется *апостериорным распределением вероятности* по X. В силу Выражения (2.14), теорема Байеса даёт удобный способ вычислить апостериорную вероятность $p(x|y)$ на основе «перевёрнутой» условной вероятности $p(y|x)$ и априорной вероятности $p(x)$. Другими словами, если необходимо получить значение x на основе данных датчиков y, теорема Байеса позволяет сделать это с помощью обратной вероятности, определяющей значение y при условии, что имело место событие x. В робототехнике вероятность $p(y|x)$ часто называют *порождающей моделью*, поскольку она, на некотором уровне абстракции, описывает, каким образом переменные состояния X *становятся причиной* изменения показаний датчиков Y.

АПРИОРНАЯ ВЕРОЯТНОСТЬ

АПОСТЕРИОРНАЯ ВЕРОЯТНОСТЬ

ПОРОЖДАЮЩАЯ МОДЕЛЬ

Важно отметить, что знаменатель $p(y)$ в теореме Байеса не зависит от x. Таким образом, множитель $p(y)^{-1}$ в уравнениях (2.13) и (2.14) будет одинаковым при любом значении x апостериорной вероятности $p(x|y)$. В силу этого, $p(y)^{-1}$ часто именуют *нормирующим членом* в теореме Байеса и, в общем виде, записывают как η:

(2.15)

$$p(x|y) = \eta p(y|x)p(x)$$

Преимуществом такой записи является ее лаконичность. Вместо явной записи точной формулы нормирующей константы (а она может вырасти до весьма больших размеров в некоторых математических выводах), просто запишем символ нормировки η, чтобы показать, что конечный результат должен быть нормирован до единицы. В этой книге подобные нормировщики будут записываться как η(или η', η'', . . .). **Важно:** Мы будем произвольно использовать в различных равенствах одно и то же обозначение нормирующего члена η, даже если его реальное значение различается.

Заметим, что будет совершенно справедливым переписать любую из обсуждаемых теорем, используя произвольные случайные переменные, скажем, переменную Z. Например, запись теоремы Байеса, при $Z = z$, даст следующее равенство:

(2.16)
$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)}$$

пока $p(y|z) > 0$.

Аналогично, можем переписать правило для комбинирования вероятностей независимых случайных переменных (2.7), используя переменную z:

(2.17)
$$p(x,y|z) = p(x|z)p(y|z)$$

УСЛОВНАЯ НЕЗАВИСИМОСТЬ

Такое отношение называется *условной независимостью*. Читатель может легко убедиться, что (2.17) эквивалентно

(2.18)
$$p(x|z) = p(x|z,y)$$

(2.19)
$$p(y|z) = p(y|z,x)$$

Условная независимость играет важную роль в вероятностной робототехнике. Она применяется, когда переменная y не несёт никакой информации о переменной x, при условии, что известно значение другой переменной z. Условная независимость не предполагает (абсолютной) независимости, а значит

(2.20)
$$p(x,y|z) = p(x|z)p(y|z) \nRightarrow p(x,y) = p(x)p(y)$$

Обратное также, в общем случае, неверно, поскольку абсолютная независимость не предполагает условной независимости:

(2.21)
$$p(x,y) = p(x)p(y) \nRightarrow p(x,y|z) = p(x|z)p(y|z)$$

В отдельных случаях условная и абсолютная независимость могут совпадать.

В ряде вероятностных алгоритмов требуется вычислять признаки или статистики вероятностных распределений. *Математическое ожидание* случайной величины X обозначается как

МАТЕМАТИЧЕСКОЕ ОЖИДАНИЕ СЛУЧАЙНОЙ ПЕРЕМЕННОЙ

(2.22)
$$E[X] = \sum_x xp(x) \qquad \text{(дискретный случай)}$$

(2.23)
$$E[X] = \int xp(x)dx \qquad \text{(непрерывный случай)}$$

Некоторые случайные переменные не имеют конечного математического ожидания, однако, такие исключения в материале книги рассматриваться не будут.

Математическое ожидание – это линейная функция случайной переменной. В частности, можно записать

(2.24)
$$E[aX + b] = aE[X] + b$$

для произвольных числовых значений a и b. Ковариация X получается следующим образом

(2.25)
$$Cov[X] = E[X - E[X]]^2 = E[X^2] - E[X]^2$$

Ковариация означает квадрат ожидаемого отклонения от значения математического ожидания. Как было указано ранее, математическое ожидание многомерного нормального распределения $N(x; \mu, \Sigma)$ равно μ, тогда его ковариация будет Σ.

ЭНТРОПИЯ

Последней важной концепцией этой книги является *энтропия*. Энтропия вероятностного распределения задаётся следующим выражением:

(2.26)
$$H_p(x) = E[-\log_2 p(x)]$$

которое можно выразить в следующем виде

(2.27)
$$H_p(x) = -\sum_x p(x) \log_2 p(x) \qquad \text{(дискретный случай)}$$

(2.28)
$$H_p(x) = -\int p(x) \log_2 p(x) dx \qquad \text{(непрерывный случай)}$$

Рис.2.1. Взаимодействие робота с окружающей средой.

Концепция энтропии происходит из теории информации. Энтропия- это ожидаемое количество информации, которое содержит значение x. В дискретном случае, для $-\log_2 p(x)$ означает количество бит, требуемое для кодирования x оптимальным образом, подразумевая, что $p(x)$ – вероятность появления значения x. В данной книге понятие энтропии будет использовано в контексте сбора информации роботом, выражая, таким образом, количество информации, которую робот может получить, совершая определённые действия.

2.3 Взаимодействие робота с окружающей средой

ОКРУЖАЮЩАЯ СРЕДА РОБОТА

На Рис. 2.1 показано взаимодействие робота с окружающей средой. *Окружающая среда* или *«мир»* - это динамическая система, обладающая внутренним состоянием. Робот способен получать информацию об окружающем мире, используя датчики. Но с датчиков поступает зашумленный сигнал, и, вдобавок, имеется множество факторов, которые невозможно измерить напрямую. Как следствие, робот формулирует некое внутреннее суждение относительно состояния окружающей среды на основании собранных и имеющихся данных, что показано слева на схеме.

Робот также способен воздействовать на окружающую среду с помощью приводов. Эффект таких действий очень часто несёт некоторую непредсказуемость, поэтому каждое действие управления влияет как на состояние окружающей среды, так и на внутреннее представление этого состояния роботом.

Опишем это взаимодействие более формально.

2.3.1 Состояние

СОСТОЯНИЕ

Окружающая среда характеризуется *состоянием*. В материале, представленном в книге, удобно понимать термин «состояние» как обозначение набора всех отличительных черт робота и его окружения, способных повлиять на будущее. Некоторые переменные состояния изменяются со временем, например, местонахождение людей в непосредственной близости от робота. Другие остаются постоянными, такие, как расположение стен в большинстве зданий. Изменяющиеся состояния назовём *динамическими*, чтобы отличать их от *статических*, то есть неизменных, состояний. Состояние также включает переменные, описывающие самого робота, такие как его положение, скорость, исправность датчиков и так далее.

В книге состояние будет обозначено x, несмотря на то, что конкретные переменные, включённые в x, могут различаться, в зависимости от контекста. Состояние в момент времени t будет обозначаться как x_t. В книге используются следующие типичные переменные состояния:

ПОЛОЖЕНИЕ

• *Положение* робота, определяет его местоположение и ориентацию в пространстве относительно глобальной координатной сетки. Мобильные роботы жёсткой конструкции имеют шесть переменных состояния, три, обозначающие координаты в прямоугольной системе, и три - для углов положения в пространстве (крен, тангаж, рысканье). Для мобильных роботов, действующих в плоских окружающих средах, положение обычно задаётся тремя переменными - двумя координатами на плоскости и углом направления движения (рысканье).

• При управлении роботом описание его положения включает переменные для *конфигурации приводов*. Это могут быть, например, углы поворота для вращающихся сочленений. Каждая степень свободы манипулятора робота характеризуется одномерной конфигурацией в произвольный момент времени и является составляющей кинематического состояния робота. Конфигурация робота часто обозначается как *кинематическое состояние*.

• *Скорость робота* и *скорости сочленений* обычно обозначаются как *динамическое состояние*. Для робота жёсткой конструкции, передвигающегося в пространстве, можно вывести до шести переменных скорости, по одной для каждого направления перемещения. Динамическое состояние иг-

рает очень небольшую роль в книге.

• *Местоположение и признаки объектов окружающего мира* также являются переменными состояния. Объектом может быть дерево, стена или даже пиксель на более обширной поверхности, а их характеристикой, например, внешний вид (цвет, текстура). В зависимости от степени детализации моделируемого состояния, описание окружающей среды может иметь от нескольких десятков до сотен миллионов переменных состояния (или даже больше). Только представьте, сколько бит потребуется для точного описания вашего окружения! Для многих задач, обсуждаемых в книге, местоположение объектов в окружающей будет неизменным. Для некоторых задач объекты могут считаться разновидностью *ориентиров*, которые представляют собой хорошо различимые, стационарные признаки окружающей среды, которые можно надёжно распознать.

ОРИЕНТИР

• *Местоположение и скорости движущихся объектов и людей* также являются потенциальными переменными состояния. Часто робот является не единственным движущимся объектом в окружающей среде. Для других движущихся сущностей выделяются собственные наборы из кинематического и динамического состояний.

• Существует множество других переменных состояния, которые могут повлиять на ориентацию робота в пространстве. Например, переменной состояния может стать факт исправности датчика, или же уровень заряда для робота, работающего от аккумуляторов. Список потенциальных переменных состояния бесконечен!

ПОЛНОЕ СОСТОЯНИЕ

Состояние x_t назовём *полным*, если оно наилучшим образом может быть использовано для предсказания будущего состояния. Другими словами, "полнота"означает, что знание прошлых состояний, измерений или управляющих действий не даёт дополнительной информации, которая могла бы помочь ещё более точно предсказать будущие изменения. Важно заметить, что наше определение полноты не требует того, чтобы будущее было *детерминированной* функцией состояния. Оно может быть и стохастическим, но никакие переменные, кроме x_t, не могут повлиять на стохастическое развитие будущих состояний, если только эта зависимость не управляется состоянием x_t.

МАРКОВСКИЕ ЦЕПИ

Временные процессы, отвечающие этим условиям, широко известны под названием *марковских цепей*.

Категория полноты состояния, в основном, имеет лишь теоретическую важность. На практике невозможно определить полное состояние для любой реалистичной робототехнической системы. По настоящему "полное"состояние будет включать не только все аспекты окружающего, которые могут повлиять на будущее, но также и самого робота, содержимое памяти его компьютера, образы мозга окружающих людей, и так далее. Некоторые из этих данных довольно трудно получить, поэтому практические реализации ограничены лишь небольшим набором из всех возможных переменных состояния. Такое состояние называется *неполным*.

НЕПОЛНОЕ СОСТОЯНИЕ

В большинстве задач практического использования роботов состояние x_t непрерывно и определено на непрерывном множестве. Хорошим примером непрерывного пространства состояний является положение робота, представляющее совокупность местоположения и ориентации во внешней системе координат. Иногда, впрочем, состояние дискретно. Примером дис-

кретного пространства состояний является, скажем, бинарная переменная состояния, моделирующая факт исправности датчика. Пространства состояний, которые содержат как непрерывные, так и дискретные переменные, называются *гибридными*.

В большинстве интересующих нас задач робототехники пространство изменяются со временем. В данной книге время будет считаться дискретным. Это означает, что все интересующие нас события произойдут в конкретные моменты времени $t = 0, 1, 2....$ Если робот начинает функционировать в определённый момент времени, будем считать, что для этого момента $t = 0$.

2.3.2 Взаимодействие со средой

Есть два основных типа взаимодействия между роботом и средой: робот может изменять состояние своего окружения с помощью приводов и собирать информацию об этом состоянии с помощью датчиков. Оба типа взаимодействия могут происходить одновременно, но, для удобства изложения, в книге они будут разделены. Взаимодействие показано на Рис. 2.1.

• **Измерение параметров окружающей среды датчиками**. Восприятие - это процесс получения данных о состоянии окружающей среды с помощью датчиков робота. Например, в целях получения необходимых данных, можно принимать изображение с камеры, определить дальность до препятствия, использовать тактильные датчики и так далее. Результат такого взаимодействия в целях восприятия будем называть *измерением*, хотя иногда будут использоваться такие термины, как *наблюдение* или *представление*. Обычно, информация измерений поступает с некоторой задержкой, и описывает состояние, которое было несколько мгновений назад.

ИЗМЕРЕНИЕ

• **Действия управления** изменяют состояние окружающего мира с помощью приложения сил к предметам окружающей среды. Примеры *управляющих действий* включают движение робота и манипулирование объектами. Даже если робот не выполняет никаких действий, состояние обычно изменяется. Поэтому, для сохранения целостности изложения, будем считать, что робот *всегда* выполняет управляющее действие вне зависимости от того, было ли принято решение перемещать какой-либо из приводов. На практике, робот непрерывно и одновременно выполняет управляющие действия и измерения.

УПРАВЛЯЮЩЕЕ ДЕЙСТВИЕ

Теоретически, робот может сохранять записи всех прошлых измерений и управляющих действий. Условимся называть такую совокупность информации *данными* (вне зависимости от того, были ли они сохранены). Поскольку выполняется два типа взаимодействий с окружающей средой, робот имеет доступ к двум разным потокам данных.

• **Данные измерений параметров окружающей среды** предоставляют информацию о текущем состоянии окружения. Примерами данных измерений служат изображение с камеры, определение расстояния и так далее. В большинстве разделов книги мы будем просто игнорировать малые эффекты запаздывания (например, большинство лазерных датчиков последовательно сканируют окружающее пространство с очень высокой скоростью, но мы просто будем считать, что имеется измерение, относящееся к конкретному моменту времени). Данные измерений в момент времени t бу-

дет обозначаться как z_t.
При изложении будем просто считать, что робот выполняет единственное измерение в один момент времени. Это допущение служит, в основном, для удобства обозначения, поскольку почти все алгоритмы книги могут быть легко расширены для роботов, которые получают произвольное число измерений в течение одного такта времени. Запись

(2.29)
$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, ..., z_{t_2}$$

обозначает набор всех переменных, полученных с момента времени t_1 до t_2, при этом $t_1 \leq t_2$.

• **Данные управления** содержат информацию об *изменении состояния* в окружающей среде. В мобильной робототехнике типичным примером данных управления является скорость робота. Установка значения скорости 10 см в секунду в течение пяти секунд предполагает, что после выполнения команды на движение робот переместится в положение, находящееся, приблизительно, на 50 см впереди от начального. Таким образом, сигнал управления передаёт информацию об изменении состояния. Дополнительным источником данных управления являются *одометры* – датчики, которые измеряют количество оборотов колес робота и, таким образом, передают информацию об изменении состояния. Хотя, строго говоря, одометры – это датчики, условимся считать показания одометра данными управления, поскольку с их помощью измеряется эффект управляющего действия.

ОДОМЕТР

Данные управления будут обозначаться как u_t. Переменная u_t будет всегда связана с изменением состояния в интервале времени $(t-1;t]$. Как и ранее, обозначим последовательности данных управления через $u_{t_1:t_2}$, для $t1 \leq t2$:

(2.30)
$$u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, ..., u_{t2}$$

Поскольку состояние среды может изменяться, даже если робот не выполняет никакого определённого действия, то, технически говоря, сам факт изменения времени содержит информацию об управлении. В силу этого, условимся, что каждый такт времени t происходит ровно одно действие управления, при этом возможно действие *«не делать ничего»*.
Разница между измерением и управлением критически важна, поскольку в материале, который будет излагаться, оба типа данных играют фундаментально различные роли. Восприятие окружающей среды предоставляет дополнительную информацию о её состоянии, и, таким образом, увеличивает осведомлённость робота. Движение, с другой стороны, часто приводит к потере информации из-за наличия собственных шумов приводов и случайных факторов окружающей среды. Такое разделение ни в коей мере не предполагает разделения во времени восприятия и действий. Восприятие и управление осуществляются одновременно, а их разделение преследует лишь цели удобства изложения.

2.3.3 Вероятностные генеративные правила

Процесс изменения состояний и измерений управляется правилами теории вероятности. В общем случае, состояние x_t стохастически возникает из состояния x_{t-1}. В силу этого, имеет смысл определить вероятностное распределение, из которого было сгенерировано x_t. На первый взгляд, появление

состояния x_t может быть обусловлено всеми предыдущими состояниями, измерениями и действиями управления. Следовательно, вероятностные правила, характеризующие эволюцию состояния, могут быть изложены в виде вероятностного распределения следующего вида: $p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t})$. Обратите внимание, что нет никакой причины считать, что робот сначала совершает действие управления u_1, а после этого – производит измерение z_1.

Важным является следующее суждение: Если состояние x полное, значит в нём в достаточной степени суммируется всё произошедшее на предыдущих тактах времени. В частности, x_{t-1} является достаточной статистической величиной для всех предыдущих действий управления и измерений вплоть до указанного момента времени, $u_{1:t-1}$ и $z_{1:t-1}$, соответственно. Из всех переменных, приведённых выше, если известно состояние x_{t-1}, только управление u_t будет иметь значение.

В терминах теории вероятности, это рассуждение можно выразить следующим равенством:

(2.31)
$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

Выраженное этим равенством свойство является примером *условной независимости*. Утверждается, что некоторые переменные независимы от других, если известны значения третьей группы переменных, называемых условными переменными. Условная независимость будет широко использована в книге, поскольку она является главным фактором вычислительной разрешимости большинства приведённых алгоритмов.

Может возникнуть желание смоделировать процесс на основании сгенерированных измерений. Здесь снова, если состояние x_t полное, наблюдается важная условная независимость:

(2.32)
$$p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t)$$

Другими словами, переменной состояния x_t достаточно для предсказания измерения z_t, даже если оно, вероятно, зашумлено. Знание любых других переменных, таких, как прошлые измерения, управление или даже прошлых состояний, роли не играет, пока соблюдается условие полноты x_t.

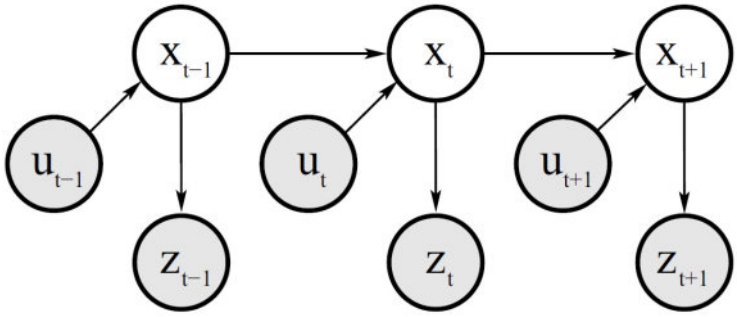


Рис. 2.2. Динамическая байесовская сеть, характеризующая эволюцию управления, состояний и измерений.

Это обсуждение раскрывает смысл двух результирующих условных вероятностей: $p(x_t|x_{t-1}, u_t)$ и $p(z_t|x_t)$. Вероятность $p(x_t|x_{t-1}, u_t)$ называется

ВЕРОЯТНОСТЬ ПЕРЕХОДА СОСТОЯНИЙ

вероятностью перехода состояний. Она определяет изменение состояния среды со временем в виде функции от управления роботом u_t. Окружающие среды робота имеют стохастический характер, поэтому $p(x_t|x_{t-1}, u_t)$ представляет собой вероятностное распределение, а не детерминированную функцию. В некоторых случаях распределение вероятности перехода состояний не зависит от временного индекса t, и в этом случае можно написать $p(x'|u, x)$, где x' последующее, а x – предыдущее состояние. Вероятность

ВЕРОЯТНОСТЬ ИЗМЕРЕНИЯ

$p(z_t|x_t)$ называется *вероятностью измерения*. Она также может не зависеть от показателя времени t, и, в этом случае, записываться как $p(z|x)$. Вероятность измерения определяет вероятностное соотношение, согласно которому измерения z генерируются из окружающей среды с состоянием x. Справедливо считать измерения зашумлёнными проекциями состояния.

Динамическая стохастическая система, состоящая из робота и его окружения описываются вероятностями перехода состояний и измерений. На Рис. 2.2 показаны развитие состояний и измерений, определённых этими вероятностями. Состояние в момент времени t стохастически зависимо от состояния в момент времени $t-1$ и управления u_t. Измерение z_t стохастически зависит от состояния в момент времени t. Такая временная генеративная модель также известна как *скрытая марковская модель* (hidden Markov model - HMM) или *динамическая байесовская сеть* (dynamic Bayes network - DBN).

2.3.4 Распределения оценок

ОЦЕНКА

Другой ключевой концепцией вероятностной робототехники является оценка. Оценка отражает внутреннюю осведомлённость робота о состоянии окружающей среды. Мы уже говорили, что состояние невозможно измерить напрямую. Например, положение робота может быть обозначено как $\langle x_t = 14.12, 12.7, 45^o\rangle$ в некоторой глобальной системе координат, но своего настоящего положения он, обычно, определить неспособен, поскольку положение невозможно измерить напрямую (даже с помощью GPS!). Вместо этого, робот вынужден делать *предположение* о своём положении на основании данных. Таким образом, необходимо разделять настоящее состояние и внутреннюю оценку этого состояния. Синонимами оценки в литературе являются термины *«знание о состоянии»* (state of knowledge) и *«информационное состояние»* (information state) (не путать с информационным вектором и информационной матрицей, которые будут обсуждаться ниже).

ИНФОРМАЦИОННОЕ СОСТОЯНИЕ

В вероятностной робототехнике оценки рассматриваются через распределения условной вероятности. Распределение оценки назначает вероятность (или значение плотности) для каждой возможной гипотезы относительно реального состояния. Распределения оценок являются апостериорными вероятностями переменных состояния, вычисленными на основании доступных данных. Условимся обозначать оценку переменной состояния x_t как $bel(x_t)$, в качестве сокращения для апостериорной вероятности

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{2.33}$$

Апостериорная вероятность - это вероятностное распределение по состоянию x_t в момент времени t, вычисленная на основании всех прошлых измерений $z_{1:t}$ и всех прошлых действий управления $u_{1:t}$.

Читатель может заметить, что оценка производится *после* получения измерения z_t. Время от времени более полезным будет вычислить апосте-

риорную вероятность сразу после выполнения управляющего действия u_t, но *до* включения в расчёт z_t. Такая апостериорная оценка будет иметь следующий вид:

(2.34)
$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$$

ПРОГНОЗ

Такое вероятностное распределение в контексте вероятностной фильтрации часто обозначается как *«прогноз»*. Этот термин отражает факт того, что $\overline{bel}(x_t)$ «прогнозирует» состояние в момент времени t, основываясь на предыдущих апостериорных состояниях, но до принятия в расчёт измерения в момент времени t. Вычисление $bel(x_t)$ из $\overline{bel}(x_t)$ называется коррекцией или обновлением измерения

2.4 Байесовские фильтры

2.4.1 Алгоритм байесовского фильтра

ФИЛЬТР БАЙЕСА

Самый общий алгоритм для вычисления оценок – это алгоритм *байесовского фильтра*. Этот алгоритм позволяет вычислить распределение оценок bel на основе данных измерения и управления. Рассмотренный в начале базовый алгоритм будет сначала проиллюстрирован на примере, а, затем мы выведем его математическим путём на основании сделанных допущений.

В Таблице 2.1 в псевдоалгоритмической форме приведён основной алгоритм байесовского фильтра. Байесовский фильтр рекурсивный, поэтому оценка $bel(x_t)$ в момент времени t вычислена

1: **Algorithm Bayes_filter** $(bel(x_{t-1}), u_t, z_t)$:
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$
4: $bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

Таблица 2.1 Общий алгоритм байесовской фильтрации.

на основании оценки $bel(x_{t-1})$ для момента времени $t-1$. На вход поступает оценка bel в момент времени $t-1$, наряду с последним значением управления u_t и последним измерением z_t. На выходе вычисляется значение оценки $bel(x_t)$ в момент времени t. В Таблице 2.1 приведена одна итерация алгоритма байесовского фильтра: *правило обновления*.

ОБНОВЛЕНИЕ БАЙЕСОВСКОГО ФИЛЬТРА

Обновление применяется рекурсивно для того, чтобы вычислить оценку $bel(x_t)$ из предварительно вычисленной оценки $bel(x_{t-1})$.

Алгоритм байесовского фильтра последовательно проходит два важных шага. В строке 3 обрабатывается управляющее воздействие u_t. Это происходит через оценку состояния по x_t на основе предыдущей оценки по состоянию x_{t-1} и управляющему воздействию u_t. В частности, оценка $\overline{bel}(x_t)$ перехода робота в состояние x_t, получается интегрированием (фактически,

суммированием) произведений двух распределений: априорного, полученного для x_{t-1}, и вероятности того, что управляющее воздействие u_t вызовет переход от x_{t-1} к x_t. Читатель может заметить схожесть такта обновления с Выражением (2.12). Как было сказано ранее, этот такт обновления называется обновлением управления или *прогнозом*.

Второй такт работы байесовского фильтра называется *обновлением измерения*. В строке 4 алгоритма выполняется умножение оценки $\overline{bel}(x_t)$ на вероятность того, что будет обнаружено наблюдение z_t. Это действие повторяется для каждого гипотетического апостериорного состояния x_t. Как станет очевидным далее, при выводе основных равенств алгоритма фильтрации, результирующее произведение, вообще-то, вероятностью не является, и может не интегрироваться до 1. В силу этого, результат необходимо нормализовать с помощью нормализующего члена η. Это даёт итоговую оценку $bel(x_t)$, значение которой и возвращается в строке 6 алгоритма.

Для рекурсивного вычисления апостериорной оценки, алгоритму требуется начальная оценка $bel(x_0)$ в момент времени $t = 0$ в качестве граничного условия. Если известно точное значение x_0, $bel(x_0)$ следует инициализировать переменную, сконцентрировав вероятность на верном значении x_0, и

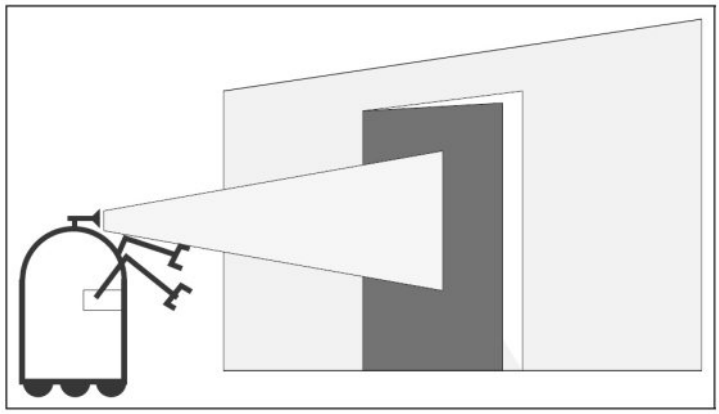


Рис. 2.3 Мобильный робот определяет состояние двери.

установив нуль во всех остальных точках. Если же значение x_0 совершенно неизвестно, оценку $bel(x_0)$ можно инициализировать равномерным распределением в окрестности x_0 (или применимым распределением Дирихле). Частичную информацию о значении x_0 возможно выразить с помощью неравномерного распределения, хотя на практике наиболее часто встречаются именно эти два случая полной осведомлённости или полного незнания.

В приведённой форме алгоритм байесовского фильтра пригоден только для очень простых оценочных задач. В частности, необходимо или гарантировать закрытую форму интегрирования в строке 3 и умножения в строке 4, или ограничиться конечными пространствами состояний, чтобы интеграл в строке 3 свёлся к конечной сумме.

2.4.2 Пример

Наша иллюстрация алгоритма байесовской фильтрации основана на сценарии из Рис. 2.3, где показан робот, который оценивает состояние двери с помощью камеры. Чтобы упростить задачу, давайте предположим, что

дверь может быть только в одном из двух возможных состояний, открытом или закрытом, при этом только робот может изменить ее состояние. Допустим также, что, изначально, робот ничего не знает о текущем состоянии двери. Чтобы выразить это, назначим равную априорную вероятность для двух возможных состояний двери:
$bel(X_0 = \text{is\_open}) = 0,5$
$bel(X_0 = \text{is\_closed} = 0,5$
Также допустим, что датчики робота подвержены зашумлению. Шум характеризуется следующими условными вероятностями:
$p(Z_t = \text{sense\_open}|X_t = \text{is\_open}) = 0,6$
$p(Z_t = \text{sense\_closed}|X_t = \text{is\_open}) = 0,4$
и
$p(Z_t = \text{sense\_open}|X_t = \text{is\_closed}) = 0,2$
$p(Z_t = \text{sense\_closed}|X_t = \text{is\_closed}) = 0,8$
Очевидно, что датчики робота довольно надёжно распознают закрытую дверь, и вероятность ошибки составляет 0,2. Однако, когда дверь открыта, имеется существенная вероятность ошибочного измерения, равная 0,4 .

И, наконец, давайте допустим, что робот может толкать дверь манипулятором, чтобы открыть ее. Если дверь уже была открыта, она остаётся открытой. Если же она была закрыта, после действия манипулятором робота она откроется с вероятностью 0,8:
$p(X_t = \text{is\_open}|U_t = \text{push}, X_{t\_1} = \text{is\_open}) = 1$
$p(X_t = \text{is\_closed}|U_t = \text{push}, X_{t\_1} = \text{is\_open}) = 0$
$p(X_t = \text{is\_open}|U_t = \text{push}, X_{t\_1} = \text{is\_closed}) = 0,8$
$p(X_t = \text{is\_closed}|U_t = \text{push}, X_{t\_1} = \text{is\_closed}) = 0,2$
Робот также может принять решение не использовать манипулятор, и тогда состояние окружающего мира не меняется. Этот случай описывается следующими вероятностями:
$p(X_t = \text{is\_open}|U_t = \text{do\_nothing}, X_{t\_1} = \text{is\_open}) = 1$
$p(X_t = \text{is\_closed}|U_t = \text{do\_nothing}, X_{t\_1} = \text{is\_open}) = 0$
$p(X_t = \text{is\_open}|U_t = \text{do\_nothing}, X_{t\_1} = \text{is\_closed}) = 0$
$p(X_t = \text{is\_closed}|U_t = \text{do\_nothing}, X_{t\_1} = \text{is\_closed}) = 1$
Допустим, в момент времени $t = 1$, робот не выполняет никаких управляющих действий, но обнаруживает открытую дверь. В результате, апостериорная оценка вычисляется байесовским фильтром, используя в качестве входных значений априорную оценку $bel(X_0)$, управляющее воздействие $u1 = \text{do\_nothing}$, и измерение sense_open. Поскольку пространство состояний конечно, интеграл в строке 3 сводится до конечной суммы:
$\overline{bel}(x_1)$
$= \int p(x_1|u_1, x_0)bel(x_0)dx_0$
$= \sum_{x_0} p(x_1|u_1, x_0)bel(x_0)$
$= p(x_1|U_1 = \text{do\_nothing}, X_0 = \text{is\_open})bel(X_0 = \text{is\_open})$
$+ p(x_1|U_1 = \text{do\_nothing}, X_0 = \text{is\_closed})bel(X_0 = \text{is\_closed})$

Теперь можно заменить два возможных значения переменной состояния X_1. Для гипотезы $X_1 = \text{is\_open}$, получаем
$\overline{bel}(X_1 = \text{is\_open})$
$= p(X_1 = \text{is\_open}|U_1 = \text{do\_nothing}, X_0 = \text{is\_open})bel(X_0 = \text{is\_open})$
$+ p(X_1 = \text{is\_open}|U_1 = \text{do\_nothing}, X_0 = \text{is\_closed})bel(X_0 = \text{is\_closed})$
$= 1 \times 0.5 + 0 \times 0.5 = 0.5$

Аналогично, для $X_1 = \text{is\_closed}$ получается
$\overline{bel}(X_1 = \text{is\_closed})$

$= p(X_1 = \text{is\_closed}|U_1 = \text{do\_nothing}, X_0 = \text{is\_open})bel(X_0 = \text{is\_open})$
$+ p(X_1 = \text{is\_closed}|U_1 = \text{do\_nothing}, X_0 = \text{is\_closed})bel(X_0 = \text{is\_closed})$
$= 0 \times 0.5 + 1 \times 0.5 = 0.5$

Факт равенства оценки $\overline{bel}(x_1)$ и априорной оценки $bel(x_0)$ не должен удивлять, поскольку действие do_nothing не влияет на состояние окружающего мира, а сам окружающий мир в нашем примере неизменен по времени.

Однако, когда в расчёт принимается измерение, оценка изменяется. В строке 4 алгоритма байесовского фильтра

$$bel(x_1) = \eta p(Z_1 = \text{sense\_open}|x_1)\overline{bel}(x_1)$$

Для двух возможных случаев, $X_1 = \text{is\_open}$ и $X_1 = \text{is\_closed}$, получаем
$bel(X_1 = \text{is\_open})$
$= \eta p(Z_1 = \text{sense\_open}|X_1 = \text{is\_open})\overline{bel}(X_1 = \text{is\_open})$
$= \eta \cdot 0.6 \times 0.5 = \eta \cdot 0.3$

и, соответственно
$bel(X_1 = \text{is\_closed})$
$= \eta p(Z_1 = \text{sense\_open}|X_1 = \text{is\_closed})\overline{bel}(X_1 = \text{is\_closed})$
$= \eta \cdot 0.2 \times 0.5 = \eta \cdot 0.1$

Теперь легко вычислить нормализующий член η:
$\eta = (0.3 + 0.1)^{-1} = 2.5$

Отсюда, получаем:
$bel(X_1 = \text{is\_open}) = 0,75$
$bel(X_1 = \text{is\_closed}) = 0,25$

Эти вычисления очень просто повторяются на следующем такте времени. Как читатель может убедиться, для $u_2 = \text{push}$ и $z_2 = \text{sense\_open}$, получаем

$\overline{bel}(X_2 = \text{is\_open}) = 1 \times 0,75 + 0,8 \times 0,25 = 0,95$
$\overline{bel}(X_2 = \text{is\_closed}) = 0 \times 0,75 + 0,2 \times 0,25 = 0,05$
и
$bel(X_2 = \text{is\_open}) = \eta 0,6 \times 0,95 \approx 0,983$
$bel(X_2 = \text{is\_closed}) = \eta 0,2 \times 0,05 \approx 0,017$

К этому моменту оценка робота указывает на то, что, дверь открыта с вероятностью 0,983.

На первый взгляд, эта вероятность может показаться достаточно большой, чтобы просто принять эту гипотезу в качестве состояния окружающего мира и действовать соответствующим образом. Однако, такой подход может стать неоправданно дорогим. Если ошибка перепутать закрытую дверь с открытой имеет цену (например, робот врежется в закрытую дверь), очень важно принять во внимание обе гипотезы, какой бы невероятной одна из них не была. Просто представьте, каково будет лететь на самолёте, автопилот которого способен избежать катастрофы с вероятностью 0,983!

2.4.3 Математический вывод для байесовского фильтра

Продемонстрируем правильность алгоритма байесовской фильтрации с помощью индукции. Чтобы это сделать, необходимо показать, что он верно вычисляет апостериорное распределение $p(x_t|z_{1:t}, u_{1:t})$ из соответствующего апостериорного же распределения $p(x_{t-1}|z_{1:t-1}, u_{1:t-1})$, но взятого на один такт ранее. Правильность доказывается с помощью индукции при условии верной инициализации априорной оценки $bel(x_0)$ в момент времени $t = 0$.

Для вывода требуется, чтобы состояние x_t было полным, как было определено в Главе 2.3.1, а управляющие воздействия – выбирались случайным образом. Первый этап нашего вывода включает применение формулы Байеса (2.16) к целевой апостериорной вероятности:

(2.35)

$$p(x_t|z_{1:t}, u_{1:t}) = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})}$$

$$= \eta p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})$$

Теперь используем допущение о том, что состояние полное. В Главе 2.3.1, мы определили x_t как полное, если никакие переменные до x_t не могут повлиять на стохастическую эволюцию будущих состояний. В частности, если мы (предположительно) знаем состояние x_t, и заинтересованы в предсказании измерения z_t, никакие измерения или управляющие действия не предоставят дополнительной информации. Математически это можно выразить следующей условной независимостью:

(2.36)

$$p(z_t|x_t, z_{1:t-1}, u_{1:t}) = p(z_t|x_t)$$

Это утверждение – ещё один пример условной независимости. Оно позволяет упростить (2.35) следующим образом:

(2.37)

$$p(x_t|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t})$$

и получить

(2.38)

$$bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$$

Это равенство используется в строке 4 алгоритма байесовского фильтра в Таблице 2.1.
Далее, расширим значение $\overline{bel}(x_t)$, используя (2.12):

(2.39)

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t})$$

$$= \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})p(x_{t-1}|z_{1:t-1}, u_{1:t})dx_{t-1}$$

И снова используется допущение о том, что состояние – полное. Это предполагает, что, если мы знаем x_{t-1}, прошлые измерения и действия управления не дают дополнительной информации относительно состояния x_t. Это позволяет

(2.40)

$$p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t)$$

Переменная u_t сохраняется, поскольку она не относилась к состоянию x_{t-1}. Фактически, читатель может быстро убедиться, что $p(x_t|x_{t-1}, u_t) \neq p(x_t|x_{t-1})$.

Наконец, заметим, что переменную управляющего воздействия для случайно выбранных управляющих воздействий u_t можно исключить из набора условных переменных в $p(x_{t-1}|z_{1:t-1}, u_{1:t})$.
Это даёт рекурсивно обновляемое равенство

(2.41)

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|z_{1:t-1}, u_{1:t-1})dx_{t-1}$$

Легко убедиться, что это же выражение используется в строке 3 алгоритма байесовской фильтрации, приведённого в Таблице 2.1.

Подведём итог. Байесовский алгоритм фильтра вычисляет апостериорную вероятность по состоянию x_t, при условии наличия данных измерений и управления вплоть до момента времени t. Вывод подразумевает, что окружающий мир задан согласно марковской модели, а, значит, состояние полное. Любая практическая реализация этого алгоритма требует трёх вероятностных распределений: первоначальной оценки $p(x_0)$, вероятности измерения $p(z_t|x_t)$, и вероятности изменения состояния $p(x_t|u_t, x_{t-1})$. Мы ещё не определили эти плотности для реальных робототехнических систем, но скоро это сделаем: Глава 5 полностью посвящена $p(x_t|u_t, x_{t-1})$, а Глава 6 - $p(z_t|x_t)$. Нам также понадобится выражение для оценки $bel(x_t)$, которое будет обсуждаться в Главах 3 и 4.

2.4.4 Марковское свойство

МАРКОВСКОЕ СВОЙСТВО

Настало время рассказать о *марковском свойстве* или *свойстве полного состояния*, поскольку оно играет фундаментальную роль в материале, представленном в книге. Марковское свойство постулирует независимость прошлых и будущих данных, если известно текущее состояние x_t. Чтобы увидеть, насколько важно это свойство, вернёмся к нашему примеру локализации мобильного робота. Напомним, что x_t – это расположение робота, а для оценки местоположения на неизменяемой карте используются байесовские фильтры. Марковское свойство может нарушаться в результате воздействия на датчик следующих факторов:

• часть динамики окружающей среды не была смоделирована и не включена в x_t (например, движущиеся люди и эффекты их действия на измерения датчиков в нашем примере локализации),

• неточности вероятностных моделей $p(z_t|x_t)$ и $p(x_t|u_t, x_{t-1})$ (например, ошибки в карты для определения местоположения робота)

• ошибки аппроксимации при использовании приблизительных представлений функций оценки (например, сетки или гауссовы функции, которые будут обсуждаться ниже), а также

• программные переменные в управляющем программном обеспечении робота, которые влияют на несколько управляющих действий (например,

переменная «расположение цели» обычно влияет на всю последовательность управляющих команд).

В принципе, многие из этих переменных могут быть включены в представления состояния. Однако, представления неполного состояния часто предпочтительны более полным из-за уменьшения вычислительной сложности алгоритма байесовского фильтра. На практике байесовские фильтры показывают удивительную устойчивость к такого рода нарушениям. При определении состояния x_t рекомендуется уделить этому вопросу особое внимание, поскольку не включённые в модель переменные могут вызывать непредсказуемые, почти случайные эффекты.

2.5 Представление и вычисление

В вероятностной робототехнике байесовские фильтры применяются несколькими разными способами. Как мы увидим в двух следующих главах, существует достаточно много разнообразных методов и алгоритмов на основе байесовского фильтра. Каждый из этих методов основан на различных допущениях относительно вероятностей измерений и перехода состояний, а также первоначальной оценки. Используя эти допущения, порождаются разнообразные типы апостериорных распределений для алгоритмов, отличающихся вычислительными характеристиками. В общем и целом, точные методы вычислений оценок существуют только для очень узких задач и, почти всегда, оценки предстоит аппроксимировать. Способ этой аппроксимации оказывает определяющее воздействие на сложность алгоритма. Нахождение подходящего способа аппроксимации представляет собой весьма сложную проблему, в которой нет единого универсального решения для всех задач робототехники.

При выборе аппроксимации приходится находить баланс между несколькими показателями:

1. **Вычислительная эффективность.** Некоторые способы аппроксимации, такие как линейные гауссовские методы, которые будут обсуждаться ниже, дают возможность вычисления оценки за время, кратное размерности пространства состояний. Другие алгоритмы могут потребовать времени, экспоненциально зависящего от размерности. Настраиваемые методы многочастичной фильтрации имеют переменную характеристику затрат времени, позволяя жертвовать точностью ради вычислительной эффективности.

2. **Точность аппроксимации.** Некоторые виды приближения могут аппроксимировать широкий набор распределений с большей точностью. Например, линейное гауссовское приближение ограничено одномодальными распределениями, а частотные представления способны аппроксимировать мультимодальные распределения, хотя и с потерей точностью. Многочастичные представления дают возможность аппроксимации в широком диапазоне распределений, но количество частиц, необходимых для получения требуемой точности, может быть весьма большим.

3. **Простота использования.** Трудность применения вероятностных алгоритмов на практике зависит от целого ряда факторов, таких, как форма записи вероятности измерения $p(z_t|x_t)$ и вероятности перехода состояния

$p(x_t|u_t, x_{t-1})$. Многочастичные представления для сложных нелинейных систем часто имеют удивительно простые реализации, что является одной из причин их сегодняшней популярности.

В следующих двух главах будут представлены конкретные алгоритмы реализации, которые довольно сильно различаются по описанным выше критериям.

2.6 Выводы

В этом разделе была представлена общая идея использования в робототехнике байесовских фильтров для оценки состояния окружающей среды и робота.

• Взаимодействие робота с окружающей средой смоделировано в виде связанной динамической системы, в которой робот управляет окружающей средой путём выбора управляющих действий и воспринимает ее характеристики с помощью датчиков.

• В вероятностной робототехнике динамика робота и окружающей среды описывается в виде двух вероятностных закономерностей: распределения перехода между состояниями и распределения измерения. Распределение перехода между состояниями показывает зависимость изменения состояния в результате управляющих действий со временем. Распределение измерений показывает, каким образом измерения зависят от состояния. Оба правила имеют вероятностную природу и позволяют принимать в расчёт изначальную неопределённость в оценке состояния и восприятия.

• *Оценка* робота – это апостериорное распределение по состоянию в окружающей среде (включая состояние робота) на основании всех прошлых измерений и управляющих действий. *Байесовский фильтр* представляет собой основной алгоритм для вычисления оценки в робототехнике. Он имеет рекурсивную природу: оценка в момент времени t вычисляется на основе оценок в предыдущий момент времени $t-1$.

• Байесовский фильтр основан на *марковском свойстве*, согласно которому текущее состояние является полным описанием всех прошлых. Это допущение предполагает, что оценка достаточно полно отображает все произошедшее с роботом. В робототехнике марковское свойство обычно является лишь приближением и есть определённые условия, при которых оно нарушается.

• Поскольку байесовский фильтр не является практическим алгоритмом и, в таком виде, не может быть реализован на компьютере, вероятностные алгоритмы используют управляемые приближения. Эти приближения могут быть оценены по различным критериям, например, точности, эффективности и простоте реализации.

В следующих двух главах мы обсудим два популярных семейства рекурсивных методов оценки состояния на основе байесовского фильтра.

2.7 Библиографические сведения

Базовый материал по статистике из этой главы освещается в большинстве учебников начального уровня по статистике и теории вероятности. Некоторые ранние классические тексты ДеГрута (DeGroot, 1975), Сабраманиана (Subrahmaniam, 1979) и Торпа (Thorp, 1966) позволяют очень быстро овладеть основными понятиями. Подробности можно найти в работах Феллера, Каселла и Бергера, Таннера (Feller 1968; Casella и Berger 1990; Tanner 1996), а также Девро и Дуда (Devroye et al. 1996; Duda et al. 2000). Общепринятая в робототехнике парадигма взаимодействия робота с окружающей средой обсуждается с точки зрения категорий искусственного интеллекта Расселом и Норвигом (Russell and Norvig, 2002).

2.8 Упражнения

1. Робот использует датчик, определяющий расстояния в диапазоне 0 - 3 м. Для простоты, допустим, что реальные значения расстояния равномерно распределены в этом интервале. К сожалению, датчик может быть неисправен. При неисправности датчика постоянно выдаются значения менее 1 м, вне зависимости от реального расстояния до объекта в конусе измерения робота. Известно, что априорная вероятность неисправности датчика $p = 0,01$.
Допустим, робот запрашивает датчик N раз, и каждое полученное значение измерения составляет менее 1м. Какова апостериорная вероятность неисправности датчика для $N = 1, 2, ..., 10$? Сформулируйте соответствующую вероятностную модель.

2. Допустим, мы проживаем в месте, где погода в течение дня может быть только солнечной, облачной или дождливой. Функция изменения погоды представляет собой цепь Маркова со следующей таблицей переходов:

	Завтра будет. . .		
	солнечно	облачно	дождливо
солнечно	0.8	0.2	0
сегодня. . .облачно	0.4	0.4	0.2
дождливо	0.2	0.6	0.2

(а) Допустим, День 1 - солнечный. Какова вероятность появления следующей последовательности дней: День 2 = *облачный*, День 3 = *облачный*, День 4 = *дождливый*?

(б) Написать программу-симулятор, которая случайным образом генерирует последовательности «погоды» из функции перехода состояний.

(в) Использовать симулятор для определения стационарного распределения данной марковской цепи. Стационарное распределение измеряет вероятность того, что случайно взятый день окажется солнечным, облачным или дождливым.

(г) Возможно ли определить закрытую форму решения для вычисления стационарного распределения на основании матрицы перехода состояний, приведённой выше?

(д) Какова энтропия стационарного распределения?

(е) Используя формулу Байеса, вычислить таблицу вероятности для вчерашней погоды по известной погоде на сегодня. (Разрешено представлять вероятности в числовом виде, а также использовать результаты предыдущих вопросов этого упражнения.)

(ж) Допустим, к модели добавили времена года. Имеющаяся функция перехода состояний выше применима только для Лета, а для Зимы, Весны и Осени используются другие функции. Нарушит ли это марковское свойство этого процесса? Объяснить ответ.

3. Допустим, узнать погоду явным образом невозможно, и мы вынуждены полагаться на датчик. Проблема состоит в том, что показания датчика зашумлены и определяются следующей моделью измерений:

	Показания датчика. . .		
	солнечно	облачно	дождливо
солнечно	0.6	0.4	0
текущая погода. . .облачно	0.3	0.7	0
дождливо	0	0	1

(а) Допустим, День 1 солнечный (это известно точно), а в последующие четыре дня датчик показал *облачно, облачно, дождливо, солнечно*. Какова вероятность, что День 5 действительно солнечный, как показывает датчик?

(б) Допустим, День 1 солнечный. В течение дней 2-4 датчик показывает *солнечно, солнечно, дождливо*. Для каждого из дней со второго по четвёртый, какая погода будет наиболее вероятна? Ответить на вопрос двумя способами: основываясь только на данных, доступных на нужный день, и используя перспективу, когда данные будущих дней также доступны.

(в) В той же самой ситуации (День 1 солнечный, измерения для Дней 2, 3, и 4 *солнечный, солнечный, дождливый*). Какая последовательность погоды для дней со второго по четвёртый наиболее вероятна? Какова вероятность для наиболее вероятной последовательности?

4. В этом упражнении применим теорему Байеса к гауссовским распределениям. Допустим, имеется мобильный робот, который находится на длинной прямой дороге и x обозначает его положение. Будем считать, что, по первоначальной оценке, робот находится на отметке $x_{init} = 1000$ м, но известно, что оценка неточная. Основываясь на этой неопределённости, смоделируем первоначальную оценку для гауссовского распределения со среднеквадратичным отклонением $\sigma^2_{init} = 900m^2$.

Чтобы уточнить местоположение, запросим данные с GPS приёмника. Показания GPS $z_{GPS} = 1100$ м. Известно, что GPS приёмник имеет среднеквадратичную ошибку $\sigma^2_{init} = 100m^2$.

(а) Найдите функции плотности вероятности для априорной вероятности $p(x)$ и измерений $p(z|x)$.

(б) Используя теорему Байеса, определить апостериорную вероятность $p(x|z)$. Можно ли доказать, что функция гауссова?

(в) Насколько правдоподобно было априорное измерение x_{GPS}= 1100 м и какова вероятность ошибки GPS приёмника?

Подсказка: Это упражнение на использование квадратичных выражений.

5. Вывести Уравнения (2.18) и (2.19) из (2.17) и законов теории вероятности, описанных в тексте.

6. Доказать Уравнение (2.25). Какие выводы можно сделать из этого выражения?

3 Фильтры Гаусса

3.1 Введение

В этой главе описывается важное семейство рекурсивных методов оценки состояния под общим названием «фильтры Гаусса». Исторически, гауссовы фильтры образуют класс самых ранних управляемых реализаций байесовских фильтров для непрерывных пространств. Несмотря на имеющиеся недостатки, они также являются наиболее популярным семейством методов на сегодняшний день.

Все гауссовы методы используют основную идею представления оценок в виде многомерных нормальных распределений. Мы уже сталкивались с определением многомерного нормального распределения в выражении (2.4), но, для удобства, приведём его ещё раз

(3.1)
$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

Эта плотность по переменной x характеризуется двумя наборами параметров: математическим ожиданием μ и ковариационной матрицей Σ. Математическое ожидание μ - это вектор, имеющий ту же размерность, что и вектор состояний x. Ковариационная матрица представляет собой квадратную симметричную неотрицательно определённую матрицу, размерность которой равна квадрату размерности вектора состояний x. В силу этого число элементов в ковариационной матрице квадратично зависит от количества элементов в векторе состояний.

Стремление представить апостериорные вероятности в виде нормального распределения имеет важные последствия. Важнее всего то, что гауссовы распределения одномодальны – они имеют единственный максимум. Такая апостериорнная вероятность характерна для многих проблем отслеживания состояния в робототехнике, она концентрируется вокруг реального состояния с небольшим диапазоном неопределённости. Нормальные апостериорные распределения плохо подходят для глобальных задач оценки, где имеется множество отдельных гипотез, каждая из которых образует собственную моду в апостериорном распределении.

ПАРАМЕТРИЗАЦИЯ В ВИДЕ МОМЕНТОВ

Задание параметров нормального распределения в виде математического ожидания и ковариационной матрицы называется параметризацией в виде моментов, поскольку математическое ожидание и ковариация – это моменты первого и второго порядка вероятностного распределения (для нормальных распределений все остальные моменты равны нулю).

КАНОНИЧЕСКАЯ ПАРАМЕТРИЗАЦИЯ

В данной главе мы также обсудим другой способ задания параметров, называемый канонической параметризацией, или, иногда, обычной параметри-

зацией. Оба вида параметризации, каноническая и в виде моментов, функционально эквиваленты и образуют взаимно однозначные выражения, которые можно трансформировать друг в друга, но имеют несколько различные вычислительные характеристики. Как мы увидим, канонический и натуральный способ задания параметров лучше всего воспринимать как взаимодополняющие: то, что выглядит вычислительно лёгким в одном виде параметризации, используется в другом, и наоборот. В этой главе вводятся два основных алгоритма фильтра Гаусса.

• В подразделе 3.2 описан фильтр Калмана, реализующий байесовский фильтр с заданием параметров в виде моментов для решения ограниченного класса задач с линейной динамикой и функциями измерения.

• В подразделе 3.3 описан обобщенный фильтр Калмана, алгоритм которого приспособлен для решения нелинейных задач.

• В подразделе 3.4 приводится описание другого нелинейного фильтра Калмана, известного как unscented Kalman filter.

• В подразделе 3.5 описан информационный фильтр, дополняющий фильтр Калмана, используя каноническую параметризацию гауссовых функций.

3.2 Фильтр Калмана

3.2.1 Линейные гауссовы системы

Возможно, наиболее известный метод использования байесовских фильтров – это фильтр Калмана или ФК (KF). Фильтр Калмана был открыт Свирлингом (Swerling, 1958) и Калманом (Kalman, 1960) как способ фильтрации и прогнозирования для *линейных гауссовых систем*, которые определение которых сейчас будет приведено. Фильтр Калмана выполняет вычисление оценок только для непрерывных состояний и неприменим для дискретных или гибридных пространств состояний.

АПОСТЕРИОРНАЯ ГАУССОВА ВЕРОЯТНОСТЬ

Фильтр Калмана отображает оценки в виде моментов: для момента времени t, оценка представлена математическим ожиданием μ_t и ковариационной матрицей Σ_t. Апостериорные вероятности представлены *функциями Гаусса* если, вдобавок к марковским допущениям для байесовских фильтров, соблюдаются 3 следующих свойства:

1. Аргументы переходной вероятности состояния $p(x_t|u_t, x_{t-1})$ должны быть линейными функциями с добавочным гауссовским шумом. Это выражается следующим равенством:

(3.2)
$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

Здесь x_t и x_{t-1} - векторы состояния, а u_t – вектор управления в момент времени t. В данной записи оба вектора имеют вид вертикальной матрицы

(3.3)
$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{pmatrix} \quad \text{и} \quad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{pmatrix}$$

A_t – это квадратная матрица размера $n \times n$, где n - размерность вектора состояний x_t. Прямоугольная матрица B_t имеет размер $n \times m$, где m

–размерность вектора управления u_t. После перемножения векторов состояния и управления с матрицами A_t и B_t, соответственно, функции перехода состояний приобретает *линейные* аргументы. Таким образом, в фильтрах Калмана достигается линейная динамика системы.

Случайная переменная ε_t в (3.2) представляет собой случайный гауссовый вектор, предназначенный для моделирования неопределённости, привносимой переходом состояний. Он имеет ту же размерность, что и вектор состояний, его математическое ожидание равно нулю, а ковариацию обозначим через R_t. Вероятность перехода состояний вида (3.2) называется *линейным гауссианом*, отражая факт линейности аргументов с добавлением гауссового шума. Технически, в (3.2) возможно прибавить и дополнительную константу, но в излагаемом материале она никакой роли не играет, поэтому её можно опустить.

Выражение (3.2) определяет переходную вероятность состояния $p(x_t|u_t, x_{t-1})$. Эта вероятность получается после вставки уравнения (3.2) в определение многомерного нормального распределения (3.1). Математическое ожидание апостериорного состояния задано в виде $A_t x_{t-1} + B_t u_t$, а ковариация - R_t:

(3.4)

$$p(x_t|u_t, x_{t-1}) = det(2\pi R_t)^{-\frac{1}{2}} \\ exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\}$$

1: **Algorithm Kalman_filter** $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

2: $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
3: $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

4: $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
7: *return* μ_t, Σ_t

Таблица 3.1 Алгоритм фильтра Калмана для линейных гауссовых переходов состояний и измерений.

2. Вероятность измерения $p(z_t|x_t)$ также должна иметь *линейные аргументы* с добавленным гауссовым шумом:

(3.5)

$$z_t = C_t x_t + \delta_t$$

Здесь C_t - это матрица размера $k \times n$, где k – размерность вектора измерений z_t. Вектором δ_t описывается шум измерений. Распределение δ_t представляет собой многомерную нормальную функцию с нулевым математическим ожиданием и ковариацией Q_t. Отсюда, вероятность измерения задаётся следующим многомерным нормальным распределением:

(3.6)

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\}$$

3. И, наконец, первоначальная оценка $bel(x_0)$ должна быть нормально распределена. Обозначим математическое ожидание этой величины μ_0, а ковариацию - Σ_0:

(3.7)

$$bel(x_0) = p(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\right\}$$

Этих трёх условий достаточно, чтобы гарантировать, что апостериорное распределение $bel(x_t)$ для любого момента времени t всегда будет гауссовой функций. Доказательство этого нетривиального утверждения можно найти ниже, в математическом выводе для фильтра Калмана (подраздел 3.2.4).

3.2.2 Алгоритм фильтра Калмана

Алгоритм фильтра Калмана приводится в Таблице 3.1. Фильтры Калмана выражают оценку $bel(x_t)$ в момент времени t через математическое ожидание μ_t и ковариацию Σ_t. На вход принимается оценка в момент времени $t-1$, выраженная через μ_{t-1} и Σ_{t-1}. Для обновления этих параметров требуются также управляющее воздействие u_t и измерение z_t. Выводом является оценка в момент времени t, выраженная через μ_t и Σ_t.

В строках 2 и 3 вычисляются прогнозируемые значения $\bar{\mu}$ и $\bar{\Sigma}$, выражая оценку $\overline{bel}(x_t)$ для следующего такта времени, но до учёта измерений z_t. Эта оценка получается путём учёта управляющего воздействия u_t. Математическое ожидание обновляется, используя детерминированную версию функции перехода состояния (3.2), с математическим ожиданием μ_{t-1}, заменённую для состояния x_{t-1}. Обновление ковариации учитывает факт зависимости новых состояний от предыдущих, отражённый в линейной матрице A_t. Эта матрица дважды умножается на значение ковариации, поскольку ковариация выражена квадратичной матрицей.

В строках с 4 по 6 происходит последовательное преобразование оценки $\overline{bel}(x_t)$ в требуемую оценку $bel(x_t)$ с учётом измерения z_t. Переменная K_t, вычисленная в строке 4, называется *усилением фильтра Калмана*. Она определяет степень, до которой измерение учитывается в новой оценке состояния. Как именно это происходит - будет разъяснено в подразделе 3.2.4. Строка 5 управляет математическим ожиданием, меняя его пропорционально усилению фильтра Калмана K_t, отклонению текущего измерения z_t, а также прогнозируемому измерению (3.5).

УСИЛЕНИЕ ФИЛЬТРА КАЛМАНА

КОРРЕКЦИЯ

Ключевой концепцией здесь является *коррекция*, которая представляет собой разницу между реальным измерением z_t и ожидаемым измерением $C_t\bar{\mu}_t$ в строке 5. Наконец, в строке 6 вычисляется новая ковариационная матрица апостериорной оценки, учитывающая добавочную информацию, полученную в результате измерения.

Фильтр Калмана довольно эффективен в вычислительном смысле. Для лучших, на сегодняшний день, алгоритмов сложность инверсии матрицы размером $d \times d$ составляет, приблизительно, $O(d^{2.4})$. Каждая итерация фильтра Калмана, как уже было сказано, ограничена снизу сложностью (приблизительно) $O(k^{2.4})$, где k – размерность вектора измерений z_t. Эта, (снова,

приблизительно), кубическая степень сложности получается в силу инверсии матрицы в строке 4. Даже для некоторых случаев с разреженным обновлением, описанных в будущих главах, она составляет, по крайней мере $O(n^2)$, где n – размерность пространства состояний, из-за перемножения в строке 6 (матрица $K_t C_t$ может быть разреженной). Для многих реализаций (например, задачи построения карт, обсуждаемой в более поздних главах) пространство измерений имеет намного меньшую размерность по сравнению с пространством состояний, и такт обновления, в основном, состоит из операций сложности $O(n^2)$.

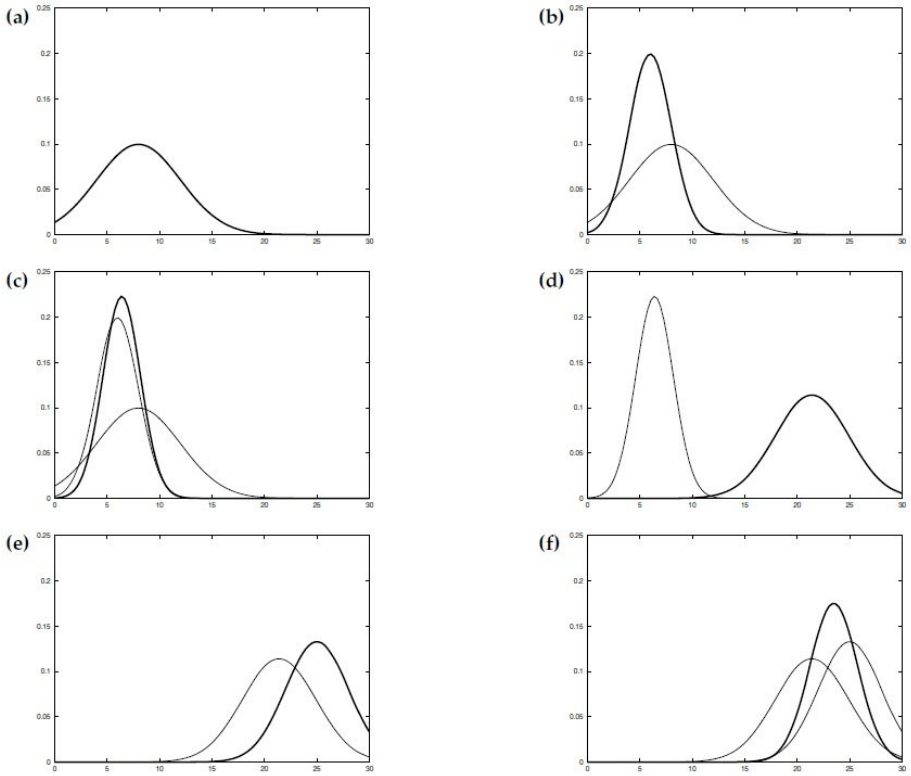


Рис. 3.2 Иллюстрация работы фильтров Калмана: (a) первоначальная оценка, (b) измерение (выделено жирным) с соответствующей неопределённостью, (c) оценка после учёта измерения алгоритмом фильтра Калмана, (d) оценка после движения вправо (которое вносит дополнительную неопределённость), (e) новое измерение с соответствующей неопределённостью, и (f) результирующая оценка.

3.2.3 Иллюстрация

На Рис. 3.2 показан алгоритм фильтра Калмана для простейшего одномерного сценария локализации. Допустим, на каждой диаграмме Рис. 3.2 робот двигается вдоль горизонтальной оси. Пусть априорная оценка местоположения робота задана нормальным распределением, показанным на Рис. 3.2a. Робот запрашивает данные своего местоположения с датчиков (например, системы GPS), и возвращает результат измерения с центром на пике выделенной на графике жирным функции Гаусса на Рис. 3.2b. График гауссовой

функции раскрывает представление измерения: пик обозначает прогнозируемое согласно показаниям датчиков значение, а ширина (дисперсия) связана с неопределённостью измерения. Результат комбинирования априорной оценки и измерения, выполненный в строках с 4 по 6 алгоритма фильтра Калмана в Таблице 3.1, приведён в виде жирного графика гауссианы на Рис. 3.2c. Значение математического ожидания этой оценки лежит между двух первичных пиков, а радиус неопределённости - меньше, чем у обоих предыдущих гауссовых функций. Факт того, что остаточная неопределённость меньше, чем у исходных нормальных распределений, может показаться контринтуитивным, но он является общей характеристикой интеграции информации в калмановских фильтрах.

Далее, допустим, робот движется направо. Степень неопределённости возрастает в силу стохастической природы перехода состояния. Строки 2 и 3 калмановского фильтра дают гауссову кривую, показанную жирным на Рис. 3.2d. Эта кривая сдвинута на величину перемещения робота, и, в силу только что озвученных причин, шире. Робот получает данные второго измерения, показанные жирной чертой на графике гауссовой функции на Рис. 3.2e, которое, затем, используется в вычислении апостериорной оценки, показанной на Рис. 3.2f.

Как показано на примере, в алгоритме фильтра Калмана последовательно чередуются такты обновления измерений (строки 5-7), в которых данные датчиков интегрируются в текущую оценку, и прогнозирования (или, иначе, такты обновления управления), которые модифицируют оценку в соответствии с действием. Такт обновления уменьшает, а такт прогнозирования – увеличивает неопределённость оценки робота.

3.2.4 Математический вывод фильтра Калмана

В этом разделе выполняется вывод алгоритмов фильтрации Калмана из Таблицы 3.1. Его можно смело пропускать при первом прочтении, поскольку он приводится из соображений полноты изложения.

Во-первых, вывод KF, по большей части, это упражнение по манипулированию квадратичными выражениями. Экспоненты складываются, например, при перемножении двух гауссовых функций. Поскольку обе первичные экспоненты квадратичные, результирующая сумма тоже. После этого останется только представить результат в форме, допускающей определение целевых параметров.

Часть 1: Прогнозирование

Вывод начинается со строк 2 и 3 алгоритма, в которых оценка $\overline{bel}(x_t)$ вычисляется на основе оценки, полученной на предыдущем шаге, $bel(x_{t-1})$. В строках 2 и 3 выполняется такт обновления, описанный в выражении (2.41), и приведённого здесь для удобства читателя:

(3.8)

$$\overline{bel}(x_t) = \int \underbrace{p(x_t|x_{t-1},u_t)}_{\sim N(x_t;A_t x_{t-1}+B_t u_t,R_t)} \underbrace{bel(x_{t-1})}_{\sim N(x_{t-1};\mu_{t-1},\Sigma_{t-1})} \quad dx_{t-1}$$

Оценка $bel(x_{t-1})$ представлена математическим ожиданием μ_{t-1} и ковариацией Σ_{t-1}. Переходная вероятность $p(x_t|x_{t-1},u_t)$ была дана в (3.4)

в виде нормального распределения по x_t с математическим ожиданием $A_t x_{t-1} + B_t u_t$ и ковариацией R_t. Как сейчас будет продемонстрировано, результат (3.8) тоже представляет собой гауссову функцию с математическим ожиданием $\bar{\mu}_t$ и ковариацией $\bar{\Sigma}_t$, как указано в Таблице 3.1.

Начнём с записи (3.8) в виде гауссовой функции:

(3.9)

$$\begin{aligned}\overline{bel}(x_t) \\ = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\} \\ \exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})\right\} dx_{t-1}\end{aligned}$$

После сокращения, получим

(3.10)

$$\overline{bel}(x_t) = \eta \int \exp\{-L_t\}\, dx_{t-1}$$

где

(3.11)

$$\begin{aligned}L_t = \tfrac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t) \\ + \frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})\end{aligned}$$

Заметим, что L_t квадратично как для x_{t-1}, так и для x_t.

Выражение (3.10) содержит интеграл, решение которого требует переопределения членов на интервале интегрирования. Способ разложения, поначалу, кажется контринтуитивным. В частности, выполним разложение L_t на две функции, $L_t(x_{t-1}, x_t)$ и $L_t(x_t)$:

(3.12)

$$L_t = L_t(x_{t-1}, x_t) + L_t(x_t)$$

Это разложение будет просто результатом перегруппировки членов в L_t и главной его целью должно стать разделение переменных в L_t на два множества, из которых только одно будет зависеть от переменной x_{t-1}. Другое, $L_t(x_t)$, от x_{t-1} зависеть не будет. В результате, второй набор переменных можно вынести из-под интеграла по переменной x_{t-1}.

Это показано следующим преобразованием:

(3.13)

$$\begin{aligned}\overline{bel}(x_t) = \eta \textstyle\int \exp\{-L_t\}\, dx_{t-1} \\ = \eta \int \exp\{-L_t(x_{t-1}, x_t) - L_t(x_t)\}\, dx_{t-1} \\ = \eta \exp\{-L_t(x_t)\} \int \exp\{-L_t(x_{t-1}, x_t)\}\, dx_{t-1}\end{aligned}$$

Конечно, существует множество способов разложить L_t на два множества, удовлетворяющие этому критерию. Ключевой мыслью будет выбор

$L_t(x_{t-1}, x_t)$ таким образом, чтобы значение интеграла в (3.13) не зависело от x_t. Если нам удастся определить такую функцию $L_t(x_{t-1}, x_t)$, весь интеграл по $L_t(x_{t-1}, x_t)$ в задаче определения распределения по x_t, станет просто константой. Константы обычно учитываются в нормализующей постоянной η, поэтому в нашем разложении возможно включить константу в η (её значение будет отличаться от константы η, приводимой выше):

(3.14)
$$\overline{bel}(x_t) = \eta \exp\{-L_t(x_t)\}$$

Таким образом, разложение делает возможным исключение интеграла из оценки (3.10). Результатом является нормализованная экспонента квадратичной функции, представляющая собой нормальное распределение.

Давайте выполним это разложение. Мы ищем функцию $L_t(x_{t-1}, x_t)$, квадратичную для x_{t-1}. (Эта функция будет также зависеть от x_t, но сейчас это неважно.) Для определения коэффициентов квадратичной функции вычислим первые две производные L_t:

(3.15)
$$\frac{\partial L_t}{\partial x_{t-1}} = -A_t^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t) + \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})$$

(3.16)
$$\frac{\partial^2 L_t}{\partial x_{t-1}^2} = A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1} =: \Psi_t^{-1}$$

Ψ_t определяет кривизну $L_t(x_{t-1}, x_t)$. Приняв первую производную L_t за 0 получим следующее математическое ожидание:

(3.17)
$$A_t^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t) = \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})$$

Сейчас выражение решено для x_{t-1}

(3.18)
$$\iff A_t^T R_t^{-1}(x_t - B_t u_t) - A_t^T R_t^{-1} A_t x_{t-1} = \Sigma_{t-1}^{-1} x_{t-1} - \Sigma_{t-1}^{-1}\mu_{t-1}$$

$$\iff A_t^T R_t^{-1} A_t x_{t-1} + \Sigma_{t-1}^{-1} x_{t-1} = A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}$$

$$\iff (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1}) x_{t-1} = A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}$$

$$\iff \Psi_t^{-1} x_{t-1} = A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}$$

$$\iff x_{t-1} = \Psi_t \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}\right]$$

Итак, теперь имеется квадратичная функция $L_t(x_{t-1}, x_t)$, определённая следующим образом:

(3.19)

$$L_t(x_{t-1}, x_t) = \tfrac{1}{2}(x_{t-1} - \Psi_t \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}\right])^T \Psi^{-1} \\ (x_{t-1} - \Psi_t \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}\right])$$

Вообще-то, это не единственная квадратичная функция, удовлетворяющая разложению в (3.12). Однако, $L_t(x_{t-1}, x)t)$ – общая квадратичная форма отрицательной экспоненты нормального распределения. Фактически, функция

(3.20)

$$\det(2\pi\Psi)^{-\frac{1}{2}} \exp\left\{-L_t(x_{t-1}, x_t)\right\}$$

представляет собой действительную функцию плотности вероятности (ФПВ) для переменной x_{t-1}. Как читатель может легко убедиться, эта функция имеет форму, определённую в (3.1). Из уравнения (2.5) известно, что ФПВ интегрируются до 1. Отсюда, получаем

(3.21)

$$\int \det(2\pi\Psi)^{-\frac{1}{2}} \exp\left\{-L_t(x_{t-1}, x_t)\right\} dx_{t-1} = 1$$

из чего следует

(3.22)

$$\int \exp\left\{-L_t(x_{t-1}, x_t)\right\} dx_{t-1} = \det(2\pi\Psi)^{\frac{1}{2}}$$

Важно заметить, что значение этого интеграла *не зависит* от x_t, нашей целевой переменной. Поэтому, в задаче вычисления распределения по x_t, значение интеграла будет константой. Включая константу в нормализующий член η, получаем следующее выражение для уравнения (3.13):

(3.23)

$$\overline{bel}(x_t) \;=\; \eta \exp\left\{-L_t(x_t)\right\} \int \exp\left\{-L_t(x_{t-1}, x_t)\right\} dx_{t-1} \;=\; \eta \exp\left\{-L_t(x_t)\right\}$$

Это разложение подтверждает правильность (3.14). Снова заметим, что нормализующие члены n в обоих случаях имеют разное значение.

Остаётся только определить функции $L_t(x_t)$, означающей разницу L_t и определённую в (3.11), и $L_t(x_{t-1}, x_t)$, которая была определена в (3.19):

(3.24)

$$L_t(x_t) = L_t - L_t(x_{t-1}, x_t)$$
$$= \tfrac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)$$
$$+\tfrac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})$$
$$-\tfrac{1}{2}(x_{t-1} - \Psi_t \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}\right])^T \Psi^{-1}$$
$$(x_{t-1} - \Psi_t \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1}\mu_{t-1}\right])$$

Давайте быстро проверим, что $L_t(x_t)$ действительно не зависит от x_{t-1}. Для этого произведём обратную замену $\Psi_t = (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1}$, и перемножим указанные множители. Для удобства читателя, члены, содержащие

x_{t-1} выделены подчёркиванием (двойным, если они квадратичны по отношению к x_{t-1}).

(3.25)

$$\begin{aligned} L_t(x_t) = & \underline{\underline{\frac{1}{2} x_{t-1}^T A_t^T R_t^{-1} A_t x_{t-1}}} \underline{-x_{t-1}^T A_t^T R_t^{-1}(x_t - B_t u_t)} \\ & + \tfrac{1}{2}(x_t - B_t u_t)^T R_t^{-1}(x_t - B_t u_t) \\ & + \underline{\underline{\tfrac{1}{2} x_{t-1}^T \Sigma_{t-1}^{-1} x_{t-1}}} \underline{-x_{t-1}^T \Sigma_{t-1}^{-1} \mu_{t-1}} + \tfrac{1}{2} \mu_{t-1}^T \Sigma_{t-1}^{-1} \mu_{t-1} \\ & \underline{\underline{-\tfrac{1}{2} x_{t-1}^T (A_t^T R_r^{-1} A_t + \Sigma_{t-1}^{-1}) x_{t-1}}} \\ & \underline{+ x_{t-1}^T \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right]} \\ & - \tfrac{1}{2} \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right]^T (A_t^T R_r^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} \\ & \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right] \end{aligned}$$

Несложно заметить, что все члены, содержащие x_{t-1}, сокращаются. Неудивительно, поскольку именно это было условием для создания функции $L_t(x_{t-1}, x_t)$.

(3.26)

$$\begin{aligned} L_t(x_t) = & + \frac{1}{2}(x_t - B_t u_t)^T R_t^{-1}(x_t - B_t u_t) + \frac{1}{2} \mu_{t-1}^T \Sigma_{t-1}^{-1} \mu_{t-1} \\ & - \tfrac{1}{2} \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right]^T (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} \\ & \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right] \end{aligned}$$

Далее, $L_t(x_t)$ квадратично на x_t. Это наблюдение означает, что $\overline{bel}(x_t)$ действительно имеет нормальное распределение. Математическое ожидание и ковариация этого распределения, конечно, будут минимумом и уклоном функции $L_t(x_t)$, которые мы теперь легко можем получить, вычислив первую и вторую производные $L_t(x_t)$ по x_t:

(3.27)

$$\begin{aligned} \frac{\partial L_t(x_t)}{\partial x_t} &= R_t^{-1}(x_t - B_t u_t) - R_t^{-1} A_t (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} \\ &\quad \left[A_t^T R_t^{-1}(x_t - B_t u_t) + \Sigma_{t-1}^{-1} \mu_{t-1} \right] \\ &= \left[R_t^{-1} - R_t^{-1} A_t (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} A_t^T R_t^{-1} \right] (x_t - B_t u_t) \\ &\quad - R_t^{-1} A_t (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} \Sigma_{t-1}^{-1} \mu_{t-1} \end{aligned}$$

Лемма об обращении матриц, определённая (и показанная) в Таблице 3.2, позволяет выразить первый множитель следующим образом:

(3.28)

$$R_t^{-1} - R_t^{-1} A_t (A_t^T R_t^{-1} A_t + \Sigma_{t-1}^{-1})^{-1} A_t^T R_t^{-1} = (R_t + A_t \Sigma_{t-1} A_t^T)^{-1}$$

Следовательно, искомая производная задана следующим выражением:

(3.29)
$$\frac{\partial L_t(x_t)}{\partial x_t} = (R_t + A_t\Sigma_{t-1}A_t^T)^{-1}(x_t - B_t u_t) \\ -R_t^{-1}A_t(A_t^T R_t^{-1}A_t + \Sigma_{t-1}^{-1})^{-1}\Sigma_{t-1}^{-1}\mu_{t-1}$$

Лемма об обращении матриц. Для любых инвертируемых квадратных матриц R и Q и произвольной матрицы P соответствующих размерностей, следующее утверждение истинно

$$(R + PQP^T)^{-1} = R^{-1} - R^{-1}P(Q^{-1} + P^T R^{-1}P)^{-1}P^T R^{-1}$$

При условии, что все три указанные матрицы обратимы указанным образом.

Доказательство. Определим $\Psi = (Q^{-1} + P^T R^{-1}P)^{-1}$.
Этого достаточно, чтобы показать, что

$$(R^{-1} - R^{-1}P\Psi P^T R{-}1)(R + PQP^T) = I$$

Это можно показать путём следующей последовательности преобразований:

$$\begin{aligned}
&= \underbrace{R^{-1}R}_{=I} + R^{-1}PQP^T - R^{-1}P\Psi P^T \underbrace{R^{-1}R}_{=I} \\
&\quad - R^{-1}P\Psi P^T R^{-1}PQP^T \\
&= I + R^{-1}PQP^T - R^{-1}P\Psi P^T - R^{-1}P\Psi P^T R^{-1}PQP^T \\
&= I + R^{-1}P[QP^T - \Psi P^T - \Psi P^T R^{-1}PQP^T] \\
&= I + R^{-1}P[QP^T - \Psi \underbrace{Q^{-1}Q}_{=I} P^T - \Psi P^T R^{-1}PQP^T] \\
&= I + R^{-1}P[QP^T - \underbrace{\Psi\Psi^{-1}}_{=I}QP^T] \\
&= I + R^{-1}P\underbrace{[QP^T - QP^T]}_{=0} = I
\end{aligned}$$

Таблица 3.2 Специализированная лемма об обращении матриц, иногда называемая *формулой Шермана/Моррисона*.

Минимум $L_t(x_t)$ достигается, когда первая производная равна нулю.

(3.30)
$$(R_t + A_t\Sigma_{t-1}A_t^T)^{-1}(x_t - B_t u_t) \\ = R_t^{-1}A_t(A_t^T R_t^{-1}A_t + \Sigma_{t-1}^{-1})^{-1}\Sigma_{t-1}^{-1}\mu_{t-1}$$

Решение для целевой переменной x_t имеет, на удивление, компактный вид

(3.31)
$$x_t = B_t u_t + \underbrace{(R_t + A_t\Sigma_{t-1}A_t^T)R_t^{-1}A_t}_{A_t + A_t\Sigma_{t-1}A_t^T R_t^{-1}A_t} \underbrace{(A_t^T R_t^{-1}A_t + \Sigma_{t-1}^{-1})^{-1}\Sigma_{t-1}^{-1}}_{(\Sigma_{t-1}A_t^T R_t^{-1}A_t + I)^{-1}} \mu_{t-1}$$

$$= B_t u_t + A_t \underbrace{(I + \Sigma_{t-1} A_t^T R_t^{-1} A_t)(\Sigma_{t-1} A_t^T R_t^{-1} A_t + I)^{-1}}_{=I} \mu_{t-1}$$

$$= B_t u_t + A_t \mu_{t-1}$$

Таким образом, математическое ожидание оценки $\overline{bel}(x_t)$ после учёта команды на движение u_t имеет вид $B_t u_t + A_t \mu_{t-1}$. Это доказывает правильность строки 2 алгоритма фильтра Калмана в Таблице 3.1.

Сейчас выражение в строке 3 получается вычислением второй производной функции $L_t(x_t)$:

(3.32)
$$\frac{\partial^2 L_t(x_t)}{\partial x_t^2} \; = \; (A_t \Sigma_{t-1} A_t^T + R_t)^{-1}$$

Это кривизна квадратичной функции $L_t(x_t)$, инверсия которой даст ковариацию оценки $\overline{bel}(x_t)$.

Подведём итог. Нами было показано, что такты прогнозирования в строках 2 и 3 алгоритма фильтра Калмана, в действительности, реализуют такт прогнозирования для байесовского фильтра. Для этого экспонента оценки $\overline{bel}(x_t)$ была разделена на две функции, $L_t(x_{t-1}, x_t)$ и $L_t(x_t)$. Затем было показано, что $L_t(x_{t-1}, x_t)$ изменяет прогнозируемую оценку $\overline{bel}(x_t)$ лишь на постоянный коэффициент, который может быть учтён нормализующим членом η. Наконец, была определена функция $L_t(x_t)$, и показано, что её результат представляет собой математическое ожидание $\bar{\mu}_t$ и ковариацию $\bar{\Sigma}_t$ такта прогнозирования фильтра Калмана $\overline{bel}(x_t)$.

Часть 2: Обновление измерения

Сделаем вывод выражений для обновления измерения в строках 4, 5 и 6 (Таблица 3.1) алгоритма фильтра Калмана. Начнём с общего механизма учёта измерения в байесовском фильтре, определённом в выражении (2.38) и приведённом здесь в аннотированном виде:

(3.33)
$$bel(x_t) = \; \eta \quad \underbrace{p(z_t|x_t)}_{\sim N(z_t; C_t x_t, Q_t)} \quad \underbrace{\overline{bel}(x_t)}_{\sim N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)}$$

Очевидно, что математическое ожидание и ковариация $\overline{bel}(x_t)$ заданы как $\bar{\mu}_t$ и $\bar{\Sigma}_t$. Вероятность измерения $p(z_t|x_t)$ была определена в (3.6), и так же является нормальным распределением с математическим ожиданием $C_t x_t$ и ковариацией Q_t, поэтому, произведение даёт экспоненту

(3.34)
$$bel(x_t) \; = \; \eta \exp\{-J_t\}$$

где

(3.35)
$$J_t = \frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t) + \frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)$$

Эта функция квадратична по x_t, следовательно, $bel(x_t)$ представляет собой нормальное распределение. Чтобы вычислить его параметры, снова возьмём первые две производные J_t по x_t:

(3.36)
$$\frac{\partial J}{\partial x_t} = -C_t^T Q_t^{-1}(z_t - C_t x_t) + \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)$$

(3.37)
$$\frac{\partial^2 J}{\partial x_t^2} = C_t^T Q_t^{-1} C_t + \bar{\Sigma}_t^{-1}$$

Второй член выражения– это инверсия ковариации $bel(x_t)$:

(3.38)
$$\Sigma_t = (C_t^T Q_t^{-1} C_t + \bar{\Sigma}_t^{-1})^{-1}$$

Математическое ожидание $bel(x_t)$ представляет собой минимум квадратичной функции, который вычисляется путём приравнивания J_t к нулю (и замены μ_t на x_t):

(3.39)
$$C_t^T Q_t^{-1}(z_t - C_t \mu_t) = \bar{\Sigma}_t^{-1}(\mu_t - \bar{\mu}_t)$$

Выражение слева от знака равенства можно преобразовать следующим образом:

(3.40)
$$\begin{aligned} & C_t^T Q_t^{-1}(z_t - C_t \mu_t) \\ & = C_t^T Q_t^{-1}(z_t - C_t \mu_t + C_t \bar{\mu}_t - C_t \bar{\mu}_t) \\ & = C_t^T Q_t^{-1}(z_t - C_t \bar{\mu}_t) - C_t^T Q_t^{-1} C_t(\mu_t - \bar{\mu}_t) \end{aligned}$$

Обратная замена в (3.39) даёт

(3.41)
$$C_t^T Q_t^{-1}(z_t - C_t \bar{\mu}_t) = \underbrace{(C_t^T Q_t^{-1} C_t + \bar{\Sigma}_t^{-1})}_{=\Sigma_t^{-1}}(\mu_t - \bar{\mu}_t)$$

откуда получается

(3.42)
$$\Sigma_t C_t^T Q_t^{-1}(z_t - C_t \bar{\mu}_t) = \mu_t - \bar{\mu}_t$$

Теперь можно определить *усиление фильтра Калмана* в виде

(3.43)
$$K_t = \Sigma_t C_t^T Q_t^{-1}$$

и получить

(3.44)
$$\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$$

Это доказывает правильность выражения в строке 5 алгоритма фильтра Калмана из Таблицы 3.1.

Усиление фильтра Калмана, согласно определению в (3.43), является функцией E_t, но это расходится с фактом использования E_t для вычисления Σ_t в строке 6 алгоритма. Следующее преобразование демонстрирует способ выражения K_t через ковариацию, отличную от Σ_t. Начнём с определения K_t в (3.43):

(3.45)

$$\begin{aligned}
K_t &= \Sigma_t C_t^T Q_t^{-1} = \Sigma_t C_t^T Q_t^{-1} \underbrace{(C_t \bar{\Sigma}_t C_t^T + Q_t)(C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}}_{=I} \\
&= \Sigma_t (C_t^T Q_t^{-1} C_t \bar{\Sigma}_t C_t^T + C_t^T \underbrace{Q_t^{-1} Q_t}_{=I})(C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
&= \Sigma_t (C_t^T Q_t^{-1} C_t \bar{\Sigma}_t C_t^T + C_t^T)(C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
&= \Sigma_t (C_t^T Q_t^{-1} C_t \bar{\Sigma}_t C_t^T + \underbrace{\bar{\Sigma}_t^{-1} \bar{\Sigma}_t}_{=I} C_t^T)(C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
&= \Sigma_t \underbrace{(C_t^T Q_t^{-1} C_t + \bar{\Sigma}_t^{-1})}_{=\Sigma_t^{-1}} \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
&= \underbrace{\Sigma_t \Sigma_t^{-1}}_{=I} \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\
&= \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}
\end{aligned}$$

Это выражение доказывает правильность строки 4 алгоритма фильтра Калмана.

Строка 6 получается в результате выражения ковариации через калмановское усиление K_t. Преимущество метода вычисления из Таблицы 3.1 по сравнению с определением в выражении (3.38) заключается в возможности избежать инверсии ковариационной матрицы состояния. Это важно для использования калмановских фильтров в пространствах состояний высокой размерности.

Преобразование опять производится с использованием леммы об обращении матриц, которая уже приводилась в Таблице 3.2. Повторим её в нотации уравнения (3.38):

(3.46)

$$(\bar{\Sigma}_t^{-1} + C_t^T Q_t^{-1} C_t)^{-1} = \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T (Q_t + C_t \bar{\Sigma}_t C_t^T)^{-1} C_t \bar{\Sigma}_t$$

Это позволяет прийти к следующему выражению для ковариационной матрицы:

(3.47)

$$\begin{aligned}
\Sigma_t &= (C_t^T Q_t^{-1} C_t + \bar{\Sigma}_t^{-1})^{-1} \\
&= \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T (Q_t + C_t \bar{\Sigma}_t C_t^T)^{-1} C_t \bar{\Sigma}_t \\
&= [I - \underbrace{\bar{\Sigma}_t C_t^T (Q_t + C_t \bar{\Sigma}_t C_t^T)^{-1}}_{=K_t, \text{см. ур-е (3.45)}} C_t] \bar{\Sigma}_t
\end{aligned}$$

$$= (I - K_t C_t)\bar{\Sigma}_t$$

На этом доказательство корректности завершается, поскольку правильность строки 6 алгоритма калмановского фильтра продемонстрирована.

3.3 Обобщенный фильтр Калмана

3.3.1 Зачем производить линеаризацию?

Допущения о том, что наблюдения являются линейными функциями состояния (то есть каждое следующее состояние - линейная функция предыдущего) являются основополагающими для алгоритма калмановского фильтра. Факт того, что любое линейное преобразование случайной гауссовой переменной даёт, в результате, другую гауссову случайную переменную, играет важную роль в выводе алгоритма фильтра Калмана, при этом эффективность фильтра Калмана зависит от возможности вычислить параметры результирующей гауссовой функции в закрытом виде.

В этой и последующих главах будут проиллюстрированы свойства различных представлений плотности с использованием преобразований одномерной случайной гауссовой переменной. На Рис. 3.3а показано *линейное* преобразование такой случайной переменной. На графике справа внизу изображена плотность случайной переменной $X \sim N(x; \mu, \sigma^2)$. Допустим, X проходит через линейную функцию $y = ax + b$, показанную на графике справа вверху. Результирующая случайная переменная, Y, имеет нормальное распределение, с математическим ожиданием $a\mu + b$ и дисперсией $a^2\sigma^2$. Эта гауссова функция показана в виде заштрихованной серым области в верхней левой части графика на Рис. 3.3а. Читатель может заметить тесную связь данного примера с тактом обновлением фильтра Калмана, где $X = x_{t-1}$, а $Y = x_t$, но без добавочной переменной зашумления. Также стоит обратить внимание на Выражение (3.2).

К сожалению, на практике переходы между состояниями и измерениями редко линейны. Например, робот, который движется с постоянной поступательной и вращательной скоростью, обычно перемещается по круговой траектории, которую невозможно описать с помощью линейных переходов состояния. Это наблюдение, а также допущение необходимости одномодовых оценок, делает простые калмановские фильтры в обсуждаемом виде неприменимыми для всех робототехнических задач, кроме самых тривиальных.

Обобщенный фильтр Калмана, или EKF, расширяет использование одного из допущений, линейности. Вместо него вводится понятие переходной вероятности состояний и измерений с помощью *нелинейных* функций g и h, соответственно:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{3.48}$$

$$z_t = h(x_t) + \delta_t \tag{3.49}$$

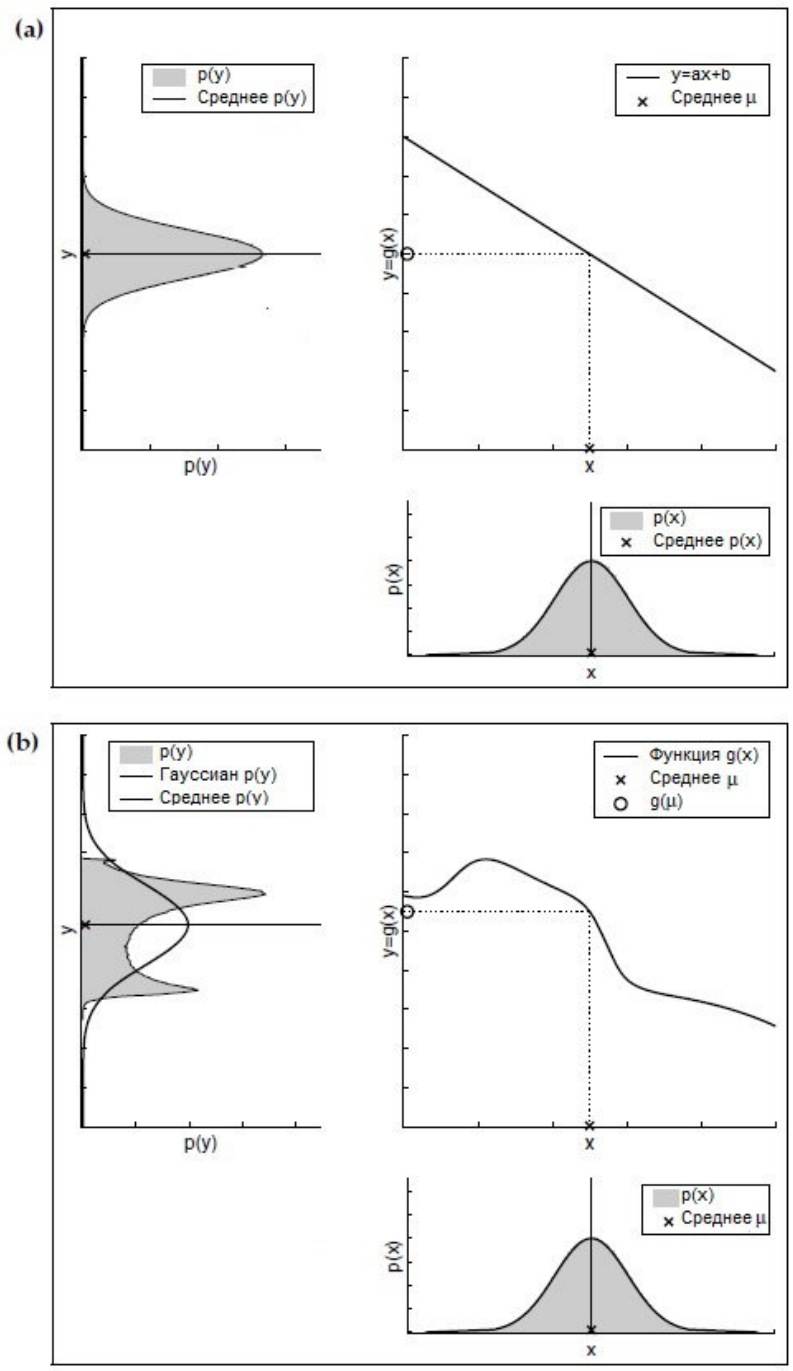


Рис 3.3 Линейное (a) и нелинейное (b) преобразование случайной гауссовой переменной. На нижнем правом графике показана плотность начальной случайной переменной, X. Эта случайная переменная проходит через функцию, показанную на верхних правых графиках (пунктиром показано преобразование математического ожидания). Плотность результирующей случайной переменной Y показана на верхних графиках слева.

Эта модель строго обобщает линейное описание в виде гауссовых функ-

ций, лежащее в основе фильтров Калмана, как допускается выражениями (3.2) и (3.5). Функция g заменяет матрицы A_t и B_t в (3.2), а h - заменяет матрицу C_t в (3.5). К сожалению, для произвольных функций g и h оценка больше не будет нормальным распределением. На практике, обновление оценки обычным образом с помощью нелинейных функций g и h часто невозможно, а решения в закрытом виде для байесовского фильтра нет.

На Рис. 3.3b показано воздействие нелинейного преобразования на случайную гауссову переменную. На графиках снизу справа и сверху справа показаны случайная переменная X и нелинейная функция g, соответственно. Плотность преобразованной случайной переменной, $Y = g(X)$, обозначена серой зоной на верхнем левом графике Рис. 3.3b. Поскольку эта плотность не может быть вычислена в закрытом виде, оценочно извлечём 500000 элементов из $p(x)$, передадим функции g, а затем отразим на гистограмме в диапазоне значений g. Как видно, Y нормальным распределением не является, в силу того, что нелинейности функции g искажают плотность X.

Обобщенный фильтр Калмана (EKF) вычисляет гауссову аппроксимацию реальной оценки. Пунктиром на левом верхнем графике на Рис. 3.3b показано гауссово приближение плотности случайной переменной Y. Соответственно, EKF представляет оценку $bel(x_t)$ в момент времени t в виде математического ожидания μ_t и ковариационной матрицы Σ_t. Таким образом, EKF наследуют от фильтра Калмана общее представление оценки, но отличается в том, что эта оценка лишь приближенная, а не точная, как в простых калмановских фильтрах. Таким образом, смысл EKF не в вычислении точной апостериорной вероятности, а в эффективной приближенной оценке ее математического ожидания и ковариации. Поскольку эти статистики не могут быть вычислены в закрытом виде, в алгоритме EKF необходимо прибегать к дополнительной аппроксимации.

3.3.2 Линеаризация разложением в ряд Тейлора

Ключевая идея в основе EKF называется *линеаризацией*, общая концепция показана на Рис. 3.4. Линеаризация аппроксимирует нелинейную функцию g с помощью касательной в точке математического ожидания гауссиана линейной функции (пунктирная линия на верхнем правом графике). Проекция гауссовой функции с помощью линейной аппроксимации даёт, в результате, плотность, что показано пунктирной линией на верхнем левом графике. Сплошной линией показаны математическое ожидание и ковариация после аппроксимации методом Монте-Карло. Разница между этими двумя гауссианами составляет ошибку, вызванную линейной аппроксимацией функции g.

Ключевым преимуществом линеаризации является ее эффективность.

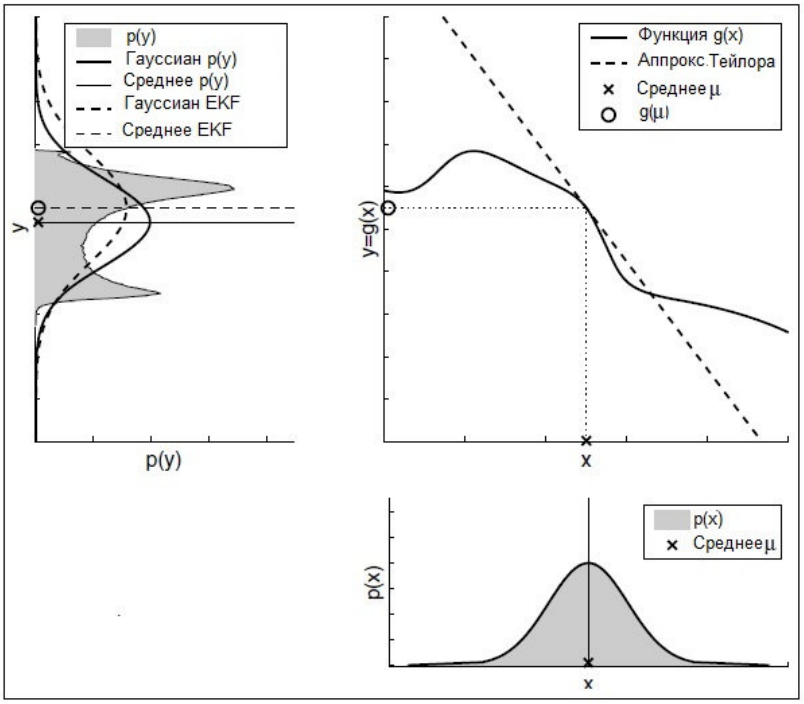


Рис. 3.4 Иллюстрация линеаризации, применяемой для EKF. Вместо представления гауссовой функции через нелинейную функцию g производится линейная аппроксимация g с помощью касательной в точке математического ожидания линейной функции. Результирующая гауссова функция показана пунктиром на левом верхнем графике. Линеаризация влечёт за собой ошибку приближения, наблюдаемую в виде разницы между линеаризованной гауссовой функцией (пунктирной) и гауссианом, вычисленным с помощью точной оценки методом Монте-Карло (сплошная линия).

Оценка гауссовской функции методом Монте-Карло была получена заданием 500000 точек функции g с последующим вычислением математического ожидания и ковариации. С другой стороны, линеаризация, применяемая в EKF, требует лишь определения линейного приближения с последующим вычислением результирующей гауссовой функции в закрытом виде. Фактически, после линеаризации g, механика распространения оценки в EKF и фильтрах Калмана одинакова.

Этот метод применим также при перемножении гауссовых функций, когда участвует функция измерения h. И снова в EKF выполняется приближение h с помощью касательной линейной функции для сохранения нормального распределения апостериорной оценки.

РАЗЛОЖЕНИЕ В РЯД ТЕЙЛОРА

Существует множество методов для линеаризации нелинейных функций. В EKF используется способ, называемый *разложением в ряд Тейлора* (первого порядка). Разложение в ряд Тейлора создаёт линейное приближение функции g на основании её значения и наклона. Наклон задаётся частной производной

(3.50)
$$g'(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$$

Разумеется, и значение g, и наклон зависят от аргументов функции. Логичным выбором аргумента является состояние, которое, вероятнее всего, наступит на момент линеаризации. Для гауссовых функций наиболее вероятное состояние - среднее апостериорной вероятности μ_{t-1} (математическое ожидание). Другими словами, функция g аппроксимируется по значению в μ_{t-1} (и u_t), а линейная экстраполяция достигается с помощью коэффициента, пропорционального градиенту g для μ_{t-1} и u_t:

(3.51)

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=:G_t}(x_{t-1} - \mu_{t-1})$$

$$= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

Записанная в виде гауссовой функции, переходная вероятность в приближенном виде выглядит следующим образом:

(3.52)

$$p(x_t|u_t, x_{t-1}) \approx \det(2\pi R_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T$$

$$R_t^{-1}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]\}$$

ЯКОБИАН

Заметим, что G_t – матрица размера $n \times n$, где n означает размерность состояния. Такая матрица часто называется *якобианом*. Значение якобиана зависит от u_t и μ_{t-1}, и, поэтому, различается для разных моментов времени.

В EKF используется аналогичная линеаризация для функции измерения h. Здесь разложение в ряд Тейлора происходит по $\bar{\mu}_t$, наиболее вероятному состоянию, определённому роботом в момент линеаризации по h:

(3.53)

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{=:H_t}(x_t - \bar{\mu}_t)$$

$$= h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)$$

С частной производной $h'(x_t) = \frac{\partial h(x_t)}{\partial x_t}$ и в виде гауссовой функции, получим

(3.54)

$$p(z_t|x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}[z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)]^T$$

$$Q_t^{-1}[z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)]\}$$

1:	**Algorithm Extended_Kalman_filter** $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2:	$\bar{\mu}_t = g(u_t, \mu_{t-1})$
3:	$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4:	$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
5:	$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6:	$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
7:	*return* μ_t, Σ_t

Таблица 3.3 Алгоритм обобщённого фильтра Калмана.

3.3.3 Алгоритм EKF

В Таблице 3.3 приводится *алгоритм обобщённого фильтра Калмана*. Во многих отношениях, этот алгоритм похож на алгоритм калмановского фильтра, приведённый в Таблице 3.1. Наиболее важные различия указаны в следующей таблице:

	Фильтр Калмана	EKF
прогноз состояния (строка 2)	$A_t \mu_{t-1} + B_t u_t$	$g(u_t, \mu_{t-1})$
прогноз измерения (строка 5)	$C_t \bar{\mu}_t$	$h(\bar{\mu}_t)$

Таким образом, линейные прогнозы калмановских фильтров заменяются в EKF нелинейными обобщениями. Более того, вместо системы линейных матриц A_t, B_t, и C_t, используемой в калмановских фильтрах, в алгоритме EKF использованы якобианы G_t и H_t, соответственно. Якобиан G_t соответствует матрицам A_t и B_t, а якобиан H_t - матрице C_t. Обобщённые калмановские фильтры будут детально разобраны на примерах в Главе 7.

3.3.4 Математический вывод для EKF

Математический вывод EKF аналогичен выводу для фильтра Калмана в подразделе 3.2.4, поэтому, приводится в сокращённом виде. Прогноз вычисляется следующим образом (см. (3.8)):

(3.55)

$$\overline{bel}(x_t) = \int \underbrace{p(x_t|x_{t-1}, u_t)}_{\sim N(x_t; g(u_t,\mu_{t-1}) + G_t(x_{t-1}-\mu_{t-1}), R_t)} \underbrace{bel(x_{t-1})}_{\sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})} \quad dx_{t-1}$$

Это распределение для EKF является аналогом прогнозного распределения калмановского фильтра, приведённом в (3.8). Гауссова функция $p(x_t|x_{t-1}, u_t)$ приводится в выражении (3.52). Функция L_t задана с помощью (см. (3.11))

(3.56)

$$L_t = \frac{1}{2}(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1}))^T$$

$$R_t^{-1}(x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1}))$$

$$+\tfrac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})$$

и является квадратичной как по x_{t-1}, так и по x_t, что указано выше. По аналогии с (3.12), разложим L_t на $L_t(x_{t-1}, x_t)$ и $L_t(x_t)$:

(3.57)

$$L_t(x_{t-1}, x_t) = \frac{1}{2}(x_{t-1} - \Phi_t[G_t^T R_t^{-1}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}])^T \Phi^{-1}$$

$$(x_{t-1} - \Phi_t[G_t^T R_t^{-1}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}])$$

где

(3.58)

$$\Phi_t = (G_t^T R_t^{-1} G_t + \Sigma_{t-1}^{-1})^{-1}$$

следовательно,

(3.59)

$$L_t(x_t) = \frac{1}{2}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1})^T R_t^{-1}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1})$$

$$+\tfrac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1}(x_{t-1} - \mu_{t-1})$$

$$-\tfrac{1}{2}[G_t^T R_t^{-1}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}]^T$$

$$\Phi_t[G_t^T R_t^{-1}(x_t - g(u_t, \mu_{t-1}) + G_t\mu_{t-1}) + \Sigma_{t-1}^{-1}\mu_{t-1}]$$

Как читатель может легко удостовериться, обнуление первой производной $L_t(x_t)$ даст функцию обновления вида $\mu_t = g(u_t, \mu_{t-1})$, по аналогии с выводом выражений с (3.27) по (3.31). Вторая производная задана $(R_t + G_t\Sigma_{t-1}G_t^T)^{-1}$ (см. (3.32)).

Обновление измерения выводится так же, как и для фильтра Калмана в подразделе 3.2.4. По аналогии с (3.33), для EKF получается

(3.60)

$$bel(x_t) = \eta \underbrace{p(z_t|x_t)}_{\sim N(z_t; h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t), Q_t)} \underbrace{\overline{bel}(x_t)}_{\sim N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)}$$

Используем линеаризованую функцию перехода состояния из (3.53), что даёт следующую экспоненту (см. (3.35))

(3.61)

$$J_t = \frac{1}{2}(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t))^T Q_t^{-1}(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t))$$

$$+\tfrac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)$$

Получившиеся в результате математическое ожидание и ковариация заданы в виде

(3.62)
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$

(3.63)
$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

а калмановское усиление равно

(3.64)
$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_{t-1} H_t^T + Q_t)^{-1}$$

Вывод этих равенств аналогичен последовательности выражений с (3.36) по (3.47).

3.3.5 Практические соображения

EKF стал едва ли не самым популярным инструментом оценки состояния в робототехнике. Его сильными сторонами является простота и вычислительная эффективность. Как и для фильтров Калмана, каждое обновление требует времени $O(k^{2.4} + n^2)$, где k – размерность вектора измерений z_t, а n – размерность вектора состояний x_t. Другие алгоритмы, например, обсуждаемые ниже многочастичные фильтры, могут потребовать времени, экспоненциально зависящего от n.

Вычислительная эффективность EKF основана на факте представления оценки с помощью многомерного гауссового распределения. Гауссова функция представляет собой одномодальное распределение, которое можно считать единственной догадкой с неопределённостью в виде эллипса. Нормальные распределения являются надёжными функциями оценки для многих практических задач и, скажем, использование калмановского фильтра на пространствах состояний от 1000 и более измерений ещё будет обсуждаться. EKF с большим успехом были использованы в ряде задач оценки состояния с нарушениями указанных допущений.

Важное ограничение EKF возникает в силу аппроксимации перехода состояний и измерений с использованием линейных разложений в ряд Тейлора. Для большинства задач робототехники переходы между состояниями и измерениями нелинейны, в результате, качество применяемой в EKF линейной аппроксимации зависит от двух основных факторов: неопределённости и локальной нелинейности аппроксимируемых функций. На двух графиках на Рис. 3.5 показана зависимость от неопределённости. На них две случайные гауссовы переменные пропускаются через одну и ту же нелинейную функцию (также см. Рис. 3.4). Хотя обе гауссовы функции имеют одинаковое математическое ожидание, переменная, показанная на графике (a) имеет более высокую степень неопределённости по сравнению с (b).

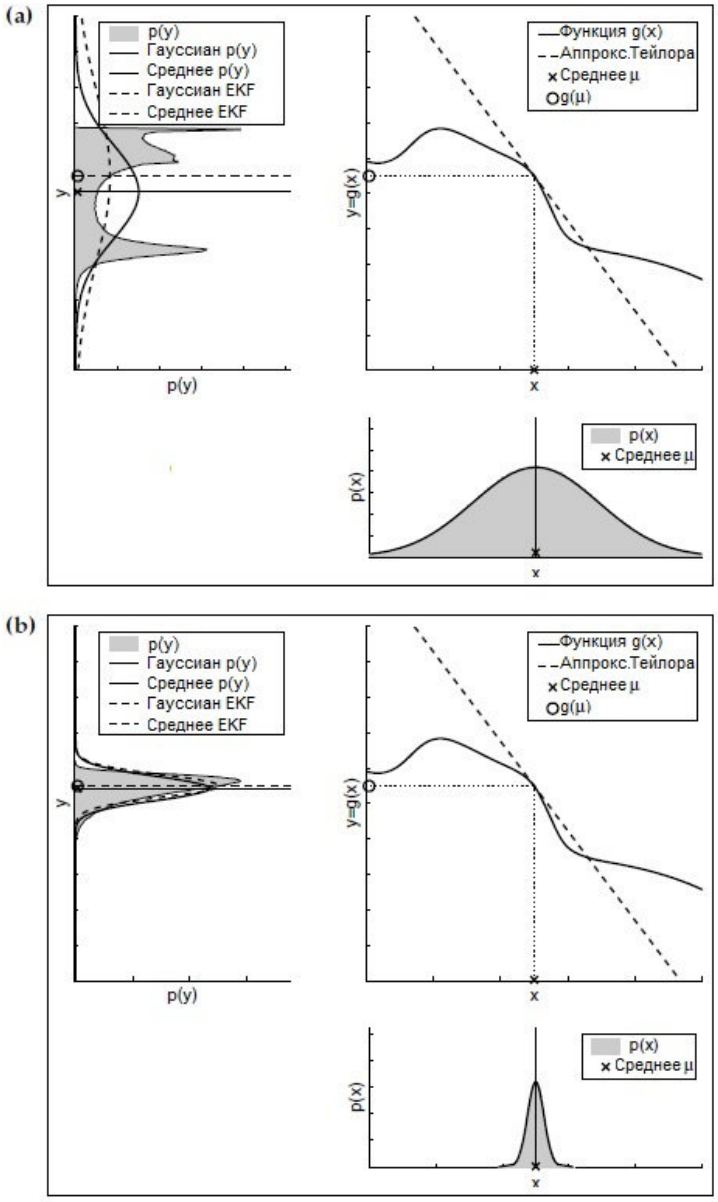


Рис. 3.5 Зависимость качества аппроксимации от степени неопределённости. Обе гауссовы функции (справа снизу) имеют одинаковое математическое ожидание и подаются на одну и ту же нелинейную функцию (сверху справа). Большая неопределённость левой гауссовой функции даёт, в результате, более искажённое распределение плотности результирующей случайной переменной (серая область слева сверху). Сплошными линиями на графиках слева сверху показаны функции плотностей. Пунктиром обозначены гауссовы функции, полученные в результате линеаризации EKF.

Поскольку разложение в ряд Тейлора зависит только от математического ожидания, обе гауссовы функции проходят одинаковую линейную аппроксимацию. Закрашенные серым области на графике слева сверху обо-

значают плотности результирующей случайной переменной, вычисленной с помощью оценки методом Монте-Карло. Плотность, выраженная более широкой гауссовой кривой, значительно более искажена. Нормальные приближения плотностей показаны на схемах сплошными линиями, пунктиром обозначены линеаризации. Сравнение гауссовых функций, полученных в результате аппроксимации методом Монте-Карло, показывает, что большая степень неопределённости обычно ведёт к менее точной оценке математического ожидания и ковариации результирующей случайной переменной.

Второй фактор, влияющий на качество линейного приближения гауссовых функций, и показанный на Рис. 3.6, это локальная нелинейность функции g. Две гауссовы функции с одинаковой дисперсией проходят через одну нелинейную функцию. На схеме (a), математическое ожидание гауссовой функции попадает в диапазон функции g с большей степенью нелинейности по сравнению с изображённой на графике (b). Разница между точной оценкой гауссовой функции методом Монте-Карло (сплошная линия, сверху слева) и полученной в результате линейной аппроксимации (пунктирная линия) показывает, что более высокая нелинейность влечёт возрастание ошибки аппроксимации. Совершенно очевидно, что в гауссовой функции EKF размах получившейся плотности недооценивается.

Иногда необходимо иметь дело с несколькими отдельными гипотезами. Например, у робота может быть две отдельные гипотезы относительно своего местонахождения, но среднее арифметическое этих гипотез не выглядит правдоподобным. Для таких ситуаций требуется мультимодальные представления апостериорной оценки, неприменимые для EKF в описанном виде. Обычным обобщением EKF является отображение апостериорных вероятностей, в виде смеси (или суммы) гауссовых функций. Сумма гауссовых функций может иметь вид

СМЕСЬ ГАУССОВЫХ ФУНКЦИЙ

$$bel(x_t) = \frac{1}{\Sigma_l \psi_{t,l}} \sum_l \psi_{t,l} \det(2\pi\Sigma_{t,l})^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_t - \mu_{t,l})^T \Sigma_{t,l}^{-1}(x_t - \mu_{t,l})\right\} \tag{3.65}$$

Здесь $\psi_{t,l}$ – это смесь параметров, где $\psi_{t,l} \geq 0$. Эти параметры служат весами компонентов смеси. Они оцениваются на основе подобия наблюдений для соответствующих гауссовых функций. EKF, в которых используются такие представления в виде смесей называются *обобщёнными фильтрами Калмана с несколькими гипотезами* или MHEKF.

EKF С НЕСКОЛЬКИМИ ГИПОТЕЗАМИ

Подводя итог, в случае, если нелинейные функции приблизительно линейны в области среднего значения оценки, то аппроксимация EKF будет, в общем, хорошим выбором для вычисления апостериорной оценки с удовлетворительной точностью.

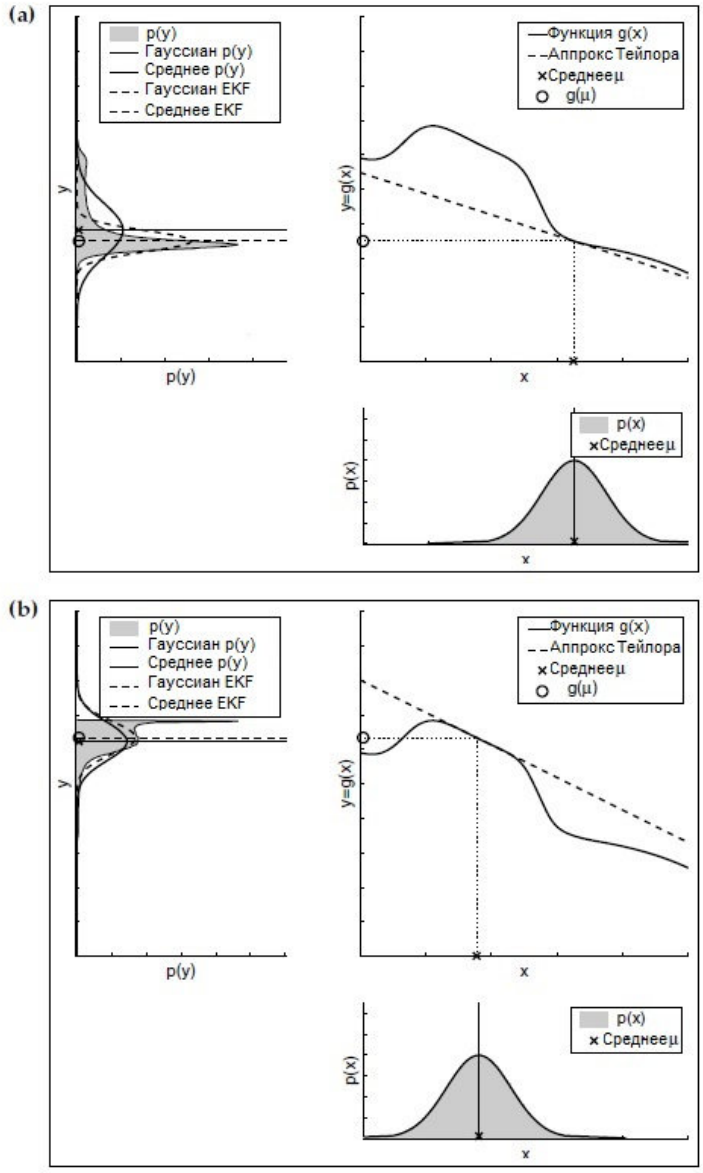


Рис. 3.6 Зависимость качества аппроксимации от локальной нелинейности функции g. Обе гауссовы функции (изображены внизу справа на обоих схемах) имеют одинаковую ковариацию и передаются через одну и ту же функцию (вверху справа). Линейная аппроксимация, применяемая в EKF, показана пунктиром на графиках справа сверху. Сплошные линии на графиках слева сверху обозначают гауссовы функции, полученные из высокоточных оценок по методу Монте-Карло. Пунктирными линиями обозначены гауссовы функции, сгенерированные в результате линеаризации с помощью EKF.

Более того, чем меньше степень уверенности робота, тем шире будет гауссова оценка, и тем сильнее она подвержена нелинейности переходных функций состояний и измерений. На практике, когда применяются EKF, важно следить за тем, чтобы неопределённость оценки состояния остава-

лась мала.

3.4 Unscented Kalman Filter

Разложение в ряд Тейлора, применяемое для EKF, является не единственным способом линеаризации преобразования гауссовой функции. Были найдены два других подхода, которые дают лучшие результаты. Один известен, как *выравнивание моментов* (moments matching) (а фильтр известен под названием *фильтр предполагаемой плотности* (assumed density filter - ADF)), в котором линеаризация вычислена так, чтобы сохранить настоящее значение математического ожидания и ковариации апостериорного распределения (чего не происходит в случае EKF). Другой метод линеаризации использован в unscented Kalman filter, или UKF, который выполняет стохастическую линеаризацию, используя процесс взвешенной статистической линейной регрессии. Обсудим UKF без математического вывода. Мы призываем читателя более детально ознакомиться с темой, воспользовавшись библиографическим примечанием.

Unscented Kalman Filter

3.4.1 Линеаризация с помощью Unscented преобразования

На Рис. 3.7 показана линеаризация, которая используется в UKF, под названием *unscented преобразование*. Вместо приближения функции g с помощью разложения в ряд Тейлора UKF предопределенным образом извлекает так называемые сигма-точки гауссовой кривой и передаёт их функции g. В общем случае, эти сигма-точки расположены на среднем значении и симметричны вдоль главных осей ковариации (по две на измерение). Для n-мерного гауссиана с математическим ожиданием μ и ковариацией Σ, в результате получается $2n+1$ сигма-точек $\mathcal{X}^{[i]}$, выбранных согласно следующему правилу:

СИГМА-ТОЧКА

(3.66)
$$\begin{aligned}\mathcal{X}^{[0]} &= \mu \\ \mathcal{X}^{[i]} &= \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i \qquad \text{для } i = 1, ..., n \\ \mathcal{X}^{[i]} &= \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n} \qquad \text{для } i = n+1, ..., 2n\end{aligned}$$

Здесь $\lambda = \alpha^2(n+\kappa) - n$, где α и κ –параметры масштабирования, определяющие, как далеко сигма-точки отстоят от среднего значения. У каждой сигма-точки $\mathcal{X}^{[i]}$ имеется два связанных с нею веса. Один из весов, $\omega_m^{[i]}$, используется для вычисления математического ожидания, а другой, $\omega_c^{[i]}$, применяется при восстановлении ковариации гауссовой функции.

(3.67)
$$\begin{aligned}w_m^{[0]} &= \frac{\lambda}{n+\lambda} \\ w_c^{[0]} &= \tfrac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \\ w_m^{[i]} &= w_c^{[i]} = \tfrac{1}{2(n+\lambda)} \qquad \text{для } i = 1, ..., 2n\end{aligned}$$

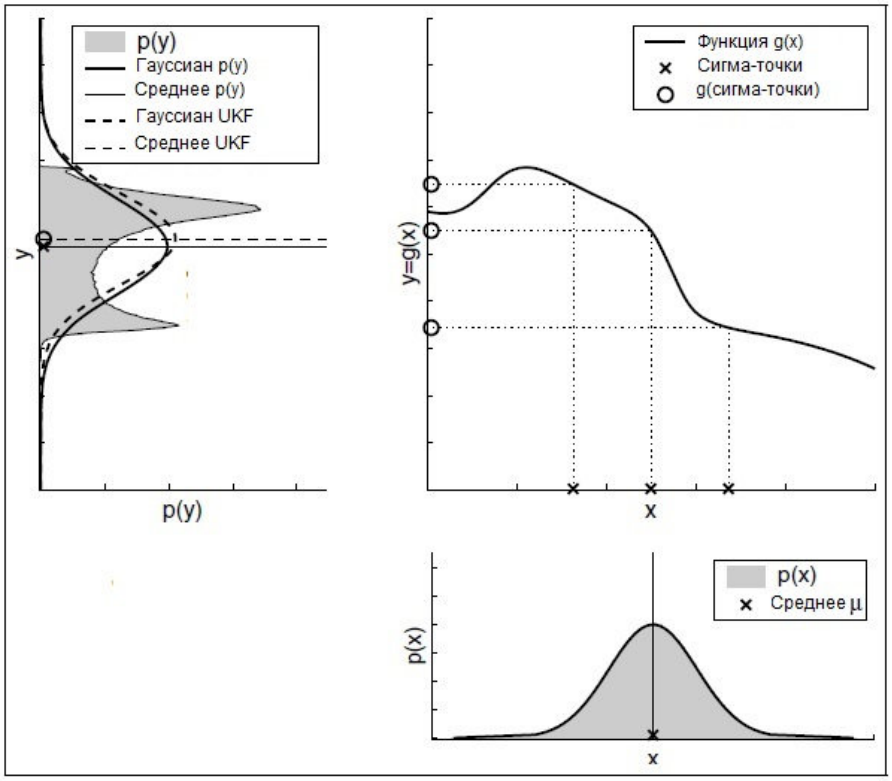


Рис. 3.7 Иллюстрация линеаризации, применяемой в UKF. Сначала алгоритм фильтра извлекает $2n + 1$ взвешенных сигма-точек из n-мерного гауссиана (в данном примере $n = 1$). Эти сигма-точки передаются нелинейной функции g. Из множества сигма-точек извлекается гауссова функция (показана маленькими кружками на правом верхнем графике). Аналогично EKF, линеаризация вызывает ошибку приближения, наблюдаемую в виде разницы между линеаризованной гауссовой функцией (пунктирная линия) и гауссианом, вычисленном с помощью высокоточной оценки методом Монте-Карло (сплошная линия).

Параметр β можно подобрать таким образом, чтобы закодировать дополнительные данные более высокого порядка относительно распределения, лежащего в основе гауссовой функции. Для распределения, заданного точной гауссовой функцией, оптимальным выбором будет $\beta = 2$.

Затем сигма-точки передаются функции g, определяя, таким образом, изменение формы гауссиана.

(3.68)
$$\mathcal{Y}^{[i]} = g(\mathcal{X}^{[i]})$$

Параметры $(\mu' \, \Sigma')$ результирующей гауссовой функции извлекаются из помеченных сигма-точек $\mathcal{Y}^{[i]}$ в соответствии с

(3.69)
$$\mu' = \sum_{i=0}^{2n} w_m^{[i]} \mathcal{Y}^{[i]}$$

$$\Sigma' = \sum_{i=0}^{2n} w_c^{[i]} (\mathcal{Y}^{[i]} - \mu')(\mathcal{Y}^{[i]} - \mu')^T$$

На Рис. 3.8 показана зависимость unscented transform от степени неопределённости оригинальной гауссовой функции. Результаты разложения в ряд Тейлора для EKF для сравнения помещены на графике рядом с результатами преобразования UKF.

На Рис. 3.9 показано дополнительное сравнение между способами приближения UKF и EKF, в зависимости от локальной нелинейности функции g. Как легко заметить, unscented transform имеет большую точность по сравнению с разложением в ряд Тейлора первого порядка, используемое в EKF. Фактически, можно показать, что unscented transform дает точную оценку в пределах первых двух членов разложения Тейлора, а EKF охватывает только первый порядок. (Заметим, что и EKF, и UKF можно модифицировать для учёта членов более высоких порядков.)

3.4.2 Алгоритм UKF

Алгоритм UKF, использующий unscented transform представлен в Таблице 3.4. Формат входных и выходных данных аналогичен алгоритму EKF. В строке 2 с помощью равенства (3.66) определены сигма-точки предыдущей оценки, при этом γ обозначает $\sqrt{n+\lambda}$. Эти точки передаются через алгоритм прогнозирования в строке 3, без учёта шумов. Затем прогнозируемые математическое ожидание и дисперсия вычисляются из результирующей совокупности сигма-точек (строки 4 и 5). Слагаемое R_t в строке 5 добавляется к вычисленной с помощью сигма-точек ковариации с целью моделирования неопределённости, вносимой дополнительными шумами прогнозирования (сравнимо со строкой 3 алгоритма EKF в Таблице 3.3). Шумы прогнозирования R_t считаются аддитивными. Позже, в Главе 7, будет представлена версия алгоритма UKF, который выполняет более точную оценку прогнозируемых шумов и шумов измерений.

Новый набор сигма-точек извлекается из прогнозируемого гауссиана в строке 6. Этот набор $\bar{\mathcal{X}}_t$ уже выражает общую неопределённость после такта прогнозирования. В строке 7 для каждой сигма-точки вычисляется прогнозируемое наблюдение. Результирующие сигма-точки наблюдения $\bar{\mathcal{Z}}_t$ используются для вычисления прогнозируемого наблюдения $\hat{z}_t$ и величины его неопределённости, S_t. Матрица Q_t представляет собой ковариационную матрицу аддитивных шумов измерения. Заметим, что S_t отображает ту же неопределённость, что и $H_t \bar{\Sigma}_t H_t^T + Q_t$ в строке 4 алгоритма EKF. В строке 10 Таблицы 3.3 определяется взаимная ковариация между состоянием и наблюдением, которая затем используется в строке 11 для вычисления усиления фильтра Калмана K_t. Взаимная ковариация $\bar{\Sigma}_t^{x,z}$ соответствует члену $\bar{\Sigma}_t H)t^T$ в строке 4 алгоритма EKF. Учитывая все сказанное, легко показать, что обновление оценки, выполненное в строках 12 и 13, эквивалентно обновлению, выполняемому алгоритмом EKF.

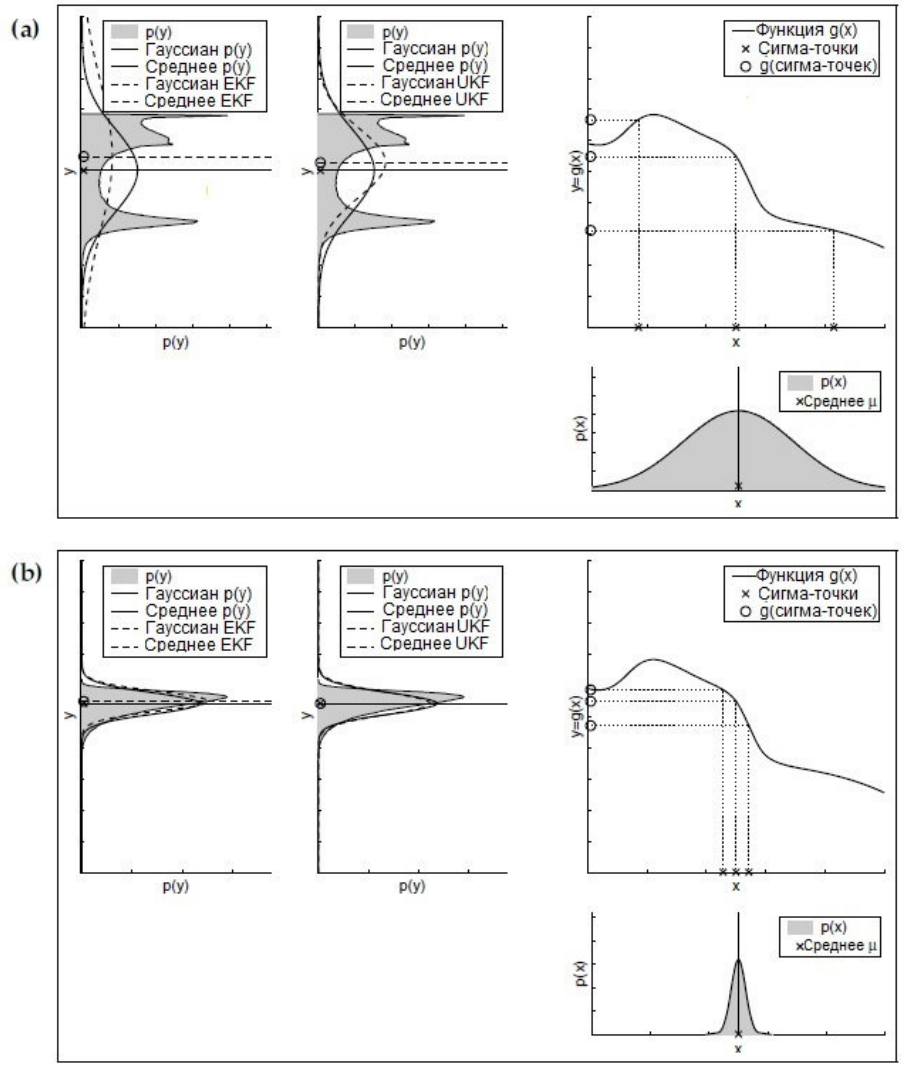


Рис. 3.8 Результаты линеаризации для UKF в зависимости от неопределённости оригинальной гауссовой функции. Результаты линеаризации EKF показаны для сравнения на Рис. 3.5. Unscented transform имеет меньшую ошибку аппроксимации, что видно по заметной схожести между пунктирным и сплошным графиком гауссовых функций.

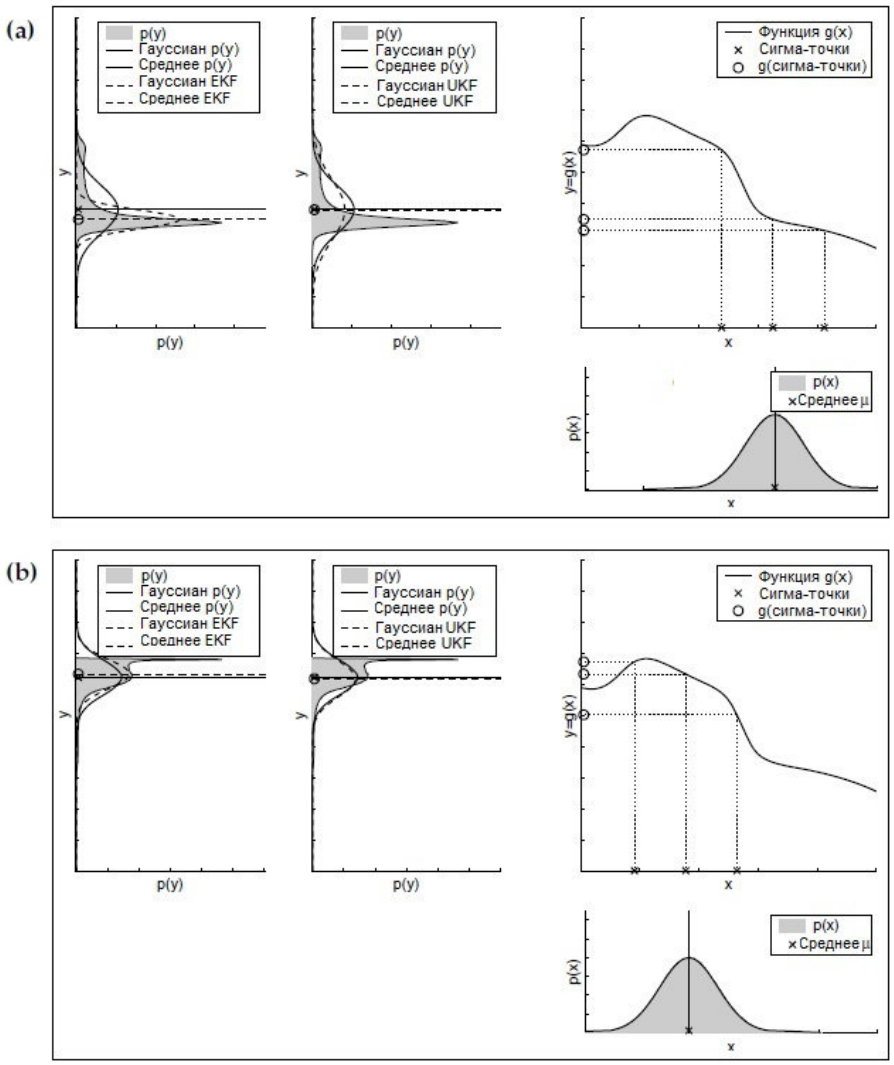


Рис. 3.9 Результаты линеаризации UKF в зависимости от математического ожидания оригинальной гауссовой функции. Результаты линеаризации EKF также показаны для сравнения (см. Рис. 3.6). Линеаризация с помощью сигма-точек имеет меньшую ошибку аппроксимации, что видно по заметной схожести между пунктирным и сплошным графиком гауссовых функций.

Асимптотическая сложность алгоритма UKF аналогична EKF. На практике EKF часто несколько быстрее, но алгоритм UKF остаётся очень эффективным, даже учитывая пропорциональное увеличение затрат времени. Более того, UKF наследует преимущества линеаризации unscented transform. Для чисто линейных систем будет показано, что оценки, сгенерированные с помощью UKF, идентичны оценкам калмановского фильтра.

1: **Algorithm Unscented_Kalman_filter** $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

2: $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$
3: $\bar{\mathcal{X}}_t^* = g(u_t, \mathcal{X}_{t-1})$
4: $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$
5: $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^T + R_t$
6: $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t})$
7: $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$
8: $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$
9: $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$
10: $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$
11: $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$
12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
13: $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$
14: *return* μ_t, Σ_t

Таблица 3.4 Алгоритм unscented Kalman filter. Переменная n обозначает размерность вектора состояний.

Для нелинейных систем UKF даёт аналогичные или лучшие результаты по сравнению с EKF, при этом разница по отношению к EKF зависит от нелинейности и разброса неопределённости предыдущего состояния. Во многих практических реализациях разница между EKF и UKF ничтожна.

Другим преимуществом UKF является факт того, что они не требуют вычисления якобианов, которые, в некоторых случаях, трудно определить. В силу этого UKF часто называют *фильтром без производных*.

ФИЛЬТР БЕЗ ПРОИЗВОДНЫХ

Наконец, unscented transform имеет некоторое сходство с представлением на основе выборки, используемом для многочастичного фильтра, которых будет обсуждаться в следующей главе. Однако, ключевое различие состоит в том, что сигма-точки в unscented transform точно определены, а в многочастичных фильтрах выборка выполняется случайным образом и приводит к важным последствиям. Если лежащее в основе распределение приближено к нормальному, тогда UKF представление значительно более эффективно по сравнению с представлением многочастичного фильтра. Если, с другой стороны, оценка очень далека от нормальной, тогда UKF представление слишком ограничено и фильтр плохо выполняет определение точек.

3.5 Информационный фильтр

Дополнением калмановского фильтра является *информационный фильтр* (information filter- IF). Аналогично KF и его нелинейным версиям, EKF и UKF, информационный фильтр отображает оценку в виде гауссовой функции. Поэтому стандартный информационный фильтр использует те же допущения, что и фильтр Калмана. Ключевым различием между KF и IF является способ представления гауссовой оценки. Во всем семействе ал-

горитмов фильтров Калмана гауссовы функции представлены моментами (математическим ожиданием, ковариацией), а в информационных фильтрах – в каноническом представлении параметров, состоящем из информационной матрицы и информационного вектора. Разница параметризации приводит к разным уравнениям обновления. В частности, то, что вычислительно сложно в одном виде параметризации оказывается простым в другом (и наоборот). Каноническая параметризация и параметризация в виде моментов часто считаются взаимодополняющими, как и IF и KF.

3.5.1 Каноническая параметризация

КАНОНИЧЕСКАЯ ПАРАМЕТРИЗАЦИЯ

Каноническая параметризация многомерной гауссовой функции задаётся матрицей Ω и вектором ξ. Матрица Ω – это инвертированная ковариационная матрица

(3.70)
$$\Omega = \Sigma^{-1}$$

ИНФОРМАЦИОННАЯ МАТРИЦА

Ω называется *информационной матрицей*, или, иногда, *матрицей точности*. Вектор ξ называется *информационным вектором*. Он определяется как

(3.71)
$$\xi = \Sigma^{-1}\mu$$

Легко заметить, что Ω и ξ образуют полную параметризацию гауссовой функции. В частности, математическое ожидание и ковариацию гауссовой функции можно легко получить из канонической параметризации путём инверсии (3.70) и (3.71):

(3.72)
$$\Sigma = \Omega^{-1}$$

(3.73)
$$\mu = \Omega^{-1}\xi$$

Каноническая параметризация часто выводится путём умножения экспоненты гауссовой функции. Согласно (3.1) определим многомерное нормальное распределение следующим образом:

(3.74)
$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

Очевидная последовательность преобразований ведёт к следующей параметризации:

(3.75)
$$\begin{aligned} p(x) &= \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu - \frac{1}{2}\mu^T\Sigma^{-1}\mu\right\} \\ &= \underbrace{\det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\mu^T\Sigma^{-1}\mu\right\}}_{const.} \exp\left\{-\tfrac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu\right\} \end{aligned}$$

Множитель, помеченный “const.”, не зависит от целевой переменной x. Поэтому, его можно заменить нормализующим членом η.

(3.76)
$$p(x) = \eta \exp\left\{-\frac{1}{2}x^T \Sigma^{-1} x + x^T \Sigma^{-1}\mu\right\}$$

В таком виде удобнее использовать параметризацию в виде канонических параметров Ω и ξ.

(3.77)
$$p(x) = \eta \exp\left\{-\frac{1}{2}x^T \Omega x + x^T \xi\right\}$$

Во многих отношениях, каноническая параметризация более элегантна по сравнению с параметризацией в виде моментов. В частности, отрицательный логарифм гауссовой функции представляет собой квадратичную функцию в x, с каноническими параметрами Ω и ξ:

(3.78)
$$-\log p(x) = const. + \frac{1}{2}x^T \Omega x - x^T \xi$$

Здесь "const."означает константу. Читатель может заметить, что мы не можем использовать символ η для обозначения этой константы, поскольку отрицательные логарифмы вероятностей не нормализуются до 1. Отрицательный логарифм распределения $p(x)$ квадратичен по x, его квадратичный член параметризован по Ω, а линейный – по ξ. Фактически, для гауссовых функций Ω должна быть неотрицательно полуопределена, поэтому $-\log p(x)$ является квадратичной функцией расстояния с математическим ожиданием $\mu = \Omega^{-1}\xi$ £. Это легко проверяется установкой первой производной (3.78) в нуль:

(3.79)
$$\frac{\partial[-\log p(x)]}{\partial x} = 0 \Leftrightarrow \Omega x - \xi = 0 \Leftrightarrow x = \Omega^{-1}\xi$$

Матрица Ω определяет скорость, с которой возрастает функция расстояния в различных измерениях переменной x. Квадратичное расстояние, взвешенное матрицей Ω, называется *расстоянием Махаланобиса.*

РАССТОЯНИЕ МАХАЛАНОБИСА

1:	**Algorithm Information_filter** $(\xi_{t-1}, \Omega_{t-1}, u_t, z_t)$:
2:	$\bar{\Omega}_t = (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1}$
3:	$\bar{\xi}_t = \bar{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t)$
4:	$\Omega_t = C_t^T Q_t^{-1} C_t + \bar{\Omega}_t$
5:	$\xi_t = C_t^T Q_t^{-1} z_t + \bar{\xi}_t$
6:	*return* ξ_t, Ω_t

Таблица 3.5 Алгоритм информационного фильтра.

3.5.2 Алгоритм информационного фильтра

Таблице 3.4 приводится модифицированный вариант алгоритма, который известен под названием *«информационного фильтра»*. На вход подаётся нормальная гауссова функция в классическом представлении, заданная вектором ξ_{t-1} и матрицей Ω_{t-1} и определяющая оценку состояния в момент времени $t-1$. Аналогично всем байесовским фильтрам, входные данные включают векторы управляющего воздействия u_t и измерения z_t. Выходными данными фильтра являются новые значения вектора ξ_t матрицы Ω_t и обновлённая гауссова функция, соответственно.

Этап обновления предусматривает использование матриц A_t, B_t, C_t, R_t, и Q_t, описанных в Разделе 3.2. Алгоритм информационного фильтра основан на предположении о том, что вероятности перехода состояний и измерений могут быть выражены с помощью следующих гауссовых уравнений, впервые представленных в (3.2) и (3.5):

(3.80)
$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

(3.81)
$$z_t = C_t x_t + \delta_t$$

В данном случае, R_t и Q_t, – это ковариации переменных нулевого среднего зашумления, ε_t и δ_t, соответственно.

Как и классический калмановский фильтр, информационный фильтр предусматривает обновление в две фазы, экстраполяции и коррекции. Фаза экстраполяции выполняется с помощью выражений в строках 2 и 3 Таблицы 3.4. Параметры $\bar{\xi}_t$ и $\bar{\Omega}_t$ описывают гауссову функцию оценки по вектору состояний x_t, после учёта управляющего воздействия u_t , но до учёта измерений z_t. Измерения принимаются во внимание в строках 4 и 5, где происходит обновление значений параметров оценки на основе измерений z_t в момент времени t.

Эти две фазы обновления могут очень сильно различаться по сложности, особенно для случая, когда пространство состояний имеет большое число измерений. Как показано в Таблице 3.5, во время фазы экстраполяции происходит инверсия двух матриц размера $n \times n$, где n – размерность пространства состояний. Инверсия матриц, по текущим оценкам, требует затрат времени, приблизительно равных $O(n^{2.4})$. Но для фильтров Калмана обновление имеет аддитивный характер и требует, максимум, сложности $O(n^2)$. Требуемое время ещё более уменьшается, если набор переменных зависит только от управляющего воздействия, или если перенос переменных выполняется независимо друг от друга. Для информационного фильтра ситуация обратная, поскольку в нем аддитивно происходит обновление измерений. Поэтому максимальные затраты времени также составят $O(n^2)$, а эффективность улучшится в случае, если измерения содержат только данные о наборе всех переменных состояния для конкретного момента времени. Обновление измерений в калмановских фильтрах очень затратно, из-за необходимости инверсии матриц, а вычислительная сложность, в худшем случае, достигает $O(n^{2.4})$. Это хорошо иллюстрирует взаимодополняющий характер калмановских и информационных фильтров.

3.5.3 Математический вывод алгоритма информационного фильтра

Вывод для информационного фильтра аналогичен выводу для фильтра Калмана.

ТАКТ ПРОГНОЗИРОВАНИЯ

Для математического вывода *такта прогнозирования* (строки 2 и 3 в Таблице 3.5), начнём с соответствующих равенств обновления для калмановских фильтров которые находятся в строках 2 и 3 алгоритма в Таблице 3.1 и повторно приведены здесь для удобства читателя:

(3.82)
$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

(3.83)
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Далее в такте прогнозирования информационного фильтра производится замена моментов μ и Σ каноническими параметрами ξ и Ω, в соответствии с их определениями в (3.72) и (3.73):

(3.84)
$$\mu_{t-1} = \Omega_{t-1}^{-1} \xi_{t-1}$$

(3.85)
$$\Sigma_{t-1} = \Omega_{t-1}^{-1}$$

Подставка этих выражений в (3.82) и (3.83) даст набор равенств для прогнозирования

(3.86)
$$\bar{\Omega}_t = (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1}$$

(3.87)
$$\bar{\xi}_t = \bar{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t)$$

Эти равенства идентичны приведённым в Таблице 3.5. Легко заметить, что такт прогнозирования включает две вложенные инверсии потенциально больших матриц. Этих инверсий можно избежать, если обновление на такте движения затрагивает только небольшое число переменных состояния, что будет обсуждаться чуть позже.

ОБНОВЛЕНИЕ ИЗМЕРЕНИЯ

Вывод для *обновления измерения* ещё проще. Начнём с гауссовой функции оценки в момент времени t, упомянутой в уравнении (3.35) и повторённой здесь:

(3.88)
$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t) - \frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\right\}$$

Для гауссовых функций, представленных в канонической форме, распределение задано следующим образом:

(3.89)
$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}x_t^T C_t^T Q_t^{-1} C_t x_t + x_t^T C_t^T Q_t^{-1} z_t - \frac{1}{2}x_t^T \bar{\Omega}_t x_t + x_t^T \bar{\xi}_t\right\}$$

будучи переписанным в виде экспоненты, выражение разрешается как

(3.90)

$$bel(x_t) = \eta \exp\left\{-\frac{1}{2} x_t^T [C_t^T Q_t^{-1} C_t + \bar{\Omega}_t] x_t + x_t^T [C_t^T Q_t^{-1} z_t + \bar{\xi}_t]\right\}$$

Сейчас можно окончательно сформулировать равенства обновления измерений, собрав все члены в квадратные скобки:

(3.91)

$$\xi_t = C_t^T Q_t^{-1} z_t + \bar{\xi}_t$$

(3.92)

$$\Omega_t = C_t^T Q_t^{-1} C_t + \bar{\Omega}_t$$

Эти равенства идентичны приведённым в строках 4 и 5, такта обновления измерения Таблицы 3.5.

3.5.4 Алгоритм обобщённого информационного фильтра

Обобщенный информационный фильтр (extended information – EIF) расширяет информационный фильтр для нелинейного случая, практически таким же образом, как EKF представляет собой нелинейное расширение фильтра Калмана. Алгоритм EIF приведён в Таблице 3.6. Прогнозирование выполняется в строках с 2 по 4, а обновление измерения - с 5 по 7. Эти уравнения такта обновления, по большей части, аналогичны линейному информационному фильтру, где функции g и h (и их якобианы G_t и H_t) заменяют параметры линейной модели A_t, B_t, и C_t. Как и прежде, g и h обозначают нелинейные переходные функции состояния и измерения, соответственно. Они были впервые определены в (3.48) и (3.49) и повторно приводятся ниже:

(3.93)

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t$$

(3.94)

$$z_t = h(x_t) + \delta_t$$

К сожалению функции g и h принимают на вход данные о состоянии, что требует восстановления оценки состояния μ на основании канонических параметров. Восстановление выполняется в строке 2, в которой состояние μ_{t-1} очевидным образом вычисляется из Ω_{t-1} и ξ_{t-1}. В строке 5 вычисляется состояние $\bar{\mu}_t$, используя уже знакомое по EKF выражение (строка 2 в Таблице 3.3). Кажется, что необходимость восстанавливать оценку состояния противоречит идее представления фильтра с каноническими параметрами. Мы вернёмся к этой теме при обсуждении использования обобщённых информационных фильтров в контексте построения карт для роботов.

1: **Algorithm Extended_ information_filter** $(\xi_{t-1}, \Omega_{t-1}, u_t, z_t)$:

2: $\mu_{t-1} = \Omega_{t-1}^{-1}\xi_{t-1}$
3: $\bar{\Omega}_t = (G_t\Omega_{t-1}^{-1}G_t^T + R_t)^{-1}$
4: $\bar{\xi}_t = \bar{\Omega}_t g(u_t, \mu_{t-1})$
5: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
6: $\Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t$
7: $\xi_t = \bar{\xi}_t + H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t]$
8: *return* ξ_t, Ω_t

Таблица 3.6 Алгоритм расширенного информационного фильтра (EIF).

3.5.5 Математический вывод обобщённого информационного фильтра

Обобщённый информационный фильтр легко вывести, используя, в основном, ту же самую линеаризацию, которая была ранее использована для обобщённых фильтров Калмана. Как в (3.51) и (3.53), в обобщённом информационном фильтре с помощью разложения в ряд Тейлора оцениваются функции g и h:

(3.95)
$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

(3.96)
$$h(x_t) \approx h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)$$

здесь G_t и H_t - якобианы g и h для μ_{t-1} и $\bar{\mu}_t$, соответственно:

(3.97)
$$G_t = g'(u_t, \mu_{t-1})$$

(3.98)
$$H_t = h'(\bar{\mu}_t)$$

Эти определения аналогичны используемым в EKF. Выражения для такта экстраполяции выводятся из строк 2 и 3 алгоритма EKF (Таблица 3.3), которые приведены ниже:

(3.99)
$$\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + R_t$$

(3.100)
$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

Замена Σ_{t-1} на Ω_{t-1}^{-1} и $\bar{\mu}_t$ на $\bar{\Omega}_t^{-1}\xi_t$ даёт уравнения экстраполяции для расширенного информационного фильтра:

(3.101)
$$\bar{\Omega}_t = (G_t\Omega_{t-1}^{-1}G_t^T + R_t)^{-1}$$

(3.102)
$$\bar{\xi}_t = \bar{\Omega}_t g(u_t, \Omega_{t-1}^{-1}\xi_{t-1})$$

Обновление измерения выводится из уравнений (3.60) и (3.61). В частности, (3.61) определяет следующую апостериорную гауссову функцию:

(3.103)

$$bel(x_t) = \eta \exp\{-\frac{1}{2}(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t))^T Q_t^{-1}$$

$$(z_t - h(\bar{\mu}_t) - H_t(x_t - \bar{\mu}_t)) - \tfrac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\}$$

Умножение экспоненты и переопределение членов даёт следующее выражение для апостериорной вероятности:

(3.104)

$$\begin{aligned} bel(x_t) &= \eta \exp\{-\frac{1}{2}x_t^T H_t^T Q_t^{-1} H_t x_t + x_t^T H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t] \\ &\quad -\tfrac{1}{2}x_t^T \bar{\Sigma}_t^{-1} x_t + x_t^T \bar{\Sigma}_t^{-1}\bar{\mu}_t\} \\ &= \eta \exp\{-\tfrac{1}{2}x_t^T [H_t^T Q_t^{-1} H_t + \bar{\Sigma}_t^{-1}]x_t \\ &\quad +x_t^T [H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t] + \bar{\Sigma}_t^{-1}\bar{\mu}_t] \end{aligned}$$

Учитывая, что $\bar{\Sigma}_t^{-1} = \bar{\Omega}_t$, это выражение разрешается до следующего информационного вида:

(3.105)

$$\begin{aligned} bel(x_t) = \eta \exp\{-\frac{1}{2}x_t^T [H_t^T Q_t^{-1} H_t + \bar{\Omega}_t]x_t \\ +x_t^T [H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t] + \bar{\xi}_t]. \end{aligned}$$

Теперь можно сформулировать уравнения обновления измерения, собрав все члены в квадратных скобках:

(3.106)

$$\Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t$$

(3.107)

$$\xi_t = \bar{\xi}_t + H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) + H_t\bar{\mu}_t]$$

3.5.6 Практические соображения

В контексте задач робототехники информационный фильтр имеет несколько преимуществ по сравнению с фильтром Калмана. Например, глобальное представление неопределённости в информационном фильтре достаточно простое: необходимо установить $\Omega = 0$. При использовании моментов, такие величины общей неопределённости дают ковариацию бесконечной степени. Это особенно проблематично, когда измерения датчиков несут информацию об ограниченном наборе всех переменных состояния, что часто встречается в робототехнике. Для решения таких ситуаций средствами EKF необходимо предпринимать дополнительные действия. Многие реализации информационного фильтра, которые будут обсуждаться позже, демонстрируют более стабильные количественные характеристики по сравнению с калмановским фильтром.

Как мы увидим в следующих главах книги, информационный фильтр с несколькими дополнениями позволяют роботу интегрировать информацию без немедленного отражения её в вероятностном виде. Это может стать решающим преимуществом в сложных задачах оценки, включающих сотни или даже миллионы переменных. Для таких больших задач попытки интеграции по типу фильтра Калмана ведёт к серьёзным вычислительным трудностям, поскольку каждую новую порцию информации требуется распространить в большой системе переменных. С помощью соответствующих модификаций информационного фильтра возможно обойти эту проблему, локально добавив эту информацию в систему. Однако, это свойство *нехарактерно* для простого информационного фильтра, обсуждаемого сейчас. В Главе 12 мы обсудим обобщения этого фильтра.

Другое преимущество информационного фильтра перед фильтром Калмана возникает в силу его изначальной приспособленности для решения задачи нескольких роботов. Такие задачи часто включают интеграцию показаний датчиков, которые собираются децентрализовано, что обычно выполняется с помощью теоремы Байеса. Будучи представлена в логарифмическом виде, теорема Байеса становится суммой. Как было сказано выше, канонические параметры информационных фильтров выражают вероятность в логарифмической форме, поэтому интеграция информации представляет собой суммированием данных с нескольких роботов. Сложение коммутативно, поэтому с помощью информационных фильтров часто возможно интегрировать информацию в произвольном порядке, с произвольными задержками и в полностью децентрализованной манере. Хотя то же самое возможно сделать, используя параметризацию в виде моментов (в конце концов, они выражают одну и ту же информацию), необходимо будет предпринять большие дополнительные усилия. Несмотря на это достоинство, применение информационных фильтров в системах из нескольких роботов остаётся, по большей части, не исследованным. Мы вернёмся к задаче нескольких роботов в Главе 12.

Обратной стороной преимуществ информационного фильтра является наличие важных ограничений. Основным недостатком обобщённого информационного фильтра можно считать необходимость восстанавливать оценку состояния для нелинейных систем на такте обновления. В представленном виде это потребует инверсии информационной матрицы. Вдобавок, информационные фильтры требуют дополнительных операций инверсии матриц на этапе экстраполяции. Для многих задач робототехники использование EKF не требует инверсии матриц сравнимого размера. Считается, что для пространств состояний с большим числом измерений информационный фильтр проигрывает фильтру Калмана по вычислительной сложности, что является одной из причин подавляющей популярности последнего.

Как будет позже показано в книге, эти ограничения не обязательно применимы для задач, в которых информационная матрица имеет явную структуру. Во многих задачах робототехники взаимодействие переменных состояния локально, следовательно, информационная матрица разрежена. Эта разреженность *не* соотносится с разреженностью матрицы ковариации.

Можно считать информационные фильтры графами, в которых состояния соединены, когда соответствующие элементы вне главной диагонали информационной матрицы не равны нулю. Разреженные информационные матрицы соответствуют разреженным графам, которые широко известны под названием *марковских случайных полей*. Для таких полей имеется множество алгоритмов, эффективно выполняющих операции общей оценки и обновления с названиями наподобие *«алгоритма распространения дове-*

МАРКОВСКИЕ СЛУЧАЙНЫЕ ПОЛЯ

рия» (loopy belief propagation). В ходе изложения мы столкнёмся с проблемой картографирования, в которой информационная матрица будет (относительно) разрежена, и разработаем обобщенный информационный фильтр, который будет существенно эффективнее как фильтров Калмана, так и информационных фильтров с неразреженными матрицами.

3.6 Вывод

В этом разделе мы представили эффективные алгоритмы байесовских фильтров, которые выражают апостериорные оценки в виде многомерных гауссовых функций. Было отмечено, что:

• Гауссовы функции могут быть представлены двумя различными способами: параметризацией в виде моментов и канонической параметризацией. Параметризация в виде моментов состоит из математического ожидания (первый момент) и ковариации (второй момент) функции нормального распределения. Каноническая или обычная параметризация состоит из информационной матрицы и информационного вектора. Оба вида параметризации взаимно дополняют друг друга, и каждую можно восстановить из другой, с помощью инверсию матриц.

• Для обоих видов параметризации можно реализовать байесовские фильтры. При использовании параметризации в виде моментов получившийся фильтр называется фильтром Калмана. Дополнением фильтра Калмана является информационный фильтр, представляющий апостериорную вероятность с использованием канонической параметризации. Обновление фильтра Калмана на основе управляющего воздействия вычислительно просто, в то время, как учёт измерения более труден. Для информационного фильтра ситуация обратная, учёт измерения достаточно прост, но обновить фильтр на основе управляющего сигнала вычислительно сложно.

• Для вычисления правильной апостериорной вероятности в обоих фильтрах требуется соблюсти три условия. Во-первых, начальная оценка должна быть нормальным распределением. Во-вторых, переходная вероятность состояния должна выражаться функцией с линейным аргументом и добавлением независимого нормально распределенного зашумления. В третьих, то же справедливо для вероятности измерения: она должна иметь линейный аргумент с добавлением гауссового шума. Системы, отвечающие этим требованиям, называются линейными гауссовыми системами.

• Оба фильтра можно обобщить для нелинейных задач. Один из методов, описанных в этой главе, состоит в вычислении касательной к нелинейной функции. Касательные линейны, что позволяет использовать это в обоих фильтрах. Способ нахождения касательной называется разложением в ряд Тейлора и включает вычисление первой производной целевой функции с последующей оценкой её в определённой точке. Результатом этой операции является матрица, известная как якобиан, а получившиеся в результате фильтры называют «обобщенными».

• Так называемый unscented Kalman filter использует другой метод линеаризации, который называется «unscented transform». Он вычисляет значения функции в нескольких выбранных точках, и выполняет линеаризо-

ванную аппроксимацию на основе полученных результатов. Этот фильтр может быть использован без необходимости вычисления якобианов, и, потому, часто именуется «фильтром без производных». Unscented Kalman filter эквивалентен фильтру Калмана для линейных систем, но часто более эффективен для нелинейных систем. Вычислительная сложность этого фильтра и обобщённого фильтра Калмана одинаковы.

• Точность разложения в ряд Тейлора и unscented transform зависит от двух факторов: степени нелинейности системы и ширины апостериорного распределения. Обобщённые фильтры обычно показывают хорошие результаты, когда состояние системы известно с относительно высокой точностью, и оставшаяся ковариация мала. Чем больше степень неопределённости, чем больше ошибка, вносимая линеаризацией.

• Одним из главных преимуществ гауссовых фильтров является их вычислительная эффективность: обновление требует времени, кратного размерности пространства состояний, что значительно лучше некоторых методов, описанных в следующей главе. Основным недостатком является ограничение применимости одномодальными гауссовыми распределениями.

• Обобщение гауссовых функций для мультимодальных апостериорных распределений известно под названием фильтра Калмана с несколькими гипотезами. Этот фильтр отображает апостериорную вероятность в виде смеси гауссовых функций, представляющих собой взвешенную сумму нормальных распределений. Специфика обновления этого фильтра требует механизмов разделения, слияния или отсечения отдельных гауссианов. Фильтры Калмана с несколькими гипотезами хорошо подходят, в частности, для задач с дискретной интеграцией данных, часто встречающихся в робототехнике.

• В случае многомерных гауссовых функций и фильтр Калмана, и информационный фильтр имеют взаимно обратные сильные и слабые стороны. Однако, калмановский фильтр и его нелинейное обобщение, обобщенный калмановский фильтр, значительно более популярны по сравнению с информационным фильтром.

Выбор материалов для этой главы основывается на самых популярных, на сегодняшний день, методах робототехники. Существует огромное количество вариантов и обобщений гауссовых фильтров, предназначенных для обхода различных ограничений и недостатков конкретных реализаций.

Значительное число алгоритмов в этой книге основано на гауссовых фильтрах. Множество практических проблем робототехники требует использования обобщений, с разреженными структурами или факторизацией апостериорного распределения.

3.7 Библиографические примечания

Фильтры Калмана были отрыты Свирлингом (1958) и Калманом (1960). Обычно, они представляются как оптимальная функция оценки методом наименьших квадратов, и, реже, в качестве метода вычисления апостериорных распределений, хотя, при соответствующих допущениях, обе точки зрения идентичны. Есть множество великолепных учебников по калманов-

ским и информационным фильтрам, включая работу Мэйбека (Maybeck, 1990) и Язвински (Jazwinsky, 1970). На сегодняшний день развитие калмановских фильтров с включением данных было описано в работах Бар-Шалома и Фортманна (Bar-Shalom и Fortmann,1988), Бар-Шалома и Ли (Bar- Shalom и Li, 1998).

Лемму об обращении матриц можно найти в работе Голаба и Лоана (Golub и Loan,1986). Согласно Копперсмиту и Винограду (Coppersmith, Winograd,1990), инверсия матриц может быть выполнена за время $O(n^2, 376)$. Этот результат стал самым заметным в серии работ по улучшению вычислительной сложности для алгоритма отбора переменных ниже $O(n^3)$. Серия началась с основополагающей работы Страссена (Strassen, 1969), в которой он сформулировал алгоритм, требующий $O(n^2, 807)$. Кавер и Томас (Cover и Thomas, 1991) выполнили обзор теории информации с акцентом на дискретные системы. Unscented Kalman filter был создан Юлиером и Ульманном (Julier и Uhlmann, 1997). Сравнение UKF и EKF в контексте различных задач оценки состояния можно найти в работе ван де Мерве (van der Merwe, 2004). Минка (Minka, 2001) выполнил работу по подбору моментов (moments matching) и допустимой фильтрации плотностей для гауссовых смесей.

3.8 Упражнения

1. В этом и последующих упражнениях требуется разработать фильтр Калмана для простой динамической системы: автомобиль с линейной динамикой двигается в линейной окружающей среде. Для простоты допустим, что $\Delta t = 1$. Положение автомобиля в момент t задано функцией x_tA. Его скорость $\dot{x}_t$, ускорение - $\ddot{x}_t$. Допустим, ускорение в каждый момент времени определяется случайно, согласно гауссовой функции с нулевым математическим ожиданием и ковариацией $\sigma^2 = 1$.

(a) Каков минимальный вектор состояний фильтра Калмана, чтобы результирующая система была марковской?

(b) Для полученного вектора состояний найти переходную вероятность состояния $p(x_t|u_t, x_{t-1})$. Подсказка: эта функция перехода будет состоять из линейных матриц A и B, а ковариации зашумления R (см. выражение (3.4) и Таблицу 3.1).

(c) Реализовать такт экстраполяции состояния для фильтра Калмана. Допустим, известно, что в момент времени $t = 0$, $x_0 = \dot{x}_0 = \ddot{x}_0 = 0$. Вычислить распределения состояний для моментов времени $t = 1, 2, ..., 5$.

(d) Для каждого значения t, изобразить точками на графике апостериорное распределение по x и $\dot{x}$, где x – горизонтальная, а $\dot{x}$ – вертикальная ось. Для каждого апостериорного распределения необходимо изобразить *эллипс неопределённости*, представляющий собой эллипс из точек на расстоянии одного стандартного отклонения от среднего. Подсказка: Если нет доступа к математической библиотеке, можно найти эллипсы путём анализа характеристического значения матрицы ковариации.

ЭЛЛИПС НЕОПРЕДЕЛЁННОСТИ

(e) Что произойдёт с корреляцией между x_t и $\dot{x}_t$ при $t \uparrow \infty$?

2. Добавим к фильтру Калмана измерения. Допустим, в момент времени

t возможно получить зашумленное наблюдение x. Предполагается, что датчик измеряет истинное местоположение, искажённое гауссовским шумом с ковариацией $\sigma^2 = 10$.

(a) Определить модель измерения. Подсказка: Необходимо определить матрицу C и ещё одну матрицу Q (см. выражение (3.6) и Таблицу 3.1).

(b) Реализовать обновление измерения. Допустим, в момент времени $t = 5$, наблюдается значение измерения $z = 5$. Указать параметры гауссовой функции до и после обновления KF. Изобразить на графике эллипс неопределённости до и после учёта измерения (выше показано, каким образом изобразить эллипс неопределённости).

3. В подразделе 3.2.4 был выведен такт экстраполяции для KF. Этот такт часто выводится с помощью Z –преобразования или преобразования Фурье, используя теорему свёртки. Заново вывести выражения для такта прогнозирования, используя преобразования. *Примечание: Это упражнение требует знания преобразования свёртки, что выходит за пределы материала этой книги.*

4. В тексте было отмечено, что линеаризация EKF является приближением. Для того, чтобы выяснить качество аппроксимации, выполните следующее упражнение. Допустим, имеется мобильный робот, действующий в плоской среде. Его состояние описано координатами его местоположения $x - y$ и глобальным направлением ориентации θ. Допустим, с высокой степенью определённости известны x и y, но неизвестна ориентация θ. Это отражается следующей первоначальной оценкой

$$\mu = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \quad \text{и} \quad \Sigma = \begin{pmatrix} 0,01 & 0 & 0 \\ 0 & 0,01 & 0 \\ 0 & 0 & 10000 \end{pmatrix}$$

Графически изобразить лучшую модель апостериорного распределения для положения робота в момент после передвижения робота на $d = 1$ единиц вперёд. Для этого упражнения предположим, что робот движется плавно и без зашумления. Отсюда, прогнозируемое положение робота после движения будет

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + \cos\theta \\ y + \sin\theta \\ \theta \end{pmatrix}$$

На графике можно опустить θ, изобразив только апостериорное распределение для координат $x - y$.

(b) Учесть это движение в такте обновления EKF. Для этого необходимо определить функцию перехода состояний и линеаризовать ее. Затем необходимо сгенерировать новую гауссову оценку положения робота, используя линеаризованную модель. Необходимо дать точные математические равенства для каждого такта и определить результирующий гауссиан.

(c) Изобразить эллипс неопределённости гауссовой функции и сравнить его с интуитивным решением.

(d) Добавить в модель измерение. Пусть измерением будет зашумлённая проекция координаты x робота с ковариацией $Q = 0,01$. Необходимо определить модель измерения. Затем применить измерение к найденному апостериорному распределению и к оценке EKF, используя стандартные механики обобщённых фильтров Калмана. Дать точный результат для EKF и сравнить его с результатом найденной модели.

(e) Объясните разницу между найденной оценкой апостериорного распределения и гауссовой функцией, получившейся в результате работы EKF. Насколько существенны эти различия? Что можно изменить, чтобы сделать приближение более точными? Что произойдёт, если вместо координаты y будет известна начальная ориентация θ?

5. В фильтре Калмана в Таблице 3.1 отсутствует постоянный аддитивный член в моделях движения и измерения. Расширить алгоритм таким образом, чтобы он учитывался.

6. Доказать (на примере) существование разреженной информационной матрицы в многомерных гауссовых распределениях (размерности d), которые связывают все d переменных с коэффициентом корреляции ε, близким к 1. Назовём информационную матрицу разреженной, если все, кроме небольшого постоянного количества элементов каждой строки и столбца, равны нулю.

4 Непараметрические фильтры

Популярной альтернативой гауссовским методам являются *непараметрические фильтры*. Непараметрические фильтры ограничены фиксированной функциональной формой апостериорного распределения, скажем, в виде гауссиана. Вместо этого, в них производится аппроксимация апостериорной вероятности конечным числом значений, каждое из которых примерно соответствует области в пространстве состояний. Некоторые непараметрические байесовские фильтры основаны на декомпозиции пространства состояний, когда каждое значение соответствует суммарной вероятности апостериорной плотности в компактной области пространства состояний. Другие алгоритмы аппроксимируют пространство состояний случайной выборкой из апостериорного распределения. Во всех случаях количество параметров, используемых для аппроксимации, может различаться. Качество аппроксимации напрямую зависит от количества параметров представления апостериорной вероятности. По мере увеличения количества параметров до бесконечности, при определённых условиях сглаживания, непараметрические методы равномерно сходятся к истинному значению апостериорной вероятности.

В этой главе обсуждаются два непараметрических подхода для аппроксимации апостериорных распределений в непрерывных пространствах с конечным количеством значений. В первом пространство состояний разбивается на конечное число областей и отображается на апостериорном распределении в виде гистограммы. На гистограмме каждому участку присваивается только одно значение суммарной вероятности, поэтому лучше всего представлять метод гистограмм как кусочно-постоянное приближение

непрерывной плотности. Второй метод представляет апостериорные распределения в виде конечного множества элементов выборки. Результирующий фильтр известен под названием *«многочастичного фильтра»* и приобрёл большую популярность в робототехнике.

В обоих методах, гистограмм и многочастичных фильтров, не делается строгих параметрических заключений относительно апостериорной плотности, поэтому они хорошо приспособлены для представления сложных мультимодальных гипотез. Непараметрические методы часто являются наилучшим выбором, если робот сталкивается с проявлениями полной неопределённости и в случае возникновения серьёзных проблем ассоциации данных, порождающих отдельные, выраженные гипотезы. Однако, мощность представления эти методов достигается за счёт дополнительной вычислительной сложности.

К счастью, в обоих непараметрических методах, описанных в этой главе, возможна настройка количества параметров согласно ожидаемой сложности апостериорного распределения. Когда апостериорное распределение имеет низкую сложность (например, сосредоточено вокруг одного значения с малым отклонением), используется лишь небольшое число параметров. Для комплексных апостериорных распределений с несколькими модами, разбросанными по пространству состояний, количество параметров сильно возрастёт.

РЕСУРС-АДАПТИВНЫЕ АЛГОРИТМЫ

Методы, способные адаптировать количество параметров для отображения апостериорного распределения в реальном времени называются *адаптивными*. Они называются *ресурс-адаптивными*, если позволяют настраивать алгоритм соответственно доступным вычислительными ресурсам. Ресурс-адаптивные методы играют важную роль в робототехнике. Они открывают возможность для роботов принимать решения в реальном времени независимо от доступных вычислительных ресурсов. Многочастичные фильтры часто реализуются в виде ресурс-адаптивного алгоритмов, в которых количество частиц настраивается в процессе работы, основываясь на доступных вычислительных ресурсах.

4.1 Фильтр на основе гистограммирования

Алгоритм *фильтра на основе гистограммирования* разбивает пространство состояний на конечное число областей и отображает суммарное апостериорное распределение для каждой области в виде единственного значения вероятности. Для конечных пространств такие фильтры известны как *дискретные байесовские фильтры*, а для непрерывных пространств обычно называют *фильтры на основе гистограмм*. Сначала опишем дискретные байесовские фильтры, а затем обсудим их использование в непрерывных пространствах состояний.

4.1.1 Алгоритм дискретного байесовского фильтра

Дискретные байесовские фильтры применяются для задач с *конечными* пространствами состояний, в которых случайная переменная X_t может принимать конечное множество значений. Мы уже сталкивались с дискретным фильтром Байеса в Главе 2.4.2, обсуждая пример робота, оценивающего вероятность того, что дверь открыта. Некоторые из задач построения карт, обсуждаемые в более поздних главах, также включают дискретные случайные переменные. Например, в алгоритмах картографирования на основе

карт занятости подразумевается, что каждая точка среды может быть либо занятой или свободной. Соответствующая случайная переменная имеет бинарный характер и может принимать одно из двух различных значений. Как видите, конечные пространства состояний играют важную роль в робототехнике.

В Таблице 4.1 показан псевдокод дискретного байесовского фильтра. Этот код выводится из общего алгоритма байесовского фильтра в Таблице 2.1 путём замены интегрирования конечной суммой. Переменные x_i и x_k определяют отдельные состояния, которых может быть только конечное множество.

```
1:  Algorithm Discrete_ Bayes_filter (\{p_{k,t-1}\}, u_t, z_t) :
2:    for all k do
3:        \bar{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t-1}
4:        p_{k,t} = \eta p(z_t | X_t = x_k) \bar{p}_{k,t}
5:    endfor
6:    return \{p_{k,t}\}
```

Таблица 4.1 Дискретный байесовский фильтр. Здесь x_i, x_k определяют отдельные состояния.

Гипотеза в момент времени t – это представление вероятности для каждого состояния x_k, и обозначается $p_{k,t}$. Поэтому, на вход алгоритма поступает дискретное распределение вероятности $p_{k,t}$, наряду с последними данными управляющего воздействия u_t и измерения z_t. В строке 3 вычисляется прогноз, гипотеза относительно нового состояния, пока на основе одного лишь управляющего воздействия. Этот прогноз затем обновляется в строке 4 с учётом измерения.

Алгоритм дискретного байесовского фильтра популярен во многих областях обработки сигналов, где часто именуется проходом вперёд в *скрытой марковской модели*, или *СММ*.

СКРЫТАЯ МАРКОВСКАЯ МОДЕЛЬ

4.1.2 Непрерывное состояние

Особенный интерес представляет использование дискретных фильтров Байеса в качестве инструмента логической оценки для *непрерывных* пространств состояний. Как было сказано ранее, такие фильтры называются *фильтры на основе гистограмм*. На Рис. 4.1 показано, как с помощью гистограмм отображается случайная переменная и её нелинейное преобразование. Показанная кривая - это приближение гауссиана с помощью нелинейной функции в виде гистограммы. Начальное гауссовское распределение разбито на 10 интервалов. То же самое было сделано для предполагаемого распределения, но в двух интервалах вероятность так близка к нулю, что не видна на схеме. На Рис. 4.1 для сравнения также показаны истинные непрерывные распределения.

Гистограммы разбивает непрерывное пространство состояний на конечное множество *интервалов*, или *областей*:

(4.1)
$$dom(X_t) = \mathrm{x}_{1,t} \cup \mathrm{x}_{2,t} \cup ... \mathrm{x}_{K,t}$$

Здесь X_t означает уже знакомую случайную переменную, описывающую состояние робота в момент времени t. Функция $dom(X_t)$ определяет пространство состояний, которое является множеством возможных значений, которые может принимать X_t. Каждое значение $\mathrm{x}_{k,t}$ описывает выпуклую область. Вместе эти области образуют разбиение пространства состояний. Для каждого $i \neq k$ получаем $\mathrm{x}_{i,t} \cap \mathrm{x}_{k,t} = 0$ и $\bigcup_k \mathrm{x}_{k,t} = dom(X_t)$.

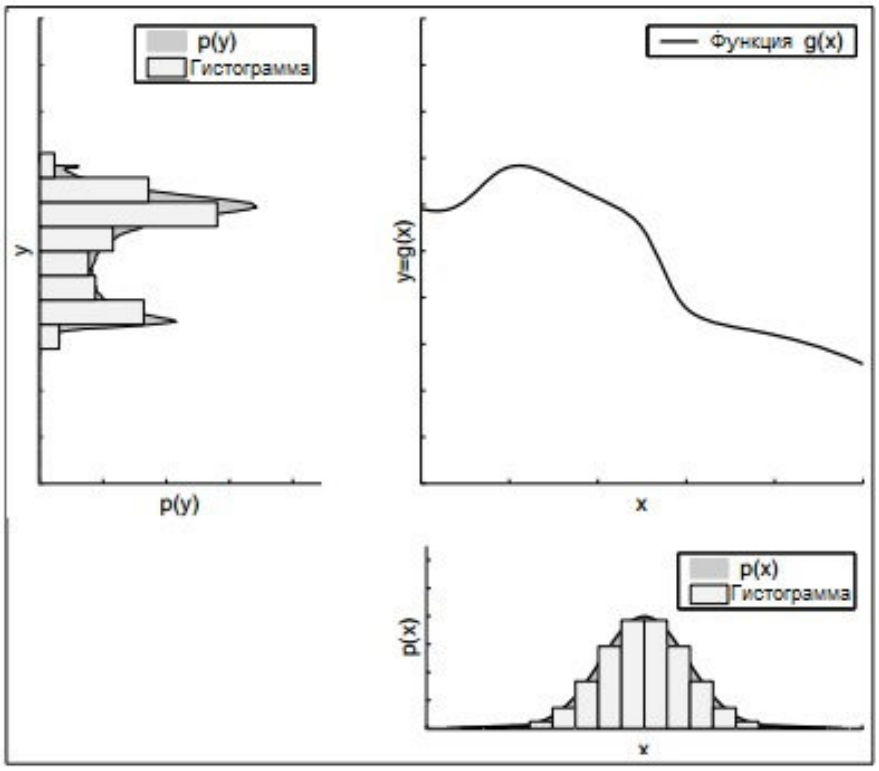


Рис. 4.1 Отображение непрерывной случайной переменной в виде гистограммы. Закрашенная серым зона на нижней правой схеме изображает плотность непрерывной случайной переменной, X. Распределение плотности в виде гистограммы изображено с заполнением светло-серым цветом. Случайная переменная пропускается через функцию, изображённую на верхней правой схеме. Плотность и аппроксимация на гистограмме результирующей случайной переменной, Y, изображены на верхней левой схеме. Гистограмма преобразованной случайной переменной была вычислена путём передачи нескольких значений точек каждого интервала X нелинейной функции.

Очевидным разложением непрерывного пространства состояний является многомерная сетка, где каждое значение $\mathrm{x}_{k,t}$ определяет ячейку сетки. Управляя разбиением такого представления, возможно настраивать баланс между точностью и вычислительной эффективностью. Разбиения с большим количеством ячеек имеют меньшую ошибку аппроксимации по сравнению с более грубыми, но, исключительно, за счёт возросшей вычислительной сложности.

Как уже обсуждалось, дискретный байесовский фильтр назначает каждой области $\mathrm{x}_{k,t}$ единственную вероятность $p_{k,t}$, не сохраняя дополнитель-

ной информации о распределении оценки. Таким образом, апостериорное распределение становится кусочно-постоянной ФРВ, где вероятность каждого состояния x_t в каждой области $\mathrm{x}_{k,t}$ постоянна:

(4.2)
$$p(x_t) = \frac{p_{k,t}}{|\mathrm{x}_{k,t}|}$$

Здесь $|\mathrm{x}_{k,t}|$ означает площадь области $\mathrm{x}_{k,t}$.

Если пространство состояний дискретно, условные вероятности $p(z_t|\mathrm{x}_{k,t})$, $p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1})$ чётко определены, и алгоритм может быть реализован в представленном виде. В непрерывных пространствах состояний обычно заданы плотности $p(x_t|u_t, x_{t-1})$ и $p(z_t|x_t)$, определённые для отдельных состояний (а не для областей пространства состояний). В тех случаях, когда все области $\mathrm{x}_{k,t}$ малы и имеют одинаковый размер, эти плотности обычно аппроксимируются подставкой $\mathrm{x}_{k,t}$ заменяющей функции. Например, можно просто «измерить» их с помощью среднего значения состояния в $\mathrm{x}_{k,t}$

(4.3)
$$\hat{x}_{k,t} = |\mathrm{x}_{k,t}|^{-1} \int_{\mathrm{x}_{k,t}} x_t dx_t$$

А затем произвести следующую замену

(4.4)
$$p(z_t|\mathrm{x}_{k,t}) \approx p(z_t|\hat{x}_{k,t})$$

(4.5)
$$p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1}) \approx \eta\,|\mathrm{x}_{k,t}|\,p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1})$$

Эти аппроксимации являются результатом кусочно-однородной интерпретации дискретного байесовского фильтра в (4.2), и аналога приближения разложением в ряд Тейлора, который используется в EKF.

4.1.3 Математический вывод приближения с помощью гистограммы

Чтобы доказать применимость приближения (4.4), нужно обратить внимание, что $p(z_t|\mathrm{x}_{k,t})$ можно выразить следующим интегралом:

(4.6)

$$\begin{aligned}
p(z_t|\mathrm{x}_{k,t}) &= \frac{p(z_t, \mathrm{x}_{k,t})}{p(\mathrm{x}_{k,t})} \\
&= \frac{\int_{\mathrm{x}_{k,t}} p(z_t, x_t) dx_t}{\int_{\mathrm{x}_{k,t}} p(x_t) dx_t} \\
&= \frac{\int_{\mathrm{x}_{k,t}} p(z_t|x_t) p(x_t) dx_t}{\int_{\mathrm{x}_{k,t}} p(x_t) dx_t} \\
&\overset{(4.2)}{=} \frac{\int_{\mathrm{x}_{k,t}} p(z_t|x_t) \frac{p_{k,t}}{|\mathrm{x}_{k,t}|} dx_t}{\int_{\mathrm{x}_{k,t}} \frac{p_{k,t}}{|\mathrm{x}_{k,t}|} dx_t} \\
&= \frac{\frac{p_{k,t}}{|\mathrm{x}_{k,t}|}}{\frac{p_{k,t}}{|\mathrm{x}_{k,t}|}} \frac{\int_{\mathrm{x}_{k,t}} p(z_t|x_t) dx_t}{\int_{\mathrm{x}_{k,t}} 1 dx_t} \\
&= |\mathrm{x}_{k,t}|^{-1} \int_{\mathrm{x}_{k,t}} p(z_t|x_t) dx_t
\end{aligned}$$

Это выражение является точным описанием искомой вероятности для модели кусочно-однородного распределения в (4.2). Если сейчас аппроксимировать $p(z_t|x_t)$ через $p(z_t|\hat{x}_{k,t})$ для $x_t \in \mathrm{x}_{k,t}$, получим

(4.7)

$$\begin{aligned}
p(z_t|\mathrm{x}_{k,t}) &\approx |\mathrm{x}_{k,t}|^{-1} \int_{\mathrm{x}_{k,t}} p(z_t|\hat{x}_{k,t}) dx_t \\
&= |\mathrm{x}_{k,t}|^{-1} p(z_t|\hat{x}_{k,t}) \int_{\mathrm{x}_{k,t}} 1 dx_t \\
&= |\mathrm{x}_{k,t}|^{-1} p(z_t|\hat{x}_{k,t}) |\mathrm{x}_{k,t}| \\
&= p(z_t|\hat{x}_{k,t})
\end{aligned}$$

что и даёт, в результате, выражение, указанное выше в (4.4).

Вывод аппроксимации для $p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1})$ в (4.5) несколько сложнее, поскольку области находятся по обеим сторонам искомой кривой. По аналогии с преобразованием выше, получим:

(4.8)

$$\begin{aligned}
p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1}) &= \frac{p(\mathrm{x}_{k,t}, \mathrm{x}_{i,t-1}|u_t)}{p(\mathrm{x}_{i,t-1}|u_t)} \\
&= \frac{\int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t, x_{t-1}|u_t) dx_t, dx_{t-1}}{\int_{\mathrm{x}_{i,t-1}} p(x_{t-1}|u_t) dx_{t-1}} \\
&= \frac{\int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t|u_t, x_{t-1}) p(x_{t-1}|u_t) dx_t, dx_{t-1}}{\int_{\mathrm{x}_{i,t-1}} p(x_{t-1}|u_t) dx_{t-1}}
\end{aligned}$$

Используем марковское свойство, постулирующее независимость между x_{t-1} и u_t, получив $p(x_{t-1}|u_t) = p(x_{t-1})$:

(4.9)

$$
\begin{aligned}
&p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1}) \\
&= \frac{\int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t|u_t, x_{t-1}) p(x_{t-1}) dx_t, dx_{t-1}}{\int_{\mathrm{x}_{i,t-1}} p(x_{t-1}) dx_{t-1}} \\
&= \frac{\int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t|u_t, x_{t-1}) \frac{p_{i,t-1}}{|\mathrm{x}_{i,t-1}|} dx_t, dx_{t-1}}{\int_{\mathrm{x}_{i,t-1}} \frac{p_{i,t-1}}{|\mathrm{x}_{i,t-1}|} dx_{t-1}} \\
&= \frac{\int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t|u_t, x_{t-1}) dx_t, dx_{t-1}}{\int_{\mathrm{x}_{i,t-1}} 1 dx_{t-1}} \\
&= |\mathrm{x}_{i,t-1}|^{-1} \int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(x_t|u_t, x_{t-1}) dx_t, dx_{t-1}
\end{aligned}
$$

Если теперь аппроксимировать $p(x_t|u_t, x_{t-1})$ с помощью $p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1})$ как было сделано раньше, получим следующее приближение. Обратите внимание на необходимость нормализующего члена η, который гарантирует, что аппроксимация будет допустимым вероятностным распределением:

(4.10)

$$
\begin{aligned}
&p(\mathrm{x}_{k,t}|u_t, \mathrm{x}_{i,t-1}) \\
&\approx \eta |\mathrm{x}_{i,t-1}|^{-1} \int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1}) dx_t, dx_{t-1} \\
&= \eta |\mathrm{x}_{i,t-1}|^{-1} p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1}) \int_{\mathrm{x}_{k,t}} \int_{\mathrm{x}_{i,t-1}} 1 dx_t, dx_{t-1} \\
&= \eta |\mathrm{x}_{i,t-1}|^{-1} p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1}) |\mathrm{x}_{k,t}| \, |\mathrm{x}_{i,t-1}| \\
&= \eta |\mathrm{x}_{k,t}| \, p(\hat{x}_{k,t}|u_t, \hat{x}_{i,t-1})
\end{aligned}
$$

Если все области имеют одинаковый размер, (то есть $|\mathrm{x}_{k,t}|$ одинаково для всех k), множитель $|\mathrm{x}_{k,t}|$ можно просто вычеркнуть, учтя с помощью нормализующего члена. Результирующий дискретный байесовский фильтр эквивалентен алгоритму в Таблице 4.1. В приведённой реализации вспомогательные параметры $\bar{p}_k$ не образуют вероятностного распределения, поскольку они не нормализированы (ср. строку 3 с (4.10)). Тем не менее, нормализация выполняется в строке 4, поэтому выходные параметры представляют собой верное вероятностное распределение.

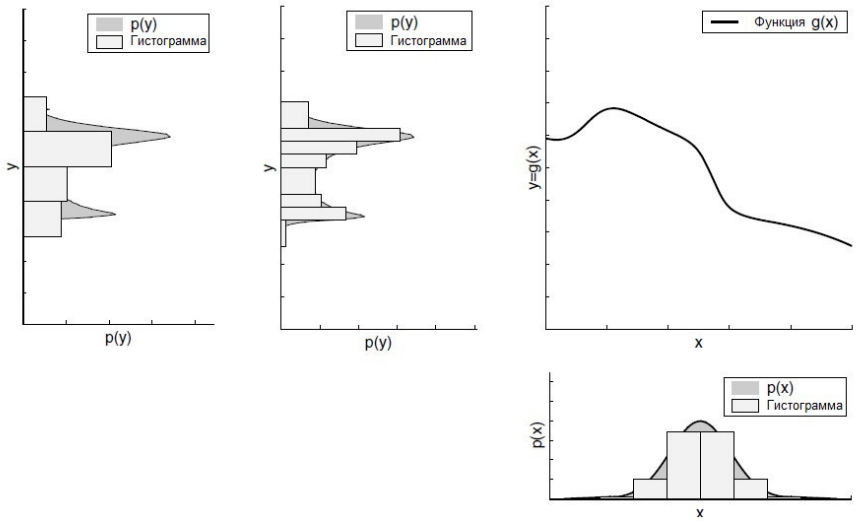


Рис. 4.2 Динамическое и статическое разбиение. На верхней левой схеме показана гистограмма статической аппроксимации случайной переменной Y, построенная из 10 разбиений для покрытия площади Y (6 разбиений имеют вероятность около нуля). На верхней схеме в середине показано представление некой случайной переменной в виде дерева, используя то же разбиение.

4.1.4 Методы разбиения

В робототехнике используется два основных способа разбиения непрерывных пространств состояний: *статические* и *динамические*. Статические методы полагаются на фиксированное, предварительное выбранное разбиение, вне зависимости от формы апостериорного разбиения, которое необходимо аппроксимировать. Динамические методы используют адаптацию разбиения для конкретной формы апостериорного распределения. Статические методы обычно проще применять, но они могут весьма расточительно расходовать вычислительные ресурсы.

ДЕРЕВЬЯ ПЛОТНОСТИ

Показательным примером метода динамического разбиения является семейство *деревьев плотности*. В деревьях плотности пространство состояний разбивается рекурсивно, отражая вероятностную меру апостериорного распределения. Идея такого разбиения состоит в том, чтобы уровень детализации разбиения отражал свойства апостериорного распределения: чем меньше вероятность региона, тем грубее разбиение. На Рис. 4.2 показана разница между представлением в виде статичной сетки и дерева плотностей. В силу более компактного представления дерево плотностей позволяет, при одинаковом разбиении, получить более высокое качество аппроксимации. Кроме того, динамические методы, такие как деревья плотностей, часто позволяют на порядок снизить вычислительную сложность по сравнению со статическими, хотя и требуют дополнительных усилий в реализации.

ВЫБОРОЧНОЕ ОБНОВЛЕНИЕ

Похожий эффект может быть достигнут *выборочным обновлением*. При обновлении апостериорного распределения, выраженного в виде сетки, затрагивается только часть всех ячеек сетки. В обычной реализации такого метода обновляются только те ячейки, апостериорная вероятность которых превосходит определяемый пользователем порог.

Выборочное обновление можно рассматривать как гибридное разбиение пространства состояний на мелкую сетку и единственное большое множество, содержащее все области не затрагиваемые процедурой выборочного обновления. В этом свете его можно воспринимать как метод динамического разбиения, поскольку решение о том, какие ячейки следует обновлять, принимается в реальном времени, на основе формы апостериорного распределения. Методы выборочного обновления позволяют на порядки уменьшить задействованные при обновлении оценок вычислительные мощности и использовать разбиение в виде сеток для пространств с тремя или более измерениями.

В литературе по мобильной робототехнике часто различаются *топологические* и *метрические* представления пространства. Хотя точного формального определения этих терминов нет, топологические отображения часто представляют в виде графов, где узлы соответствуют некоторым значимым местам (или признакам) окружающей среды. В замкнутых пространствах это могут быть пересечения, разветвления, тупики и так далее. Таким образом, разрешение разбиений зависит от структуры окружающей среды. Альтернативным способом является разбиение пространства состояний сетками с регулярной ячейкой. Такое разбиение не зависит от формы и расположения ориентиров окружающей среды. Отображения в виде сеток часто именуют метрическими, хотя, строго говоря, метрическими являются состояния, а не разбиения. В мобильной робототехнике пространственное разрешение сеток обычно выше, чем у топологических представлений. Скажем, в некоторых примерах в Главе 7 используются сетки с размером ячейки 10 см или менее. Дополнительная точность при этом достигается ценой возросшей вычислительной сложности.

1: **Algorithm binary_ Bayes_filter** (l_{t-1}, z_t) :
2: $l_t = l_{t-1} + \log \frac{p(x|z_t)}{1-p(x|z_t)} - \log \frac{p(x)}{1-p(x)}$
3: *return* l_t

Таблица 4.2 Бинарный фильтр Байеса в логарифмическом виде с инвертированной моделью измерений. Здесь l_t - это логарифм шансов апостериорной оценки по бинарной переменной состояния, постоянной во времени.

4.2 Бинарные байесовские фильтры со статическим состоянием.

Некоторые проблемы робототехники лучше всего формулировать в виде задач оценки с бинарным состоянием, постоянным во времени. Для решения этих задач предназначен *бинарный фильтр Байеса*. Такие задачи возникают, если робот оценивает фиксированный бинарный параметр окружающей среды с помощью последовательности измерений датчика. Например, имеется необходимость узнать, открыта дверь или закрыта, в контексте, когда состояние двери в процессе измерения не меняется. Другим примером бинарных фильтров Байеса со статическим состоянием являются *карты сеток занятости*, с которыми мы столкнёмся в Главе 9.

КАРТЫ СЕТКИ ЗАНЯТОСТИ

Когда состояние статично, оценка является лишь функцией измерений:

(4.11)
$$bel_t(x) = p(x|z_{1:t}, u_{1:t}) = p(x|z_{1:t})$$

где состояние выбирается из двух возможных значений, обозначаемых x и $\neg x$. Очевидно, что $bel_t(\neg x) = 1 - bel_t(x)$. Отсутствие индекса времени для состояния x отображает тот факт, что состояние не изменяется со временем.

Обычно, задачи бинарной оценки такого вида возможно разрешить, используя дискретный фильтр Байеса из Таблицы 4.1. Оценка, обычно, задаётся как *логарифм отношения шансов*. Шансы состояния x определяются в виде отношения вероятности данного события и вероятности его противоположности $p(x)$

ЛОГАРИФМ ОТНОШЕНИЯ ШАНСОВ

(4.12)
$$\frac{p(x)}{p(\neg x)} = \frac{p(x)}{1 - p(x)}$$

Логарифм шансов - это логарифм следующего выражения

(4.13)
$$l(x) := \log \frac{p(x)}{1 - p(x)}$$

Логарифм шансов принимает значения от$-\infty$ до ∞. Байесовский фильтр для обновления оценок в представлении логарифма шансов имеет вычислительно лаконичный вид, обходя проблему отсекания вероятностей, близких к 0 или 1.

В Таблице 4.2 приведён основной алгоритм обновления. Он аддитивен, и, практически, любой алгоритм, в котором происходит приращение или уменьшение переменной в качестве реакции на измерения, может быть интерпретирован как байесовский фильтр в форме логарифма шансов. Этот бинарный байесовский фильтр использует *инвертированную модель измерений* $p(x|z_t)$, вместо известной прямой модели $p(z_t|x)$. Инвертированная модель измерений определяет распределение бинарной переменной состояния в виде функции измерения z_t.

ИНВЕРТИРОВАННАЯ МОДЕЛЬ ИЗМЕРЕНИЙ

Инверсные модели часто используются в ситуациях с более сложными, чем бинарное состояние, измерениями. Примером такой ситуации является проблема определения закрыта дверь или нет на основе изображений с камеры. Само состояние очень простое, но пространство всех измерений огромно. Проще найти функцию, которая вычисляет вероятность того, что дверь закрыта, на основании изображения с камеры, чем описывать распределение по всем изображениям с камеры, на которых имеется закрытая дверь. Другими словами, легче применить инвертированную модель датчика, чем прямую.

Как читатель может легко проверить, воспользовавшись определением логарифма шансов в (4.13), оценку $bel_t(x)$ можно восстановить из отношения логарифма шансов l_t с помощью следующего равенства:

(4.14)
$$bel_t(x) = 1 - \frac{1}{1 + \exp\{l_t\}}$$

Чтобы проверить корректность нашего алгоритма бинарного фильтра Байеса, кратко переформулируем основное уравнение фильтра, явно указав нормализующий член:

(4.15)

$$p(x|z_{1:t}) = \frac{p(z_t|x, z_{1:t-1})\, p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})} = \frac{p(z_t|x)\, p(x|z_{1:t-1})}{p(z_t|z_{1:t-1})}$$

Применим теорему Байеса к модели измерений $p(z_t|x)$:

(4.16)

$$p(z_t|x) = \frac{p(x|z_t)\, p(z_t)}{p(x)}$$

и получим

(4.17)

$$p(x|z_{1:t}) = \frac{p(x|z_t)\, p(z_t)\, p(x|z_{1:t-1})}{p(x)\, p(z_t|z_{1:t-1})}$$

По аналогии, есть противоположное событие $\neg x$:

(4.18)

$$p(\neg x|z_{1:t}) = \frac{p(\neg x|z_t)\, p(z_t)\, p(\neg x|z_{1:t-1})}{p(\neg x)\, p(z_t|z_{1:t-1})}$$

Деление (4.17) на (4.18) позволяет сократить различные сложные для вычисления вероятности:

(4.19)

$$\frac{p(x|z_{1:t})}{p(\neg x|z_{1:t})} = \frac{p(x|z_t)}{p(\neg x|z_t)} \frac{p(x|z_{1:t-1})}{p(\neg x|z_{1:t-1})} \frac{p(\neg x)}{p(x)} = \frac{p(x|z_t)}{1 - p(x|z_t)} \frac{p(x|z_{1:t-1})}{1 - p(x|z_{1:t-1})} \frac{1 - p(x)}{p(x)}$$

Определим отношение логарифма шансов оценки $bel_t(x)$ через $l_t(x)$. Оценка в момент времени t задаётся следующим логарифмом (4.19).

(4.20)

$$l_t(x) = \log \frac{p(x|z_t)}{1 - p(x|z_t)} + \log \frac{p(x|z_{1:t-1})}{1 - p(x|z_{1:t-1})} + \log \frac{1 - p(x)}{p(x)} = \log \frac{p(x|z_t)}{1 - p(x|z_t)} - \log \frac{p(x)}{1 - p(x)} + l_{t-1}(x)$$

Здесь $p(x)$ - априорная вероятность состояния x. Как и в (4.20), каждое обновление измерения включает добавление априорной вероятности (в форме логарифма шансов). Априорная вероятность также определяет логарифм шансов начальной оценки перед обработкой каких-либо измерений датчиков:

(4.21)

$$l_0(x) = \log \frac{p(x)}{1 - p(x)}$$

4.3 Многочастичный фильтр

4.3.1 Общий алгоритм

Многочастичный фильтр является альтернативой непараметрической реализации байесовского фильтра. Также как фильтр на основе гистограмм, многочастичный фильтр аппроксимирует апостериорную вероятность с помощью конечного числа параметров. Различие состоит в способе генерации и распределения в пространстве состояний. Ключевая идея многочастичного фильтра состоит в отображении апостериорной вероятности $bel(x_t)$ с помощью выборки случайных значений состояния этого апостериорного распределения. На Рис. 4.3 принцип показан на примере гауссовой функции. Вместо представления распределения в параметрической форме, например, экспоненциальной функции, определяющей плотность нормального распределения, многочастичные фильтры представляют распределения с помощью выборки из них. Такое распределение является приближенным, но оно непараметрическое, а, значит, способно отображать значительно более широкое пространство распределений, чем, к примеру, гауссовы функции. Другим преимуществом отображения на основе выборки является возможность моделирования нелинейных преобразований случайных переменных, что показано на Рис. 4.3.

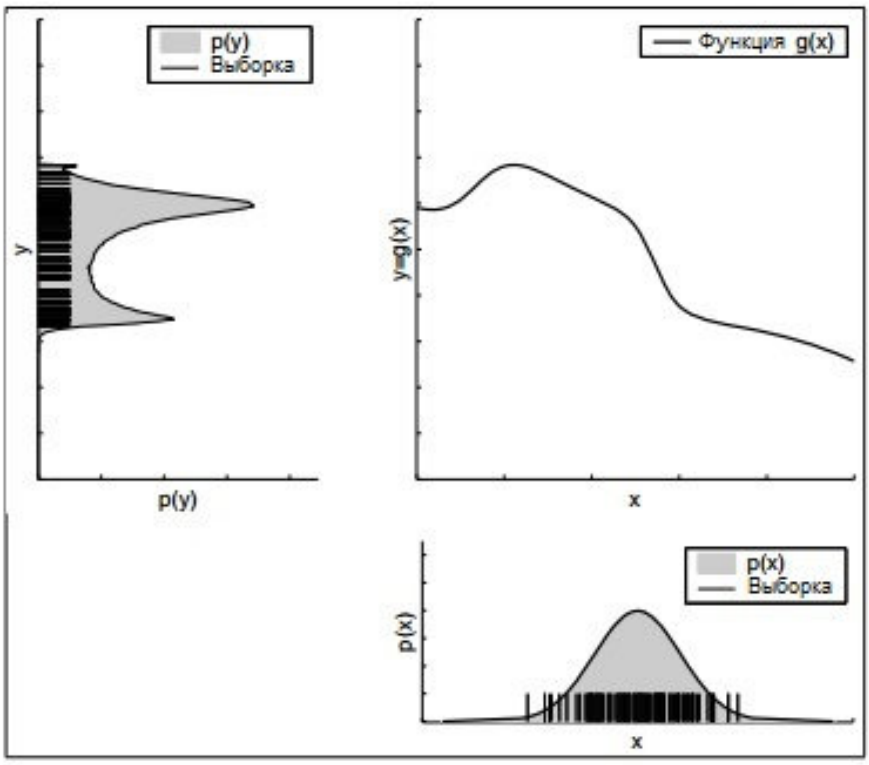


Рис. 4.3 Отображение в виде «частицы», используемое в многочастичных фильтрах. На нижней правой схеме показаны элементы выборки, извлечённые из гауссовской случайной переменной, X. Эти элементы пропускаются через нелинейную функцию, показанную сверху справа. Результирующие элементы распределены согласно случайной переменной Y.

В многочастичных фильтрах выборка из апостериорного распределения называется частицами и обозначается как

(4.22)
$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}$$

Каждая частица $x_t^{[m]}$ (с $1 \leq m \leq M$) является конкретным экземпляром состояния в момент времени t. Другими словами, частица – это гипотеза относительно настоящего значения переменной среды в момент времени t. Здесь M означает число частиц в множестве частиц $\mathcal{X}_t$. На практике, количество частиц M часто достаточно велико, например, $M = 1000$. В некоторых реализациях M является функцией времени или других количественных характеристик, относящихся к оценке $bel(x_t)$.

1: **Algorithm Particle_filter** $(\mathcal{X}_{t-1}, u_t, z_t)$:
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = 0$
3: *для* $m = 1$ *до* M *выполнять*
4: *выборка* $x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} = p(z_t|x_t^{[m]})$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: *endfor*
8: *для* $m = 1$ *до* M *выполнять*
9: *извлечь* i *с вероятностью* $\propto w_t^{[i]}$
10: *добавить* $x_t^{[i]}$ *к* $\mathcal{X}_t$
11: *endfor*
3: *return* $\mathcal{X}_t$

Таблица 4.3 Алгоритм многочастичного фильтра, вариант байесовского фильтра, основанного на выборке по значимости.

Основная идея фильтров частиц лежит в аппроксимации оценки $bel(x_t)$ с помощью набора частиц $\mathcal{X}_t$. В идеальном случае, правдоподобие гипотезы состояния x_t на основе набора частиц $\mathcal{X}_t$, будет пропорционально апостериорному распределению байесовского фильтра $bel(x_t)$:

(4.23)
$$x_t^{[m]} \sim p(x_t|z_{1:t}, u_{1:t})$$

Как следствие (4.23), чем плотнее область пространства состояний заполнена выборкой, тем более вероятно, что настоящее состояние попадёт в эту область. Как будет обсуждаться ниже, свойство (4.23) для стандартного алгоритма многочастичного фильтра сохраняется лишь асимптотически при $M \uparrow \infty$. Для конечного M частицы выбираются из несколько другого распределения, но на практике разница несущественна, пока количество частиц не станет слишком малым (то есть, $M \geq 100$).

Так же, как во всех других алгоритмах байесовских фильтров, обсуждаемых ранее, алгоритм многочастичного фильтра рекурсивно строит оценку $bel(x_t)$ из оценки $bel(x_{t-1})$, взятой на шаг ранее. Поскольку оценки отображены наборами частиц, это означает, что многочастичные фильтры рекурсивно строят набор частиц $\mathcal{X}_t$ из набора $\mathcal{X}_{t-1}$.

Самый общий вариант многочастичного фильтра приведён в Таблице 4.3. На вход этого алгоритма подаётся набор частиц $\mathcal{X}_{t-1}$, вместе с последними данными управляющего воздействия u_t и последними данными измерений z_t. Затем алгоритм строит временный набор частиц $\bar{\mathcal{X}}$, отображающий оценку $\overline{bel}(x_t)$. Это выполняется путём систематической обработки каждой частицы $x_{t-1}^{[m]}$ во входном наборе частиц $\mathcal{X}_{t-1}$. Далее эти частицы трансформируются в набор $\mathcal{X}_t$, который аппроксимирует апостериорное распределение $bel(x_t)$. Это происходит следующим образом:

1. В строке 4 генерируется гипотетическое состояние $x_t^{[m]}$ для момента времени t, используя частицу $x_{t-1}^{[m]}$ и управляющее воздействие u_t. Результирующий образец индексируется по m, чтобы показать, что он был сгенерирован из m-ой частицы в $\mathcal{X}_{t-1}$. На этом шаге выполняется выборка из распределения переходов состояний $p(x_t|u_t, x_{t-1})$. Для выполнения этого

шага необходимо взять выборку из распределения. Набор частиц, полученный после M итераций представляет собой отображение оценки фильтра $\overline{bel}(x_t)$.

ФАКТОР ЗНАЧИМОСТИ

2. В строке 5 для каждой частицы $x_t^{[m]}$ вычисляется так называемый *фактор значимости*, обозначаемый $w_t^{[m]}$. Факторы значимости используются для учёта измерения z_t в наборе частиц. Таким образом, значимость – это вероятность измерения z_t в частице $x_t^{[m]}$, заданной $w_t^{[m]} = p(z_t|x_t^{[m]})$. Если интерпретировать $w_t^{[m]}$ как вес частицы, набор взвешенных частиц представляет (приближённо) апостериорную вероятность байесовского фильтра $bel(x_t)$.

ПЕРЕВЫБОРКА

3. Настоящий «фокус» многочастичного фильтра происходит в строках с 8 по 11 в Таблице 4.3 в процессе, известном как перевыборка или выборка по значимости. Алгоритм извлекает с заменой M частиц из временного набора $\bar{\mathcal{X}}_t$, причём вероятность извлечения частицы из набора задана ее весом значимости. При перевыборке происходит преобразование набора M в другой набор частиц такого же размера. Распределение частиц изменяется с учётом весов значимости: там, где до перевыборки частицы были распределены согласно $\overline{bel}(x_t)$, после неё распределение будет примерно соответствовать апостериорному $bel(x_t) = \eta\, p(z_t|x_t^{[m]})\overline{bel}(x_t)$. В результирующей выборке появляется много дубликатов, поскольку частицы извлекаются с заменой. Более важно, что имеются и частицы не включённые в $\mathcal{X}_t$: это частицы с весом значимости ниже порогового.

Этап перевыборки выполняет важную функцию возврата частиц обратно в апостериорное распределение $bel(x_t)$. Фактически, альтернативная (и обычно худшая) версия многочастичного фильтра может никогда не выполнять перевыборку, а, вместо этого, поддерживать для каждой частицы многократно обновляемый вес значимости, инициализированный единицей:

(4.24)
$$w_t^{[m]} = p(z_t|x_t^{[m]})\, w_{t-1}^{[m]}$$

Такой алгоритм многочастичного фильтра будет в состоянии аппроксимировать апострериорное распределение, но многие частицы окажутся в областях с малой апостериорной вероятностью. В результате, потребуется значительно больше частиц, а их точное количество зависит от апостериорного распределения. Шаг перевыборки – это вероятностная реализация дарвинской идеи *выживания сильнейшего*: он перенаправляет набор частиц в области пространства состояний с высокой апостериорной вероятностью. Таким образом, вычислительные ресурсы алгоритма фильтрации сосредоточены на тех областях пространства состояний, где они нужнее всего.

4.3.2 Перевыборка на основе значимости

Для вывода многочастичного фильтра, будет полезным обсудить этап перевыборки более детально.

На первый взгляд, мы сталкиваемся с задачей вычисления ожидания по функции плотности вероятности f, по имеющейся другой функции плотности вероятности, g. Например, необходимо найти ожидание $x \in A$. Можно выразить эту вероятность в виде ожидания по g. Здесь I – это индикаторная функция, которая принимает значение 1, если аргумент верен, и 0 - в

остальных случаях.

(4.25)

$$\begin{aligned} E_f[I(x \in A)] &= \int f(x) I(x \in A) dx \\ &= \int \underbrace{\frac{f(x)}{g(x)}}_{=:w(x)} g(x)\, I(x \in A) dx \\ &= E_g[w(x)\, I(x \in A)] \end{aligned}$$

Здесь $w(x) = \frac{f(x)}{g(x)}$ это весовой коэффициент, который отражает "несты-ковку"между f и g. Чтобы это равенство было верным, необходимо, чтобы $f(x) > 0 \longrightarrow g(x) > 0$.

Это преобразование выполняется алгоритмом перевыборки по значимости. На Рис. 4.4a показана функция плотности f вероятностного распределения, которая здесь и далее будет называться *целевым распределением.* Как и прежде, мы хотели бы получить значение из f, но, напрямую сделать это невозможно. Вместо этого будем генерировать частицы из плотности g, как показано на Рис. 4.4b.

ЦЕЛЕВОЕ РАСПРЕДЕЛЕНИЕ

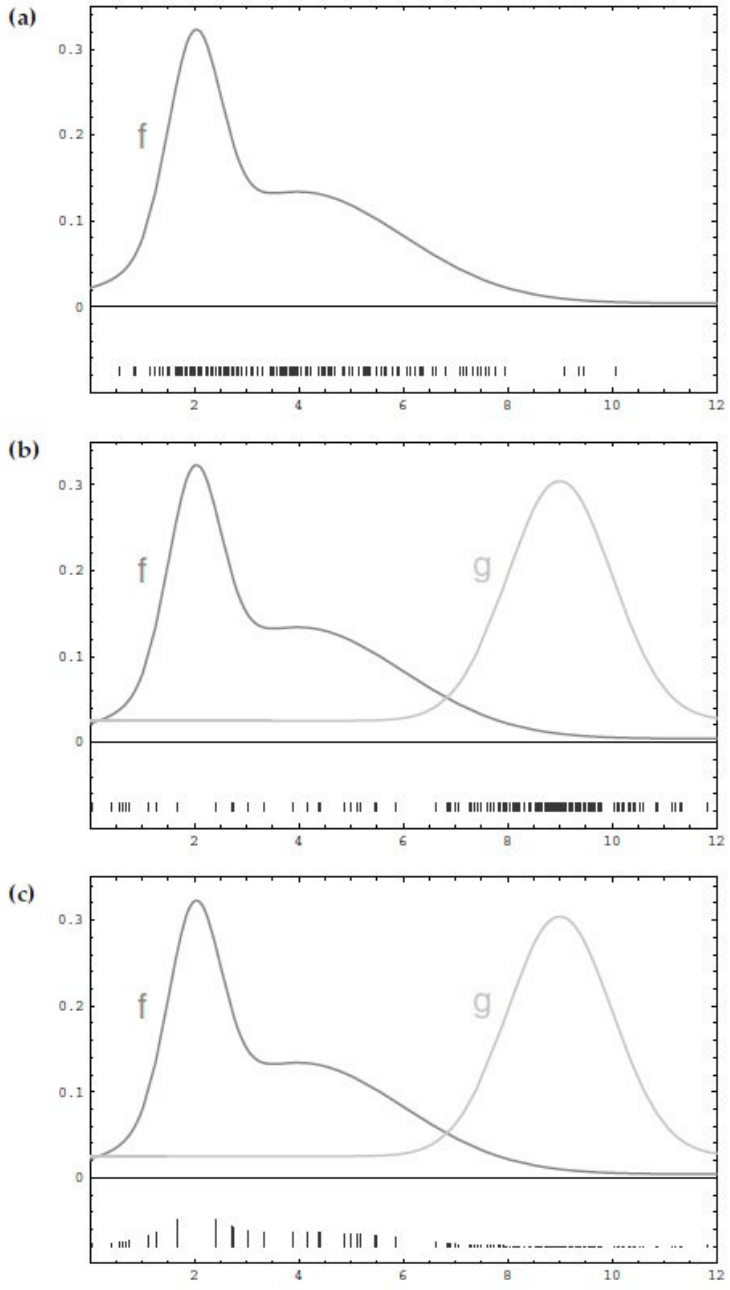


Рис. 4.4 Иллюстрация факторов значимости в многочастичных фильтрах: (a) Необходимо аппроксимировать целевую плотность f. (b) Вместо получения значений из f напрямую, мы можем только генерировать выборку из другой функции, g. Выборка из g показана внизу схемы. (c) Выборка из f получается назначением веса $f(x)/g(x)$ каждому значению x. В многочастичных фильтрах f соответствует оценке $bel(x_t)$, а g – оценке $\overline{bel}(x_t)$.

ПРЕДПОЛАГАЕМОЕ РАСПРЕДЕЛЕНИЕ

Распределение, которое соответствует плотности g, называется *предполагаемое распределение*. Плотность g должна быть такой, чтобы $f(x) > 0$ и $g(x) > 0$, гарантируя ненулевую вероятность сгенерировать частицу при

формировании выборки из g для любого состояния, которое может быть сгенерировано при выборке из f. Таким образом, результирующий набор частиц, показанный внизу на Рис. 4.4b, распределён согласно g, а не f. В частности, для любого интервала $A \subseteq dom(X)$ (или, в более общем виде, любого борелевского множества A) эмпирическое количество частиц, попадающих в A, сходится к интегралу g под A:

(4.26)
$$\frac{1}{M}\sum_{m=1}^{M} I(x^{[m]} \in A) \longrightarrow \int_A g(x)dx$$

Чтобы распределить эту разность между f и g, частицы $x^{[m]}$ взвешиваются следующим коэффициентом

(4.27)
$$w^{[m]} = \frac{f(x^{[m]})}{g(x^{[m]})}$$

Это показано на Рис 4.4c: вертикальные полоски на схеме обозначают величину весов значимости. Веса значимости – это ненормализованная масса вероятности каждой частицы. В частности, получается выражение

(4.28)
$$\left[\sum_{m=1}^{M} w^{[m]}\right]^{-1} \sum_{m=1}^{M} I(x^{[m]} \in A)w^{[m]} \longrightarrow \int_A f(x)dx$$

в котором первый член служит нормализующим для всех весов значимости. Другими словами, несмотря на то, что мы генерируем частицы из плотности g, приблизительно взвешенные частицы сходятся к плотности f. Можно показать, что в общих условиях, эта аппроксимация сходится к искомой $E_f[I(x \in A)]$ для произвольных наборов A. В большинстве случаев, скорость сходимости находится в диапазоне $O(1/\sqrt{M})$, где M – количество элементов. Постоянный множитель зависит от сходства $f(x)$ и $g(x)$.

В многочастичных фильтрах плотность f связана с целевой оценкой $bel(x_t)$. При асимптотически верном допущении, что частицы в $\mathcal{X}_{t-1}$ распределены согласно $bel(x_{t-1})$, плотность g связана с распределением произведения:

(4.29)
$$p(x_t|u_t, x_{t-1})bel(x_{t-1})$$

Это распределение оказывается предлагаемым распределением.

4.3.3 Математический вывод многочастичного фильтра

Для того, чтобы математически вывести многочастичные фильтры, полезно воспринимать частицы в виде элементов последовательностей состояний

(4.30)
$$x_{0:t}^{[m]} = x_0^{[m]}, x_1^{[m]}, ..., x_t^{[m]}$$

Легко модифицировать алгоритм соответствующим образом: просто добавим частицу $x_t^{[m]}$ к последовательности элементов состояния, из которой

было сгенерировано $x_{0:t-1}^{[m]}$. Этот многочастичный фильтр вычисляет апостериорное распределение по всем последовательностям состояния:

(4.31)
$$bel(x_{0:t}) = p(x_{0:t}|u_{1:t}, z_{1:t})$$

вместо оценки $bel(x_t) = p(x_t|u_{1:t}, z_{1:t})$. Очевидно, что пространство по всем последовательностям состояния имеет огромный размер, и пытаться полностью покрыть его частицами, как правило, не слишком продуктивная идея. Однако, здесь это нас не остановит, поскольку определение предназначено только для вывода алгоритма многочастичного фильтра в Таблице 4.3.

Апостериорная оценка $bel(x_{0:t})$ получается аналогично выводу $bel(x_t)$ в подразделе 2.4.3. В частности,

(4.32)
$$\begin{aligned}
&p(x_{0:t}|z_{1:t}, u_{1:t}) \\
&\overset{\text{Bayes}}{=} \eta\, p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t})\, p(x_{0:t}|z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|x_t)\, p(x_{0:t}|z_{1:t-1}, u_{1:t}) \\
&\quad = \eta\, p(z_t|x_t)\, p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t})\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t}) \\
&\overset{\text{Markov}}{=} \eta\, p(z_t|x_t)\, p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1})
\end{aligned}$$

Обратите внимание на отсутствие знаков интеграла в выводе, что стало результатом сохранения в апостериорной оценке всех состояний, а не только последнего, как это было в подразделе 2.4.3.

Теперь вывод можно выполнить, используя одну лишь индукцию. Начальное условие проверим, допустив, что первый набор частиц получается выборкой из апостериорной плотности $p(x_0)$. Пусть набор частиц в момент времени $t-1$ распределён согласно $bel(x_{0:t-1})$. Для m-ой частицы $x_{0:t-1}^{[m]}$ в этом наборе, элемент $x_t^{[m]}$, сгенерированный на Шаге 4 нашего алгоритма, стал элементом предлагаемого распределения:

(4.33)
$$p(x_t|x_{t-1}, u_t)bel(x_{0:t-1}) = p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1})$$

где

(4.34)
$$\begin{aligned}
w_t^{[m]} &= \frac{\text{целевое распределение}}{\text{предлагаемое распределение}} \\
&= \frac{\eta\, p(z_t|x_t)\, p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{1:t-1}, u_{1:t-1})}{p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1})} \\
&= \eta\, p(z_t|x_t)
\end{aligned}$$

Константа η роли не играет, поскольку перевыборка выполняется для вероятностей, *пропорциональных* весам значимости. В результате перевыборки частиц с вероятностями, пропорциональными $w_t^{[m]}$, результирующие частицы заведомо распределены согласно произведению предположительных весов и весов значимости $w_t^{[m]}$:

(4.35)

$$\eta\, w_t^{[m]} p(x_t|x_{t-1}, u_t)\, p(x_{0:t-1}|z_{0:t-1}, u_{0:t-1}) = bel(x_{0:t})$$

(Обратите внимание, что постоянный множитель η отличается от такого, приведённого в (4.34).) Алгоритм в Таблице 4.4 следует простому наблюдению, что, если $x_{0:t}^{[m]}$ распределено согласно $bel(x_{0:t})$, тогда элемент состояния $x_t^{[m]}$, очевидно, распределён согласно $bel(x_t)$.

Как будет доказано ниже, этот вывод верен только при $M \uparrow \infty$, в силу произвольности в определении нормализующих постоянных. Тем не менее, даже при конечном числе M, он отражает идею многочастичного фильтра.

4.3.4 Практические соображения и свойства многочастичных фильтров. Извлечение плотности

Наборы элементов, поддерживаемые в многочастичных фильтрах, отображают дискретные аппроксимации непрерывных оценок. Однако, для многих реализаций требуется доступность непрерывных оценок, то есть не только состояний, представленных в наборе частиц, но произвольной точки пространства состояний. Задача извлечения непрерывной плотности из таких элементов называется оценкой плотности. Проиллюстрируем в неформальном виде некоторые подходы к оценке плотности.

На Рис. 4.5 показаны различные способы извлечения плотности из частиц. На самой левой схеме показаны частицы и плотность преобразованной гауссовой функции из нашего стандартного примера с Рис. 4.3. Простой и очень эффективный метод извлечения плотности из таких частиц - это вычисление *гауссова приближения*, как показано пунктиром на Рис. 4.5(b). В этом случае, гауссова функция, полученная на основе частиц, практически идентична гауссову приближению настоящей плотности (сплошная линия).

Очевидно, гауссово приближение отражает лишь основные свойства плотности, и применимо только для одномодальной плотности. Мультимодальные распределения элементов требуют более сложных методов, таких как

АЛГОРИТМ К-СРЕДНИХ

кластеризация по методу k-средних, которая аппроксимирует плотность, используя смеси гауссианов. Альтернативный подход показан на Рис. 4.5(c). Здесь дискретная *гистограмма* наложена на пространство состояний и вероятность каждого интервала вычисляется суммированием весов частиц, которые попадают в диапазон. По аналогии с фильтром на основе гистограмм, значительным недостатком такого подхода является факт экспоненциальной зависимости сложности пространства от числа измерений. С другой стороны, с помощью гистограмм возможно очень эффективно, в вычислительном смысле, отобразить многомерные распределения, а плотность любого состояния можно извлечь за время, не зависящее от числа частиц.

Пространственная сложность представлений в виде гистограмм может быть существенно уменьшена с помощью генерации из частиц деревьев

ДЕРЕВО ПЛОТНОСТЕЙ

плотностей, как обсуждалось в Главе 4.1.4. Однако, деревья плотностей имеют логарифмическое к глубине дерева количество вычислительных операций в операциях поиска при извлечении плотности произвольной точки в

ОЦЕНКА ПЛОТНОСТИ ЯДРА

пространстве состояний. *Оценка плотности ядра* - это ещё один способ преобразования набора частиц в непрерывную плотность. В этом методе каждая частица рассматривается в виде центра так называемого ядра, а общая плотность задана смесью плотностей ядер. На Рис. 4.5(d) показана смесь

плотностей, получившаяся в результате размещения в каждой частице гауссового ядра. Преимуществом оценок плотности ядер является гладкость и алгоритмическая простота. При этом, сложность вычисления плотности в произвольной точке линейно зависит от количества частиц или ядер.

Какой же из этих способов извлечения плотности следует использовать на практике? Это зависит от решаемой задачи. Например, во многих областях робототехники вычислительная мощность сильно ограничена, а информации о математическом ожидании частиц достаточно для управления роботом. В других областях, таких как активная локализация, требуется более полная информация о степени неопределённости в пространстве состояний. В таких ситуациях лучшим выбором будут гистограммы или смеси гауссовых функций. Комбинирование данных, собранных несколькими роботами, иногда требует перемножения плотностей из разных наборов элементов. Деревья плотностей или оценки плотности ядра хорошо приспособлены ля решения этой задачи.

Дисперсия выборки

Значительный вклад в ошибку многочастичного фильтра вносится дисперсией случайной выборки. Когда конечное число элементов извлекается из плотности вероятности, статистические величины, извлечённые из этих элементов, несколько отличаются от показателей оригинальной плотности. Например, если изобразить элементы случайной гауссовой переменной, математическое ожидание и дисперсия элементов будут отличаться от математического ожидания и дисперсии оригинальной случайной переменной

ДИСПЕРСИЯ ЭЛЕМЕНТА

Разброс из-за случайности выборки называется дисперсией алгоритма отбора.

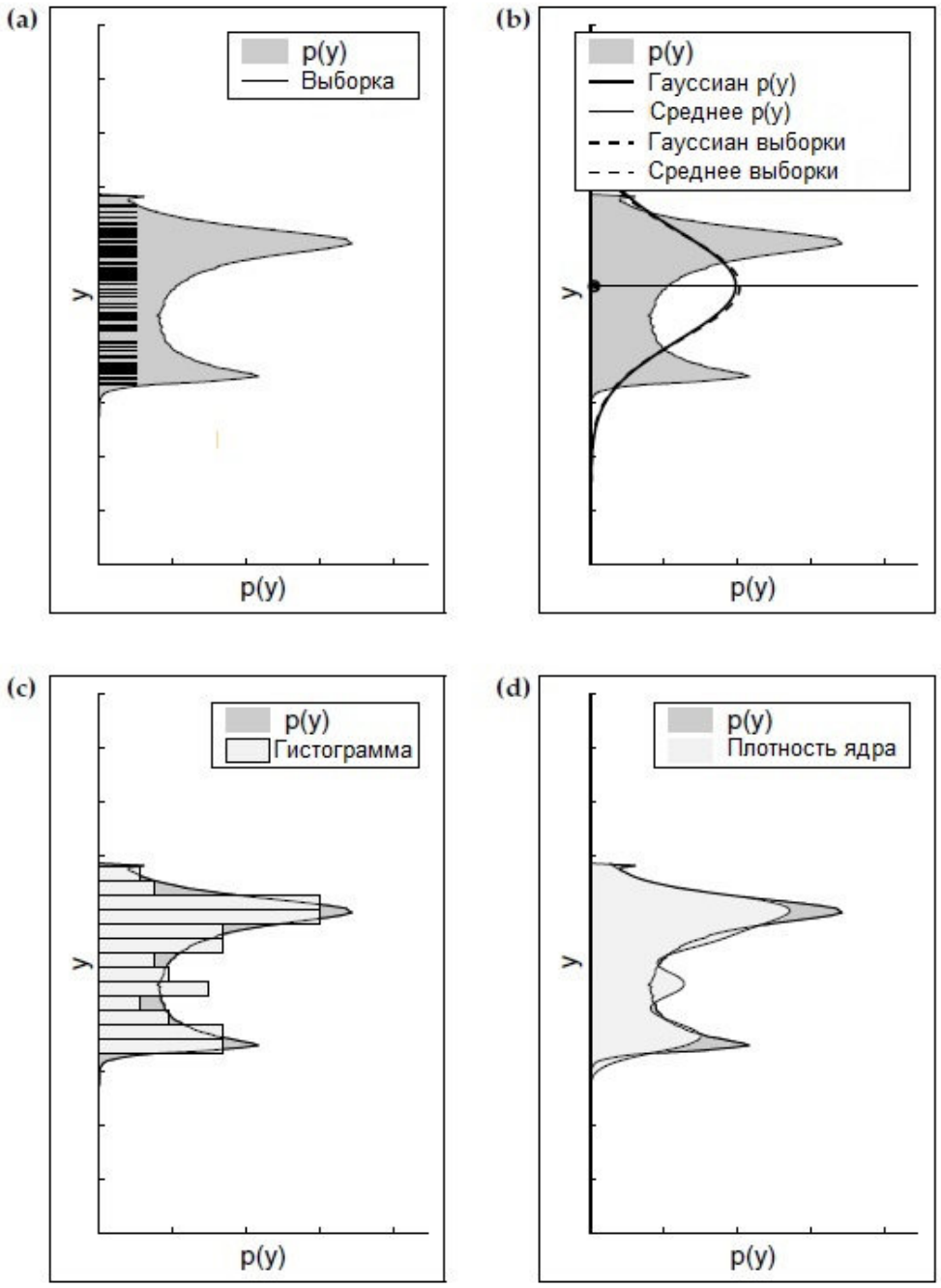


Рис. 4.5 Различные способы извлечения плотностей из частиц. (a) Плотность и аппроксимация набора элементов, (b) гауссова аппроксимация (математическое ожидание и дисперсия), (c) аппроксимация с помощью гистограммы, (d) оценка плотности ядра. Выбор метода аппроксимации сильно зависит от конкретной области применения и вычислительных ресурсов.

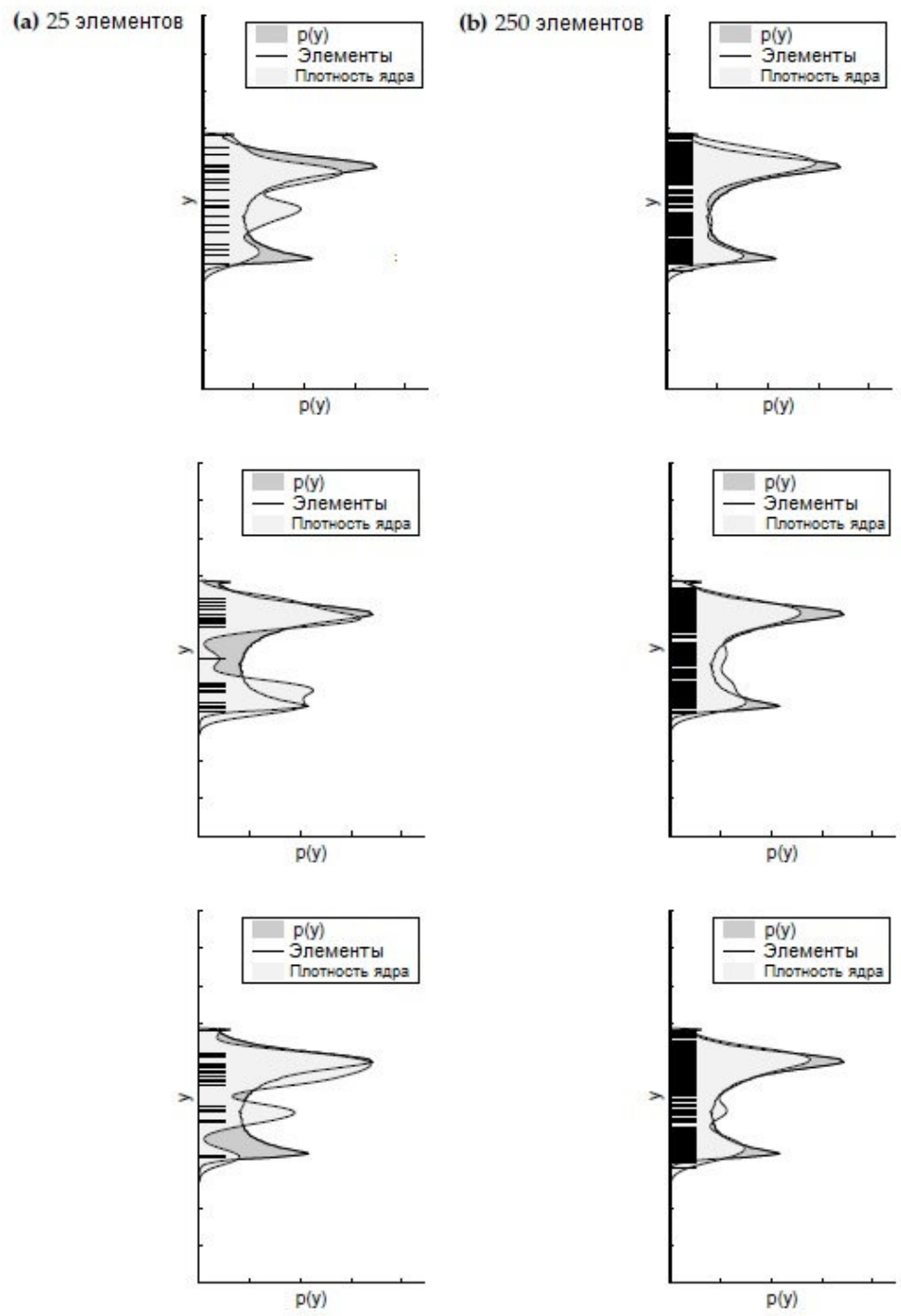


Рис. 4.6 Дисперсия случайной выборки. Элементы извлекаются из гауссовой функции и пропускаются через нелинейную функцию. Показаны элементы и оценки ядер после 25 (левый столбец) и 250 (правый столбец)повторных выборок. В каждой строке показан один эксперимент.

Представим двух одинаковых роботов с идентичными, гауссовыми оценками, выполняющих одинаковые, не подверженные зашумлению, действия.

Очевидно, после выполнения действия оба роботы будут иметь одну и ту же оценку состояния. Для имитации этой ситуации будем многократно извлекать значения из гауссовой плотности и пропускать из через нелинейное преобразование. На схемах на Рис. 4.6 показаны результирующие значения и их оценки плотности ядра, а также реальная оценка (серая область). На схемах в верхнем ряду показаны результаты извлечения из гауссовой функции 25 элементов. Вопреки ожидаемому результату, некоторые из оценок плотности ядра существенно отличаются от реальной плотности и наблюдается большая дисперсия между разными плотностями ядер. К счастью, дисперсия выборки уменьшается с ростом количества элементов выборки. В нижнем ряду на Рис. 4.6 показаны типичные результаты, полученные для 250 элементов. Очевидно, что увеличение числа элементов ведёт к более точной аппроксимации с меньшей дисперсией. На практике, если выбрано достаточное количество элементов, данных наблюдений, которые выполняет робот, достаточно, чтобы держать оценку на основе выборки «достаточно близко» к настоящей.

Перевыборка

Дисперсия выборки усиливается при повторной перевыборке. Для понимания проблемы, будет полезно разобрать граничный случай, в котором состояние робота не меняется. Иногда точно известно, что $x_t = x_{t-1}$. Хорошим примером будет задача локализации неподвижного мобильного робота. Давайте ещё более усугубим ситуацию и допустим, что у робота отсутствуют датчики и никакой информации о своём состоянии он получить не может. Очевидно, такой робот никогда ничего не сможет выяснить об окружении, поскольку оценка в момент времени t будет идентична первоначальной оценке в произвольный момент времени t.

К сожалению, многочастичный фильтр в чистом виде даст другой результат. Изначально, набор частиц будет генерироваться из предыдущего, а частицы – распределены в пространстве состояний. Однако, на шаге перевыборки (строка 8 алгоритма) элемент состояния $x^{[m]}$ в некоторых случаях восстановлен не будет. Поскольку переход состояния детерминирован, новых состояний при проходе по выборке (строка 4) добавлено не будет. С течением времени в силу случайной природы шага перевыборки все больше и больше частиц будет удаляться, при этом новые частицы создаваться не будут. Результат довольно обескураживающий: с вероятностью M будут сохраняться несколько идентичных копий одной и той же частицы. Разнообразие исчезнет в силу повторяющейся перевыборки. Для внешнего наблюдателя может показаться, что робот точно определил состояние окружающей среды – очевидное противоречие в силу того, что у него нет датчиков.

Этот пример демонстрирует ещё одно ограничение многочастичных фильтров, которое имеет серьёзные практические последствия. Дело в том, что процесс перевыборки вызывает потерю разнообразия в популяции частиц, что ведёт к ошибке аппроксимации. Несмотря на то, что дисперсия самого набора частиц уменьшается, дисперсия набора частиц, как оценивающей функции реального состояния, увеличивается. Управление этой дисперсией, или ошибкой, многочастичного фильтра, очень важно в любой практической реализации. Существует две основные стратегии *уменьшения дисперсии*. Во первых, можно уменьшить частоту перевыборки. Если известно, что состояние статическое ($x_t = x_{t-1}$), перевыборка не должна происходить никогда. На примере локализации мобильного робота: когда робот останав-

УМЕНЬШЕНИЕ ДИСПЕРСИИ

ливается, перевыборку необходимо приостанавливать (а, вдобавок, неплохо будет приостановить и интеграцию измерений). Даже если состояние меняется, часто полезно уменьшить частоту перевыборки. Несколько измерений всегда можно интегрировать множественным обновлением фактора значимости, как было показано ранее. А именно – держать веса значимости в памяти и обновлять их следующим образом:

(4.36)

$$w_t^{[m]} = \begin{cases} 1 & \text{если имеет место перевыборка} \\ p(z_t|x_t^{[m]})\, w_{t-1}^{[m]} & \text{если перевыборка отсутствует} \end{cases}$$

Решение о необходимости перевыборки достаточно сложное и требует практического опыта: слишком частая перевыборка увеличивает риск потери разнообразия. Если делать перевыборку слишком редко, множество значений в областях с низкой вероятностью может быть утрачено. Стандартным способом определения того, нужно ли выполнять перевыборку, служит измерение дисперсии весов значимости. Дисперсия весов значимости связана с эффективностью отображения на основе выборки. Если все веса идентичны, то дисперсия равна нулю и перевыборка не нужна. Если, с другой стороны, веса сконцентрированы в небольшом количестве элементов, то дисперсия весов будет велика и необходимо выполнить перевыборку.

ВЫБОРКА ПО НИЗКОЙ ДИСПЕРСИИ

Вторая стратегия уменьшения ошибки выборки известна как *выборка по низкой дисперсии.* В Таблице 4.4 приводится реализация такого алгоритма выборки. Основной идеей является то, что, при перевыборке вместо независимого выбора элементов (как в случае базового многочастичного фильтра в Таблице 4.3), выбор включает последовательный стохастический процесс.

```
1:   Algorithm Low_variance_sampler (X_t, W_t) :
2:     X̄_t = 0
3:     r = rand(0; M^{-1})
4:     c = w_t^[1]
5:     i = 1
6:     для m = 1 до M выполнять
7:       U = r + (m − 1) · M^{-1}
8:       пока U > c
9:         i = i + 1
10:        c = c + w_t^[i]
11:      endwhile
12:      добавить x_t^[i] к X̄_t
13:    endfor
14:    return X̄_t
```

Таблица 4.4 Перевыборка с низкой дисперсией для многочастичного фильтра. Этот алгоритм использует одно случайное число для выборки из набора частиц $\mathcal{X}$ с ассоциированными весами $\mathcal{W}$, хотя вероятность частицы быть выбранной все ещё пропорционально ее весу. Алгоритм выборки довольно эффективен и выборка M частиц требует количества операций $O(M)$.

Вместо выбора M случайных чисел и отбора частиц, соответствующих

этим числам, в этом алгоритме вычисляется единственное случайное число, согласно которому и отбираются элементы, хотя и с вероятностью, пропорциональной весу элемента. Это достигается выбором случайного числа r в интервале $[0; M^{-1}]$, где M - число элементов для выборки в момент времени t. Затем алгоритм в Таблице 4.4 циклически прибавляет фиксированное значение M_{-1} к r, отбирая частицу, соответствующую номеру. Любое число U в диапазоне $[0; 1]$ будет указывать только на одну частицу, например, i, для которой

(4.37)
$$i = \underset{j}{\operatorname{argmin}} \sum_{m=1}^{j} w_t^{[m]} \geq U$$

Цикл с условием в Таблице 4.4 служит двум целям. В нем вычисляется сумма с правой стороны уравнения и дополнительно проверяется, является ли i индексом первой частицы, чтобы соответствующая сумма весов превышала U. Отбор выполняется в строке 12. Этот же процесс показан на Рис. 4.7.

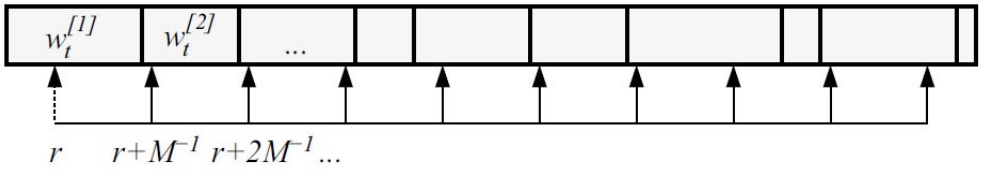


Рис. 4.7 Принцип процедуры перевыборки с низкой дисперсией. Выбирается случайное число r, а затем отбираются, только те частицы, которые соответствуют $u = r + (m-1) \cdot M^{-1}$, где $m = 1, ..., M$.

У алгоритма отбора с низкой дисперсией три преимущества. Во-первых, он значительно более равномерно покрывает пространство элементов, нежели независимый алгоритм отбора. Это очевидно следует из факта, что зависимый алгоритм отбора циклически перебирает все частицы, а не выбирает их случайным образом независимо друг от друга. Во-вторых, если все элементы имеют одинаковые факторы значимости, результирующий набор элементов $\bar{\mathcal{X}}_t$ эквивалентен $\mathcal{X}_t$, поэтому элементы не будут утеряны при перевыборке без интеграции наблюдения в $\mathcal{X}_t$. В-третьих, алгоритм отбора с низкой дисперсией имеет сложность $O(M)$. Достижение столь же малой сложности для независимой выборки затруднено, поскольку очевидные реализации требуют операций поиска со сложностью $O(\log M)$ для извлечения каждой частицы после выбора случайного числа, что ведёт к сложности всего процесса перевыборки $O(M \log M)$. При использовании многочастичных фильтров очень важно время использования, и, зачастую, эффективная реализация процесса перевыборки будет означать огромную разницу в практической производительности. Из-за этих причин реализации многочастичных фильтров в робототехнике чаще всего и основаны на обсуждаемых механизмах. В общем же, количество литературы о способах эффективного отбора выборки огромно. Другим популярным подходом является *стратифицированная выборка*, в которой частицы сгруппированы в подмножества. Отбор элементов из таких наборов выполняется в два этапа. Сначала на основе общего веса частиц в подмножестве определяется количество извлекаемых элементов. На втором этапе отдельные элементы случайным образом

СТРАТИФИЦИРОВАННАЯ ВЫБОРКА

извлекаются из каждого множества, используя, например, перевыборку с низкой дисперсией. Такой подход имеет более низкую дисперсию выборки и обычно работает лучше для случаев, когда робот отслеживает несколько отдельных, четко различимых гипотез с помощью одного многочастичного фильтра.

Смещение выборки

Из-за того, что используется только конечное число частиц, возникает систематическое *смещение* в апостериорной оценке. Возьмём, для примера, граничный случай с единственной частицей $M = 1$. В этом случае, цикл в строках с 3 по 7 в Таблице 4.3 будет выполнен только один раз, и $\bar{\mathcal{X}}_t$ будет содержать единственную частицу, извлечённую из модели движения. Ключевым моментом является то, что перевыборка (строки с 8 по 11 в Таблице 4.3) теперь будет *детерминированным образом* принимать этот элемент, вне зависимости от его фактора значимости $w_t^{[m]}$. Поскольку вероятность измерения $p(z_t|x_t^{[m]})$ не играет никакой роли в результате этого обновления, то же справедливо и для z_t. Поэтому, при $M = 1$ многочастичный фильтр генерирует частицы для вероятности $p(x_t|u_{1:t})$ вместо целевой апостериорной $p(x_t|u_{1:t}, z_{1:t})$. Он просто игнорирует все измерения. Как же это произошло?

Виновником является нормализация, выполняемая на шаге перевыборки. При выборке пропорционально весам значимости (строка 9 алгоритма), $w_t^{[m]}$ становится собственным нормализатором при $M = 1$:

(4.38)
$$p(\text{извлекаемых } x_t^{[m]} \text{ в строке 9}) = \frac{w_t^{[m]}}{w_t^{[m]}} = 1$$

В общем случае, проблема состоит в том, что ненормализованные значения $w_t[m]$ извлекаются из M-мерного пространства, но, после нормализации, располагаются в пространстве мерности $M - 1$. Так происходит потому, что после нормализации m-ый вес можно восстановить из $M - 1$ других весов вычитанием их из единицы. К счастью, для больших значений M, эффект потери размерности или степеней свободы, становится все менее выраженным.

Дефицит частиц

Даже при большом числе частиц может сложиться ситуация, когда в окрестностях верного состояния частиц нет. Эта проблема известна как *проблема дефицита частиц*. Она возникает, в основном, когда количество частиц слишком мало, чтобы покрыть все соответствующие области с высокой вероятностью. Однако, можно заметить, что так может случиться с любым многочастичным фильтром вне зависимости от размера набора частиц M.

Дефицит частиц происходит в результате дисперсии случайной выборки. Неудачная серия случайных чисел может не содержать ни одной частицы около настоящего состояния. На каждом шаге выборки вероятность, что это случится, больше нуля (хотя обычно экспоненциально меньше M). Таким образом, при достаточно продолжительном использовании многочастичного фильтра можно получить оценку со сколь угодно большой ошибкой.

На практике проблемы подобного рода возникают только при малом M, относительно пространства всех состояний с высокой вероятностью. Популярным решением проблемы дефицита частиц является добавление неболь-

шого числа случайно сгенерированных частиц в набор после каждой перевыборки, вне зависимости от текущих команд движения и измерения. Такая методология может уменьшить (но не исправить) проблему дефицита, но лишь ценой неверной оценки апостериорной вероятности. Преимущество метода добавления случайных элементов заключается в его простоте: необходима лишь минимальная модификация программного обеспечения. В качестве общего правила, добавление случайных элементов следует считать последним средством, которое следует использовать, только если остальные методы исправления проблемы дефицита частиц не работают. Альтернативные методы решения проблемы дефицита частиц будут обсуждаться в Главе 8 в контексте локализации робота.

Мы пришли к выводу, что качество отображения на основе выборки возрастает с количеством элементов. Важным вопросом является нахождение количества элементов, которое следует использовать в конкретной задаче оценки. К сожалению, универсального ответа на этот вопрос нет, и, зачастую, требуемое количество элементов может определить только пользователь. Обычно, количество элементов сильно зависит от размерности пространства состояний и степени неопределённости аппроксимируемых фильтром распределений. Так, для однородных распределений требуется намного больше элементов, чем для случаев, сконцентрированных в небольшой области пространства состояний. Более детальное обсуждение размера выборки будет дано в будущих главах в контексте локализации робота и построения карт.

4.4 Вывод

В этой главе были приведены два непараметрических фильтра Байеса, фильтр на основе гистограмм и многочастичный фильтр. Непараметрические фильтры аппроксимируют апостериорную вероятность с помощью конечного числа значений. При одинаковых допущениях модели системы и формы апостериорного распределения оба фильтра имеют свойство равномерной сходимости ошибки аппроксимации к нулю при увеличении до бесконечности количества значений, используемых для аппроксимации.

• Фильтры на основе гистограмм разбивают пространство состояний на конечное множество выпуклых областей и отображает суммарную апостериорную вероятность каждой области одним числовым значением.

• В робототехнике используется множество способов разбиения. В частности, степень разбиения может зависеть (или, наоборот, не зависеть) от структуры окружающей среды. Когда такая зависимость есть, результирующие алгоритмы часто называют «топологическими».

• Методы разбиения можно разделить на статические и динамические. Статические разбиения выполняются единообразно, вне зависимости от формы оценки. Динамические разбиения основаны на особенностях оценок робота, когда разбиением пространства состояний пытаются увеличить пространственное разрешение апостериорной вероятности. Динамические разбиения обычно дают лучшие результаты, но более трудно реализуемы.

• Альтернативным непараметрическим методом является многочастичный фильтр. Многочастичные фильтры представляют апостериорные рас-

пределения в виде случайной выборки состояний. Такие элементы называются частицами. Многочастичные фильтры очень легко реализовать, и они являются самыми надёжными из всех алгоритмов байесовских фильтров, представленных в книге.

• Существует множество способов уменьшения ошибки многочастотных фильтров. Среди наиболее популярных - методы уменьшения дисперсии оценки, возникающей в силу случайной природы алгоритма, и регулирования числа частиц в соответствии с сложностью апостериорного распределения.

Алгоритмы фильтров, обсуждаемые в этой и предыдущей главе, образуют основу большинства вероятностных алгоритмов робототехники, которые встретятся в книге. Представленный здесь материал отражает множество самых популярных сегодня алгоритмов и представлений вероятностной робототехники.

4.5 Библиографические сведения

Вест и Гаррисон (West and Harrison, 1997) провели детальное исследование нескольких методов, обсуждаемых в этой и предыдущих главах. Гистограммы использовались в статистике многие десятилетия. Стуржес (Sturges, 1926) представил одну из ранних теорем выбора разрешения для аппроксимации с помощью гистограмм, а более современные исследования были проведены Фридманом и Диаконисом (Freedman and Diaconis, 1981). Современный анализ можно найти в работах Скотта (Scott, 1992). После того, как пространство состояний размечается дискретной гистограммой, результирующая задача временного логического вывода становится экземпляром дискретной скрытой марковской модели, ставшей популярной благодаря исследованиям Рабинера и Янга (Rabiner and Juang, 1986). Два современных текста были написаны Макдональдом и Цуккини (MacDonald and Zucchini, 1997) и Элиотом с соавторами (Elliott et al., 1995).

Многочастичные фильтры можно проследить до работ Метрополиса и Улама (Metropolis and Ulam, 1949), которые и изобрели методы Монте-Карло. Более современный вводный текст принадлежит Рубинштейну (Rubinstein, 1981). Метод перевыборки по значимости, входящий в состав многочастичного фильтра, прослеживается до двух основополагающих работ Рубина (Rubin, 1988) и Смита и Гелфанда (Smith and Gelfand, 1992). Стратифицированная выборка была открыта Нейманом (Neyman, 1934). В последние несколько лет многочастичные фильтры интенсивно исследовались в контексте байесовской статистики (Doucet 1998; Kitagawa 1996; Liu and Chen 1998; Pitt and Shephard 1999). В области искусственного интеллекта многочастичные фильтры были заново введены под именем "выживание сильнейшего"(Kanazawa et al. 1995). В компьютерном зрении алгоритм, называемый конденсацией, изобретённый Исардом и Блейком (Isard and Blake, 1998) используется в задачах отслеживания объектов. Хороший современный текст по многочастичным фильтрам принадлежит Дюсе с соавторами (Doucet et al., 2001).

4.6 Упражнения

1. В этом упражнении будет необходимо реализовать фильтр на основе гистограмм для линейной динамической модели, изученный в предыдущей главе.

(a) Реализовать фильтр на основе гистограмм для динамической системы, описанной в Упражнении 1 предыдущей главы (см стр. 84). Использовать фильтр для прогнозирования последовательности апостериорных распределений для $t = 1, 2, ..., 5$. Для каждого значения t, изобразить в виде графика апостериорное распределение x и $\dot{x}$, где x горизонтальная, а $\dot{x}$ - вертикальная ось.

(b) Добавить этап обновления измерения к фильтр на основе гистограмм, как описано в Упражнении 2 предыдущей главы (стр. 84). Допустим, в момент времени $t = 5$ наблюдается измерение $z = 5$. Определить и изобразить апостериорное распределение до и после такта обновления фильтра на основе гистограмм.

2. Теперь необходимо реализовать фильтр на основе гистограмм для нелинейного случая, использовав материал Упражнения 4 предыдущей главы (стр. 85). Напомним, была изучена нелинейная система с детерминированным переходом состояний, определённая тремя переменными состояния.

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x + \cos\theta \\ y + \sin\theta \\ \theta \end{pmatrix}$$

Начальная оценка состояния задана следующим образом:

$$\mu = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} \quad \text{и} \quad \Sigma = \begin{pmatrix} 0,01 & 0 & 0 \\ 0 & 0,01 & 0 \\ 0 & 0 & 10000 \end{pmatrix}$$

(a) Предложить подходящую начальную оценку для фильтра на основе гистограмм, отражающую информацию об априорном гауссовом распределении.

(b) Реализовать фильтр на основе гистограмм и выполнить этап прогнозирования. Сравнить результирующее апостериорное распределение с аналогичным, полученным с помощью EKF и интуитивного анализа. Что можно сказать о разрешении по координатам x, y и ориентации θ с помощью фильтра на основе гистограмм?

(c) Интегрировать в оценку данные измерения. Как и прежде, измерение представляет собой зашумлённую проекцию x-координаты робота с ковариацией $Q = 0,01$. Реализовать этот шаг, вычислить, изобразить на графике, и сравнить с результатами EKF и интуитивного анализа.

Примечание: При изображении результатов фильтра на основе гистограмм на графике, показать несколько графиков вероятности, по одному для каждого дискретного диапазона в пространстве всех значений θ.

3. В тексте уже обсуждался эффект использования единственной части-

цы. Каков будет результат использования $M = 2$ частиц при фильтрации? Можно ли дать пример смещённого апостериорного распределения? Если да, то каково смещение?

4. Выполнить Упражнение 1, используя вместо гистограмм многочастичные фильтры, изобразить на графике и объяснить результат.

5. Выполнить Упражнение 2, используя вместо гистограмм многочастичные фильтры, изобразить на графике и объяснить результат. Исследовать эффект влияния различного количества частиц на результат.

5 Движение робота

5.1 Введение

Эта и следующая главы посвящены двум оставшимся компонентам практической реализации описанных алгоритмов фильтров: моделям движения и измерения. Эта глава посвящена *моделям движения*, включающим в себя переходную вероятность состояния $p(x_t|u_t, x_{t-1})$, которая играет важную роль для такта экстраполяции байесовского фильтра. В главе приводятся подробные примеры вероятностных моделей движения в том виде, в котором они используются на практике в робототехнике. В следующей главе будут описаны вероятностные модели измерений датчиков $p(z_t|x_t)$, которые являются основой для такта обновления измерения. Представленный материал составляет важную часть практической реализации алгоритмов, описанных в следующих главах.

Тема кинематики робота, которая освещается в данной главе, подвергалась всестороннему исследованию последние несколько десятилетий, но результаты практически полностью используют допущение детерминированного определения состояния. Вероятностная робототехника обобщает уравнения кинематики, учитывая возникающие в результате управляющего воздействия неточности, вызванные шумами управления или не смоделированными внешними эффектами. Следуя тематике книги, наша точка зрения будет вероятностной, а результат управляющего воздействия - описан апостериорной вероятностью. Таким образом, результирующие модели будут соответствовать освещаемым в предыдущих главах методам вероятностной оценки состояния.

Наше изложение полностью посвящено кинематике мобильных роботов, действующих в плоском окружении и, таким образом, гораздо более специфично по сравнению с современным принятым толкованием кинематики. Ни кинематика манипулятора, ни динамика робота рассматриваться не будут. Однако, ограничения в выборе материала ни в коей мере не следует интерпретировать как ограниченность вероятностного подхода лишь простыми кинематическими моделями мобильных роботов. Напротив, описываются самые современные текущие разработки, поскольку вероятностные методы с наибольшим успехом применяются в мобильной робототехнике. При этом в их основе лежат относительно простые модели, которые и будут представлены в этой главе. Использование более сложных вероятностных моделей (например, вероятностных моделей динамики робота) практически не освещено в литературе, но такие обобщения не являются неправдоподобными. Как будет показано в этой главе, детерминированные модели приводов робота можно «привести к вероятностному виду» путём прибавления

переменных шумов, характеризующих различные типы неопределённости, присутствующие в приводах робота.

Теоретически, может показаться, что целью хорошей вероятностной модели будет точное моделирование конкретных типов неопределённости, присутствующих в восприятии и действиях робота. На практике же точный вид модели часто менее важен, чем некоторое понимание воздействия неопределённости. Фактически, во многих моделях, на практике доказавших свою эффективность, неопределённость сильно переоценена, но именно поэтому результирующие алгоритмы более надёжны при нарушениях марковских свойств (подраздел 2.4.4), таких как не смоделированное состояние и эффект алгоритмических приближений. На такие особенности будет указано далее, при обсуждении практических реализаций вероятностных алгоритмов в робототехнике.

5.2 Предварительные сведения

5.2.1 Кинематическая конфигурация

КОНФИГУРАЦИЯ

Кинематика - это способ расчёта эффекта управляющих воздействий на конфигурацию робота. *Конфигурация* жёсткого мобильного робота обычно описывается шестью переменными, тремя пространственными координатами и тремя углами Эйлера (прецессия, нутация и вращение) во внешней системе координат. Материал, представленный в книге, по большей части, ограничен мобильными роботами, действующими на плоскости, кинематическое состояние которых описывается тремя переменными, и называется в тексте термином «положением робота».

ПОЛОЖЕНИЕ

Положение мобильного робота, действующего на плоскости, показано на Рис. 5.1. Оно состоит из двухмерных плоских координат во внешней системе отсчёта, а также угла поворота.

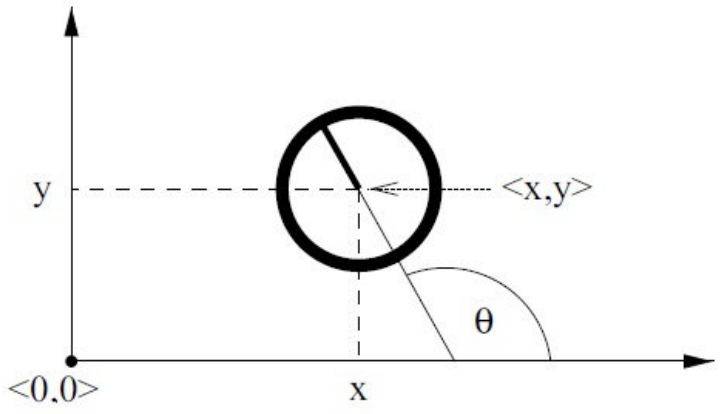


Рис. 5.1 Положение робота в глобальной системе координат.

Обозначив координаты через x и y (не следует путать с переменной состояния x_t), а угол - θ, положение можно описать следующим вектором:

(5.1)

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

КУРС

Угол ориентации робота часто называют *курс*, или *угол направления*. Как показано на Рис. 5.1, определим, что робот с курсом $\theta = 0$ направлен вдоль оси x. Робот, ориентированный с $\theta = 0,5\pi$ указывает в направлении оси y.

МЕСТОПОЛОЖЕНИЕ

Положение без учёта ориентации называется *местоположением*. Концепция местоположения играет важную роль в следующей главе, посвящённой измерениям, как способу описания окружающей среды робота. Для простоты, в книге местоположения обычно будут описываться двухмерными векторами, соответствующими координатам объекта на плоскости:

(5.2)

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

Положение робота и набор местоположений объектов окружающей среды может образовывать кинематическое состояние x_t системы «робот- окружающая среда».

5.2.2 Вероятностная кинематика

Вероятностная кинематическая модель, или модель движения, играет в мобильной робототехнике роль модели перехода состояний. Эта модель представляет собой уже знакомую условную плотность

(5.3)

$$p(x_t|u_t, x_{t-1})$$

Здесь x_t и x_{t-1} – положения робота (а не просто значения его x-координаты), а u_t – управляющая команда на выполнение движения. Этой моделью описывается апостериорное распределение кинематических состояний, которые принимает робот при выполнении команды на движение u_t в момент времени x_{t-1}. На практике, u_t часто представлено в виде одометрии робота. Из содержательных соображений будем, все-таки, считать u_t управляющим воздействием.

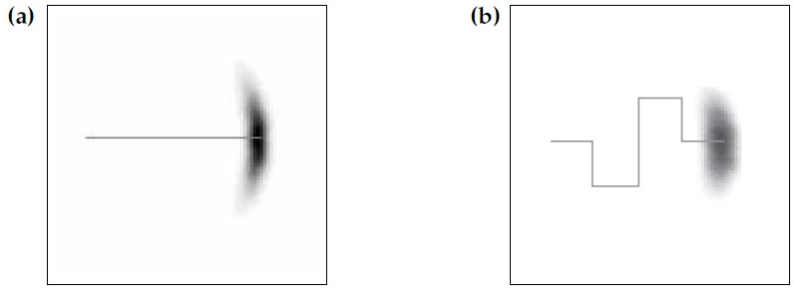


Рис. 5.2 Модель движения: Апостериорные распределения для положения робота при выполнении команды на движение, показанной сплошной линией. Чем темнее местоположение, тем больше вероятность. График был спроектирован в 2-D. В оригинале плотность трехмерна, поскольку учитывается ещё и угол направления θ робота.

На Рис. 5.2 показаны два примера, иллюстрирующие кинематическую модель жёсткого мобильного робота, действующего на плоскости. В обоих случаях начальное положение робота x_{t-1}. Распределение $p(x_t|u_t, x_{t-1})$ показано закрашенной областью: чем темнее положение, тем оно вероятнее. На рисунке вероятность апостериорного положения спроектировано на плоскость, а измерение, в котором учитывается ориентация робота, отсутствует. На Рис. 5.2a робот двигается вперёд на некоторое расстояние. Как показано на рисунке, в процессе движения могут возникнуть ошибки оценки поступательного и вращательного движения. На Рис. 5.2b показано результирующее распределение после выполнения более сложной команды на движение, что привело к большему разбросу неопределённости.

В этой главе будут детально разобраны две отдельные вероятностные модели движения $p(x_t|u_t, x_{t-1})$ для мобильных роботов, действующих на плоскости. Обе модели взаимно по типу обрабатываемой информации о движении дополняют друг друга. В первой подразумевается, что данные о движении u_t обозначают скорости вращения двигателей после подачи на них команды. Многие коммерческие мобильные роботы (с дифференциальным или синхронным приводом) независимо управляются по поступательным и вращательным скоростям, или же лучше всего моделируются таким образом. Вторая модель подразумевает наличие доступа к данным одометрии. Большинство коммерческих шасси предоставляют данные одометрии, в виде кинематической информацию, например, пройденного пути и суммарного угла поворота. В результате, вероятностная модель для интеграции таких данных несколько отличается от модели скоростей.

На практике модели одометрии часто более точны по сравнению с моделями скоростей просто в силу неспособности большинства коммерческих роботов выполнять команды на достижение определённой скорости с той же точностью, которую можно достичь, измеряя количество оборотов колес. Однако, данные одометрии доступны только после выполнения команды на движение, и, поэтому непригодны для планирования движения. Алгоритмы планирования, такие как методы предотвращения столкновений, должны прогнозировать эффект движения. Поэтому, одометрия обычно используется для оценки, а модели скорости – для вероятностного планирования движения.

5.3 Модель движения на основе скорости

Модель движения на основе скорости подразумевает, что робот управляется установкой двух скоростей, вращательной и поступательной. Множество коммерческих роботов имеют интерфейсы управления, позволяющие программистам определяет скорости. Приводные механизмы, обычно управляемые таким образом, включают дифференциальные приводы, приводы Аккермана и синхронные приводы. Модель непригодна для приводов с неголономным набором ограничений, например, колесом Илона или шагающих роботов.

Определим поступательную скорость в момент времени t через v_t, а *вращательную скорость* - как ω_t. Отсюда,

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} \tag{5.4}$$

Произвольно условимся, что положительные вращательные скорости ω_t означают вращение против часовой стрелки, то есть повороты налево. Положительные поступательные скорости v_t соответствуют движению вперёд.

5.3.1 Вычисление в закрытом виде

Возможный алгоритм вычисления вероятности $p(x_t|u_t, x_{t-1})$ показан в Таблице 5.1. На вход подаются начальное положение $x_{t-1} = (x\, y\, \theta)^T$, управляющее воздействие $u_t = (v\, \omega)^T$ и предполагаемое последующее положение $x_t = (x'\, y'\, \theta')^T$. На выходе возвращается вероятность $p(x_t|u_t, x_{t-1})$ нахождения системы в состоянии x_t после выполнения команды u_t из начального состояния x_{t-1}. Будем считать, что управление осуществляется в течение фиксированного промежутка времени Δt. Серией параметров от α_1 до α_6 определяются конкретные виды ошибок движения. Алгоритм в Таблице 5.1 вычисляет управляющее воздействие идеального робота. Значения каждой переменной при вычислении будет описано в разделе математического вывода. Параметры заданы через $\hat{v}$ и $\hat{\omega}$.

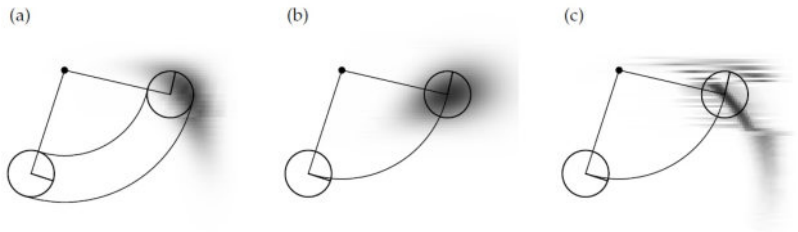


Рис. 5.3 Модель движения на основе скорости при различных установках параметра зашумления.

Функцией $\text{prob}(x, b^2)$ моделируется ошибка движения с помощью вычисления вероятности параметра x случайной переменной с центром в нуле и дисперсией b^2. Две возможные реализации показаны в Таблице 5.2, где переменные ошибок представлены нормальным и треугольным распределением, соответственно.

На Рис. 5.3 показаны примеры модели движения на основе скорости в виде проекций на плоскость. Во всех трёх случаях для робота устанавливаются одинаковые поступательные и вращательные скорости. На Рис. 5.3a показано результирующее распределение с умеренным значением параметров ошибки с α_1 до α_6. Распределение, показанное на Рис. 5.3b, получено с меньшей угловой ошибкой (параметры α_3 и α_4), но большей ошибкой поступательного движения (параметры α_1 и α_2). На Рис. 5.3c показано распределение с большими значениями ошибок.

5.3.2 Алгоритм выборки

Для многочастичных фильтров (см. подраздел 4.3) допустимо выполнять выборку из модели движения $p(x_t|u_t, x_{t-1})$ вместо вычисления апостериорного распределения произвольных x_t, u_t и x_{t-1}. Взятие выборки условной плотности отличается от её вычисления набором начальных данных: при выполнении выборки даны u_t и x_{t-1} и необходимо сгенерировать случайное x_t из выборки согласно модели движения $p(x_t|u_t, x_{t-1})$. При вычислении вероятности, кроме указанных величин, дано ещё и x_t, сгенерированное с помощью других значений математического ожидания, а требуется вычислить вероятность x_t при $p(x_t|u_t, x_{t-1})$.

Алгоритм **sample_motion_model_velocity** в Таблице 5.3 генерирует случайные элементы выборки $p(x_t|u_t, x_{t-1})$ для фиксированного воздействия управления u_t и положения x_{t-1}. Он принимает на вход x_{t-1} и u_t, а генерирует случайное положение x_t в соответствии с распределением $p(x_t|u_t, x_{t-1})$. В строках с 2 по 4 выполняется «искажение» параметров команды управления шумами, извлечёнными из параметров ошибки кинематической модели движения. Значения шумов затем используются для генерации нового положения, в строках с 5 по 7.

1:	**Algorithm motion_model_velocity** (x_t, u_t, x_{t-1}) :
2:	$\mu = \frac{1}{2}\frac{(x-x')\cos\theta+(y-y')\sin\theta}{(y-y')\cos\theta-(x-x')\sin\theta}$
3:	$x^* = \frac{x+x'}{2} + \mu(y-y')$
4:	$y^* = \frac{y+y'}{2} + \mu(x'-x)$
5:	$r^* = \sqrt{(x-x^*)^2+(y-y^*)^2}$
6:	$\Delta\theta = \text{atan2}(y'-y^*, x'-x^*) - \text{atan2}(y-y^*, x-x^*)$
7:	$\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$
8:	$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$
9:	$\hat{\gamma} = \frac{\theta'-\theta}{\Delta t} - \hat{\omega}$
10:	*return* $\text{prob}(v-\hat{v}, \alpha_1 v^2 + \alpha_2\omega^2) \cdot \text{prob}(\omega-\hat{\omega}, \alpha_3 v^2 + \alpha_4\omega^2)$ $\cdot\, \text{prob}(\hat{\gamma}, \alpha_5 v^2 + \alpha_6\omega^2)$

Таблица 5.1 Алгоритм вычисления $p(x_t|u_t, x_{t-1})$ на основании данных скорости. Подразумевается, что x_{t-1} выражено вектором $(x\, y\, \theta)^T$, x_t выражено через $(x'\, y'\, \theta')^T$, а u_t - вектором скоростей $(v\, \omega)^T$. Функция $\text{prob}(a, b^2)$ вычисляет вероятность аргумента в распределении с центром в нуле и дисперсией b^2. Она может быть реализована на практике, используя любой из алгоритмов в Таблице 5.2.

1: **Algorithm prob_normal_distribution** (a, b^2) :

2: *return* $\frac{1}{\sqrt{2\pi\, b^2}}\, \exp\left\{-\frac{1}{2}\frac{a^2}{b^2}\right\}$

3: **Algorithm prob_triangular_distribution** (a, b^2) :

4: *return* $\max\left\{0,\ \frac{1}{\sqrt{6}\, b} - \frac{|a|}{6\, b^2}\right\}$

Таблица 5.2 Алгоритмы вычисления плотностей нормального и треугольного распределений с нулевым математическим ожиданием и дисперсией b^2.

1: **Algorithm sample_motion_model_velocity** (u_t, x_{t-1}) :

2: $\hat{v} = v + \text{sample}(\alpha_1 v^2 + \alpha_2 \omega^2)$
3: $\hat{\omega} = \omega + \text{sample}(\alpha_3 v^2 + \alpha_4 \omega^2)$
4: $\hat{\gamma} = \text{sample}(\alpha_5 v^2 + \alpha_6 \omega^2)$
5: $x' = x - \frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t)$
6: $y' = y + \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t)$
7: $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$
8: *return* $x_t = (x', y', \theta')^T$

Таблица 5.3 Алгоритм выполнения выборки по положению $x_t = (x'\, y'\, \theta')^T$ из положения $x_{t-1} = (x\, y\, \theta)^T$ и управляющего воздействия $u_t = (v\, \omega)^T$. Обратите внимание на искажение значения конечной ориентации добавлением случайного слагаемого $\hat{\gamma}$. Переменные с α_1 до α_6 - параметры шумов движения. Функция sample(b^2) генерирует случайный элемент выборки из распределения с нулевым математическим ожиданием и дисперсией b^2. Она может быть реализована, например, с помощью алгоритма в Таблице 5.4.

1: **Algorithm sample_normal_distribution** (b^2) :

2: *return* $\frac{1}{2}\sum_{i=1}^{12} \text{rand}(-b, b)$

3: **Algorithm sample_triangular_distribution** (b^2) :

4: *return* $\frac{\sqrt{6}}{2}[\text{rand}(-b, b) + \text{rand}(-b, b)]$

Table 5.4 Алгоритм для извлечения элемента выборки из (приблизительно) нормальных и треугольных распределений с нулевым математическим ожиданием и дисперсией b^2. См. Винклер (Winkler, 1995: стр. 293). Функция $\text{rand}(x, y)$ считается генератором псевдослучайных чисел с однородным распределением в диапазоне $[x, y]$.

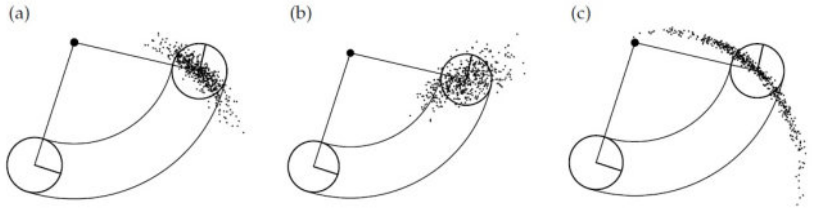


Рис. 5.4 Выборка из модели движения на основе скоростей, используя тот же набор параметров, что и на Рис. 5.3. На каждой схеме показано 500 элементов.

Таким образом, процедура выборки реализует упрощённую модель движения физического робота, которая учитывает шумы управления на шаге экстраполяции. На Рис. 5.4 показан итог работы последовательности выполнения выборки. На нем изображено 500 элементов выборки, сгенерированные алгоритмом **sample_motion_model_velocity**. Читатель может самостоятельно сравнить этот рисунок с плотностью, изображённой на Рис. 5.3.

Заметим, что во многих случаях легче получить выборку x_t, чем высчитывать плотность вероятности для данного x_t. Это происходит потому, что выборка требует только прямого прохода по физической модели движения. Для вычисления вероятности предполагаемого положения необходимо восстановить параметры ошибки, что требует вычисления обратной физической модели. Факт того, что многочастичные фильтры основаны на выборке, делает их особенно привлекательными для практической реализации.

5.3.3 Математический вывод модели движения на основе скоростей

Давайте выведем алгоритмы **motion_model_velocity** и **sample_motion_model_velocity**. Как обычно, читатель, не заинтересованный в математических подробностях, может пропустить этот раздел при первом чтении

и продолжить чтение с подраздела 5.4 (страница 130). Вывод начинается с генеративной модели движения робота, с последующим определением формул для выборки и вычисления $p(x_t|u_t, x_{t-1})$ для произвольных x_t, u_t, и x_{t-1}.

Идеальное движение

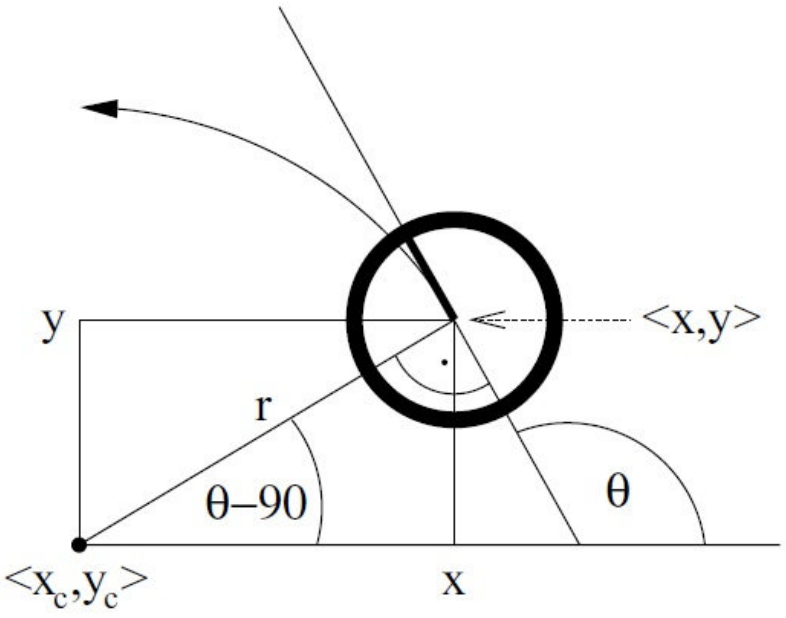


Рис. 5.5 Движение, выполняемое идеальным роботом передвигающимся с постоянными скоростями v и ω из начальной точки $(x\, y\, \theta)^T$ без зашумления.

Перед тем, как перейти к вероятностному описанию, начнём с кинематики идеального, не подверженного зашумлению робота. Пусть $u_t = (v\, \omega)^T$ определяет управляющее воздействие в момент времени t. Если сохраняется фиксированное значение обеих скоростей на всем интервале времени $(t-1, t)$, робот движется по окружности радиуса

(5.5)
$$r = \left|\frac{v}{\omega}\right|$$

Это следует из общей взаимосвязи между поступательной и вращательной скоростями v и ω для произвольного объекта, движущегося по круговой траектории с радиусом r:

(5.6)
$$v = \omega \cdot r$$

Выражение (5.5) описывает случай, когда робот не поворачивает совсем (то есть $\omega = 0$), и движется по прямой линии. Прямая соответствует кругу бесконечного радиуса, поэтому заметим, что r может быть бесконечным.

Пусть $x_{t-1} = (x, y, \theta)^T$ будет начальным положением робота, и, допустим, скорость остаётся постоянной по $(v\,\omega)^T$ в течение некоторого промежутка времени Δt. Как легко показать, центр окружности находится в месте с координатами

$$x_c = x - \frac{v}{\omega}\sin\theta \tag{5.7}$$

$$y_c = y + \frac{v}{\omega}\cos\theta \tag{5.8}$$

Переменные $(xc\,yc)^T$ обозначают эти координаты. После движения в течение промежутка времени Δt идеальный робот будет находиться в $x_t = (x', y', \theta')^T$ с

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ y_c - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix} \tag{5.9}$$

$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

Вывод этого выражения следует из простых тригонометрических зависимостей: после Δt единиц времени, идеальный робот передвинется по кругу на $v \cdot \Delta t$, что приведёт к повороту угла направления на $\omega \cdot \Delta t$. В то же время, его координаты x и y заданы пересечением окружности с центром $(x_c\,y_c)^T$ и луча, имеющего начало в $(x_c\,y_c)^T$ и перпендикулярном $\omega \cdot \Delta t$. Во втором преобразовании в результирующие уравнения движения просто подставляется выражение (5.8).

Конечно, реальные роботы неспособны мгновенно изменять скорость и сохранять её постоянной в течение всего промежутка времени, поэтому для вычисления кинематики обычно используются малые значения Δt, в течение которых скорость считается постоянной. Приближенное конечное положение можно получить, связывая соответствующие круговые траектории с помощью математических уравнений, которые только что были приведены.

Реальное движение

В реальности движение робота подвержено зашумлению, поэтому реальные скорости отличаются от указанных (или измеренных, если робот оснащён датчиками измерения скорости). Смоделируем это различие случайной переменной с нулевым математическим ожиданием и конечной дисперсией. Более формально, примем, что скорости заданы в виде

$$\begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v \\ \omega \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2} \\ \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2} \end{pmatrix} \tag{5.10}$$

Здесь ε_{b^2} - переменная с нулевым математическим ожиданием и дисперсией b^2. Таким образом, реальная скорость равна сумме заданной скорости и некоторой малой, аддитивной ошибки (шумов). В нашей модели стандартное отклонение ошибки пропорционально заданной скорости. Параметры от α_1 до α_4 (где $\alpha_i \geq 0$ для $i = 1, ..., 4$) являются специфическими для робота

параметрами ошибки. С их помощью моделируется точность робота, и чем она меньше, тем больше значения параметров.

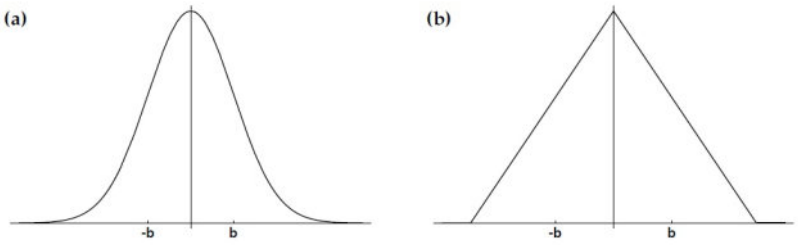


Рис. 5.6 Функции плотности вероятности с дисперсией b^2: (a) Нормальное распределение,(b)треугольное распределение.

Общепринятым способом задания ошибки ε_{b^2} является использование нормального или треугольного распределений.

НОРМАЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Нормальное распределение с нулевым средним и дисперсией b^2 задано функцией плотности

(5.11)
$$\varepsilon_{b^2}(a) = \frac{1}{\sqrt{2\pi b^2}} e^{-\frac{1}{2}\frac{a^2}{b^2}}$$

На Рис. 5.6a показана функция плотности нормального распределения b^2. Нормальные распределения обычно используются для моделирования шумов в непрерывных стохастических процессах. Её носитель, то есть набор точек a для которых $p(a) > 0$, равен $\Re$.

ТРЕУГОЛЬНОЕ РАСПРЕДЕЛЕНИЕ

Плотность *треугольного распределения* с нулевым математическим ожиданием и дисперсией b^2 задана в виде

(5.12)
$$\varepsilon_{b^2}(a) = \max\left\{0, \frac{1}{\sqrt{6}\,b} - \frac{|a|}{6\,b^2}\right\}$$

и не равна нулю только в промежутке $(-\sqrt{6\,b}; \sqrt{6\,b})$. Как видно из Рис. 5.6b, график плотности по форме напоминает равнобедренный треугольник, отсюда и название.

Наилучшей моделью реального положения $x_t = (x'\, y'\, \theta')$ после выполнения команды на движение $u_t = (v\, \omega)^T$ в момент времени $x_{t-1} = (x\, y\, \theta)^T$, таким образом, будет

(5.13)
$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}}\sin\theta + \frac{\hat{v}}{\hat{\omega}}\sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}}\cos\theta - \frac{\hat{v}}{\hat{\omega}}\cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t \end{pmatrix}$$

Это равенство получено заменой указанной скорости $u_t = (v\, \omega)^T$ зашумленным движением $(\hat{v}\, \hat{\omega})^T$ в выражении (5.9). Однако, модель все ещё не слишком реалистична в силу описанных ниже причин.

Конечная ориентация

Два приведённых выше равенства точно описывают конечное местоположение робота при условии, что робот движется по строго круговой траектории радиуса $r = \frac{\hat{v}}{\hat{\omega}}$. Хотя значения радиуса этого сегмента окружности и пройденное расстояние подвержены шумам управления, сам факт движения по круговой траектории остаётся в силе. Допущение о том, что траектория является круговой, ведёт к важному преобразованию, которое приведет к вырожденному случаю. В частности, носитель функции плотности $p(x_t|u_t, x_{t-1})$ является двухмерным, при этом находится внутри трехмерного пространства состояний. Факт нахождения всех апостериорных положений в двухмерном множестве трехмерного пространства положений является прямым следствием того факта, что было использовано только две переменных зашумления, одна для v и одна - для ω. К сожалению, это вырождение имеет важные последствия при использовании байесовских фильтров для оценки состояния.

Конечно, в реальности любое осмысленное апостериорное распределение не будет вырожденным и положения будут расположены в трехмерном пространстве с дисперсией по x, y и θ. Чтобы соответствующим образом обобщить рассматриваемую модель, допустим, что робот выполняет поворот $\hat{\gamma}$, когда достигает конечного положения. Поэтому, вместо вычисления θ' в соответствии с (5.13), смоделируем конечную ориентацию

(5.14)
$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$$

с

(5.15)
$$\hat{\gamma} = \varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2}$$

Здесь α_5 и α_6 – дополнительные параметры робота, определяющие дисперсию добавочного зашумления вращения. Результирующая модель будет выглядеть следующим образом:

(5.16)
$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}}\sin\theta + \frac{\hat{v}}{\hat{\omega}}\sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}}\cos\theta - \frac{\hat{v}}{\hat{\omega}}\cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{pmatrix}$$

Вычисление $p(x_t|u_t, x_{t-1})$

Алгоритм **motion_model_velocity** в Таблице 5.1 реализует вычисление $p(x_t|u_t, x_{t-1})$ для заданных значений $x_{t-1} = (x\, y\, \theta)^T$, $u_t = (v\, \omega)^T$, и $x_t = (x\, y\, \theta)^T$. Вывод алгоритма несколько неочевиден, поскольку в нем эффективно реализуется обратная модель движения. В частности, алгоритм **motion_model _velocity** на основании положений x_{t-1} и x_t определяет параметры движения $\hat{u}_t = (\hat{v}\, \hat{\omega})^T$ а также соответствующий конечный поворот $\hat{\gamma}$. Вывод, очевидно, доказывает необходимость конечного поворота, поскольку без него вероятность движения почти для всех значений x_{t-1}, u_t и x_t будет равна нулю.

Давайте вычислим вероятность $p(x_t|u_t, x_{t-1})$ действия управления $u_t = (v\, \omega)^T$, которое перенесёт робота из положения $x_{t-1} = (x\, y\, \theta)^T$ в положение $x_t = (x'\, y'\, \theta')^T$ в течение времени Δt. Чтобы это сделать, определим управляющее воздействие $\hat{u} = (\hat{v}\, \hat{\omega})^T$, требуемое для переноса робота из положе-

ния x_{t-1} в положение $(x'\, y')$ безотносительно конечного угла направления. Далее, станет возможным определить конечный поворот $\hat{\gamma}$, который необходимо выполнить роботу для перехода к углу направления θ'. На основании этих вычислений легко найти вероятность $p(x_t|u_t, x_{t-1})$.

Читатель может вспомнить, что в нашей модели подразумевалось перемещение робота с фиксированной скоростью по круговой траектории в течение времени Δt. Для робота, который переместился из $x_{t-1} = (x\, y\, \theta)^T$ к $x_t = (x'\, y')^T$, центр окружности будет определён как $(x^*\, y^*)^T$ и задан в виде

$$\text{(5.17)} \quad \begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -\lambda \sin\theta \\ \lambda \cos\theta \end{pmatrix} = \begin{pmatrix} \frac{x+x'}{2} + \mu(y - y') \\ \frac{y+y'}{2} + \mu(x' - x) \end{pmatrix}$$

для некоторых неизвестных $\lambda, \mu \in \Re$. Первое равенство стало результатом ортогональности направления на центр окружности первоначальному углу направления робота. Второе является очевидным заключением о том, что центр окружности лежит на прямой, которая пролегает посередине между $(x\, y)^T$ и $(x'\, y')^T$ и ортогональна соединяющей их прямой.

Обычно, выражение (5.17) имеет единственное решение, за исключением вырожденного случая, когда $\omega = 0$, а центр окружности находится в бесконечности. Как читатель может захотеть проверить,

$$\text{(5.18)} \quad \mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$

поэтому

$$\text{(5.19)} \quad \begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} \frac{x+x'}{2} + \frac{1}{2} \frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}(y - y') \\ \frac{y+y'}{2} + \frac{1}{2} \frac{(x-x')\cos\theta + (y-y')\sin\theta}{(y-y')\cos\theta - (x-x')\sin\theta}(x' - x) \end{pmatrix}$$

радиус окружности задан евклидовым расстоянием

$$\text{(5.20)} \quad r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2} = \sqrt{(x' - x^*)^2 + (y' - y^*)^2}$$

Теперь стало возможным вычислить изменение угла направления

$$\text{(5.21)} \quad \Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*)$$

Здесь atan2 - это широко известное расширение арктангенса y/x, расширенного до $\Re^2$ (в большинстве языков программирования есть реализация этой функции):

$$\text{(5.22)} \quad \text{atan2}(y, x) = \begin{cases} \text{atan}(y/x) & \text{если } x > 0 \\ \text{sign}(y)(\pi - \text{atan}(|y/x|)) & \text{если } x < 0 \\ 0 & \text{если } x = y = 0 \\ \text{sign}(y)\pi/2 & \text{если } x = 0, y \neq 0 \end{cases}$$

Поскольку было принято, что робот движется по круговой траектории, расстояние между x_t и x_{t-1} по дуге равно

(5.23)
$$\Delta\text{dist} = r^* \cdot \Delta\theta$$

Из Δdist и $\Delta\theta$ легко вычислить скорости $\hat{v}$ и $\hat{\omega}$:

(5.24)
$$\hat{u}_t = \begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \Delta t^{-1} \begin{pmatrix} \Delta\text{dist} \\ \Delta\theta \end{pmatrix}$$

Скорость вращения $\hat{\gamma}$, необходимая для достижения конечного угла направления θ' робота в координатах $(x'\, y')$ за время Δt можно определить из (5.14) как:

(5.25)
$$\hat{\gamma} = \Delta t^{-1}(\theta' - \theta) - \hat{\omega}$$

Ошибка движения - это отклонение от указанных скоростей по $\hat{u}_t$ и $\hat{\gamma}$ $u_t = (v\, \omega)^T$ и $\gamma = 0$, как было определено выражениями (5.24) и (5.25).

(5.26)
$$v_{\text{err}} = v - \hat{v}$$

(5.27)
$$\omega_{\text{err}} = \omega - \hat{\omega}$$

(5.28)
$$\gamma_{\text{err}} = \hat{\gamma}$$

В нашей модели ошибок, определённой выражениями (5.10) и (5.15), эти ошибки имеют следующие вероятности:

(5.29)
$$\varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2}(v_{\text{err}})$$

(5.30)
$$\varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2}(\omega_{\text{err}})$$

(5.31)
$$\varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2}(\gamma_{\text{err}})$$

где ε_{b^2}, как и прежде, означает переменную ошибки с нулевым математическим ожиданием и дисперсией b^2. Поскольку подразумевается независимость между разными источниками ошибки, искомая вероятность $p(x_t|u_t, x_{t-1})$ является произведением отдельных ошибок:

(5.32)
$$p(x_t|u_t, x_{t-1}) = \varepsilon_{\alpha_1 v^2 + \alpha_2 \omega^2}(v_{\text{err}}) \cdot \varepsilon_{\alpha_3 v^2 + \alpha_4 \omega^2}(\omega_{\text{err}}) \cdot \varepsilon_{\alpha_5 v^2 + \alpha_6 \omega^2}(\gamma_{\text{err}})$$

Чтобы убедиться в корректности алгоритма **motion_model_velocity** в Таблице 5.1, читатель может заметить, что в алгоритме реализовано именно это выражение. А именно, содержание строк со 2 по 9 эквивалентно выражениям (5.18), (5.19), (5.20), (5.21), (5.24), и (5.25). В строке 10 реализуется преобразование (5.32), заменяя выражения для ошибки, как указано в (5.29) до (5.31).

Выборка из $p(x'|u, x)$

Алгоритм выборки **sample_motion_model_velocity** в Таблице 5.3 реализует прямую модель, описанную ранее в этом разделе. Строки с 5 по 7 соответствуют выражению (5.16). Зашумленные значения, вычисленные в строках со 2 по 4, соответствуют выражениям (5.10) и (5.15).

Алгоритм **sample_normal_distribution** в Таблице 5.4 реализует общий подход к выполнению выборки из нормального распределения. Это приближение использует центральную предельную теорему, которая утверждает, что любое среднее невырожденной случайной переменной сводится до нормального распределения. Путём усреднения 12 однородных распределений в алгоритме **sample_normal_distribution** генерируются значения с, примерно, нормальным распределением. Технически, результирующие значения будут всегда лежать в промежутке $[-2b, 2b]$. Наконец, в **sample_triangular_distribution** в Таблице 5.4 реализуется выборка из множества для треугольных распределений.

5.4 Модель движения на основе одометрии

Модель движения на основе скорости, обсуждаемая выше, использует значения скорости робота для вычисления апостериорных распределений положений. В качестве альтернативного варианта для вычисления движения робота возможно использовать измерения одометрии. Данные одометрии обычно получаются путем интегрирования информации с энкодеров на колесах. Большинство коммерческих роботов выдают такие оценки положения через определённые интервалы времени (например, 10 раз в секунду). Это приводит к методу определения другой модели движения, обсуждаемой в этом главе – модели на основе одометрии. В ней вместо данных управления используется одометрия.

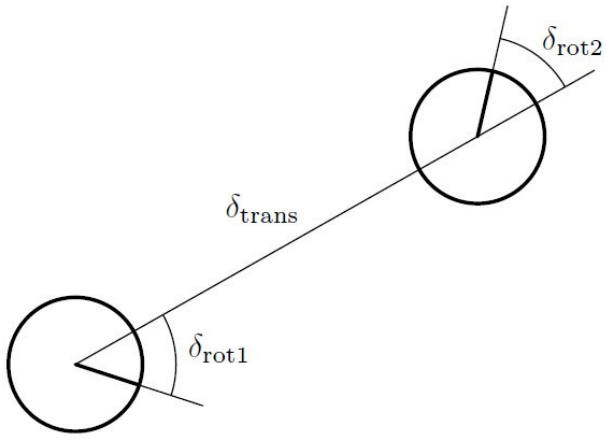


Рис. 5.7 Модель одометрии: Робот движется в интервале времени $(t-1, t]$ по пути, приблизительно определённом вращением δ_{rot1}, перемещением δ_{trans} и вторым вращением δ_{rot2}. Повороты и перемещения зашумлены.

Практический опыт показывает, что одометрия, даже при наличии ошибок, обычно даёт более точные данные, чем скорость. Точность обоих по-

казателей падает из-за пробуксовки и скольжения, но скорость, вдобавок, подвержена ещё и несоответствиям между реальными контроллерами движения и их (грубой) математической моделью. Кроме этого, данные одометрии доступны только в ретроспективе, после окончания перемещения робота. Это не является проблемой для алгоритмов фильтров, таких, как методы локализации и построения карт, обсуждаемых в следующих главах. Однако, это делает данные одометрии недоступными для задач точного планирования движения и управления.

5.4.1 Вычисление в закрытом виде

Технически, данные одометрии представляют собой измерения датчиков, а не управление, но при их моделировании в виде измерений результирующий байесовский фильтр должен включать текущие скорости в виде переменных состояния, что увеличивает размерность пространства состояний. Для сохранения малого размера пространства состояний, общепринято считать данные одометрии сигналами управления, поэтому и в книге будем рассматривать измерения одометров как управляющие воздействия. Результирующая модель входит в состав программных ядер многих лучших современных вероятностных робототехнических систем.

Определим формат информации управления. В момент времени t, верное положение робота моделируется случайной переменной x_t. Одометрия робота является оценкой положения, но, из-за пробуксовки и проскальзывания, точный метод перевода координат между внутренней одометрией робота и координатами физического мира отсутствует. Фактически, знание такого преобразования может решить задачу локализации робота!

В модели на основе одометрии используются относительные данные движения, измеренные внутренним одометром робота. В течение интервала времени $(t-1, t]$ робот перемещается из положения x_{t-1} в положение x_t. Одометр сообщает показания относительного перемещения из $\bar{x}_{t-1} = (\bar{x}\ \bar{y}\ \bar{\theta})^T$ в $\bar{x}_t = (\bar{x}'\ \bar{y}'\ \bar{\theta}')^T$. Здесь черта над переменной указывает на то, что измерения одометрии внесены во внутреннюю систему координат робота с неизвестным отношением к глобальной мировой системе координат .

1: **Algorithm motion_model_odometry** (x_t, u_t, x_{t-1}) :
2: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
3: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
4: $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

5: $\hat{\delta}_{\text{rot1}} = \text{atan2}(y' - y, x' - x) - \theta$
6: $\hat{\delta}_{\text{trans}} = \sqrt{(x - x')^2 + (y - y')^2}$
7: $\hat{\delta}_{\text{rot2}} = \theta' - \theta - \hat{\delta}_{\text{rot1}}$

8: $p_1 = \text{prob}(\delta_{\text{rot1}} - \hat{\delta}_{\text{rot1}}, \alpha_1 \hat{\delta}^2_{\text{rot1}} + \alpha_2 \hat{\delta}^2_{\text{trans}})$
9: $p_2 = \text{prob}(\delta_{\text{trans}} - \hat{\delta}_{\text{trans}}, \alpha_3 \hat{\delta}^2_{\text{trans}} + \alpha_4 \hat{\delta}^2_{\text{rot1}} + \alpha_4 \hat{\delta}^2_{\text{rot2}})$
10: $p_3 = \text{prob}(\delta_{\text{rot2}} - \hat{\delta}_{\text{rot2}}, \alpha_1 \hat{\delta}^2_{\text{rot2}} + \alpha_2 \hat{\delta}^2_{\text{trans}})$

11: *return* $p_1 \cdot p_2 \cdot p_3$

Таблица 5.5 Алгоритм для вычисления $p(x_t|u_t, x_{t-1})$ на основе данных одометрии. Здесь управление u_t задано в виде $(\bar{x}_{t-1}\, \bar{x}_t)^T$, с $\bar{x}_{t-1} = (\bar{x}\, \bar{y}\, \bar{\theta})$ и $\bar{x}_t = (\bar{x}'\, \bar{y}'\, \bar{\theta}')$.

Ключевой момент использования этой информации при оценке состояния состоит в том, что относительная разница между $\bar{x}_{t-1}$ и $\bar{x}_t$, при соответствующем определении термина, является хорошей функцией для оценки разницы реальных положений x_{t-1} и x_t. Информация о движении u_t задана в виде

$$u_t = \begin{pmatrix} \bar{x}_{t-1} \\ \bar{x}_t \end{pmatrix} \tag{5.33}$$

Чтобы извлечь относительные данные одометрии u_t необходимо выполнить преобразование для трёх последовательных этапов: вращения, за которым следует движение по прямой линии (поступательное), и ещё одного вращения. На Рис. 5.7 это показано следующим образом: первоначальный поворот назван δ_{rot1}, поступательное движение обозначено как δ_{trans}, а второй поворот - δ_{rot2}. Как читатель может легко убедиться, для каждой пары положений $(\bar{s}\, \bar{s}')$ имеется уникальный вектор параметров $(\delta_{\text{rot1}}\, \delta_{\text{trans}}\, \delta_{\text{rot2}})^T$, и этих параметров достаточно для реконструкции относительного перемещения между $\bar{s}$ и $\bar{s}'$. Таким образом, $\delta_{\text{rot1}}, \delta_{\text{trans}}, \delta_{\text{rot2}}$ образуют достаточные статистические показатели относительного движения, закодированного в виде одометрии.

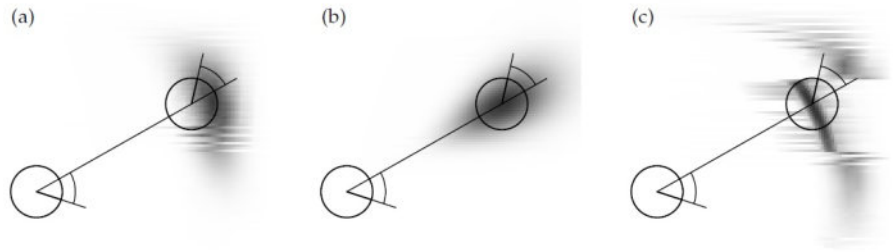


Рис. 5.8 Модель движения на основе одометрии при различных значениях параметров зашумления.

Вероятностная модель движения предполагает искажение этих трёх параметров независимыми шумами. Читатель может заметить, что при описании движения на основе одометрии используется на один параметр больше, чем в векторе скорости, определённом в предыдущем разделе. Поэтому вырождение, из-за которого мы были вынуждены использовать метод «последнего поворота», отсутствует.

Перед тем, как погрузиться в математические подробности, определим базовый метод вычисления этой плотности в закрытом виде. В Таблице 5.5 приведён алгоритм вычисления $p(x_t|u_t, x_{t-1})$ из данных одометрии. Он принимает на вход данные о начальном положении x_{t-1}, пару положений $u_t = (\bar{x}_{t-1}\, \bar{x}_t)^T$, полученных из одометрии робота, и гипотетическое конечное положение x_t. На выход возвращается числовая вероятность $p(x_t|u_t, x_{t-1})$.

В строках со 2 по 4 в Таблице 5.5 из показаний одометрии восстанавливаются параметры относительного перемещения $(\delta_{\text{rot1}}\, \delta_{\text{trans}}\, \delta_{\text{rot2}})^T$. Как и прежде, используется *обратная модель движения*. Соответствующие параметры относительного перемещения $(\hat{\delta}_{\text{rot1}}\, \hat{\delta}_{\text{trans}}\, \hat{\delta}_{\text{rot2}})^T$ для заданных положений x_{t-1} и x_t вычисляются в строках с 5 по 7 алгоритма. В строках с 8 по 10 вычисляются вероятности ошибок для отдельных параметров движения. Как и прежде, функция prob (a, b^2) реализует распределение ошибок по a с нулевым математическим ожиданием и дисперсией b^2. При реализации алгоритма необходимо следить, чтобы все угловые разницы находились в диапазоне $[-\pi, \pi]$. В силу этого результат $\delta_{\text{rot2}} - \bar{\delta}_{\text{rot2}}$ следует соответствующим образом сокращать — распространённая ошибка реализации, которую трудно обнаружить. Наконец, в строке 11 возвращается вероятность комбинированной ошибки, полученная перемножением отдельных вероятностей ошибки p_1, p_2 и p_3. На этом шаге подразумевается независимость различных источников ошибок. Переменные от α_1 до α_4 – специфические параметры робота, определяющие шумы движения.

1: **Algorithm sample_motion_model_odometry** (u_t, x_{t-1}) :
2: $\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
3: $\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$
4: $\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

5: $\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \text{sample}(\alpha_1 \delta^2_{\text{rot1}} + \alpha_2 \delta^2_{\text{trans}})$
6: $\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \text{sample}(\alpha_3 \delta^2_{\text{trans}} + \alpha_4 \delta^2_{\text{rot1}} + \alpha_4 \delta^2_{\text{rot2}})$
7: $\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \text{sample}(\alpha_1 \delta^2_{\text{rot2}} + \alpha_2 \delta^2_{\text{trans}})$

8: $x' = x + \hat{\delta}_{\text{trans}} \cos(\theta + \hat{\delta}_{\text{rot1}})$
9: $y' = y + \hat{\delta}_{\text{trans}} \sin(\theta + \hat{\delta}_{\text{rot1}})$
10: $\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

11: *return* $x_t = (x', y', \theta')^T$

Таблица 5.6 Алгоритм выборки из $p(x_t|u_t, x_{t-1})$ на основе данных одометрии. Здесь положение в момент времени t представлено в виде $x_{t-1} = (x\, y\, \theta)^T$. Управление представляет собой дифференцируемый набор двух оценок положения, полученных одометром робота, $u_t = (\bar{x}_{t-1}\, \bar{x}_t)^T$, где $\bar{x}_{t-1} = (\bar{x}\, \bar{y}\, \bar{\theta})$, а $\bar{x}_t = (\bar{x}'\, \bar{y}'\, \bar{\theta}')$.

На Рис. 5.8 показаны примеры модели движения на основе одометрии для разных значений параметров ошибки от α_1 до α_4. Если распределение на Рис. 5.8a выглядит знакомо, то на Рис. 5.8b и 5.8c показаны необычно большие ошибки перемещения и вращения. Читатель может самостоятельно сравнить эти схемы с приведёнными на Рис. 5.3 на странице 120. Чем меньше время между двумя последовательными измерениями, тем более схожи между собой эти две модели движения. Поэтому, если оценка обновляется достаточно часто, например 10 раз в секунду для робота, действующего внутри помещений, разница между моделями движения не слишком существенна.

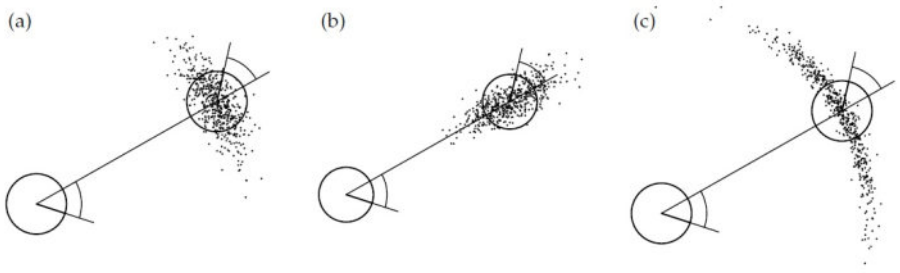


Рис. 5.9 Выборка из модели движения на основе одометрии, используя те же параметры, что на Рис. 5.8. На каждой схеме показаны 500 элементов выборки.

5.4.2 Алгоритм выборки

При использовании многочастичных фильтров в задаче локализации было бы желательно получить алгоритм выборки из $p(x_t|u_t, x_{t-1})$. Напомним, что многочастичные фильтры (подраздел 4.3) для любых x_{t-1}, u_t и x_t требуют элементов выборки из $p(x_t|u_t, x_{t-1})$, а не выражения для вычисления в закрытой виде $p(x_t|u_t, x_{t-1})$. Алгоритм **sample_motion_model_odometry**, показанный на Рис. 5.6, реализует метод выполнения выборки. На вход принимается начальное положение x_{t-1} и показания одометрии u_t, а на выход возвращается случайная переменная x_t, распределенная согласно $p(x_t|u_t, x_{t-1})$. Отличие от предыдущего алгоритма состоит в том, что вместо вычисления вероятности данного x_t, случайным образом выбирается гипотеза относительно положения x_t (строки 5-10). Как и прежде, алгоритм выборки **sample_motion_model_odometry** несколько проще в реализации, чем алгоритм в закрытом виде **motion_model_odometry**, поскольку он обходит необходимость в обратной модели.

На Рис. 5.9 показаны примеры наборов элементов выборки алгоритма **sample_motion_model_odometry**, с использованием тех же параметров, что и для показанной на Рис.5.8 модели. На Рис. 5.10 показано применение модели движения с наложением выборок нескольких тактов времени. Эти данные были сгенерированы с использованием уравнений обновления движения алгоритма **particle_filter** (Таблица 4.3). На рисунке одометрия пути робота показана сплошной линией. Кроме того, явно виден рост неопределённости при движении робота, в результате чего выборка распределена по все более возрастающей площади.

5.4.3 Математический вывод модели движения на основе одометрии

Вывод алгоритмов довольно прямолинеен, и может быть пропущен при первом чтении. Для вывода вероятностной модели движения с использованием одометрии вспомним, что относительная разница между двумя положениями отображена суммой трёх основных движений: поворота, движения по прямой линии (поступательного), и ещё одного поворота. Следующие уравнения демонстрируют способ вычисления значений двух поворотов и поступательного движения на основе показаний одометрии $u_t = (\bar{x}_{t-1}\, \bar{x}_t)^T$, где $\bar{x}_{t-1} = (\bar{x}\, \bar{y}\, \bar{\theta})$, а $\bar{x}_t = (\bar{x}'\, \bar{y}'\, \bar{\theta}')$:

$$\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \tag{5.34}$$

$$\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2} \tag{5.35}$$

$$\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}} \tag{5.36}$$

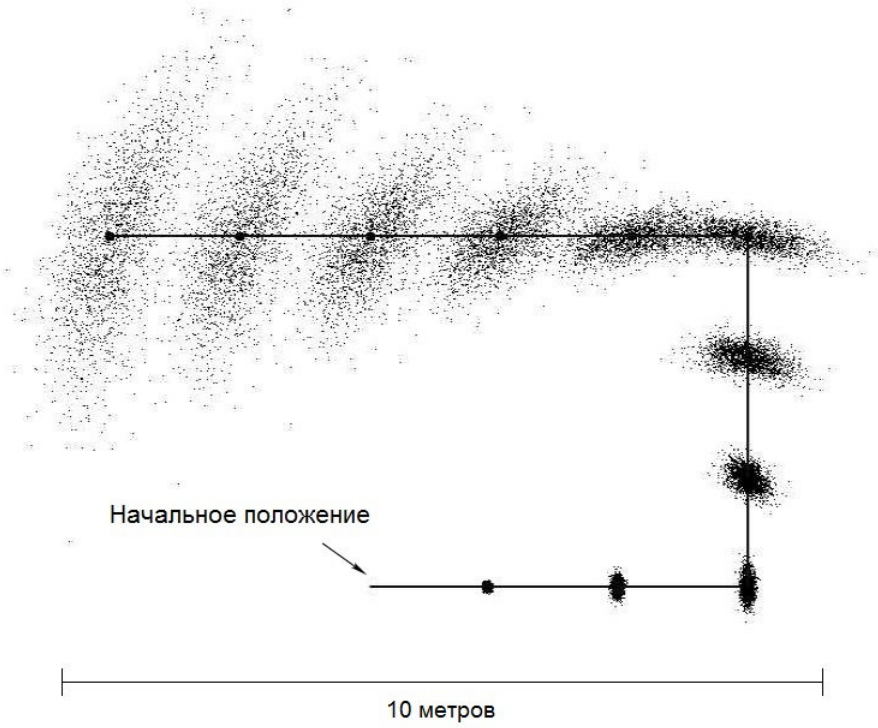


Рис. 5.10 Аппроксимация выборки гипотезы местоположения для робота без датчиков. Сплошной линией показаны действия, а элементы выборки отображают относительные местоположения робота в различные моменты времени.

Для моделирования ошибки движения, допустим, что «истинные» значения поворота и поступательного движения получены из измеренных путём вычитания независимого шума ε_{b^2} с нулевым математическим ожиданием и дисперсией b^2:

(5.37)
$$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \varepsilon_{\alpha_1\delta^2_{\text{rot1}}+\alpha_2\delta^2_{\text{trans}}}$$

(5.38)
$$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \varepsilon_{\alpha_3\delta^2_{\text{trans}}+\alpha_4\delta^2_{\text{rot1}}+\alpha_4\delta^2_{\text{rot2}}}$$

(5.39)
$$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \varepsilon_{\alpha_1\delta^2_{\text{rot2}}+\alpha_2\delta^2_{\text{trans}}}$$

Как и в предыдущем разделе, ε_{b^2} - переменная зашумления с нулевым математическим ожиданием и дисперсией b^2. Параметры от α_1 до α_2 являются специфическими параметрами ошибки робота, определяющими ошибку движения.

Следовательно, истинное местоположение x_t, получается из x_{t-1} путём первого поворота на угол $\hat{\delta}_{\text{rot1}}$, последующего линейного движения $\hat{\delta}_{\text{trans}}$, и второго поворота на угол $\hat{\delta}_{\text{rot2}}$. Таким образом,

$$(5.40)\quad \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} \hat{\delta}_{\text{trans}} \cos(\theta + \hat{\delta}_{\text{rot1}}) \\ \hat{\delta}_{\text{trans}} \sin(\theta + \hat{\delta}_{\text{rot1}}) \\ \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}} \end{pmatrix}$$

Заметим, что в алгоритме **sample_motion_model_odometry** реализованы выражения с (5.34) по (5.40).

Алгоритм **motion_model_odometry** получен путём сравнения строк 5-7, вычисляющих параметры движения $\hat{\delta}_{\text{rot1}}$, $\hat{\delta}_{\text{trans}}$, и $\hat{\delta}_{\text{rot2}}$ гипотетического местоположения x_t относительно начального местоположения x_{t-1}. Разница между ними

$$(5.41)\quad \delta_{\text{rot1}} - \hat{\delta}_{\text{rot1}}$$

$$(5.42)\quad \delta_{\text{trans}} - \hat{\delta}_{\text{trans}}$$

$$(5.43)\quad \delta_{\text{rot2}} - \hat{\delta}_{\text{rot2}}$$

составляет ошибку одометрии, считая, что x_t - истинное конечное местоположение. Модель ошибки, приведённая в уравнениях от (5.37) до (5.39) подразумевает, что вероятность этих ошибок задана как

$$(5.44)\quad p_1 = \varepsilon_{\alpha_1 \delta^2_{\text{rot1}} + \alpha_2 \delta^2_{\text{trans}}}(\delta_{\text{rot1}} - \hat{\delta}_{\text{rot1}})$$

$$(5.45)\quad p_2 = \varepsilon_{\alpha_3 \delta^2_{\text{trans}} + \alpha_4 \delta^2_{\text{rot1}} + \alpha_4 \delta^2_{\text{rot2}}}(\delta_{\text{trans}} - \hat{\delta}_{\text{trans}})$$

$$(5.46)\quad p_3 = \varepsilon_{\alpha_1 \delta^2_{\text{rot2}} + \alpha_2 \delta^2_{\text{trans}}}(\delta_{\text{rot2}} - \hat{\delta}_{\text{rot2}})$$

с определёнными выше распределениями ε. Эти вероятности вычислены в строках 8-10 алгоритма **motion_model_odometry**, и, поскольку ошибки подразумеваются независимыми, совместная вероятность является произведением $p_1 \cdot p_2 \cdot p_3$ (см. строку 11).

5.5 Движение и карты

Найдя $p(x_t|u_t, x_{t-1})$, мы определим "движение робота в вакууме", поскольку этой моделью описывается движение робота при отсутствии любых данных о природе окружающей среды. Во многих случаях также будет дана карта m, которая может содержать информацию относительно мест, через которые робот способен или не способен двигаться. Например, *карты занятости*, описанные в Главе 9, разделяют *свободную* (проходимую) и *занятую* территорию. Местоположение робота всегда должно располагаться на свободном месте. Таким образом, знание m даёт дополнительную информацию о местоположении робота x_t до, во время, и после выполнения действия управления u_t.

Эти соображения ведут к модели движения, принимающей во внимание карту m. Определим модель $p(x_t|u_t, x_{t-1}, m)$, добавив карту m в дополнение к стандартным переменным. Если m содержит информацию относительно оценки местоположения, получим

(5.47)
$$p(x_t|u_t, x_{t-1}) \neq p(x_t|u_t, x_{t-1}, m)$$

Модель движения $p(x_t|u_t, x_{t-1}, m)$ должна давать лучшие результаты по сравнению с моделями движения без карт $p(x_t|u_t, x_{t-1})$. Будем называть $p(x_t|u_t, x_{t-1}, m)$ *моделью движения на основе карт*. Модель движения на основе карт вычисляет правдоподобность оценки того, что робот, расположенный в среде с картой m после выполнения команды u_t из начального местоположения x_{t-1} прибывает в местоположение x_t. К сожалению, вычисление этой модели движения в закрытом виде затруднено, поскольку для вычисления правдоподобности положения в x_t после выполнения действия u_t необходимо учесть вероятность существования свободного пути между x_{t-1} и x_t, и возможность робота следовать этому незанятому пути при выполнении управляющего действия u_t, что является весьма нетривиальной задачей.

МОДЕЛЬ ДВИЖЕНИЕ НА ОСНОВЕ КАРТ

К счастью, для модели движения на основе карт существует эффективный метод приближения, который хорошо работает, если расстояние между x_{t-1} и x_t мало (менее половины диаметра робота). Метод приближения разлагает модель движения на основе карт на две компоненты:

(5.48)
$$p(x_t|u_t, x_{t-1}, m) = \eta \frac{p(x_t|u_t, x_{t-1})p(x_t|m)}{p(x_t)}$$

где η – нормализующий член. Обычно, $p(x_t)$ также однородна и может быть включена в нормализующие константы. Необходимо только перемножить оценку без карты $p(x_t|u_t, x_{t-1})$ и, $p(x_t|m)$, который выражает «соответствие» оценки x_t и карты m.

```
1:  Algorithm motion_model_with_map (x_t, u_t, x_{t-1}, m) :
2:    return p(x_t|u_t, x_{t-1}) · p(x_t|m)

1:  Algorithm sample_motion_model_with_map (u_t, x_{t-1}, m) :
2:    do
3:        x_t = sample_motion_model(u_t, x_{t-1})
3:        π = p(x_t|m)
4:    until π > 0
5:    return ⟨x_t, π⟩
```

Таблица 5.7 Алгоритм для вычисления $p(x_t|u_t, x_{t-1}, m)$, использующий карту местности m. Этот алгоритм дополняет предыдущие модели движения (Таблицы 5.1, 5.3, 5.5, и 5.6) возможностью отслеживания невозможности нахождения робота на занятой ячейке карты m.

Для карт занятости $p(x_t|m) = 0$ тогда и только тогда, когда робот располагается на занятой ячейке сетки на карте и константе – в других случаях. Перемножением $p(x_t|m)$ и $p(x_t|u_t, x_{t-1})$, получим распределение, которое назначает всю массу вероятности местоположениям x_t , соответствующим данным карты, которые, в другом случае, будут иметь ту же форму, что и

$p(x_t|u_t, x_{t-1})$. Поскольку η может быть учтён при нормализации, это приближение модели движения на основе карты может эффективно вычисляться без существенных потерь по сравнению с моделью движения без карты.

В Таблице 5.7 перечислены основные алгоритмы для вычисления и выборки из моделей движения на основе карты. Обратим внимание, что алгоритм выборки возвращает взвешенный элемент, который включает фактор значимости по отношению $p(x_t|m)$. Следует соблюдать осторожность при реализации версии выборки, чтобы гарантировать выход из внутреннего цикла. Пример модели движения показан на Рис. 5.11. Плотность вероятности на Рис. 5.11a составляет $p(x_t|u_t, x_{t-1})$ и вычислена согласно модели движения на основе скорости. Теперь допустим, карта m содержит длинное прямоугольное препятствие (Рис. 5.11b). Вероятность $p(x_t|m)$ равна нулю для всех местоположений робота x_t , где он пересекается с препятствием. Поскольку робот в примере круглый, эта область равна размеру препятствия, увеличенного на радиус робота.

КОНФИГУРАЦИОННОЕ ПРОСТРАНСТВО

Это эквивалентно препятствиям из *рабочего пространства* в *конфигурационном пространстве* робота или пространстве местоположений. Результирующая вероятность $p(x_t|u_t, x_{t-1}, m)$ показана на Рис. 5.11b, и представляет нормализованное произведение $p(x_t|m)$ и $p(x_t|u_t, x_{t-1})$. Она равна нулю в расширенной области препятствия, и пропорциональна $p(x_t|u_t, x_{t-1})$ во всех остальных точках.

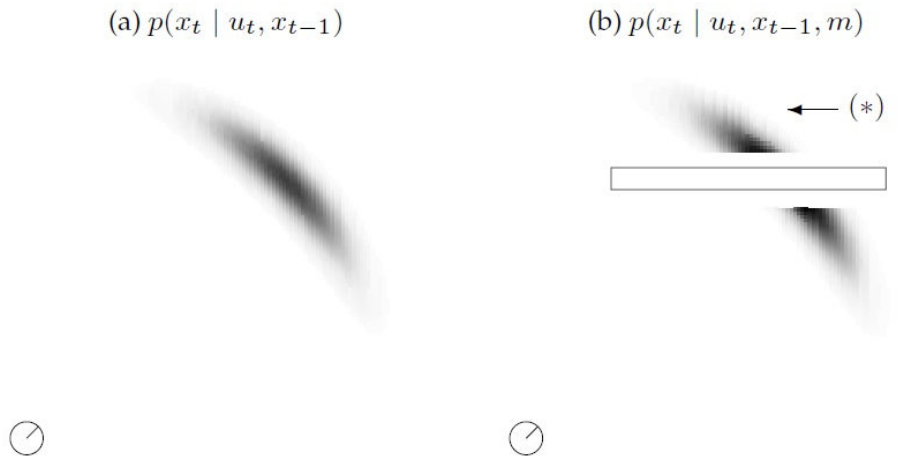


Рис. 5.11 Модель движения на основе скорости (a) без карты, (b) перенесённая на карту m.

На Рис. 5.11 также показана проблема такой аппроксимации. Область, отмеченная $(*)$, имеет ненулевое правдоподобие, поскольку и $p(x_t|u_t, x_{t-1})$ и $p(x_t|m)$ в этой области не равны нулю. Однако, чтобы робот смог попасть в эту область, он должен пройти сквозь стену, что в реальном мире невозможно. Эта ошибка является результатом проверки целостности модели только в конечном местоположении x_t, вместо проверки на пути к цели. На практике такие ошибки происходят только при сравнительно больших значениях u_t и могут быть устранены увеличением частоты обновления.

Чтобы прояснить природу этой аппроксимации, приведём ее краткий вывод. Выражение (5.48) можно получить из теоремы Байеса:

(5.49)
$$p(x_t|u_t, x_{t-1}, m) = \eta\, p(m|x_t, u_t, x_{t-1}) p(x_t|u_t, x_{t-1})$$

Если аппроксимировать $p(m|x_t, u_t, x_{t-1})$ с помощью $p(m|x_t)$ и заметить, что $p(m)$ – константа, связанная в искомой апостериорной вероятностью, получим следующее искомое равенство:

(5.50)
$$\begin{aligned} p(x_t|u_t, x_{t-1}, m) &= \eta\, p(m|x_t) p(x_t|u_t, x_{t-1}) \\ &= \eta\, \frac{p(x_t|m)p(m)}{p(x_t)}\, p(x_t|u_t, x_{t-1}) \\ &= \eta\, \frac{p(x_t|m)p(x_t|u_t, x_{t-1})}{p(x_t)} \end{aligned}$$

Здесь η – нормализующий член (заметим, что значение η различно на разных этапах преобразования). Этот короткий анализ показывает, что наша модель на основе карты подтверждается при грубом допущении, что

(5.51)
$$p(m|x_t, u_t, x_{t-1}) = p(m|x_t)$$

Очевидно, что эти выражения не равны. При вычислении условной вероятности по m, из аппроксимации вычёркиваются два члена: u_t и x_{t-1}. В результате этого теряется вся информация относительно пути, ведущего к x_t. Все, что известно, это конечное местоположение x_t. Заметим, что условием вычёркивания в приведённом примере является ненулевая вероятность для робота оказаться за стеной. Если начальное и конечное местоположение находятся на свободном пространстве, приближенная модель движения на основе карты может ошибочно разрешить роботу пройти сквозь стену. Насколько критично это может быть? Как было замечено ранее, все зависит от величины интервала обновления. Фактически, при достаточно высоких скоростях обновления, и с учётом ограниченности переменных зашумления в модели движения, можно гарантировать, что отрицательный эффекта не возникнет, а аппроксимация останется достаточно близкой.

В этом анализе показана одна из неочевидных особенностей реализации алгоритма - следует обращать внимание на частоту обновления. Байесовский фильтр с очень высокой частотой обновления может показать совершенно отличный результат по сравнению с аналогичным фильтром, который обновляется лишь от случая к случаю.

5.6 Вывод

В этом разделе приводится вывод двух основных вероятностных моделей движения для мобильных роботов, действующих на плоскости.

• Был выведен выполняемый на ограниченном интервале времени Δt алгоритм вероятностной модели движения $p(x_t|u_t, x_{t-1})$, выражающий параметр управления u_t через поступательную и угловую скорости. При реализации модели было определено, что двух параметров шумов управления, (одного для поступательной и одного – для угловой скорости) недостаточно для генерации апостериорного распределения соответствующей размерности. В силу этого был добавлен третий параметр учёта шумов, выраженный

в виде добавочного последнего действия поворота после окончания любого движения.

• Была представлена альтернативная модель движения, использующая одометрию робота в качестве входных данных. Измерения одометрии выражаются тремя параметрами – начальным поворотом, линейным перемещением и конечным поворотом. Вероятностная модель движения была реализована на основе зашумления всех трёх параметров. Было замечено, что показания одометра, технически говоря, не являются управляющим сигналом, однако, при использовании их в интерпретации "управления", формулировка задачи оценки сильно упростится.

• Для обеих моделей движения были предоставлены два типа реализаций, в одной из которых вероятность $p(x_t|u_t, x_{t-1})$ вычисляется в закрытом виде, а во втором - выполнялась генерация выборки из $p(x_t|u_t, x_{t-1})$. Выражение в закрытой форме принимает на вход x_t, u_t и x_{t-1}, возвращая числовое значение вероятности. Чтобы сравнить действительные и заданные параметры управления для вычисления этой вероятности алгоритмы эффективно обращают модель движения. Модель с выборкой не требует такого обращения, поскольку в ней реализована прямая модель движения $p(x_t|u_t, x_{t-1})$. На вход принимаются значения u_t и x_{t-1}, а на выходе находится случайный элемент x_t, извлечённый согласно выборке $p(x_t|u_t, x_{t-1})$. Модели в закрытом виде требуются лишь для некоторых вероятностных алгоритмов. Другие алгоритмы, в частности, многочастичные фильтры, основаны на моделях с выборкой.

• Наконец, все модели движения были обобщены для включения карт окружающей среды, когда в результирующей вероятности $p(x_t|u_t, x_{t-1}, m)$ учтена карту m. Это обобщение следует интуитивному соображению о использовании карты для определения мест, где робот может находиться, и расчёта возможности перемещения из положения x_{t-1} в x_t. Результирующий алгоритм приведён в приближенном виде, поскольку проверяется только допустимость конечного положения.

Представленные модели движения являются лишь примерами. Конечно, область изучения действий роботов намного шире задачи перемещения мобильных роботов по плоской поверхности. Даже в области мобильной робототехники существует множество устройств, которые не были освещены в изложении материала. В качестве примера можно привести голономных роботов, способных двигаться в стороны без поворота или транспортные средства с подвеской. Обсуждение не включает динамику роботов, что очень важно для быстро движущихся транспортных средств, например, автомашин на шоссе. Большинство этих роботов может быть смоделировано похожим образом, для чего следует только определить физические законы движения робота и соответствующие параметры зашумления. В динамических моделях это потребует расширения состояния робота вектором скорости, который будет отображать динамическое состояние транспортного средства. По большей части, эти обобщения достаточно очевидны.

Что же касается измерения собственного движения, во многих роботах для измерения движения используются инерционные датчики для дополнения или замены одометрии. Проектированию фильтров с использованием инерционных датчиков посвящены целые книги. Мы призываем читателя использовать более сложные модели и датчики в тех случаях, когда данных

одометрии недостаточно.

5.7 Библиографические примечания

Представленный материал обобщает базовые кинематические уравнения некоторых типов мобильных роботов(Cox и Wilfong, 1990) добавлением вероятностного компонента. Используемой моделью описываются дифференциальный привод, привод Аккермана и синхронный привод (Borenstein et al., 1996). За пределами рассмотрения модели находятся приводы без неголономных ограничений (Latombe 1991), например колёсные роботы Mecanum (Ilon 1975) или шагающие роботы, наподобие описанных в перспективных работах Райберта (Raibert et al.,1986, Raibert, 1991, Saranli and Koditschek, 2002).

В робототехнике движение и взаимодействие роботов с окружающей средой изучалось весьма широко. Современные публикации по мобильной робототехнике, освещающие аспекты кинематики и динамики были написаны Мерфи (Murphy,2000c), Дудеком и Дженкином (Dudek и Jenkin, 2000), Сигвартом и Норбакшем (Siegwart и Nourbakhsh, 2004). Кокс и Вилфонг (Cox and Wilfong, 1990) подготовили сборник статей передовых (на момент публикации) исследователей. Также можно обратиться к работе Кортенкампа (Kortenkamp et al.,1998). Классический подход к кинематике и динамике робота можно найти у Крейга (Craig, 1989), Вукобратовича (Vukobratovic, 1989), Пола (Paul, 1981) и Йошикавы (Yoshikawa, 1990). Более современная работа по теме динамики робота была написана Физерстоуном (Featherstone, 1987). Податливое движение как одна из форм взаимодействия с окружающей средой было исследовано Мэйсоном (Mason, 2001). Механика грунтов, в части взаимодействия колёсных роботов с поверхностью была изучена в фундаментальных работах Беккера (Bekker, 1956, 1969) и Вонга (Wong, 1989). Современная публикация о взаимодействии колеса с грунтом была написана Иагнемма и Дубовски (Iagnemma and Dubowsky, 2004). Обобщение таких моделей в рамках единой вероятностной концепции – перспективное направление для будущих исследований.

5.8 Упражнения

ДИНАМИКА

1. Все модели роботов, представленные в главе - кинематические. В этом упражнении будет описан робот с *динамикой*. Представим робота, который находится в одномерной системе координат. Его местоположение задано координатой x, скорость - $\dot{x}$, ускорение - $\ddot{x}$. Допустим также, что управление влияет лишь на ускорение $\ddot{x}$. Разработать математическую модель движения, вычисляющую апостериорное положение x' и скорость $\dot{x}'$ на основании начального положения x и скорости $\dot{x}$. Считать, что ускорение $\ddot{x}$ является суммой указанного ускорения и гауссового шума с нулевым математическим ожиданием и дисперсией σ^2 (при условии, что ускорение в течение такта моделирования Δt остаётся постоянным). Коррелируют ли x' и $\dot{x}'$ в апостериорном распределении? Объяснить, почему.

2. Снова представим робота из упражнения 1. Написать математическую формулу вычисления апостериорного распределения по конечной скорости $\dot{x}'$, основываясь на начальном положении робота x, начальной скорости $\dot{x}$, и конечному положению x'. Что примечательного в этом распределении?

3. Допустим, робот управляется случайными ускорениями в течение T временных интервалов, при некоем большом значении T. Будут ли скоррелированы конечное положение x и скорость $\dot{x}$? Если да, будут ли они *полностью* коррелировать при $T \uparrow \infty$, так, что одна переменная станет детерминированной функцией другой?

4. Представим простую кинематическую модель идеального велосипеда. Оба колеса имеют диаметр d, и закреплены на раме длины l. Переднее колесо может поворачиваться вокруг вертикальной оси, угол поворота обозначим как α. Заднее колесо всегда параллельно раме велосипеда и поворачиваться не может.

В этом упражнении положение велосипеда будет определено с помощью трёх переменных: координат $x - y$ центра переднего колеса, угла направления θ (поворота) рамы велосипеда по отношению к внешней системе координат. Скорость движения велосипеда вперёд v и угол поворота колеса α будем считать постоянными в каждом такте прогнозирования.

Сформулировать математическую модель прогнозирования в интервале времени Δt, считая, что угол поворота колеса α и скорость движения вперёд v подвержены воздействию гауссовского шума. Модель должна прогнозировать апостериорное состояние велосипеда после промежутка времени Δt от известного состояния. Если точную модель найти не удаётся, сформулировать приближенную и объяснить приближения.

5. Вспомним кинематическую модель велосипеда из упражнения 4. Реализовать функцию выборки для апостериорных положений велосипеда при некоторых соображениях зашумленности.

Для модели можно считать, что $l = 100\text{см}, d = 80\text{см}, \Delta t = 1\text{сек}, |\alpha| \leq 80^\circ, v \in [0; 100]\text{см/сек}$. Допустим, что дисперсия угла поворота колеса $\sigma_\alpha^2 = 25^{\circ 2}$, а дисперсия скорости $\sigma_v^2 = 50\text{см}^2/\text{сек}^2 \cdot v^2$. Заметим, что дисперсия скорости зависит от указанной скорости.

Изобразить на графике результирующий набор элементов выборки для велосипеда, стартующего в начале координат со следующими параметрами управления:

problem number	α	v
1	25°	20см/сек
2	-25°	20см/сек
3	25°	90см/сек
4	80°	10см/сек
1	85°	90см/сек

На всех графиках изобразить оси с единицами измерения

6. Снова используем модель велосипеда из упражнения 4. На основании начального состояния x, y, θ и конечных координат x' и y' (но без конечного угла θ'), сформулировать математическую формулу для определения наиболее вероятных значений α, v и θ'. Если решение в закрытом виде найти не удаётся, предложить метод аппроксимации искомых значений.

ГОЛОНОМНЫЙ ПРИВОД

7. Обычно роботы, действующие внутри помещений, оснащены какой-либо разновидностью *голономного* привода. Робот с голономным приводом имеет количество управляемых степеней свободы, равное количеству измерений пространства его конфигурации (или положения). В этом упражнении требуется обобщить модель на основе скорости для голономного робота, действующего на плоскости. Условимся, что робот может управляться изменением скорости продольного и поперечного перемещения, а также скорости поворота. Произвольно будем считать положительными значения скорости поперечного перемещения, направленные влево, а отрицательными – направленные вправо.

• Определить математическую модель такого робота, считая что его управляющие действия подвержены независимому гауссовому шуму.

• Сформулировать алгоритм вычисления $p(x_t|u_t, x_{t-1})$.

• Сформулировать алгоритм выборки для $x_t \sim p(x_t|u_t, x_{t-1})$.

8. Доказать, что треугольное распределение в выражении (5.12) имеет нулевое математическое ожидание и дисперсию b^2. Доказать то же самое для алгоритма выборки в Таблице 5.4.

6 Восприятие робота

6.1 Введение

Модели измерения окружающей среды вместе с моделями движения составляют две, специфические для вероятностной робототехники, группы. Моделями измерений описывается процесс взаимодействия, посредством которого в физическом мире генерируются измерения датчиков. Сегодня в роботах используются самые разнообразные датчики, такие как датчики касания, измерения расстояния или видеокамеры. Специфика модели зависит от конкретного вида датчика: датчики изображения лучше всего моделируются с помощью проективной геометрии, а сонары – описанием звуковой волны и ее отражения от различных поверхностей в окружающей среде.

В вероятностной робототехнике обязательно моделируется шум датчиков, чтобы учесть изначальная неопределённость, присущую самому процессу измерения. Формально, модель измерений определяется в виде распределения условной вероятности $p(z_t|x_t, m)$, где x_t – положение робота, z_t – измерение в момент времени t, а m – карта окружающей среды. Хотя в материале данной главы, в основном, описаны датчики расстояния, предлагаемые принципы и уравнения применимы и для других типов датчиков, таких, как видеокамера или детектор ориентиров на основе считывателя штрих-кодов.

СКАНИРОВАНИЕ ДАЛЬНОСТИ СОНАРОМ

Чтобы проиллюстрировать основную проблему мобильных роботов, использующих для восприятия окружающей среды датчики, на Рис. 6.1a показано типичное *сканирование дальности сонаром* в коридоре с помощью мобильного робота, оборудованного круговым массивом из 24 ультразвуковых датчиков. Значения расстояний, измеренные отдельными датчиками, выделены светло-серым цветом, карта окружающей среды показана чёрным. Большинство измерений соответствуют расстоянию до ближайшего

объекта в конусе измерения. При этом на некоторых измерениях отсутствуют обнаруженные объекты.

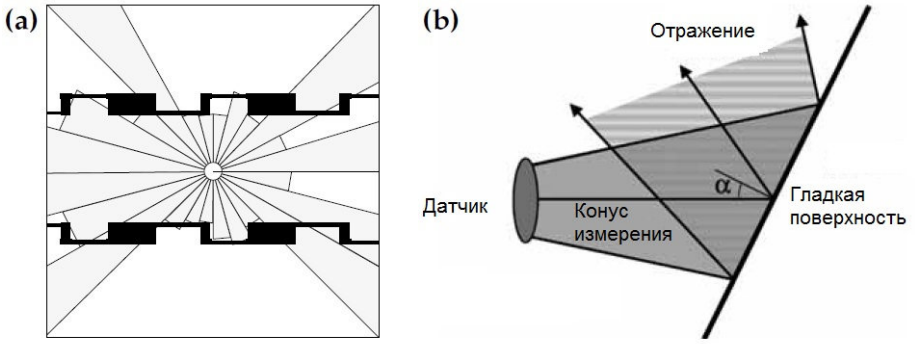


Рис. 6.1 (a) Типичное ультразвуковое сканирование роботом окружающей среды. (b) Ошибочное измерение ультразвукового измерения дальности. Такой эффект возникает при попадании сигнала сонара на отражающую поверхность под углом α, превышающим половину угла раскрытия датчика.

Неспособность датчиков надёжно измерить расстояние до близлежащих предметов часто называют шумами датчика.

ЗЕРКАЛЬНОЕ ОТРАЖЕНИЕ

Технически, такие шумы довольно предсказуемы, поскольку гладкие поверхности (например, стены), отражают звук зеркально, и можно считать, что стена для звуковой волны сонара является аналогом зеркала. Трудности возникают, в основном, когда звуковая волна сталкивается с гладкой поверхностью под углом. В таком случае эхо может распространяться в направлении, отличном от обратного, что показано на Рис. 6.1b. Этот эффект часто ведёт к слишком большим показаниям измерения, по сравнению с реальным расстоянием до ближайшего объекта в основном конусе. Вероятность возникновения такого эффекта зависит от ряда характеристик, например, материала поверхности, угла между нормалью к поверхности и направлением конуса датчика, расстояния до поверхности, ширины основного конуса датчика, чувствительности сонара. Другие ошибки, такие как слишком малые показания, могут вызываться перекрёстными наводками между разными датчиками (звук распространяется медленно!) или присутствием не смоделированных объектов, например, людей, в непосредственной близости от робота.

СКАНИРОВАНИЕ РАССТОЯНИЯ ЛАЗЕРОМ

На Рис. 6.2 показано типичное *сканирование расстояния лазером*, полученное с помощью 2D лазерного дальномера. Лазер аналогичен сонару в том смысле, что он тоже активно излучает сигнал и записывает эхо, но в случае лазера сигнал – это световой луч. Ключевая разница с сонаром состоит в том, что лазеры позволяют получить гораздо более сфокусированные лучи. Лазер, изображённый на Рис. 6.2, измеряет расстояние по величине запаздывания отражённого сигнала, при этом измерения разделены углом в один градус.

Как правило, чем точнее модель датчика, тем лучше результаты измерений, хотя некоторые важные замечания по этому поводу уже обсуждались в подразделе 2.4.4. На практике довольно часто точное моделирование датчика невозможно, в основном, в силу сложности происходящих в них физических процессов.

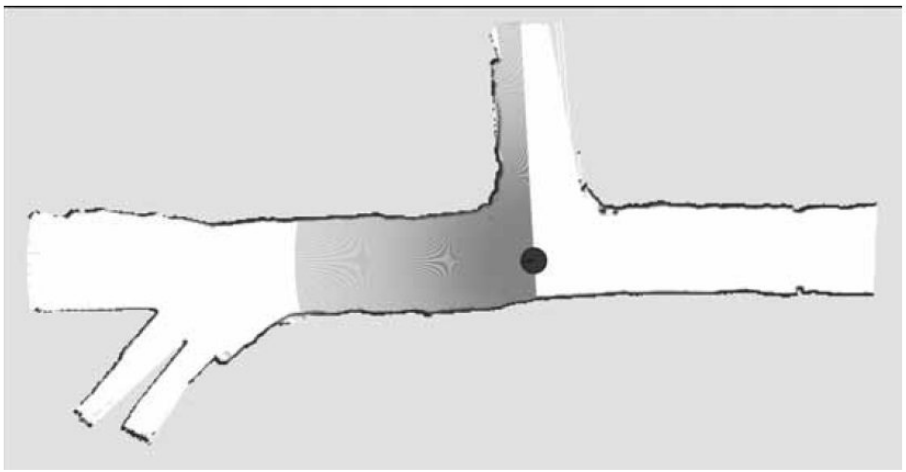


Рис. 6.2 Типичное сканирование расстояния лазерным лучом, полученное с лазера SICK LMS. Окружающая среда, показанная на рисунке – угольная шахта. Изображение принадлежит Дирку Хёнелу (Dirk Hähnel), университет Фрайбурга.

Часто характеристики отклика датчика зависят от переменных, которые не хотелось бы явно описывать в вероятностном алгоритме робота (например, материал поверхности стен, который, без особой необходимости, не указывается при составлении карт для робота). Вероятностная робототехника обрабатывает стохастические неточности моделей датчиков путём моделирования процесса измерения в виде плотности условной вероятности $p(z_t|x_t)$, а не детерминированной функции $z_t = f(x_t)$. В силу этого, неопределённость датчика возможно учесть в недетерминированной части модели. В этом проявляется ключевое преимущество вероятностных методов перед классической робототехникой: на практике возможно использовать весьма грубые модели. Тем не менее, при построении вероятностной модели следует следить за полнотой учёта различных типов неопределённости, которые могут влиять на измерения датчика.

Многие датчики при запросе генерируют более одного числового значения. Например, камеры генерируют целые массивы значений (яркость, насыщенность, цвет). Похожим образом, дальномеры также обычно генерируют целые наборы значений сканирования дальностей. Определим количество значений в измерении z_t как K, и запишем в следующем виде:

(6.1)
$$z_t = \{z_t^1, ..., z_t^K\}$$

Чтобы сослаться на конкретное измерение, воспользуемся записью z_t^k (одно значение расстояния в измерении).

Вероятность $p(z_t|x_t, m)$ получается из произведения отдельных правдоподобий измерения

(6.2)
$$p(z_t|x_t, m) = \prod_{k=1}^{K} p(z_t^k|x_t, m)$$

Технически, это определяет допущение о независимости шумов каждого отдельного луча измерений, по аналогии с тем, как марковское свойство

предполагает независимость шумов от времени (см. подпункт 2.4.4). Это допущение верно только в идеальном случае. Мы уже обсуждали возможные причины зависимости шумов в подпункте 2.4.4. Повторимся, зависимость обычно обусловлена рядом причин: наличием людей в непосредственной близости робота, которые могут исказить измерения нескольких рядом расположенных датчиков, ошибок модели карты m, приближений апостериорного распределения и так далее. Но сейчас мы просто не будем учитывать возможные нарушения независимости, поскольку вернёмся к этой теме позже.

6.2 Карты

Чтобы упорядочить процесс генерации измерений, необходимо формализовать среду, в которой они производятся. Карта среды – это список окружающих объектов и их местоположений. Карты уже обсуждались в неформальном виде в предыдущей главе при разработке модели движения мобильного робота, который бы принимал во внимание занятость отдельных мест окружающего мира. Формально, карта m представляет собой список объектов окружающей среды и их свойств:

(6.3)
$$m = \{m_1, m_2, ..., m_N\}$$

Здесь N – общее количество объектов окружающей среды, а каждая переменная m_n, при $1 \leq n \leq N$, определяет свойство. Карты обычно индексируются одним из двух способов, известных, как *индексация по признакам* и *индексация по местоположению*. В картах, индексированных по признакам, n - это индекс признака. Значение m_n содержит, наряду с характеристиками признака, координаты его местоположения. В картах, индексированных по местоположению, каждый индекс n соответствует отдельному местоположению. В плоских картах принято обозначать элемент карты в виде $m_{x,y}$ вместо m_n, чтобы явно указать на принадлежность $m_{x,y}$ конкретным координатам $(x\, y)$ окружающего мира.

ОБЪЁМНЫЕ КАРТЫ

Оба типа карт обладают как преимуществами, так и недостатками. Карты на основе местоположения *объемные*, поэтому позволяют пометить любой объект окружающего мира. Объемные карты содержат информацию не только об объектах окружающей среды, но и об отсутствии таковых, то есть о свободном пространстве. Это довольно сильно отличается от карт на основе признаков. Карты на основе признаков описывают форму окружающего пространства в конкретных местоположениях, обозначенных в виде объектов на карте. Представление в виде признаков значительно упрощает изменение положения объекта, например, в результате дополнительных данных, полученных восприятием. Карты на основе признаков завоевали популярность в мире робототехники именно из-за возможности составления на базе данных датчиков. В книге мы столкнёмся с обоими типами карт и даже, время от времени, будем переходить от одного представления к другому.

Классическое представление карты известно как *карта сетки занятости* (occupancy grid map), и будет детально обсуждаться в Главе 9. Карты занятости основаны на допущении, что для каждого местоположения $x - y$ возможно назначить бинарный признак занятости, показывающий, занято ли это местоположение каким-либо объектом. Карты занятости отлично

подходят для мобильной навигации, поскольку позволяют легко находить пути по незанятому пространству.

В ходе изложения мы не будем делать различий между физическим миром и его картой. Конечно, измерения датчиков вызываются взаимодействием с физическими объектами, а не картой этих объектов, но, традиционно, модели датчиков переносятся на карту m, потому возможно допущение зависимости измерений от карты.

6.3 Модели дальномеров на основе лучей

Дальномеры являются одними из самых популярных датчиков в робототехнике. Поэтому наша первая *модель измерений* в этой главе, представляет собой примерную физическую модель датчиков расстояния. Дальномеры измеряют расстояние до близлежащих объектов. Расстояние можно измерить вдоль луча, что хорошо имитирует работу лазерных дальномеров, или же вдоль конуса, что предпочтительнее для ультразвуковых датчиков.

6.3.1 Основной алгоритм измерений

В нашей модели будет учитываться четыре типа ошибок измерений, каждый из которых важен для работы: малые случайные шумы измерений, ошибки из-за неучтенных объектов, ошибки при обнаружении объектов и случайные ошибки невыясненной природы. Искомая модель $p(z_t|x_t, m)$ представляет собой смесь четырёх плотностей, каждая из которых соответствует конкретному типу ошибки:

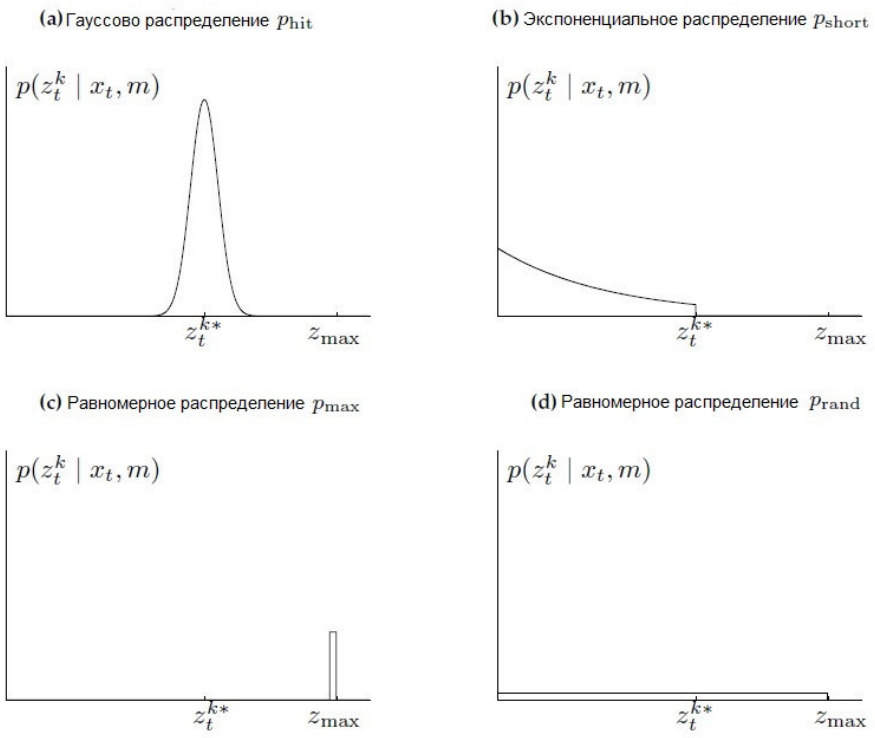


Рис. 6.3 Компоненты модели датчика определения расстояния. На каждой схеме горизонтальная ось соответствует измерению z_t^k, а вертикальная - правдоподобию.

1. **Верное измерение дистанции с локальным шумом измерений**. В идеальных условиях дальномер всегда измеряет верное расстояние до ближайшего объекта в поле измерения. Обозначим через z_t^{k*} «истинное» расстояние до объекта, измеренное z_t^k. В картах на основе местоположения расстояние z_t^{k*} можно определить, используя метод «бросания лучей» (ray casting), в картах на основе признаков они обычно получаются путём поиска ближайшего признака в конусе измерений. Однако, даже если датчик верно измеряет расстояние до ближайшего объекта, возвращаемое значение подвержено ошибке. Эта ошибка возникает в силу ограниченного разрешения дальномеров, атмосферных эффектов измеряющего сигнала и так далее.

ШУМЫ ИЗМЕРЕНИЙ

Этот вид *шумов измерений* обычно моделируется узким гауссовым распределением с математическим ожиданием z_t^{k*} и стандартным отклонением σ_{hit}. Обозначим гауссову функцию как p_{hit}. На Рис. 6.3а показана плотность p_{hit} отдельного значения z_t^{k*}.

На практике значения измерений дальномера ограничены интервалом $[0; z_{max}]$, где z_{max} - максимальное расстояние измерений. Отсюда, вероятность измерения задана в виде

(6.4)

$$p_{hit}(z_t^k|x_t, m) = \begin{cases} \eta\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) & \text{если } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{в других случаях} \end{cases}$$

где z_t^{k*} вычисляется на основании x_t и m с помощью бросания лучей,

а $\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2)$ определяет одномерное нормальное распределение с математическим ожиданием z_t^{k*} и стандартным отклонением σ_{hit}:

(6.5)

$$\mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) = \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2}\frac{(z_t^k - z_t^{k*})^2}{\sigma_{hit}^2}}$$

Нормализующий член η, оценочно

(6.6)

$$\eta = \left(\int_0^{z_{max}} \mathcal{N}(z_t^k; z_t^{k*}, \sigma_{hit}^2) dz_t^k \right)^{-1}$$

Стандартное отклонение σ_{hit} является действительным параметром зашумления модели измерения. Мы обсудим стратегии установки этого параметра ниже.

2. Неучтенные объекты. Окружающие среды мобильных роботов изменчивы, хотя сами карты m статические. В результате, объекты, отсутствующие на карте, могут заставить дальномеры выдавать неожиданно малые значения расстояний, по крайней мере, при сравнении с картой. Типичным примером таких движущихся объектов, которых нет на карте, являются люди, находящиеся в непосредственной близости от робота. Одним из способов обработки таких ситуаций является включение их в вектор состояния с последующей оценкой местоположения. Другим, гораздо более простым путём является включение их в шумы. При включении в шумы объекты, которых нет в модели, могут укорачивать измеренные расстояния z_t^{k*}, но неспособны их увеличивать.

Вероятность обнаружения неучтенных объектов уменьшается с расстоянием. Для примера, представим двух человек, которые независимо и с одинаковой вероятностью появляются в поле восприятия датчика движения. Расстояние до первого человека равно r_1, а до второго - r_2. Также допустим, что $r_1 < r_2$, без потери обобщения. Видно, что большую вероятность будет иметь измерение r_1, нежели r_2. При появлении первого человека показания датчика будут r_1. Для того, чтобы показания были r_2, должна сложиться ситуация, когда будет присутствовать только второй человек, а первый - отсутствовать.

Математически, вероятность измерения расстояния в таких ситуациях описывается *экспоненциальным распределением*. Параметр этого распределения, λ_{short}, является внутренним параметром модели измерений. В соответствии с определением экспоненциального распределения, получаем следующее равенство для $p_{short}(z_t^k | x_t, m)$:

(6.7)

$$p_{short}(z_t^k | x_t, m) = \begin{cases} \eta\, \lambda_{short}\, e^{-\lambda_{short} z_t^k} & \text{если } 0 \le z_t^k \le z_t^{k*} \\ 0 & \text{в других случаях} \end{cases}$$

Как и в предыдущем случае, требуется нормализующий член η, поскольку экспонента ограничена интервалом $[0; z_t^{k*}]$. В силу того, что суммарная вероятность на этом интервале задана в виде

(6.8)

$$\begin{aligned} \int_0^{z_t^{k*}} \lambda_{short}\, e^{-\lambda_{short} z_t^k} dz_t^k &= -e^{-\lambda_{short} z_t^{k*}} + e^{-\lambda_{short} 0} \\ &= 1 - e^{-\lambda_{short} z_t^{k*}} \end{aligned}$$

значение η можно вывести, как:

(6.9)
$$\eta = \frac{1}{1 - e^{-\lambda_{short} z_t^{k*}}}$$

На Рис. 6.3b эта плотность показана в графическом виде. Очевидно, что она экспоненциально уменьшается с расстоянием z_t^k.

3. Отказы. Иногда все препятствия одновременно "исчезают". Это часто происходит, например, с датчиками сонаров в результате зеркальных отражений. Отказы также случаются с лазерными дальномерами при сканировании чёрных, светопоглощающих объектов или же с некоторыми лазерными системами, которые неустойчиво работают при ярком солнечном свете.

ОТКАЗ ДАТЧИКА

Типичным результатом отказа датчика является возвращение показаний максимального расстояния, то есть датчик возвращает z_{max}. Такие события происходят достаточно часто, поэтому необходимо явно учитывать измерения максимального расстояния в модели измерений.

Смоделируем этот случай размещением точечной вероятности с центром в z_{max}:

(6.10)
$$p_{max}(z_t^k | x_t, m) = I(z = z_{max}) = \begin{cases} 1 & \text{если } z = z_{max} \\ 0 & \text{в других случаях} \end{cases}$$

Здесь была определена индикаторная функция, которая принимает значение 1, если аргумент имеет значение True, и 0 – в противном случае. Технически, p_{max} не определяет функцию плотности вероятности, поскольку p_{max} является дискретным распределением. Однако, это не должно нас волновать, ведь несуществующая функция плотности на математическую модель оценки вероятности измерения датчика не влияет. На наших схемах просто изобразим p_max в виде очень узкого равномерного распределения с центром в z_{max}, представив, что такая плотность существует.

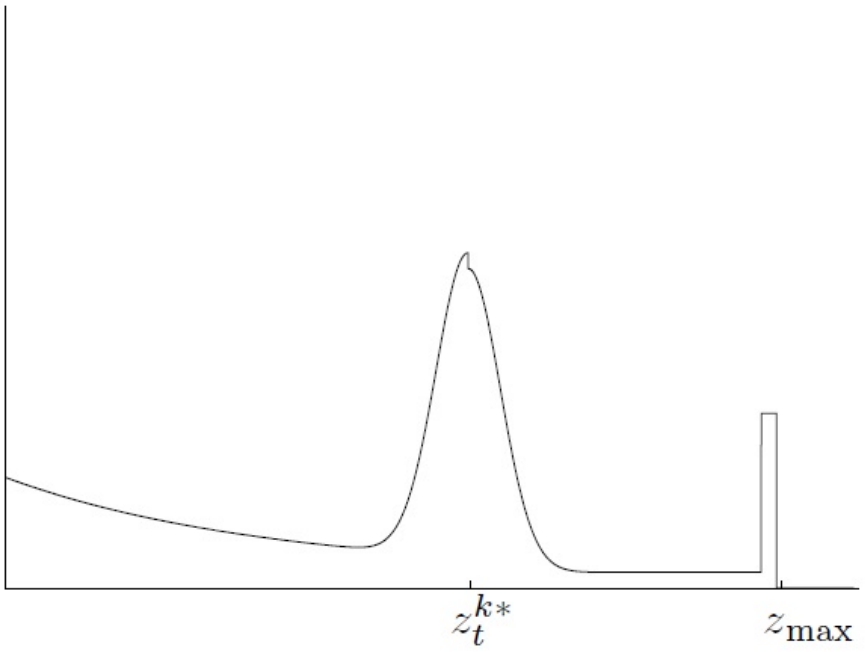


Рис. 6.4 «Псевдовероятность» типичной смеси распределений $p(z_t^k|x_t, m)$.

НЕОБЪЯСНИМЫЕ ИЗМЕРЕНИЯ

4. Случайные измерения. Наконец, дальномеры время от времени возвращают совершенно *необъяснимые измерения.* Так, сонары часто возвращают фантомные показания из-за отражения от стен или при перекрёстных наводках между различными датчиками. Для упрощения, такие измерения будут моделированы, используя равномерное распределение по всему диапазону измерений датчика $[0; z_{max}]$:

(6.11)
$$p_{rand}(z_t^k|x_t, m) = \begin{cases} \frac{1}{z_{max}} & \text{если } 0 \leq z_t^k \leq z_{max} \\ 0 & \text{в других случаях} \end{cases}$$

На Рис. 6.3d показана плотность распределения p_{rand}.

Эти четыре разных распределения учтены в виде взвешенной средней, определённой параметрами z_{hit}, z_{short}, z_{max} и z_{rand}, где $z_{hit} + z_{short} + z_{max} + z_{rand} = 1$.

(6.12)
$$p(z_t^k|x_t, m) = \begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} p_{hit}(z_t^k|x_t, m) \\ p_{short}(z_t^k|x_t, m) \\ p_{max}(z_t^k|x_t, m) \\ p_{rand}(z_t^k|x_t, m) \end{pmatrix}$$

Типичная плотность вычисляется из линейной комбинации отдельных плотностей, показанных на Рис. 6.4 (с визуализацией распределения точечной вероятности p_max в виде небольшой однородной плотности). Как может заметить читатель, основные характеристики всех четырёх базовых моделей все ещё присутствуют в этой комбинированной плотности.

1: **Algorithm beam_range_finder_model** (z_t, x_t, m) :

2: $q = 1$
3: *Для* $k = 1$ *до* K *выполнять*
4: *вычислить* z_t^{k*} *для измерения* z_t^k *используя метод бросания лучей*
5: $p = z_{hit} \cdot p_{hit}(z_t^k | x_t, m) + z_{short} \cdot p_{short}(z_t^k | x_t, m)$
6: $+ z_{max} \cdot p_{max}(z_t^k | x_t, m) + z_{rand} \cdot p_{rand}(z_t^k | x_t, m)$
7: $q = q \cdot p$
8: *return* q

Таблица 6.1 Алгоритм для вычисления схожести сканирования расстояния z_t, допуская условную независимость между отдельными измерениями расстояния в проходе сканирования.

Модель дальномера реализована алгоритмом **beam_range_finder_model** в Таблице 6.1. На вход этой модели подаётся полный проход сканирования z_t, положение робота x_t, и карта m. Во внешнем цикле (в строках 2 и 7) правдоподобия отдельных лучей датчика z_t^k перемножаются, давая равенство (6.2). В строке 4 используется «метод бросания лучей» для вычисления расстояния отдельного измерения датчика с учётом шума. Вероятность каждого отдельного измерения расстояния z_t^k вычисляется в строке 5, где выполняется смешивание плотностей из (6.12). После прохода по всем измерениям датчика z_t^k в наборе z_t, алгоритм возвращает искомую вероятность $p(z_t | x_t, m)$.

6.3.2 Настройка внутренних параметров модели

Пока в нашем обсуждении не был освещён вопрос выбора различных параметров модели датчика. Эти параметры включают параметры смешивания z_{hit}, z_{short}, z_{max}, и z_{rand}, а также σ_{hit} и λ_{short}. Обозначим набор всех внутренних параметров через Θ. Очевидно, вероятность любого измерения датчика является функцией Θ. Поэтому, обсудим алгоритм для *настройки параметров модели*.

Одним из способов определения внутренних параметров является анализ данных. На Рис. 6.5 показаны две серии из 10000 измерений, полученных мобильным роботом при передвижении в среде обычного офиса.

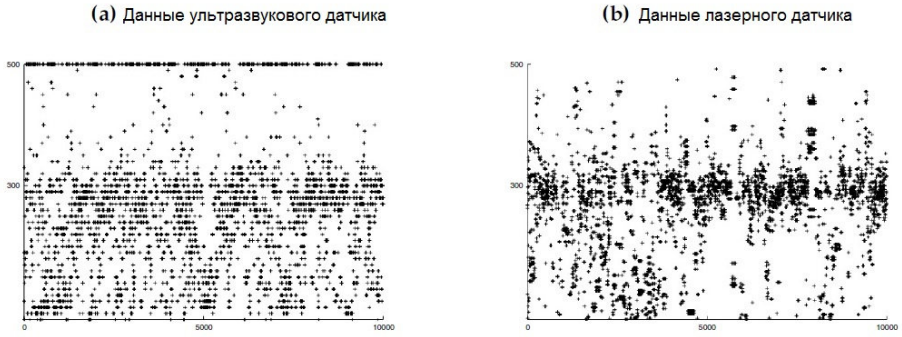


Рис. 6.5 Типичные данные, полученные с (a) датчика сонара и (b) лазерного датчика расстояния в офисе для «истинного» расстояния 300 см и максимального расстояния 500 см.

На обоих графиках изображены только измерения, предполагаемое значение которых приблизительно равно 3 метрам (между 2,9 м и 3,1 м). На левом графике показаны данные для датчика сонара, а на правом – аналогичные данные лазерного датчика. В обоих случаях по оси x показано число измерений (от 1 до 10000), а по оси y – дальность, измеренная датчиком.

Хотя большинство измерений обоих датчиков находится вблизи диапазона верных значений, поведение датчиков существенно различается. Ультразвуковой датчик, очевидно, гораздо больше подвержен шумам измерений и ошибкам обнаружения. Довольно часто он оказывается неспособен обнаружить имеющееся препятствие, возвращая максимальное расстояние. Напротив, лазерный дальномер значительно более точен, но и он, время от времени, возвращает неверные показания.

Полностью применимым на практике способом регулировки внутренних параметров Θ является их ручная установка таким образом, чтобы результирующая плотность была согласована с эмпирической картиной. Другим, более надёжным способом является изучение значений параметров на основе настоящих данных. Это достигается максимизацией правдоподобия опорного набора данных $Z = \{z_i\}$ относительно связанных положений $X = \{x_i\}$ и карты m, где каждая величина z_i представляет собой актуальное измерение, x_i - положение, из которого было получено измерение, а m - карту. Правдоподобие данных Z задано в виде

(6.13)
$$p(Z|X, m, \Theta)$$

Нашей целью является определение внутренних параметров Θ, максимизирующих это правдоподобие.

ОЦЕНКА МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ

Оценивающая функция или алгоритм, максимизации правдоподобия обычно называются *оценкой максимального правдоподобия*, ("ML estimator").

В Таблице 6.2 приводится алгоритм **learn_intrinsic_parameters** вычисления оценки максимального правдоподобия внутренних параметров.

1: **Algorithm learn_intrinsic_parameters** (Z, X, m) :

2: *повторять до удовлетворения критерия сходимости*
3: *для всех* z_i *в* Z *выполнять*
4: $\eta = [p_{hit}(z_i|x_i, m) + p_{short}(z_i|x_i, m) + p_{max}(z_i|x_i, m) + p_{rand}(z_i|x_i, m)]^{-1}$
5: *вычислить* z_i^*
6: $e_{i,hit} = \eta\, p_{hit}(z_i|x_i, m)$
7: $e_{i,short} = \eta\, p_{short}(z_i|x_i, m)$
8: $e_{i,max} = \eta\, p_{max}(z_i|x_i, m)$
9: $e_{i,rand} = \eta\, p_{rand}(z_i|x_i, m)$

10: $z_{hit} = |Z|^{-1} \sum_i e_{i,hit}$
11: $z_{short} = |Z|^{-1} \sum_i e_{i,short}$
12: $z_{max} = |Z|^{-1} \sum_i e_{i,max}$
13: $z_{rand} = |Z|^{-1} \sum_i e_{i,rand}$
14: $\sigma_{hit} = \sqrt{\frac{1}{\sum_i e_{i,hit}} \sum_i e_{i,hit}(z_i - z_i^*)^2}$
15: $\lambda_{short} = \frac{\sum_i e_{i,short}}{\sum_i e_{i,short} z_i}$
16: $return\, \Theta = \{z_{hit}, z_{short}, z_{max}, z_{rand}, \sigma_{hit}, \lambda_{short}\}$

Таблица 6.2 Алгоритм обучения на основе данных для внутренних параметров модели датчика, использующего лучи.

Как будет показано ниже, алгоритм является примером метода максимизации ожидания (expectation maximization -EM), итерационной процедуры оценки параметров максимального правдоподобия.

В начале работы алгоритма **learn_intrinsic_parameters** в Таблице 6.2 требуется качественная инициализация внутренних параметров σ_{hit} и λ_{short}. В строках с 3 по 9 оцениваются вспомогательные переменные $e_{i,xxx}$ - вероятности, что измерение z_i вызвано причиной "xxx", где "xxx" выбирается между четырьмя возможными шумами модели датчика: отражениями, короткими показаниями, максимальными показаниями и случайными шумами. Далее, в строках с 10 по 15, оцениваются внутренние параметры. Поскольку эти параметры являются функцией вычисленных ранее значений ожидания и их изменение вызывает изменение математического ожидания, алгоритм необходимо повторять. На практике процесс достаточно быстро завершаются, и, обычно, для получения хороших результатов хватает дюжины повторений.

На Рис. 6.6 в графическом виде показаны четыре примера данных и модели измерения максимального правдоподобия, вычисленные с помощью **learn_intrinsic_parameters**. В первой строке показаны приближения к данным, полученным с помощью ультразвукового датчика. Во второй строке приведены графики двух функций, сгенерированных из данных лазерного дальномера. Столбцы соответствуют различным «истинным» расстояниям. Данные организованы в виде гистограмм и легко заметить разницу между графиками. Чем меньше расстояние z_t^{k*}, тем точнее измерение. Для обоих датчиков ширина гауссовых функций меньше для более близких измерений, чем для более удалённых. Вдобавок, лазерный дальномер более точен по сравнению с ультразвуковым датчиком, что хорошо видно

из более узкого графика гауссиана и меньшего числа максимальных значений измерений. Другой важной особенностью является относительно высокое правдоподобие близких и случайных измерений. Эта большая по величине ошибка правдоподобия является, одновременно, и преимуществом, и недостатком. С одной стороны, количество информации при каждом опросе датчика уменьшается из-за малой разницы правдоподобия отражения и случайного измерения. С другой стороны, это делает модель менее подверженной не смоделированным *систематическим* искажениям, например, появлению людей, которые перекрывают путь роботу в течение длительного времени.

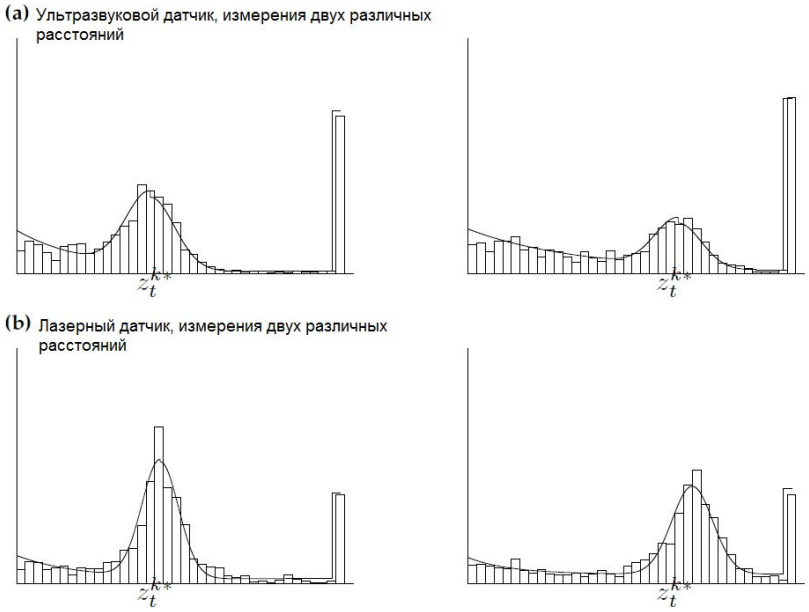


Рис. 6.6 Приближение модели на основе лучей для (a) данных сонара и (b) данных лазера. Модели датчиков, показанные слева были получены методом максимального правдоподобия наборов данных на Рис. 6.5.

На Рис. 6.7 показана в действии обученная модель датчика. На Рис. 6.7a показан проход сканирования шириной 180 градусов. Робот расположен в месте, предварительно полученном из карты занятости и его положение точно определено. На Рис. 6.7b изображена карта окружения с правдоподобностью $p(z_t|x_t, m)$ прохода сканирования расстояния, спроектированного в пространство $x - y$ (путём максимизации по ориентации Θ). Чем более тёмным цветом выделено местоположение, тем более оно вероятно. Как можно легко увидеть, все области с высоким правдоподобием расположены вдоль коридора. Неудивительно, поскольку отдельный проход сканирования лучше сохраняет геометрическую целостность в коридорах, нежели внутри комнат. Факт распределения массы вероятности вдоль коридора указывает на недостаточность отдельного прохода сканирования для определения точного положения робота. Это происходит, в основном, из-за симметричности коридора, а размещение апостериорного распределения в виде двух узких горизонтальных полос вызвано неизвестной ориентацией робота по направлению. Каждая полоса соответствует двум допустимым направлениям поворота робота.

6.3.3 Математический вывод модели на основе лучей

Для вывода оценки методом максимального правдоподобия будет полезно ввести вспомогательную переменную c_i, так называемую переменную соответствия. Каждая c_i может принимать одно из четырёх значений: близкое, максимальное, случайное расстояние и отражение, что соответствует четырём возможным механизмам, создающим измерение z_i.

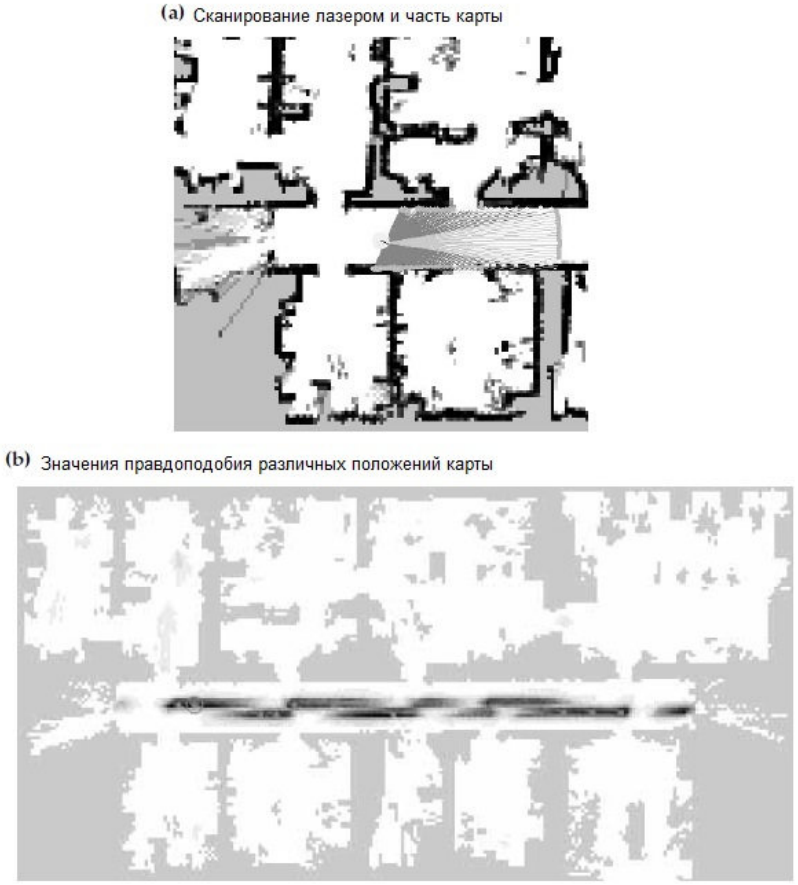


Рис. 6.7 Вероятностная модель восприятия: (a) Проход сканирования расстояния лазером, в проекции на предварительно созданную карту m. (b) Оценка правдоподобия $p(z_t|x_t, m)$, всех местоположений x_t, спроектированная на карту (показана серым). Чем темнее местоположение, тем выше вероятность $p(z_t|x_t, m)$.

Для начала разберём случай, когда c_i неизвестны, но известно, каким из четырёх указанных выше механизмов вызывается каждое измерение z_i. На основе значений c_i, можно разложить Z на четыре непересекающихся множества, Z_{hit}, Z_{short}, Z_{max}, и Z_{rand}, которые вместе и образуют полный набор данных Z. Оценка максимального правдоподобия для внутренних параметров z_{hit}, z_{short}, z_{max}, и z_{rand} представляет просто нормализованные соотношения:

(6.14)
$$\begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix} = |Z|^{-1} \begin{pmatrix} |Z_{hit}| \\ |Z_{short}| \\ |Z_{max}| \\ |Z_{rand}| \end{pmatrix}$$

Оставшиеся внутренние параметры σ_{hit} и λ_{short} получаются следующим образом. Для набора данных Z_{hit} получим из (6.5)

(6.15)
$$\begin{aligned} p(Z_{hit}|X, m, \Theta) &= \prod_{z_i \in Z_{hit}} p_{hit}(z_i|x_i, m, \Theta) \\ &= \prod_{z_i \in Z_{hit}} \frac{1}{\sqrt{2\pi\sigma_{hit}^2}} e^{-\frac{1}{2}\frac{(z_i - z_i^*)^2}{\sigma_{hit}^2}} \end{aligned}$$

Здесь z_i^* - «истинное» расстояние, вычисленное из положения x_i и карты m. Классическим способом оценки правдоподобия является максимизация логарифма правдоподобия, а не самого значения правдоподобия. Логарифм является строго монотонной функцией, поэтому максимум логарифма правдоподобия является и максимумом самого правдоподобия. Логарифм правдоподобия задан в виде

(6.16)
$$\log p(Z_{hit}|X, m, \Theta) = \sum_{z_i \in Z_{hit}} \left[-\frac{1}{2} \log 2\pi\sigma_{hit}^2 - \frac{1}{2} \frac{(z_i - z_i^*)^2}{\sigma_{hit}^2} \right]$$

и легко преобразуется следующим образом

(6.17)
$$\begin{aligned} &\log p(Z_{hit}|X, m, \Theta) \\ &\quad = -\frac{1}{2} \sum_{z_i \in Z_{hit}} \left[\log 2\pi\sigma_{hit}^2 + \frac{(z_i - z_i^*)^2}{\sigma_{hit}^2} \right] \\ &\quad = -\frac{1}{2} \left[|Z_{hit}| \log 2\pi + 2|Z_{hit}| \log \sigma_{hit} + \sum_{z_i \in Z_{hit}} \frac{(z_i - z_i^*)^2}{\sigma_{hit}^2} \right] \\ &\quad = \text{const.} - |Z_{hit}| \log \sigma_{hit} - \frac{1}{2\sigma_{hit}^2} \sum_{z_i \in Z_{hit}} (z_i - z_i^*)^2 \end{aligned}$$

Производная этого выражения по внутренним параметрам σ_{hit} выглядит следующим образом:

(6.18)
$$\frac{\partial \log p(Z_{hit}|X, m, \Theta)}{\partial \sigma_{hit}} = -\frac{|Z_{hit}|}{\sigma_{hit}} + \frac{1}{\sigma_{hit}^3} \sum_{z_i \in Z_{hit}} (z_i - z_i^*)^2$$

Максимум логарифма правдоподобия получается установкой этой производной в нуль. Отсюда и решение оценки методом максимального правдоподобия.

(6.19)
$$\sigma_{hit} = \sqrt{\frac{1}{|Z_{hit}|} \sum_{z_i \in Z_{hit}} (z_i - z_i^*)^2}$$

Оценка оставшегося внутреннего параметра λ_{short} выполняется практически таким же образом. Апостериорное распределение Z_{short} задано в виде

(6.20)

$$\begin{aligned} p(Z_{short}|X, m, \Theta) &= \prod_{z_i \in Z_{short}} p_{short}(z_i|x_i, m) \\ &= \prod_{z_i \in Z_{short}} \lambda_{short}\, e^{-\lambda_{short} z_i} \end{aligned}$$

Логарифм выражения

(6.21)

$$\begin{aligned} \log p(Z_{short}|X, m, \Theta) &= \sum_{z_i \in Z_{short}} \log \lambda_{short} - \lambda_{short} z_i \\ &= |Z_{short}| \log \lambda_{short} - \lambda_{short} \sum_{z_i \in Z_{short}} z_i \end{aligned}$$

Первая производная этого выражения по отношению к внутреннему параметру λ_{short} выглядит следующим образом:

(6.22)

$$\frac{\partial \log p(Z_{short}|X, m, \Theta)}{\partial \lambda_{short}} = \frac{|Z_{short}|}{\lambda_{short}} - \sum_{z_i \in Z_{short}} z_i$$

Установка производной в нуль даёт оценку максимального внутреннего правдоподобия λ_{short}

(6.23)

$$\lambda_{short} = \frac{|Z_{short}|}{\sum_{z_i \in Z_{short}} z_i}$$

Для вычисления производной подразумевается знание параметров c_i. Обобщим этот случай для ситуации, когда c_i неизвестно. Как будет показано, результирующая задачи оценка максимизации правдоподобия не имеет решения в закрытом виде. Однако, можно предложить метод вычисления ожидания c_i в цикле из двух шагов, с последующим вычислением внутренних параметров модели по этим ожиданиям.

АЛГОРИТМ МАКСИМИЗАЦИИ ОЖИДАНИЯ

Как было отмечено, результирующий алгоритм является экземпляром *алгоритма максимизации ожидания* обычно сокращаемом до EM.

Чтобы вывести EM, будет уместно сначала определить правдоподобие данных Z:

(6.24)

$$\begin{aligned} &\log p(Z|X, m, \Theta) \\ &\quad = \sum_{z_i \in Z} \log p(z_i|x_i, m, \Theta) \\ &\quad = \sum_{z_i \in Z_{hit}} \log p_{hit}(z_i|x_i, m) + \sum_{z_i \in Z_{short}} \log p_{short}(z_i|x_i, m) \\ &\quad + \sum_{z_i \in Z_{max}} \log p_{max}(z_i|x_i, m) + \sum_{z_i \in Z_{rand}} \log p_{rand}(z_i|x_i, m) \end{aligned}$$

Это выражение может быть переписано с использованием переменных c_i:

(6.25)

$$\begin{aligned}\log p(Z|X,m,\Theta) = \sum_{z_i \in Z} & I(c_i = \text{hit}) \log p_{hit}(z_i|x_i,m) \\ & + I(c_i = \text{short}) \log p_{short}(z_i|x_i,m) \\ & + I(c_i = \text{max}) \log p_{max}(z_i|x_i,m) \\ & + I(c_i = \text{rand}) \log p_{rand}(z_i|x_i,m)\end{aligned}$$

где I – индикаторная функция. Поскольку значения c_i неизвестны, остаётся только интегрировать их. Другими словами, с помощью EM максимизируется ожидание $E[\log\, p(Z|X,m,\Theta)]$ по неизвестным переменным c_i:

(6.26)

$$\begin{aligned}&E[\log p(Z|X,m,\Theta)] \\ &\quad = \sum_i p(c_i = \text{hit}) \log p_{hit}(z_i|x_i,m) + p(c_i = \text{short}) \log p_{short}(z_i|x_i,m) \\ &\quad + p(c_i = \text{max}) \log p_{max}(z_i|x_i,m) + p(c_i = \text{rand}) \log p_{rand}(z_i|x_i,m) \\ &\quad =: \sum_i e_{i,hit} \log p_{hit}(z_i|x_i,m) + e_{i,short} \log p_{short}(z_i|x_i,m) \\ &\quad + e_{i,max} \log p_{max}(z_i|x_i,m) + e_{i,rand} \log p_{rand}(z_i|x_i,m)\end{aligned}$$

Переменная e определяется указанным выше способом. Это выражение максимизируется в два шага. На первом шаге будем считать, что внутренние параметры σ_{hit} и λ_{short} заданы изначально, и вычислим математическое ожидание по переменной c_i.

(6.27)
$$\begin{pmatrix} e_{i,hit} \\ e_{i,short} \\ e_{i,max} \\ e_{i,rand} \end{pmatrix} := \begin{pmatrix} p(c_i = \text{hit}) \\ p(c_i = \text{short}) \\ p(c_i = \text{max}) \\ p(c_i = \text{rand}) \end{pmatrix} = \eta \begin{pmatrix} p_{hit}(z_i|x_i,m) \\ p_{short}(z_i|x_i,m) \\ p_{max}(z_i|x_i,m) \\ p_{rand}(z_i|x_i,m) \end{pmatrix}$$

Нормализующий член задан в виде

(6.28)

$$\eta = [p_{hit}(z_i|x_i,m) + p_{short}(z_i|x_i,m) + p_{max}(z_i|x_i,m) + p_{rand}(z_i|x_i,m)]^{-1}$$

Этот шаг называется "шаг-E", указывая на вычисление математического ожидания латентных переменных c_i. Оставшийся шаг теперь очевиден, поскольку ожидания выражают зависимости между разными компонентами модели датчика. Во-первых, заметим, что параметры смеси ML представляют собой просто нормализованные ожидания

(6.29)
$$\begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix} = |Z|^{-1} \sum_i \begin{pmatrix} e_{i,hit} \\ e_{i,short} \\ e_{i,max} \\ e_{i,rand} \end{pmatrix}$$

Параметры ML σ_{hit} и λ_{short} получаются аналогично, заменой жёсткого присваивания значений в (6.19) и (6.23) присваиванием взвешенных значений.

(6.30)
$$\sigma_{hit} = \sqrt{\frac{1}{\sum_{z_i \in Z} e_{i,hit}} \sum_{z_i \in Z} e_{i,hit}(z_i - z_i^*)^2}$$

и

(6.31)
$$\lambda_{short} = \frac{\sum_{z_i \in Z} e_{i,short}}{\sum_{z_i \in Z} e_{i,short} z_i}$$

6.3.4 Практические соображения

На практике, вычисление плотностей всех показаний датчиков может быть вычислительно затруднено. Например, лазерные сканеры расстояния часто возвращают сотни значений за проход, выполняя несколько проходов в секунду. Поскольку для каждого луча сканирования необходимо выполнить операцию «бросания луча» в каждое возможное местоположение, интеграцию текущего прогноза в проход сканирования в реальном времени выполнить невозможно. Обычным способом решения задачи является включение в расчёт только ограниченного набора измерений (например, восьми равномерно распределенных измерений на проход сканирования лазера вместо 360). Такой подход имеет важное дополнительное преимущество. Поскольку соседние лучи прохода сканирования часто не являются независимыми, процесс оценки состояния становится менее подвержен коррелированному шуму соседних измерений.

При выраженной зависимости между соседними измерениями применение модели ML может привести робота к излишней самоуверенности и совершенно неоптимальным результатам оценки местоположения. Один простой приём, позволяющий этого избежать, состоит в замене $p(z_t^k|x_t, m)$ более «слабой» версией $p(z_t^k|x_t, m)^\alpha$ при $\alpha < 1$. Идея состоит в уменьшении, на коэффициент α, информации, извлечённой из измерения датчика (логарифм этой вероятности задан в виде $\alpha \log p(z_t^k|x_t, m)$). Другая возможность, которая будет только упомянута здесь, это выяснение внутренних параметров в контексте приложения. Например, в мобильной локализации возможно обучить модель внутренних параметров градиентным спуском для получения хороших результатов локализации в течение нескольких тактов времени. Такая методология с несколькими шагами существенно отличается от метода оценки максимального правдоподобия, описанного выше. В практическом применении этот метод может дать отличные результаты. (Thrun, 1998a).

Самой затратной, в вычислительном смысле, частью моделей на основе лучей является операция бросания луча. Затраты на вычисление $p(z_t|x_t, m)$ могут быть существенно уменьшены предварительным кэшированием результатов работы алгоритма бросания лучей и хранением их в памяти, заменив, таким образом, операцию бросания лучей значительно более быстрым проходом по таблице. Очевидной реализацией этой идеи является разбиение пространства состояний на трехмерную сетку с мелкой ячейкой и вычисление диапазонов z_t^{k*} для каждой ячейки сети. Эта идея уже была описана в

Главе 4.1 и, в зависимости от разрешения сетки, требования к памяти могут оказаться весьма существенными. В задаче локализации мобильного робота пространство с разрешением сетки 15 см на 2 градуса хорошо показывает себя в задачах локализации в замкнутом пространстве. Такая карта хорошо помещается в памяти средних по мощности компьютеров, показывая производительность на порядок лучше бросания лучей в реальном времени.

6.3.5 Ограничения модели лучей

Модель датчика на основе лучей, хотя и пересекается с геометрией и физикой дальномеров, имеет два важных недостатка.

В частности, модель на основе лучей недостаточно гладкая. В загромождённых средах с множеством малых препятствий распределение $p(z_t^k|x_t, m)$ может быть очень негладким по x_t. Для примера возьмём среду с множеством стульев и столов (скажем, конференц-зал). Робот, показанный в Главе 1, сможет воспринимать ножки этих препятствий. Очевидно, небольшие изменения положения робота x_t могут оказать огромное влияние на верное определение расстояния лучом датчика. В результате, модель измерения $p(z_t^k|x_t, m)$ сильно прерывиста по x_t. В частности, затрагивается параметр направления поворота θ_t, поскольку малые изменения направления могут вызвать большие смещения в пространстве $x-y$ на расстояниях измерения.

Недостаток гладкости имеет два следствия. Во-первых, любое примерное представление подвержено опасности потери корректного состояния, поскольку ближайшие состояния могут иметь радикально различные апостериорные правдоподобия. Это накладывает ограничения на точность аппроксимации, которые могут привести к увеличению результирующей апостериорной ошибки. Во-вторых, методы поиска экстремума для нахождения наиболее вероятного состояния в условиях с большим количеством локальных минимумов в негладких моделях страдают от проблемы локального минимума.

Модель на основе лучей также является вычислительно затратной. Оценка $p(z_t^k|x_t, m)$ каждого единичного измерения датчика z_t^k включает бросание лучей, что требует достаточной вычислительной мощности. Как было отмечено выше, задача может быть частично упрощена предварительным вычислением диапазонов на дискретной сетке пространства положений. Такой подход смещает акцент вычислений на начальную оффлайновую фазу, ускоряя работу алгоритма. Однако, результирующие таблицы очень велики, поскольку покрывают значительные участки трехмерного пространства. В силу этого, предварительный расчёт расстояний является вычислительно затратным и требует значительного объёма памяти.

6.4 Поля правдоподобия для датчиков расстояния

6.4.1 Общий алгоритм

ПОЛЕ ПРАВДОПОДОБИЯ

Опишем альтернативную модель под названием *поле правдоподобия*, которая обходит эти ограничения, но внятного физического объяснения не имеет. Фактически, это специализированный алгоритм, который вовсе не обязательно вычисляет условную вероятность относительно применимой генеративной модели физики датчиков, но на практике работает хорошо. Результирующие апостериорные распределения значительно более гладкие даже в загромождённом пространстве, а вычисления – более эффективные.

Ключевая идея состоит в проекции конечных точек прохода сканирования датчика z_t в глобальное координатное пространство карты. Чтобы это сделать, необходимо определить, как именно в глобальной системе координат расположена локальная система отсчёта робота, откуда начинается луч датчика z_k и куда он направлен. Как обычно, пусть $x_t = (x\, y\, \theta)^T$ определяет положение робота в момент времени t. Сохраняя двухмерное представление окружающего мира, определим фиксированное относительное положение датчика робота, локальную координатную систему $(x_{k,sens}\, y_{k,sens})^T$, угловое положение луча датчика относительно направления робота $\theta_{k,sens}$. Эти значения зависят от типа датчика. Конечная точка измерения z_t^k проектируется в глобальную систему координат с помощью очевидного тригонометрического преобразования.

(6.32)

$$\begin{pmatrix} x_{z_t^k} \\ y_{z_t^k} \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{k,sens} \\ y_{k,sens} \end{pmatrix} + z_t^k \begin{pmatrix} \cos(\theta + \theta_{k,sens}) \\ \sin(\theta + \theta_{k,sens}) \end{pmatrix}$$

Эти координаты имеют смысл только для измерения, при котором датчик обнаруживает препятствие. Если датчик расстояния выдаёт максимальное значение $z_t^k = z_{max}$, координаты не имеют значения в физическом мире (хотя измерение и содержит информацию). В модели измерения на основе поля правдоподобия показания максимального расстояния просто отбрасываются.

Аналогично модели на основе лучей, обсуждаемой выше, будут приниматься во внимание три источника зашумления и неопределённости:

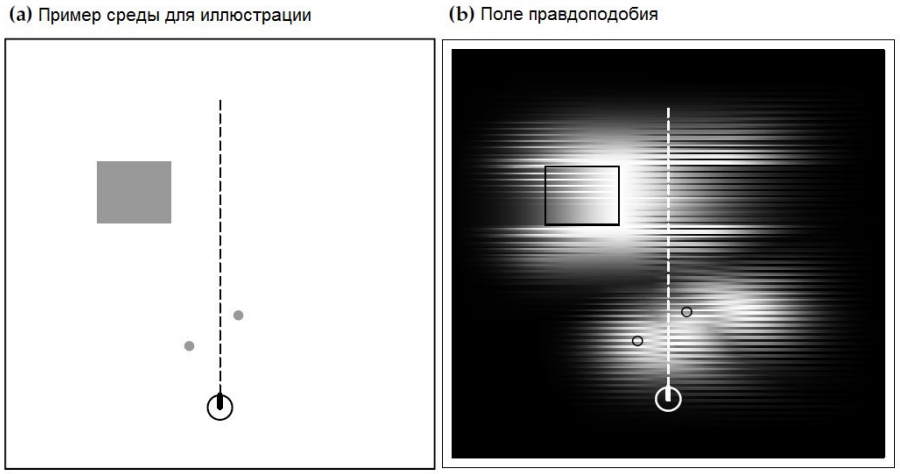


Рис. 6.8 (a) Модельная среда с тремя препятствиями (выделены серым). Робот, расположенный внизу схемы, получает измерение z_t^k, обозначенное пунктиром. (b) Поле правдоподобия для данной конфигурации препятствий: чем темнее местоположение, тем менее вероятно обнаружение препятствия в этой точке. Вероятность $p(z_t^k|x_t, m)$ для отдельного луча датчика показана на Рис. 6.9.

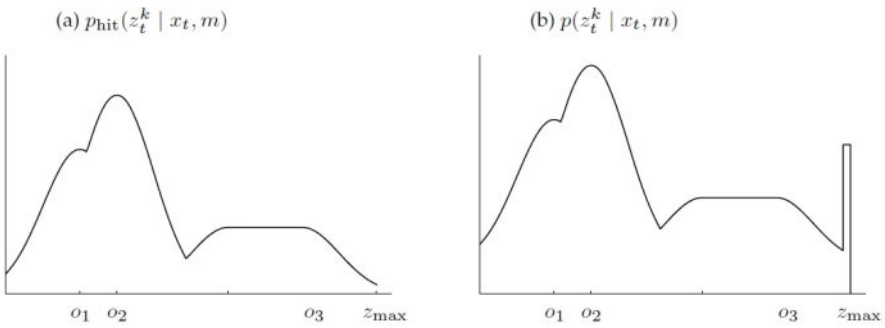


Рис. 6.9 (a) Вероятность $p_{hit}(z_t^k)$ как функция измерения z_t^k, для ситуации, показанной на Рис. 6.8. Здесь луч датчика проходит по трём препятствиям с ближайшими точками o_1, o_2, и o_3, соответственно. (b) Вероятность датчика $p(z_t^k|x_t, m)$, полученная для ситуации на Рис. 6.8, двумя дополнительными равномерными распределениями.

1. Шумы измерения. Шум при измерении смоделирован гауссовыми распределениями. В пространстве $x - y$ для этого требуется найти на карте ближайшее препятствие. Для начала просто определим евклидово расстояние между координатами измерения $(x_{z_t^k}\, y_{z_t^k})^T$ и ближайшим объектом на карте m. Тогда вероятность измерения датчика будет задано гауссовой функцией с нулевым центром, учитывающей шумы датчика:

(6.33)
$$p_{hit}(z_t^k|x_t, m) = \varepsilon_{\sigma_{hit}}(dist)$$

На Рис. 6.8a показана карта, а на Рис. 6.8b - соответствующее гауссово правдоподобие для точек измерения $(x_{z_t^k}\, y_{z_t^k})^T$ в двухмерном пространстве. Чем более светлым цветом помечено местоположение, тем более вероятно измерение объекта с помощью датчика расстояния. Теперь плотность p_{hit} получается пересечением (и нормализацией) поля правдоподобия осью датчика, обозначенной пунктиром 6.8. Результирующая функция показана на Рис. 6.9a.

2. Отказы. Как и ранее, предположим, что показания максимального расстояния имеют выраженное большое правдоподобие и снова смоделируем это с помощью точечного распределения p_{max}.

3. Необъяснимые случайные измерения. Наконец, для моделирования случайного шума измерения используется однородное распределение p_{rand}.

Аналогично модели датчика на основе лучей, искомая вероятность $p(z_t^k|x_t, m)$ объединяет все три распределения:

(6.34)
$$z_{hit} \cdot p_{hit} + z_{rand} \cdot p_{rand} + z_{max} \cdot p_{max}$$

используя уже знакомое смешивание весов z_{hit}, z_{rand}, и z_{max}. На Рис. 6.9b показан пример результирующего распределения $p(z_t^k|x_t, m)$ вдоль луча измерения. Будет легко показать, что это распределение объединяет p_{hit}, как показано на Рис. 6.9a, с распределениями p_{max} и p_{rand}. Большая часть

сказанного о настройке параметров смешивания справедливо и для новой модели датчика. Параметры можно настраивать вручную или использовать обученный алгоритм смешивания. Отображение наподобие показанного на Рис. 6.8b, показывает правдоподобие обнаружения препятствия в виде функции глобальных координат $x - y$ и называется *полем правдоподобия*.

В Таблице 6.3 предлагается алгоритм вычисления вероятности измерений с использованием поля правдоподобия. Читателю уже должен быть знаком внешний цикл, в котором выполняется перемножение отдельных значений $p(z_t^k|x_t, m)$, подразумевая независимость между шумами в разных лучах датчика. В строке 4 проверяется не является ли измерение дальности максимальным, и, если да, такое значение отбрасывается. В строках с 5 по 8 происходит интересное преобразование. Сначала вычисляется расстояние до ближайшего препятствия (строка 7), а затем результирующее правдоподобие получается в строке 8 смешением нормального и равномерного распределений. Как и прежде, функцией $\mathbf{prob}(dist, \sigma_{hit})$ вычисляется вероятность $dist$ под гауссовой функцией с нулевым математическим ожиданием и стандартным отклонением σ_{hit}.

1: **Algorithm likelihood_field_range_finder_model** (z_t, x_t, m) :

2: $q = 1$
3: *для всех* k *выполнить*
4: $\quad if\, z_t^k \neq z_{max}$
5: $\quad\quad x_{z_t^k} = x + x_{k,sens}\cos\theta - y_{k,sens}\sin\theta + z_t^k\cos(\theta + \theta_{k,sens})$
6: $\quad\quad y_{z_t^k} = y + y_{k,sens}\cos\theta + x_{k,sens}\sin\theta + z_t^k\sin(\theta + \theta_{k,sens})$
7: $\quad\quad dist = \min_{x',y'}\left\{\sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} | \langle x', y'\rangle \text{ занятых в } m\right\}$
8: $\quad\quad q = q \cdot (z_{hit} \cdot \mathbf{prob}(dist, \sigma_{hit}) + \frac{z_{random}}{z_{max}})$
9: *return* q

Таблица 6.3 Алгоритм вычисления правдоподобия датчика расстояния на основе евклидова расстояния до ближайшего соседа. Функция $\mathbf{prob}(dist, \sigma_{hit})$ вычисляет вероятность расстояния под гауссовым распределением с нулевым математическим ожиданием и стандартным отклонением σ_{hit}

Поиск ближайшего соседа на карте (строка 7) является самой затратной операцией в алгоритме **likelihood_field_range_finder_model**. Для ускорения поиска целесообразно предварительно вычислить поле правдоподобия, чтобы вычисление вероятности измерения сводилось к преобразованию координат с последующим просмотром таблицы. Конечно, при использовании дискретной сетки результат поиска достаточно приблизителен, поэтому может вернуть неверные координаты препятствия. Однако, эффект вероятности $p(z_t^k|x_t, m)$ обычно мал даже для достаточно грубых сеток.

6.4.2 Обобщения

Ключевым преимуществом модели поля правдоподобия над моделью на основе лучей, обсуждаемой раньше, является её гладкость. В силу гладкости функции евклидова расстояния, малые измерения положения робота

x_t оказывают лишь небольшой эффект на результирующее распределение $p(z_t^k|x_t, m)$. Другим ключевым преимуществом является выполнение предварительного расчёта в 2D вместо 3D, что уменьшает размер данных.

Такая модель имеет три главных недостатка. Во-первых, она неспособна явно моделировать людей и другие динамические процессы, способные вызвать слишком малые показания. Во-вторых, датчики в ней воспринимаются, как если бы они могли «видеть сквозь стены». Это происходит в результате замены операции бросания лучей функцией ближайшего соседа, неспособной определить пересечение пути к точке с препятствием. В-третьих, такой подход не принимает во внимание неопределённость карты. В частности, невозможно обработать неизвестные области, для которых карта не определена или же полностью неизвестна.

Рис. 6.10 (a) Карта сетки занятости технического музея Сан-Хосе, (b) предварительное вычисленное поле правдоподобия.

Основной алгоритм **likelihood_field_range_finder_model** может быть обобщён для минимизации эффекта этих ограничений. Например, возможно отсортировать значения занятости на карте по трём категориям: *занято*, *свободно*, и *неизвестно*, а не только по первым двум. Когда измерение датчика z_t^k попадает в категорию "неизвестно", ее вероятность $p(z_t^k|x_t, m)$ считается постоянной со значением $\frac{1}{z_{max}}$. Результирующая вероятностная модель достаточно груба, поскольку подразумевает, что в неизвестном пространстве каждое измерение датчика равновероятно.

На Рис. 6.10 показана карта и соответствующее ей поле правдоподобия. Как и прежде, выделение серым цветом местоположения $x - y$ обозначает правдоподобие получения показаний датчика в данной точке. Читатель может заметить, что расстояние до ближайшего препятствия применимо только *внутри* карты, что соответствует обследованной территории. Вне карты правдоподобность $p(z_t^k|x_t, m)$ постоянна. В целях вычислительной эффективности полезно предварительно вычислить ближайшего соседа, используя мелкую двухмерную сетку.

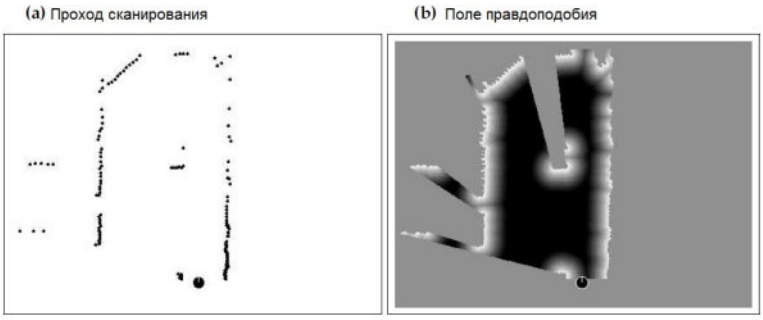


Рис. 6.11 (a) Проход сканирования при виде сверху. Робот расположен в нижней части чертежа, и выполняет сканирование близлежащей области в 180 точках впереди робота. (b) На основе прохода сканирования генерируется функция правдоподобия. Чем темнее область, тем меньше правдоподобие обнаружения объекта в этой точке. Обратим внимание, что занятые области белые, поэтому штраф не накладывается.

Поля правдоподобия по выделенному пространству могут также определяться для последнего прохода, который, фактически, и определяет карту. На Рис 6.11 показано такое поле правдоподобия. Оно играет важную роль в методах совмещениях отдельных проходов сканирования.

6.5 Модели измерений на основе корреляции

В литературе описаны несколько моделей датчиков расстояния, измеряющих корреляцию между измерением и картой.

НАЛОЖЕНИЕ КАРТ

Популярный метод называется *наложением карт*. Наложение карт требует методов, которые будут описаны позже, в частности, возможности превращать данные проходов сканирования в карты занятости. Обычно, при наложении карт небольшое число последовательных проходов сканирования преобразуются в локальные карты, обозначаемые m_{local}. На Рис. 6.12 показана такая локальная карта, здесь в виде карты сетки занятости. В модели измерения датчика сравниваются *локальная карта* m_{local} и глобальная карта m. Чем более схожи m и m_{local}, тем больше $p(m_{local}|x_t, m)$. Поскольку локальная карта отображается относительно положения робота, это сравнение требует преобразования ячеек локальной карты в координатную сетку глобальной карты.

Такое преобразование может быть выполнено аналогично выражению преобразования координат (6.32), выполненное для измерений датчиков, используемых в модели поля правдоподобия. Если робот находится в положении x_t, определим ячейку сетки на локальной карте, соответствующую глобальным координатам $(x\, y)^T$, как $m_{x,y,local}(x_t)$.

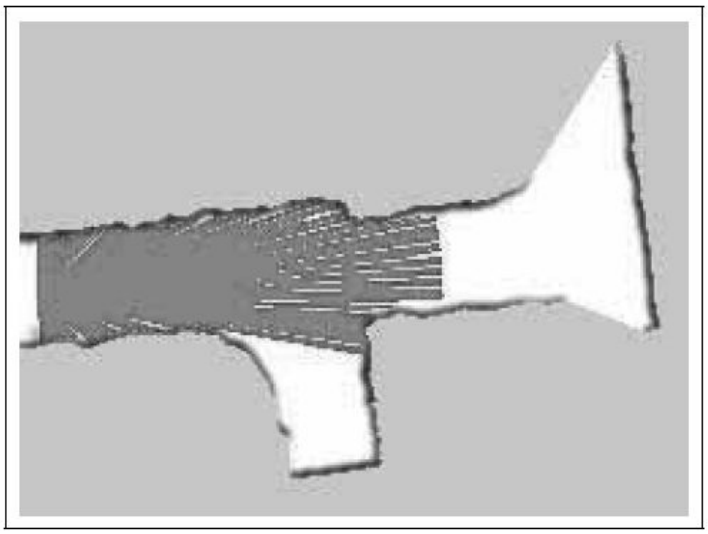


Рис. 6.12 Пример локальной карты, сгенерированной из 10 измерений расстояния, из которых показано одно.

Как только обе карты приведены к эталонной системе отсчёта, их можно сравнить с помощью функции корреляции карты, определённой следующим образом:

(6.35)

$$\rho_{m,m_{local},x_t} = \frac{\sum_{x,y}(m_{x,y} - \bar{m}) \cdot (m_{x,y,local}(x_t) - \bar{m})}{\sqrt{\sum_{x,y}(m_{x,y} - \bar{m})^2 \sum_{x,y}(m_{x,y,local}(x_t) - \bar{m})^2}}$$

Здесь сумма оценивается для всех ячеек, определённых на обеих картах, а $\bar{m}$ означает среднее значение по картам:

(6.36)

$$\bar{m} = \frac{1}{2N} \sum_{x,y}(m_{x,y} + m_{x,y,local})$$

Здесь N определяет количество перекрывающихся элементов между локальной и глобальной картами. Масштаб корреляции ρ_{m,m_{local},x_t} меняется в диапазоне ± 1. Наложение карт интерпретирует значение

(6.37)

$$p(m_{local}|x_t, m) = \max\{\rho_{m,m_{local},x_t}, 0\}$$

как вероятность локальной карты, наложенная на глобальную карту m и положение робота x_t. Если локальная карта сгенерирована из одиночного сканирования расстояния z_t, эта вероятность заменяет вероятность измерения $p(z_t|x_t, m)$.

Наложение карт имеет ряд особенностей. Аналогично модели поля правдоподобия, его легко вычислить, хотя, в результате, не получается гладких распределений вероятности параметра положения x_t. Одним из способов приблизительно оценить поле правдоподобия (и достичь гладкости) является свёртка карты m гауссовым ядром сглаживания, и выполнение наложения карт для уже сглаженной карты.

Ключевым преимуществом наложения карт над полями правдоподобия является явный учёт свободного пространства в сравнении двух карт. Метод поля правдоподобия учитывает только конечную точку сканирования, которая, по определению, соответствует занятому пространству (или шумам). С другой стороны, многие методы строят локальных карт за пределами досягаемости датчиков. Например, многие методы строят круговые карты вокруг робота, устанавливая значение вероятности 0,5 для зоны за пределами измерений датчиков. В таких случаях имеется опасность, что в результате наложения карт будут учтены зоны за пределами текущего радиуса измерений, как если бы датчик «видел сквозь стены». Такие побочные эффекты присутствуют в некоторых реализациях наложения карт.

Ещё одним недостатком наложения карт является отсутствие внятного физического обоснования. Корреляции представляют собой нормализованное квадратичное расстояние между объектами карт, а не шумовые характеристики датчиков расстояния.

6.6 Модели измерений на основе признаков

6.6.1 Извлечение признаков

ПРИЗНАКИ

Все модели датчиков, обсуждаемые до текущего момента, основаны на исходных измерениях. Альтернативным подходом является извлечение признаков из измерений. Если определить механизм извлечения признаков в виде функции f, признаки, извлечённые из измерения расстояния, будут заданы как $f(z_t)$. Большинство выделителей признаков извлекают лишь небольшое количество признаков из измерений датчика высокой размерности. Главным преимуществом такого подхода является колоссальное уменьшение вычислительной сложности. Выполнение оценок в пространстве состояний высокой размерности может оказаться весьма затратным, и подобные операции в пространстве признаков с малой размерностью на порядки эффективнее.

Обсуждение отдельных алгоритмов извлечения признаков не входит в область описания книги. В литературе предложен широкий набор признаков для самых различных датчиков. Для датчиков расстояния обычно определяют линии, углы или локальные минимумы сканирования расстояния, соответствующие стенам, углам или отдельным выделяющимся объектам, таким, как стволы деревьев. Использование камер для навигации относится к области компьютерного зрения, развитие которой породило множество методов извлечения признаков из изображений с камеры. Популярные признаки для изображений включают грани, углы, контрастные узоры и другие чётко различимые элементы. В робототехнике в качестве признаков часто применяются местоположения, например, проходы и пересечения.

6.6.2 Измерения на основе ориентиров

ОРИЕНТИРЫ

Во многих областях робототехники признаки соответствуют чётко различимым объектам физического мира. Например, в замкнутых помещениях признаками могут быть дверные косяки, подоконники, а на открытом пространстве – стволы деревьев или углы зданий. В робототехнике принято называть такие физические объекты *ориентирами*, чтобы показать, что они используются для навигации робота.

ДАТЧИК РАССТОЯНИЯ И НАПРАВЛЕНИЯ

Наиболее распространённая модель для обработки ориентиров подразумевает, что датчик способен измерить расстояние и направление на ориентир относительно локальной координаты системы робота. Такие датчики называются *датчиками расстояния и направления*. Наличие таких датчиков не является неправдоподобным допущением, поскольку любой локальный признак, извлечённый из сканирования расстояния, содержит информацию о расстоянии и направлении, а также и о визуальных признаках, полученных со стереокамеры. Кроме этого, выделитель признаков может

СИГНАТУРА ОРИЕНТИРА

генерировать *сигнатуру*. В рамках изложения в книге будем считать сигнатурой числовое значение (например, среднее значение цвета). Это может быть и целое число, характеризующее тип ориентира, и многомерный вектор, характеризующий, скажем, его высоту и цвет.

Если определить расстояние как r, ориентацию как ϕ, а сигнатуру s, вектор признаков задан набором триплетов

$$f(z_t) = \{f_t^1, f_t^2, ...\} = \{\begin{pmatrix} r_t^1 \\ \phi_t^1 \\ s_t^1 \end{pmatrix}, \begin{pmatrix} r_t^2 \\ \phi_t^2 \\ s_t^2 \end{pmatrix}, ...\} \tag{6.38}$$

Количество признаков, идентифицированных на каждом такте времени, различается. Но множество вероятностных алгоритмов подразумевает условную независимость между признаками

$$p(f(z_t)|x_t, m) = \prod_i p(t_t^i, \phi_t^i, s_t^i|x_t, m) \tag{6.39}$$

Условная независимость применима, если шум каждого отдельного измерения $(r_t^i\, \phi_t^i\, s_t^i)^T$ не зависит от шума других измерений $(r_t^i\, \phi_t^i\, s_t^i)^T$ (для $i \neq j$). При допущении условной вероятности в один момент времени возможно обработать только один признак, также, как было сделано в нескольких моделях измерения расстояния. Это значительно облегчает разработку алгоритмов, реализующих вероятностные модели измерений.

Начнём с определения модели датчика для признаков. В начале главы было выделено два типа карт: *карты на основе признаков* и *карты на основе местоположения*. Модели измерения на основе ориентиров обычно определены только для карт на основе признаков. Читатель может припомнить, что такие карты состоят из списков признаков $m = \{m_1, m_2, ...\}$, каждый из которых может иметь сигнатуру и *координату местоположения*. Местоположение признака, обозначенное $m_{i,x}$ и $m_{i,y}$, обозначает просто координаты на глобальной координатной сетке карты.

Вектор измерения для датчика ориентиров без учёта шума легко определить, используя обычные геометрические зависимости. Смоделируем шумы при восприятии ориентиров с помощью независимого гауссового шума по расстоянию, направлению и сигнатуре. Результирующая модель измерений сформулирована для случая, когда i-ый признак в момент времени t соответствует j-му ориентиру на карте. Как обычно, положение робота задано в виде $x_t = (x\, y\, \theta)^T$.

$$\begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix} = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \\ s_j \end{pmatrix} + \begin{pmatrix} \varepsilon_{\sigma_r^2} \\ \varepsilon_{\sigma_\phi^2} \\ \varepsilon_{\sigma_s^2} \end{pmatrix} \tag{6.40}$$

Здесь ε_{σ_r}, $\varepsilon_{\sigma_\phi}$, и ε_{σ_s} - переменные ошибки в виде гауссовых функций с нулевым математическим ожиданием и стандартными отклонениями σ_r, σ_ϕ, и σ_s, соответственно.

6.6.3 Модель датчика с известным соответствием

ПРОБЛЕМА АССОЦИАЦИИ ДАННЫХ

ПЕРЕМЕННАЯ СООТВЕТСТВИЯ

Одна из главных проблем датчиков расстояния/направления известна как *проблема ассоциации данных*. Она возникает, когда невозможно различить ориентиры, и появляется некоторая остаточная неопределённость по отношению к идентификации ориентира. Для разработки модели датчика расстояния/направления будет полезно ввести переменную соответствия между признаком f_t^i и ориентиром m_j на карте. Эта переменная будет определена как c_t^i с $c_t^i \in \{1, ..., N+1\}$. N – количество ориентиров на карте m. Если $c_t^i = j \leq N$, тогда i-ый признак, наблюдаемый в момент времени t, соответствует j-ому ориентиру на карте. Другими словами, c_t^i - истинный идентификатор наблюдаемого признака. Единственное исключение происходит при $c_t^i = N + 1$: В данном случае наблюдение признака не соответствует ни одному признаку карты m. Этот случай важен для обработки фантомных ориентиров, а также очень актуален для составления карт в робототехнике, когда робот может столкнуться с прежде неизвестными ориентирами.

В Таблице 6.4 показан алгоритм вычисления вероятности признака f_t^i с известным соответствием $c_t^i \leq N$. В строках 3 и 4 вычисляются верные расстояние и направление на ориентир. Вероятность для измеренных расстояния и направления вычисляются в строке 5, где подразумевая независимость шумов. Как читатель может легко убедиться, алгоритм реализует уравнение (6.40).

1: **Algorithm landmark_model_known_correspondence** (f_t^i, c_t^i, x_t, m) :

2: $j = c_t^i$

3: $\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$

4: $\hat{\phi} = \text{atan2}(m_{j,y} - y, m_{j,x} - x)$

5: $q = \textbf{prob}(r_t^i - \hat{r}, \sigma_r) \cdot \textbf{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi) \cdot \textbf{prob}(s_t^i - s_j, \sigma_s)$

6: *return* q

Таблица 6.4 Алгоритм вычисления правдоподобности измерения ориентира. Алгоритм требует на вход наблюдаемый ориентир $f_t^i = (r_t^i \, \phi_t^i \, s_t^i)^T$, истинное обозначение признака c_t^i, положение робота $x_t = (x \, y \, \theta)^T$, и карта m. На выходе вычисляется численное значение вероятности $p(f_t^i | c_t^i, m, x_t)$.

6.6.4 Определение положения на основе выборки

Иногда желательно сделать выборку местоположений робота x_t, соответствующую измерению f_t^i с признаком c_t^i. Мы уже сталкивались с такими алгоритмами выборки в прошлой главе, при обсуждении модели движения робота. Такие модели выборки также применимы для моделей датчиков. Например, при глобальной локализации робота будет полезным сгенерировать такую выборку положений робота, которая будет включать измерения датчиков для генерации начальных гипотез локализации.

Хотя, в общем случае, выборка положений x_t, соответствующая измерению датчика z_t, затруднена, для нашей модели ориентиров возможно предложить эффективный алгоритм выборки, хотя и с некоторыми допущениями. В частности, необходимо знать априорную вероятность $p(x_t|c_t^i, m)$. Для простоты, допустим, что эта априорная вероятность равномерна (в общем случае это не так!). Тогда, согласно теореме Байеса

(6.41)

$$\begin{aligned} p(x_t|f_t^i, c_t^i, m) &= \eta\, p(f_t^i|c_t^i, x_t, m)\, p(x_t|c_t^i, m) \\ &= \eta\, p(f_t^i|c_t^i, x_t, m) \end{aligned}$$

Выборка из $p(x_t|f_t^i, c_t^i, m)$ теперь может быть выполнена из «обратной» модели датчика $p(f_t^i|c_t^i, x_t, m)$. В Таблице 6.5 приводится алгоритм, выполняющий выборку положений x_t. Правда, такая выборка имеет особенность: даже в случае отсутствия шума, наблюдение ориентира не определяет точного положения робота. Согласно алгоритму, робот может находиться в произвольной точке окружности вокруг ориентира, диаметр которой равен расстоянию до ориентира. Неопределённость положения робота следует также из факта того, что параметры расстояния и направления накладывают два ограничения в трехмерном пространстве положений робота.

1: **Algorithm sample_landmark_model_known_correspondence** (f_t^i, c_t^i, m) :

2: $j = c_t^i$
3: $\hat{\gamma} = \text{rand}(0, 2\pi)$
4: $\hat{r} = r_t^i + \text{sample}(\sigma_r)$
5: $\hat{\phi} = \phi_t^i + \text{sample}(\sigma_\phi)$
6: $x = m_{j,x} + \hat{r}\cos\hat{\gamma}$
7: $y = m_{j,y} + \hat{r}\sin\hat{\gamma}$
8: $\theta = \hat{\gamma} - \pi - \hat{\phi}$
9: *return* $(x\, y\, \theta)^T$

Таблица 6.5 Алгоритм выполнения выборки положений из измерения ориентира $f_t^i = (r_t^i\, \phi_t^i\, s_t^i)^T$ с известным обозначением c_t^i.

Для реализации алгоритма выборки, необходимо выполнить выборку оставшегося свободного параметра, определяющего конкретное место окружности, в котором и находится робот. Этот параметр называется $\hat{\gamma}$ в Таблице 6.5, и выбирается случайным образом в строке 3. В строках 4 и 5 происходит изменение измеренных расстояния и направления с использованием факта симметричного отображения математического ожидания и измерения в виде нормального распределения. Наконец, в строках с 6 по 8 восстанавливается положение, соответствующее $\hat{\gamma}$, $\hat{r}$, и $\hat{\phi}$.

На Рис. 6.13 показано распределение положений $p(x_t|f_t^i, c_t^i, m)$ (левая схема) и выборки, выполненной алгоритмом **sample_landmark_model_known_correspondence** (правая схема). Апостериорное распределение проектируется на плоскость в виде кольца вокруг измеренного расстояния r_t^i. В трехмерном пространстве положений образуется спираль, которая разворачивается из кольца с углом θ.

6.6.5 Дальнейшие соображения

В обоих алгоритмах измерений на основе ориентиров подразумевается, что соответствие известно. Случай, когда соответствие неизвестно, будет детально описан в следующих главах при обсуждении алгоритмов локализации и построения карт.

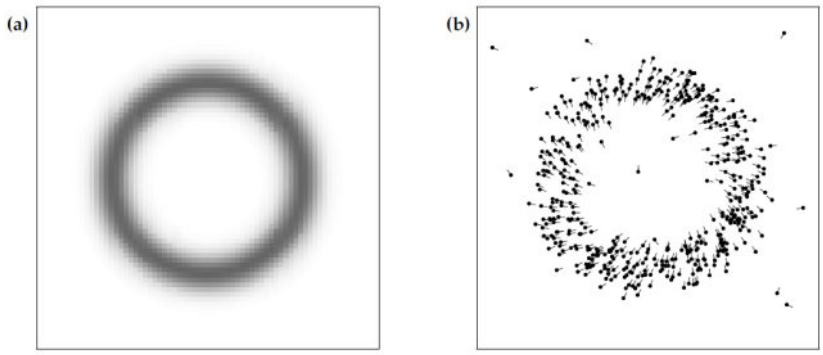


Рис. 6.13 Модель обнаружения ориентира: (a) Согласно апостериорному распределению положения робота, был обнаружен ориентир на расстоянии 5 м и под относительным углом направления 30 градусов (в проекции на плоскость). (b) Выборка положений робота, сгенерированная на основании обнаружения. Линиями обозначена ориентация положений по углу направления.

Следует добавить несколько слов о сигнатуре ориентира. В большинстве опубликованных алгоритмов черты внешнего вида в явном виде не используются. Если сигнатура не учитывается, все ориентиры выглядят одинаково, и проблема ассоциации данных при оценке переменной соответствия усугубляется. Мы включили в модель сигнатуру потому, что это значимый источник информации, которую часто можно легко извлечь из показаний датчиков.

Как отмечено выше, основной причиной использования признаков вместо полного вектора измерений является экономия вычислительных ресурсов. Гораздо проще обрабатывать несколько сот признаков, чем несколько миллионов измерений расстояния. Представленная здесь модель крайне груба и, определённо, не отражает каких-либо физических законов, лежащих в основе процесса измерений датчика. Несмотря на это, способна хорошо работать в большом количестве прикладных задач.

ДОСТАТОЧНЫЕ СТАТИСТИКИ

Важно заметить, что операция преобразование измерений в признаки также имеет свою цену. В литературе по робототехнике признаки часто (ошибочно) воспринимаются, как *достаточные статистики* вектора измерений z_t. Пусть X будет интересующей переменной (например, картой, положением, и т. д.), а Y – какой-либо другой информацией, которую хотелось бы учитывать (скажем, прошлые измерения датчиков). Тогда f будет достаточной статистикой z_t при условии

(6.42)
$$p(X|z_t, Y) + p(X|f(z_t), Y)$$

На практике большое количество информации просто утрачивается в силу использования признаков вместо полного вектора измерений. Эта утра-

ченная информация усугубляет некоторые проблемы, например, проблему ассоциации данных при определении, не оказался ли робот в ранее посещённом местоположении. Проблему извлечения признаков легко понять умозрительно: Если открыть глаза, визуальная картина окружающего, скорее всего, окажется достаточной, чтобы недвусмысленно определить местоположение, даже если до этого момента присутствовала общая неопределённость. Если же, с другой стороны, воспринимать только отдельные признаки, например, относительное расположение дверных косяков и подоконников, скорее всего, степень уверенности будет значительно ниже и получаемой информации будет недостаточно для выполнения глобальной локализации.

С появлением всё более быстрых компьютеров признаки в области робототехники постепенно теряют значимость, особенно при использовании датчиков расстояния. Многие новейшие алгоритмы полагаются на плотные векторы измерений и используют столь же плотные карты на основе местоположений для отображения окружающего пространства. Тем не менее, признаки хорошо подходят для использования в образовательных целях. Они позволяют представить базовые концепции вероятностной робототехники и, при правильном подходе к задачам, например, задаче соответствия, могут быть использованы даже для случаев, когда карты состоят из плотных наборов точек сканирования. В силу этого, часть алгоритмов, представленных в книге, сначала описываются в представлении признаков, а затем обобщаются для использования исходных измерений датчиков.

6.7 Практические соображения

В этом разделе описан ряд моделей измерения. Большое внимание было уделено моделям датчиков расстояния в силу их большой значимости для робототехники. Однако, описанные модели представляют только иллюстрацию для значительно более широкого класса вероятностного моделирования. При выборе верной модели важно найти правильное сочетание физического реализма и наиболее значимых свойств алгоритма на её основе. Например, было замечено, что физически реалистичная модель датчика расстояния может выдавать негладкие вероятности при определении положения робота, и это может стать проблемой для алгоритмов, скажем, многочастичных фильтров. Поэтому, физический реализм не является единственным критерием выбора модели датчика. Столь же важно учитывать применимость модели для алгоритма, в котором она применяется.

В большинстве случаев, чем точнее модель, тем лучше. Например, чем больше информации удаётся извлечь из показаний датчика, тем лучше. Модели на основе признаков извлекают сравнительно мало информации в силу того, что выделитель признаков проектирует показания датчиков с высокой размерностью в пространство с более низкой размерностью. В результате, методы на основе признаков демонстрируют худшие результаты, что, отчасти, компенсируется их большей вычислительной эффективностью.

При настройке внутренних параметров модели измерений часто полезно искусственно увеличить неопределённость в силу наличия основного ограничения вероятностного подхода: чтобы сделать вероятностные методы вычислительно разрешимыми, приходится игнорировать существующие в физическом мире зависимости и огромное множество переменных, которые эти зависимости образуют. Если такие зависимости не моделировать, алгоритмы, собирающие данные на малом количестве измерений, часто становятся

чрезмерно «самоуверенными».

САМОУВЕРЕННОСТЬ

Такая *самоуверенность*, в итоге, приводит к неверным заключениям, что негативно сказывается на результатах. На практике часто хорошей идеей является уменьшение количества информации, получаемой из показаний датчика. Проектирование измерения в пространство признаков с низкой размерностью – это лишь один способ выполнить такое уменьшение, который, вдобавок, подвержен описанным выше ограничениям. Равномерное экспоненциальное разложение информации в модели измерений с параметром α, как обсуждалось в разделе 6.3.4,будет значительно лучше в использовании, поскольку не увеличивает дисперсию на выходе вероятностного алгоритма.

6.8 Выводы

В этом разделе описаны вероятностные модели измерений.

• Начиная с моделей датчиков расстояния, в частности, лазерных, были описаны модели измерений $p(z_t^k|x_t, m)$. В первой модели для определения формы распределения $p(z_t^k|x_t, m)$ на основе известных картах m и положений x_t было использовано бросание лучей. Была определена модель смеси, учитывающая различные виды шумов, которые могут повлиять на измерения расстояния.

• Был описан метод максимального правдоподобия для определения внутренних параметров шума модели измерений. Поскольку модель измерения представлена методом смешения, была предложена итерационная процедура оценки максимального правдоподобия. Предложенный подход является экземпляром алгоритма максимизации ожидания, который изменяет такт вычисления ожидания в соответствии с типом ошибки измерения и тактом максимизации, который в закрытом виде находит наилучший набор внутренних параметров для этих ожиданий.

• Альтернативная модель измерений для датчиков расстояния основана на полях правдоподобия. В этом методе для моделирования вероятности $p(z_t^k|x_t, m)$ используется ближайшее расстояние в двухмерных координатах. Заметим, что такой подход даёт более гладкие распределения $p(z_t^k|x_t, m)$, но это происходит за счёт появления некоторых нежелательных побочных эффектов. Так, в методе полей правдоподобия игнорируется информацию относительно свободного пространства и не учитываются противоречия интерпретации измерений расстояния.

• Третья модель измерения основана на наложении карт. При наложении карт происходит наложение проходов сканирования датчика на локальные карты в целях корреляции их с глобальными. Такой подход недостаточно физически обоснован, зато может быть очень эффективно реализован на практике.

• Обсуждалось, каким образом предварительные вычисления могут уменьшить вычислительную эффективность в задачах реальном времени. В модели измерения на основе лучей предварительные вычисления выполняются в 3D, а поля правдоподобия требуют предварительных расчётов в 2D.

• Была представлена модель датчика на основе признаков, в которой робот выполняет извлечение данные о расстоянии, угле направления и сигнатуре близлежащих ориентиров. Методы на основе признаков извлекают признаки из исходных измерений датчиков, уменьшая размерность измерений датчиков на несколько порядков.

• В конце главы в ходе дискуссии о практических аспектах были выявлены некоторые трудности, которые могут возникнуть в конкретных реализациях.

6.9 Библиографические примечания

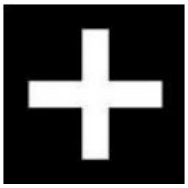
В этой главе приводятся лишь единичные примеры обширной литературы по физическому моделированию датчиков. Более точные модели сонаров можно найти в работах Блахата (Blahut et al., 1991), Грунбаума (Grunbaum et al., 1992) и Эттера (Etter, 1996). Модели лазерных датчиков расстояния были описаны Рисом (Rees, 2001). Эмпирическое обсуждение адекватных моделей шумов было проведено Сахиным (Sahin et al., 1998). В сравнении с этими моделями, обсуждаемые в книге модели весьма грубы.

Ранние разработки моделей на основе лучей для дальномеров можно найти в основополагающей работе Моравица (Moravec, 1988). Похожая модель была позже использована в локализации мобильного робота Бургардом (Burgard et al., 1996). Модель на основе лучей наподобие описанной, вместе с предварительным вычислением измерений расстояния была впервые описана Фоксом (Fox et al.,1999b). Поля правдоподобия впервые были опубликованы в работе Труна (Thrun, 2001), хотя они тесно связаны с обширной литературой по методам наложения сканирования (Besl and McKay 1992). Фактически, их можно считать мягким вариантом модели корреляции, описанной Конолигом и Чау (Konolige and Chou, 1999). Методы вычисления корреляции между картами сеток занятости также стали довольно популярными. Трун (Thrun, 1993) вычислил сумму квадратичной ошибки между отдельными ячейками двух карт сетей. Шили и Краули (Schiele and Crowley, 1994) представили сравнение разных моделей, включая методы на основе корреляции. Ямагучи и Ленгли (Yamauchi and Langley, 1997) проанализировали надёжность наложения карт в динамических средах. Дакит и Немцоу (Duckett and Nehmzow, 2001) преобразовали локальные сети занятости в гистограммы, которые можно сравнивать между собой более эффективно.

Измерения расстояния и угла направления для нахождения ориентиров общеприняты в литературе по SLAM. Возможно, впервые они были упомянуты Леонардом и Дюран-Уайтом (Leonard and Durrant-Whyte, 1991). В более ранней работе Краули (Crowley, 1989) вывел модели измерения для прямолинейных объектов.

6.10 Упражнения

1. Многие ранние образцы роботов выполняли навигацию, используя признаки на основе искусственных ориентиров, помещённые в среду так, чтобы их было легко различить. Хорошим местом для размещения таких маркеров был потолок (а почему?). Классический пример визуального маркера: допустим, на потолке закреплён маркер следующего вида:



Пусть глобальные координаты маркера будут x_m и y_m, а ориентация в глобальной системе координат θ_m. Определим положение робота через x_r, y_r, и θ_r.

Теперь допустим, что был определён маршрут, в ходе которого маркер может быть обнаружен на плоскости изображения перспективной камеры. Пусть x_i и y_i определяют координаты маркера на плоскости изображения, а θ_i – угол направления. Камера имеет фокусное расстояние f. С точки зрения проективной геометрии известно, что каждое смещение d в пространстве $x - y$ проектируется в виде пропорционального смещения $d \cdot \frac{f}{h}$ на плоскости изображения. (Необходимо сделать некоторые допущения относительно выбранной системы координат и выполнить их в явном виде).

Вопросы:

(a) Математически описать ожидаемое местоположение маркера (в глобальных координатах x_m, y_m, θ_m), при известных координатах изображения x_i, y_i, ,θ_i и положении робота в x_r, y_r, θ_r.

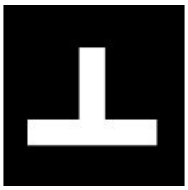
(b) Привести математическое выражение для вычисления координат изображения x_i, y_i, θ_i на основе положения робота x_r, y_r, θ_r и координат маркера x_m, y_m, θ_m.

(c) Привести математическое выражение для определения координат робота x_r, y_r, θ_r, допуская, что известны истинные координаты маркера x_m, y_m, θ_m, а координаты изображения x_i, y_i, θ_i.

(d) Пока подразумевалось, что маркер только один. Теперь допустим, что имеется несколько неразличимых между собой маркеров, как показанный выше. Сколько таких маркеров должен воспринимать робот, чтобы надёжно определить положение? Нарисовать такую конфигурацию и объяснить, почему она достаточна.

Подсказка: Нет необходимости учитывать неопределённость измерения для ответа на этот вопрос. Также, обратите внимание, что маркер симметричен, поскольку для ответа на вопрос это имеет значение!

2. В этом упражнении требуется обобщить вычисление из прошлого упражнения, чтобы учесть ошибки ковариации. Для упрощения вычислений допустим, что маркер несимметричен и можно оценить его абсолютный угол поворота :



Также, для простоты допустим, что значение угла поворота не содержит шумов, но оценки $x-y$ координат в плоскости изображения могут быть зашумлены. Определим, что измерения будут подвержены гауссовскому шуму с нулевым математическим ожиданием и ковариацией

$$\sum = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

для некоторого положительного значения σ^2.

Вычислить соответствующую ковариацию для трёх предыдущих упражнений. В частности,

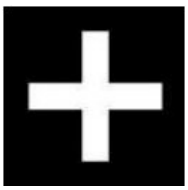
(a) Даны координаты изображения x_i, y_i, θ_i и координаты робота x_r, y_r, θ_r. Определить ковариацию ошибки для значений x_m, y_m, θ_i.

(b) Даны координаты робота x_r, y_r, θ_r и ковариация маркера x_m, y_m, θ_m. Определить ковариацию ошибки для значений x_i, y_i, θ_i.

(c) Задана ковариация маркера x_m, y_m, θ_m и координаты изображения x_i, y_i, θ_i. Определить ковариацию ошибки для значений x_r, y_r, θ_r.

Обратите внимание, что не все эти распределения могут быть гауссовыми. Для этого упражнения допустимо применить разложение в ряд Тейлора для получения гауссового апостериорного распределения, но требуется объяснить, каким образом это нужно делать.

3. Необходимо реализовать алгоритм **sample_marker_model**, который принимает на вход положение маркера x_m, y_m, θ_m и местоположение найденного маркера на плоскости изображения x_i, y_i, θ_i, и сгенерировать в виде выходных данных выборку положений робота x_r, y_r, θ_r. Использовать симметричный маркер из упражнения 1:



Изобразить график выборки для координат робота x_r и y_r со следующими параметрами (для этого распределения можно игнорировать угол ориентации θ_r).

problem #	x_m	y_m	θ_m	x_i	y_i	θ_i	h/f	σ^2
#1	0см	0см	0°	0см	0см	0°	200	0.1см^2
#2	0см	0см	0°	1см	0см	0°	200	0.1см^2
#3	0см	0см	0°	2см	0см	45°	200	0.1см^2
#4	0см	0см	0°	2см	0см	45°	200	1.0см^2
#5	50см	150см	10°	1см	6см	200°	250	0.5см^2

На всех графиках следует изобразить координатные оси с единицами измерения. Примечание: если не сможете предложить алгоритм в точном виде, предложить приближенный алгоритм и объяснить соображения приближения.

4. Для этого упражнения понадобится доступ к любому роботу для работы внутри помещений, оснащённому сонаром. Разместить датчик перед плоской стеной, на расстоянии d и под углом ϕ. Подобрать и измерить частоту, при которой датчик обнаруживает стену. Отобразить значения расстояния d на графике с шагом 0,5 м и углов ϕ с шагом 5 градусов. Что получилось в результате?

Часть II
Локализация

7 Локализация мобильного роботы: марковские и гауссовы алгоритмы

В этой главе читатель познакомится с рядом конкретных алгоритмов локализации мобильного робота. Локализация мобильного робота – это задача определения положения робота относительно данной карты окружения. Этот процесс часто называют *«оценкой местоположения» (position estimation)*. Локализация мобильного робота – это частный случай задачи общей локализации, самой базовой задачей восприятия среды, поскольку практически все задачи робототехники требуют знания местоположения объектов, с которыми производятся манипуляции. Методы, описываемые в этой и последующей главах, одинаково применимы и для задач локализации объекта.

На Рис. 7.1 показана графическая модель задачи локализации мобильного робота. Роботу дана карта окружающей среды, а его целью является определение своего положения на карте с помощью восприятия окружающей среды и оценки перемещений.

Локализацию мобильного робота можно рассматривать как проблему преобразования координат. Карты выполнены в глобальной системе координат, независимой относительно положения робота, поэтому локализацию можно свести к установлению соотношения между системой координат карты и собственной локальной системой координат робота. Знание способа преобразования координат позволяет выразить местоположение интересующих объектов в координатной системе робота, что является важным предварительным требованием для навигации. Как читатель может легко проверить, знания положения $x_t = (x\, y\, \theta)^T$ робота на карте достаточно для определения преобразования координат, при условии, что положение и карта

выражены в одной и той же системе координат.

К сожалению (в этом и состоит проблема локализации мобильного робота), положение обычно нельзя воспринять напрямую. Другими словами, большинство роботов не обладает датчиками, позволяющими без помех измерять положение. Положение приходится извлекать из данных измерений, и ключевой трудностью определения является недостаточность единственного измерения. Поэтому, чтобы определить положение, необходимо интегрировать данные измерений по времени. Чтобы понять необходимость этого, просто представим, что робот находится в здании со множеством одинаковых коридоров. В этом случае единственного измерения датчика (например, сканирования расстояния), очевидно, будет недостаточно для идентификации конкретного коридора.

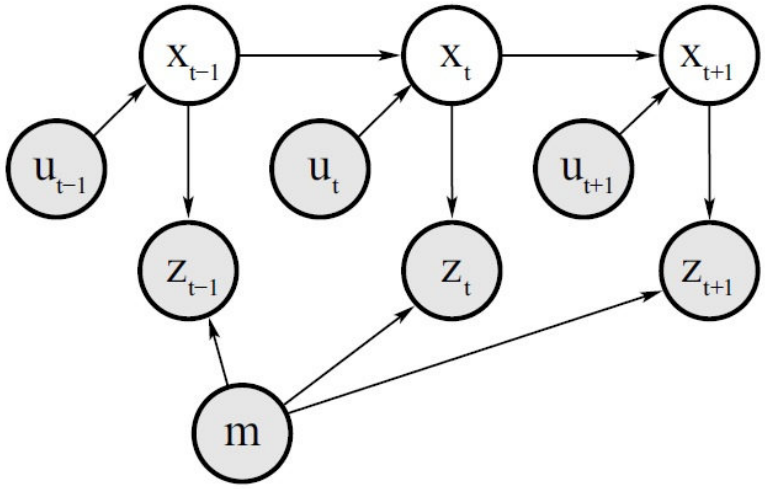


Рис. 7.1 Графическая модель локализации мобильного робота. Значения узлов с заливкой серым цветом известны. Карта обозначена как m, значения измерений - z, а управление - u. Целью локализации является определение переменных положения робота x.

Методы локализации были разработаны для широкого набора представлений карт. Мы уже обсуждали два типа карт: *на основе признаков* и *на основе местоположения*. Примером последних являются карты сеток занятости, которые обсуждаются в более поздней главе книги. Некоторые другие типы карт показаны на Рис. 7.2. На этой схеме показана созданная вручную метрическая двухмерная карта, топологическая карта в виде графа, карта сеток занятости и панорама из снимков потолка (которую тоже можно использовать в качестве карты). В более поздних главах будут детально описываться конкретные типы карт и обсуждаться алгоритмы получения карт из данных. Локализация подразумевает наличие точной карты.

В этой и следующей главах будут представлены некоторые базовые вероятностные алгоритмы мобильной локализации. Все эти алгоритмы являются вариантами базового фильтра Байеса, описанного в главе 2. Мы уже обсуждали преимущества и недостатки каждого представления и соответствующих алгоритмов. В главе также излагаются некоторые обобщения, касающиеся различных проблем локализации, определённые следующей классификацией.

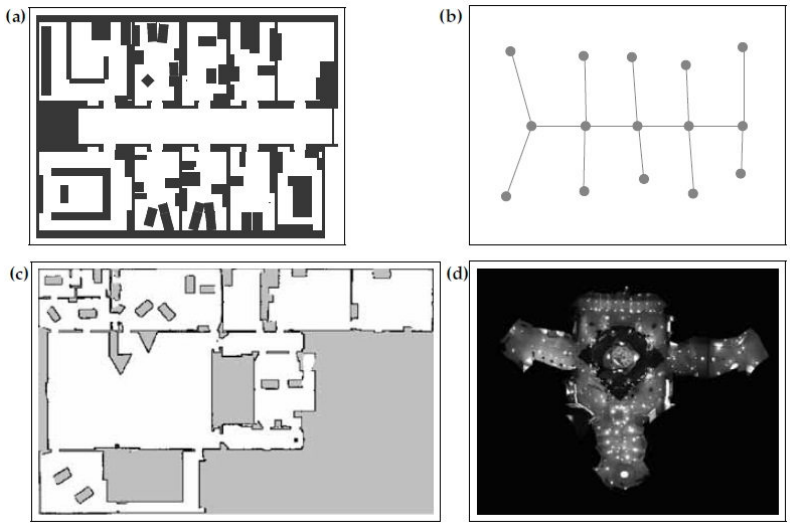


Рис. 7.2 Примеры карт, используемых для локализации робота: двухмерная метрическая схема (a), сконструированная вручную, топологическая карта в виде графа (b), карта сетки занятости (c), и панорама изображений потолка (d). Изображения принадлежат Френку Делаэрту, из института технологий штата Джорджия (Frank Dellaert, Georgia Institute of Technology).

7.1 Классификация задач локализации

Не все проблемы локализации одинаково трудны и для понимания необходимо кратко обсудить их классификацию. В этой классификации проблемы разделены по ряду важных критериев, касающихся природы окружающей среды и начальной информации, которой может обладать робот.

Локальная локализация против глобальной Проблемы локализации характеризуются типом осведомлённости, доступной изначально и при работе. Определим три типа проблем локализации с возрастающей сложностью.

ОТСЛЕЖИВАНИЕ ПОЗИЦИИ

Отслеживание положения подразумевает, что *начальное положение* робота известно. Локализация робота может быть достигнута, если суметь приспособиться к шумам при движении робота. Эффект таких шумов обычно достаточно мал, поэтому методы отслеживания позиции часто полагаются на допущение о малой ошибке положения. Неопределённость положения часто аппроксимируется одномодальным распределением (например, гауссовым). Проблема отслеживания позиции – локальная, поскольку неопределённость носит локальный характер и ограничена областью вблизи истинного положения робота.

ГЛОБАЛЬНАЯ ЛОКАЛИЗАЦИЯ

В *глобальной локализации*, первоначальное положение робота неизвестно. Изначально, робот помещён в некотором месте окружающей среды, но нет никаких данных ни о его местоположении, ни о самом окружении нет. Методы глобальной локализации не могут ограничиваться одной лишь ошибкой положения. Как позже будет показано в этой главе, одномодальные вероятностные распределения также обычно неприменимы. Глобальная

локализация более трудна, чем отслеживание позиции, и включает в себя задачу отслеживания как составную часть.

ПРОБЛЕМА ПОХИЩЕННОГО РОБОТА

Проблема похищенного робота является вариантом проблемы глобальной локализации, но ещё более трудна. Представим, что работающий робот был похищен и телепортирован в какое-то другое место. Дополнительная сложность по сравнению с проблемой глобальной локализации, состоит в том, что робот может ошибочно предполагать, что знает своё текущее положение. В глобальной локализации роботу всегда известно, что начальное местоположение неизвестно. Можно оспорить эту точку зрения и сказать, что на практике роботов похищают довольно редко. Практическая важность этой проблемы состоит в том, что даже самые современные алгоритмы локализации не защищены от сбоев, а возможность восстанавливаться после возникновения ошибки по-настоящему важна для автономных роботов. Тестирование алгоритма локализации «имитацией похищения» позволяет проверить его способность восстанавливаться после сбоя глобальной локализации.

Статическое и динамическое окружение Вторым фактором, оказывающим существенное влияние на локализацию, является окружающая среда. Окружение может быть статическим или динамическим.

СТАТИЧЕСКОЕ ОКРУЖЕНИЕ

Статическая окружающая среда – это такая среда, где единственным изменяющимся значением (состоянием) является положение робота. Другими словами, в статической среде движется только робот, а все другие объекты среды всегда остаются на своих местах. Статические среды имеют некоторые математические свойства, облегчающие эффективную вероятностную оценку.

ДИНАМИЧЕСКОЕ ОКРУЖЕНИЕ

Динамическая окружающая среда содержит объекты, помимо самого робота, местоположение или конфигурация которых изменяется со временем. Особенный интерес представляют изменения, которые сохраняются со временем и влияют более чем на одно измерение датчика. Изменения, которые невозможно измерить, конечно, не имеют отношения к локализации, а те, что затрагивают единственное измерение, лучше всего считать шумами. (см подраздел 2.4.4). Примерами более постоянных воздействий являются: наличие людей, засветки (для роботов, оборудованных камерами), перемещений мебели или дверей. Очевидно, что большинство реальных окружающих сред динамические, с изменениями состояния, происходящими в различных диапазонах скоростей.

Очевидно, локализация в динамических средах более трудна, чем в статических. Есть два принципиальных подхода к обработке динамики. Во-первых, динамические элементы можно включить в вектор состояний. В результате, марковское свойство снова будет соблюдаться, но такой подход влечёт за собой увеличение вычислительной и модельной сложности. Во–вторых, в некоторых ситуациях данные датчиков можно подвергнуть фильтрации, чтобы уничтожить негативные эффекты не смоделированной динамики. Такой подход описан ниже в подразделе 8.4.

Пассивный и активный подход Третий параметр, характеризующий проблемы локализации, связан с тем, управляет ли алгоритм локализации движением робота. Можно выделить два класса:

ПАССИВНАЯ ЛОКАЛИЗАЦИЯ

При *пассивной локализации* модуль локализации только наблюдает за действиями робота, а управление роботом осуществляется какими-то иными методами. Движение не используется в локализации и робот может перемещаться случайным образом или же выполнять какие-то регулярные действия.

АКТИВНАЯ ЛОКАЛИЗАЦИЯ

Алгоритмы *активной локализации* управляют роботом таким образом, чтобы минимизировать ошибку локализации и/или ущерб, причинённый ошибочным перемещением робота в опасное место.

Активный подход к локализации обычно показывает лучшие результаты локализации по сравнению с пассивным и мы уже обсуждали подобный пример прибрежной навигации во вводной части книги. Второй пример показан на Рис. 7.3. Здесь робот расположен в симметричном коридоре, и оценка после навигации по коридору сходится к двум (симметричным) положениям. Локальная симметрия среды делает невозможной локализацию робота в коридоре, и определение положения возможно только после захода в комнату и разрешения неопределённости состояния. Именно в таких ситуациях активная локализация даёт лучшие результаты, поскольку вместо того, чтобы просто ждать, пока робот случайно войдёт в комнату, возможно вовремя спрогнозировать безвыходную ситуацию, что позволит её предотвратить.

Главным ограничением активных методов является необходимость наличия возможности управлять роботом. На практике это требование делает использование одной лишь активной локализации недостаточным, поскольку роботу необходимо определять своё местоположение даже в процессе выполнении какой-либо задачи. Некоторые методы активной локализации создаются на основе пассивных методов. Другие комбинируют выполнение основной задачи с задачей локализации при управлении роботом.

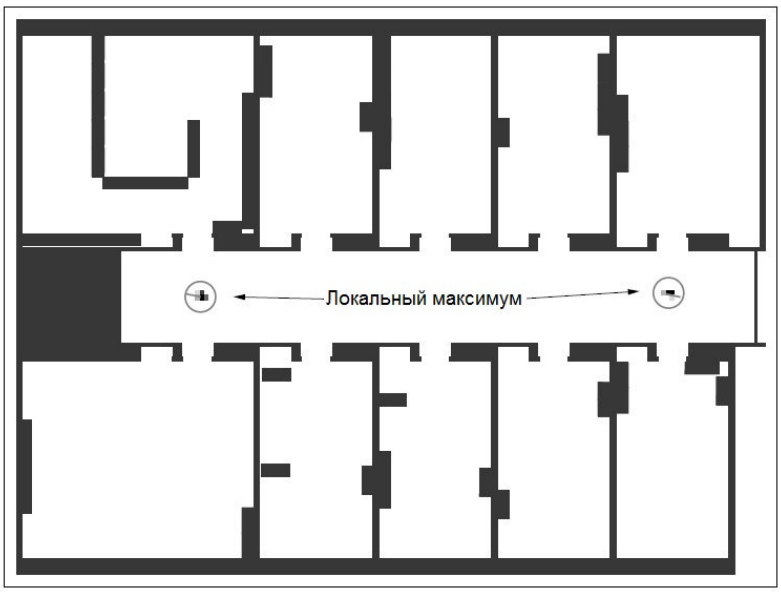


Рис. 7.3 Модельная ситуация, иллюстрирующая типичную оценку состояния при выполнении глобальной локализации в локально симметричной среде. Роботу необходимо зайти в комнату для определения своего местоположения.

В этой главе описаны исключительно алгоритмы пассивных методов локализации. Активная локализация будет обсуждаться в главе 17.

Один или несколько роботов Четвёртым критерием проблемы локализации является количество задействованных роботов.

ЛОКАЛИЗАЦИЯ ОДИНОЧНЫМ РОБОТОМ

Локализация одиночным роботом наиболее часто встречается в локализации и имеет дело с единственным роботом. В этом случае предполагается, что все данные собираются с помощью единственной робототехнической платформы и любая коммуникация отсутствует.

ЛОКАЛИЗАЦИЯ НЕСКОЛЬКИХ РОБОТОВ

Задача *локализации нескольких роботов* возникает в группах роботов. На первый взгляд, каждый робот может локализовать себя самостоятельно, поэтому проблема локализации несколькими роботами может быть решена через локализацию одним роботом. Но, если роботы в состоянии обнаружить друг друга, имеется лучшая возможность. Дело в том, что при известных относительных местоположениях нескольких роботов оценка одного робота может быть использована, чтобы уточнить оценку местоположения другого робота. Для случая локализации нескольких роботов возникают интересные и нетривиальные соображения относительно отображения оценок и природы связи между ними.

Эти четыре критерия отражают наиболее важные характеристики проблемы локализации мобильных роботов.

1: **Algorithm Markov_localization** $(bel(x_{t-1}), u_t, z_t, m)$:
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}, m) bel(x_{t-1}) dx_{t-1}$
4: $bel(x_t) = \eta\, p(z_t|x_t, m) \overline{bel}(x_t)$
5: *endfor*
9: *return* $bel(x_t)$

Таблица 7.1 Марковская локализация.

Существуют и другие характеристики, влияющие на сложность проблемы, такие, как информация, полученная из измерений и её утрата при передвижении робота. Кроме того, симметричные среды более трудны для выполнения локализации, чем асимметричные из-за более высокой степени возникающей неоднозначности.

7.2 Марковская локализация

Вероятностные алгоритмы локализации являются вариантами фильтра Байеса. Непосредственное приложение байесовских фильтров к проблеме локализации называется *марковской локализацией*. В Таблице 7.1 описан основной алгоритм. Этот алгоритм выведен из **Bayes_filter** (Таблица 2.1 на странице 32). Заметим, что для алгоритма **Markov_localization** также требуется карта m в качестве входных данных, которая используется в модели измерений $p(z_t|x_t, m)$ (строка 4). Она часто, хотя и не всегда, также включается в модель движения $p(x_t|u_t, x_{t-1}, m)$ (строка 3). Также, как и байесовский фильтр, марковская локализация преобразует вероятностную

оценку в момент времени $t-1$ в оценку в момент времени t. С помощью марковской локализации возможно решать задачи глобальной локализации, отслеживания позиции и украденного робота в статических средах.

Начальная оценка $bel(x_0)$ отражает исходную степень осведомлённости относительно положения робота. Она устанавливается различными способом, зависящим от типа задачи локализации.

• **Отслеживание позиции**. Если известно начальное положение, $bel(x_0)$ инициализируется распределением, сосредоточенным в одной точке. Пусть $\bar{x}_0$ означает (известное) начальное положение. Тогда

(7.1)
$$bel(x_0) = \begin{cases} 1 & \text{если } x_0 = \bar{x}_0 \\ 0 & \text{в других случаях} \end{cases}$$

Точечные распределения дискретны и плотностью не обладают.

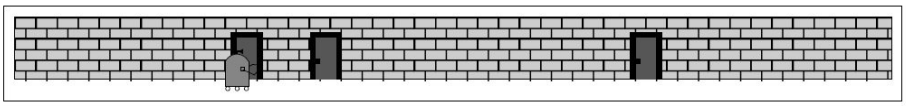


Рис. 7.4 Модельная среда для иллюстрации локализации мобильного робота: одномерный коридор с тремя неразличимыми между собой дверьми. Изначально робот не знает своего местоположения, только направление движения, и его целью является выполнение локализации.

На практике начальное положение часто известно лишь приблизительно. В этом случае оценка $bel(x_0)$ инициализируется узким нормальным распределением с центром около $\bar{x}_0$. Нормальные распределения определены выражением (2.4) на странице 21.

(7.2)
$$bel(x_0) = \underbrace{\det(2\pi\Sigma)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_0-\bar{x}_0)^T\Sigma^{-1}(x_0-\bar{x}_0)\}}_{\sim\mathcal{N}(x_0;\bar{x}_0,\Sigma)}$$

Σ–это ковариация неопределённости начального положения.

• **Глобальная локализация**. Если начальное положение неизвестно, $bel(x_0)$ инициализируется равномерным распределением в пространстве всех допустимых положений карты:

(7.3)
$$bel(x_0) = \frac{1}{|X|}$$

где $|X|$ означает объем (мера Лебега) пространства всех положений на карте.

• **Прочие**. Частичное понимание роботом своей позиции обычно может быть легко преобразовано в применимое первичное распределение. Например, если известно, что робот стартует около двери, можно инициализировать $bel(x_0)$ нулевой плотностью, кроме мест около двери, где она может

быть однородной. Если известно, что робот находится в конкретном коридоре, можно инициализировать $bel(x_0)$ однородным распределением в коридоре и нулём во всех остальных точках.

7.3 Иллюстрация марковской локализации

Мы уже обсуждали марковскую локализацию во вводной части книги в качестве иллюстративного примера вероятностной робототехники. Сейчас мы можем вернуться к примеру, используя математический подход. На Рис. 7.4 показан одномерный коридор с тремя идентичными дверями. Первичная гипотеза $bel(x_0)$ равномерна для всех положений, что показано на Рис. 7.5a равномерной плотностью. Когда робот запрашивает датчики и обнаруживает, что находится около одной из дверей, происходит умножение оценки $bel(x_0)$ на $p(z_t|x_t, m)$, как показано в строке 4 алгоритма. На верхней плотности на Рис. 7.5b показана вероятность $p(z_t|x_t, m)$ для примера с коридором. Нижняя плотность является результатом умножения этой плотности на априорную равномерную гипотезу. Здесь снова получается многомодовая результирующая гипотеза, отражающая остаточную неопределённость в этой точке.

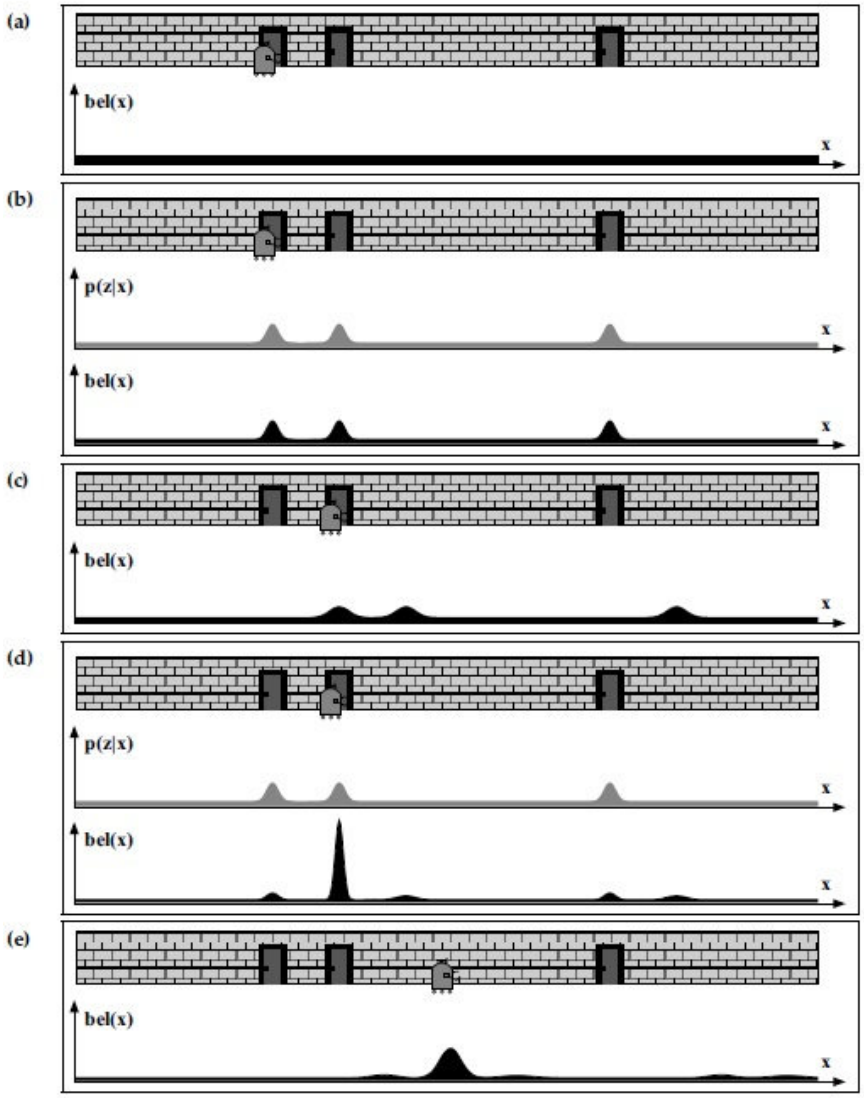


Рис. 7.5 Иллюстрация марковского алгоритма локализации. На каждой схеме показана позиция робота в коридоре и текущая гипотеза $bel(x)$. На схемах (b) и (d) также показана модель наблюдений $p(z_t|x_t)$, описывающая вероятность наблюдения двери в разных местоположениях коридора.

По мере того, как робот перемещается вправо, как показано на Рис. 7.5c, в строке 3 марковского алгоритма локализации происходит свёртка оценки с моделью движения $p(x_t|u_t, x_{t-1})$. Модель движения $p(x_t|u_t, x_{t-1})$ не сосредоточена на одном положении, но распределена на всем континууме положений с центром вокруг ожидаемого результата идеального движения. Результат приведён на Рис. 7.5c, где показано смещение оценки, которая также расширена в результате выполнения свёртки.

Последнее измерение показано на Рис. 7.5d. Здесь в алгоритме марковской локализации выполняется перемножение текущей гипотезы с вероятностью восприятия $p(z_t|x_t)$. На этой точке большая часть вероятности сфокусирована около истинного расположения и робот достаточно уверен, что сумел выполнить локализацию. На Рис. 7.5e показана гипотеза робота после

передвижения вниз по коридору.

Уже было замечено, что марковская локализация не зависит от основного отображения пространства состояний. Фактически, её возможно реализовать, используя любое из отображений, обсуждаемых в Главе 2. Давайте выведем практические алгоритмы, способные локализовать робота в реальном времени, используя три различных отображения. Начнём с фильтров Калмана, в которых гипотеза отображается в виде первого и второго моментов. Затем продолжим рассмотрение с дискретными, сеточными представлениями и, наконец, представим алгоритмы на основе многочастичных фильтров.

7.4 Локализация на основе EKF

Алгоритм *локализации на основе обобщённых фильтров Калмана* или *локализация на основе EKF* – это особый случай марковской локализации. Локализация на основе EKF отображает гипотезы $bel(x_t)$ в виде первого и второго моментов, математического ожидания μ_t и ковариации Σ_t. Базовый алгоритм EKF приведён в Таблице 3.3 в Главе 3.3 (страница 62). Локализация на основе EKF будет нашей первой практической реализацией EKF в контексте настоящей задачи робототехники.

Наш алгоритм локализации EKF подразумевает, что карта отображается в виде набора признаков. В любой момент времени t робот может принимать вектор расстояний и направлений на близлежащие признаки: $z_t = \{z_t^1, z_t^2, ...\}$. Начнём с алгоритма локализации, в котором все признаки уникально различимы. Существование уникально различимых признаков необязательно является плохим допущением. Например, Эйфелева башня в Париже – это ориентир, который сложно перепутать с чем-то ещё и который хорошо видно со многих точек Парижа. Обозначим признаки набором *переменных соответствия* c_t^i, по одной для каждого вектора признаков z_t^i. Переменные соответствия также уже обсуждались в Главе 6.6. Для начала допустим, что соответствия известны. Затем разовьём изложение до более общего случая, позволяющего использовать неразличимые признаки. В более общем случае для оценки значения латентной переменной соответствия будет использоваться алгоритм максимального правдоподобия, а результат этой оценки - использован в качестве эталонного.

ПЕРЕМЕННАЯ СООТВЕТСТВИЯ

7.4.1 Иллюстрация

На Рис. 7.6 показан алгоритм локализации на основе EKF, для примера локализации мобильного робота в одномерном коридоре (см. Рис. 7.4). В целях удобства и для обеспечения одномодальной формы гипотезы распределения в EKF, сделаем два допущения. Во-первых, допустим, что соответствия известны. Присвоим уникальные метки дверям (1, 2 и 3), и определим модель измерений $p(z_t|x_t, m, c_t)$, где m это карта, а $c_t \in \{1, 2, 3\}$ обозначает двери, наблюдаемые в момент времени t. Во-вторых, допустим, что начальное положение относительно хорошо известно. Типичная начальная оценка в виде нормального распределения, показанная на Рис. 7.6a, имеет центр в зоне около Двери 1 и гауссовскую неопределённость. По мере движения робота вправо, выполняется свёртка оценки вместе с моделью движения. Результирующей оценкой является смещённая гауссовская функция с увеличенной шириной, показанная на Рис.7.6b.

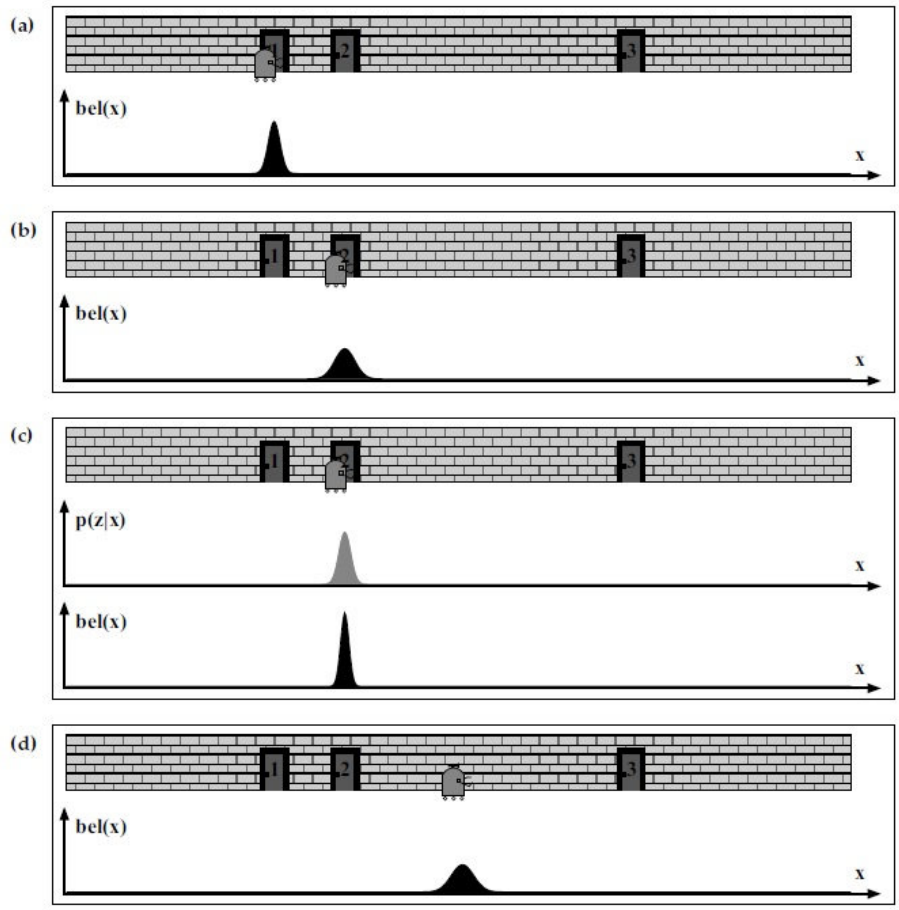


Рис. 7.6 Применение алгоритма фильтра Калмана для локализации мобильного робота. Все плотности представлены одномодальными гауссовыми функциями.

Теперь допустим, что робот обнаружил, что находится около двери $c_t = 2$. Верхняя плотность на Рис 7.6c $p(z_t|x_t, m, c_t)$ для этого наблюдения (снова в виде гауссового распределения). Свёрткой вероятности измерения в оценку робота получается апостериорная вероятность, показанная на Рис. 7.6c. Стоит заметить, что дисперсия результирующей оценки меньше, чем для обоих предыдущих наблюдений и плотности наблюдения. Это естественно, поскольку интегрирование двух независимых оценок позволяет уточнить оценку робота, по сравнению с любой из отдельных оценок. После перемещения вниз по коридору неопределённость робота в оценке местоположения снова возрастает, поскольку EKF продолжает добавлять её (Рис. 7.6d). Этим примером можно наглядно проиллюстрировать использование фильтра Калмана в ограниченных условиях задачи.

7.4.2 Алгоритм локализации EKF

До настоящего момента дискуссия носила довольно абстрактный характер. По умолчанию считалось, что подходящие модели движения и измерения доступны, а количество ключевых переменных для обновления EKF не бы-

ло определено. Давайте обсудим конкретную реализацию EKF для карт на основе признаков, состоящих из точечных ориентиров, как уже обсуждалось в подразделе 6.2. Для таких точечных ориентиров используем обычную модель измерений, обсуждаемую в подразделе 6.6. Также используем модель движения на основе скорости, определённую в подразделе 5.3. Перед тем, как продолжать чтение, читателю рекомендуется ненадолго вернуться к указанным разделам, чтобы восстановить основные уравнения измерения и движения.

В Таблице 7.2 описан алгоритм локализации на основе EKF с известными соотношениями **EKF_localization_known_correspondences**. Этот алгоритм выводится из EKF в Таблице 3.3 Главы 3. На вход принимается гауссова оценка положения робота в момент времени $t-1$ с математическим ожиданием μ_{t-1} и ковариацией Σ_{t-1}. Далее, требуется управляющее воздействие u_t, карта m, и набор переменных $z_t = \{z_t^1, z_t^2, ...\}$, измеренных в момент времени t, а также переменные соответствия $c_t = \{c_t^1, c_t^2, ...\}$. На выход возвращается новое значение μ_t, Σ_t, а также правдоподобие наблюдения признака p_{z_t}. Алгоритм не обрабатывает случай прямолинейного движения, для которого $\omega_t = 0$. Обработку этого случая оставим для рассмотрения читателю в качестве упражнения.

Отдельные вычисления в этом алгоритме описаны ниже. В строках 3 и 4 вычисляются якобианы, необходимые для линеаризованной модели движения. В строке 5 из данных управления определяется матрица ковариации шумов движения. В строках 6 и 7 реализуется уже знакомое обновление движения. Прогнозируемое положение после окончания движения $\bar{\mu}_t$ вычисляется в строке 6, а в строке 7 – соответствующий эллипс неопределённости. Обновление измерения (шаг коррекции) реализован в строках с 8 по 21. Основой такта обновления является перебор в цикле i всех признаков, наблюдаемых в момент времени t. В строке 10 назначается j-е соответствие i-му признаку в векторе измерений, а затем вычисляется прогнозируемое измерение $\hat{z}_t^i$ и якобиан H_t^i модели измерения.

1: **Algorithm EKF_localization_known_correspondences**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, m)$:

2: $\theta = \mu_{t-1,\theta}$

3: $$G_t = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

4: $$V_t = \begin{pmatrix} \frac{-\sin\theta+\sin(\theta+\omega_t\Delta t)}{\omega_t} & \frac{v_t(\sin\theta-\sin(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\cos(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta-\cos(\theta+\omega_t\Delta t)}{\omega_t} & -\frac{v_t(\cos\theta-\cos(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\sin(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

5: $$M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2\omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4\omega_t^2 \end{pmatrix}$$

6: $$\bar{\mu}_t = \mu_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

7: $\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + V_tM_tV_t^T$

8: $$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

9: *для всех наблюдаемых признаков* $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ *do*

10: $j = c_t^i$

11: $q = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$

12: $$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \\ m_{j,s} \end{pmatrix}$$

13: $$H_t^i = \begin{pmatrix} -\frac{m_{j,x}-\bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y}-\bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y}-\bar{\mu}_{t,y}}{q} & -\frac{m_{j,x}-\bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

14: $S_t^i = H_t^i\bar{\Sigma}_t[H_t^i]^T + Q_t$

15: $K_t^i = \bar{\Sigma}_t[H_t^i]^T[S_t^i]^{-1}$

16: $\bar{\mu}_t = \bar{\mu}_t + K_t^i(z_t^i - \hat{z}_t^i)$

17: $\bar{\Sigma}_t = (I - K_t^i\,H_t^i)\bar{\Sigma}_t$

18: *endfor*

19: $\mu_t = \bar{\mu}_t$

20: $\Sigma_t = \bar{\Sigma}_t$

21: $p_{z_t} = \prod_i \det(2\pi S_t^i)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t^i - \hat{z}_t^i)^T[S_t^i]^{-1}(z_t^i - \hat{z}_t^i)\}$

22: *return* μ_t, Σ_t, p_{z_t}

Таблица 7.2 Алгоритм локализации с помощью обобщённого фильтра Калмана (EKF), сформулированного для карты на основе признаков и робота с датчиком измерения расстояния и направления. Эта версия подразумевает знание точных значений соответствия.

Используя этот якобиан, вычисляется S_t^i, то есть неопределённость про-

гнозируемого измерения $\hat{z}_t^i$. Усиление Калмана K_t^i вычисляется в строке 15. Обновление оценки выполняется в строках 16 и 17, единожды для каждого признака. В строках 19 и 20 устанавливается новая оценка положения, за которой следует вычисление правдоподобия измерения в строке 21. В этом алгоритме следует обращать внимание на разницу углов, поскольку результат может выйти за пределы 2π.

7.4.3 Математический вывод локализации с помощью EKF

Шаг экстраполяции (Строки 3–7) В алгоритме локализации EKF используется модель движения, определённая уравнением (5.13). Кратко повторим определение:

$$(7.4)\quad \begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}_t}{\hat{\omega}_t}\sin\theta + \frac{\hat{v}_t}{\hat{\omega}_t}\sin(\theta + \hat{\omega}_t \Delta t) \\ \frac{\hat{v}_t}{\hat{\omega}_t}\cos\theta - \frac{\hat{v}_t}{\hat{\omega}_t}\cos(\theta + \hat{\omega}_t \Delta t) \\ \hat{\omega}_t \Delta t \end{pmatrix}$$

Здесь $x_{t-1} = (x\, y\, \theta)^T$ и $x_t = (x'\, y'\, \theta')^T$ выражают векторы состояния в момент времени $t-1$ и t, соответственно. Истинное движение описано поступательной скоростью $\hat{v}_t$, и вращательной скоростью $\hat{\omega}_t$. Как уже было указано в уравнении (5.10), эти скорости генерируются сигналами управления перемещением $u_t = (v_t\, \omega_t)^T$ с добавленным гауссовым шумом:

$$(7.5)\quad \begin{pmatrix} \hat{v}_t \\ \hat{\omega}_t \end{pmatrix} = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v_t^2 + \alpha_2 \omega_t^2} \\ \varepsilon_{\alpha_3 v_t^2 + \alpha_4 \omega_t^2} \end{pmatrix} = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix} + \mathcal{N}(0, M_t)$$

Из Главы 3 известно, что для локализации с помощью EKF поддерживается локальная апостериорная оценка состояния, выраженная в виде математического ожидания μ_{t-1} и ковариации Σ_{t-1}. Также вспомним, что базовым свойством EKF является линеаризации модели движения и измерения. Разложим модель движения на идеальную часть и случайный компонент шумов:

$$(7.6)\quad \underbrace{\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix}}_{x_t} = \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}}_{g(u_t, x_{t-1})} + \mathcal{N}(0, R_t)$$

В выражении (7.6) выполняется аппроксимация выражения (7.4) заменой истинного движения $(\hat{v}_t\, \hat{\omega}_t)^T$ фактическим управляющим воздействием $(v_t\, \omega_t)^T$, и учётом шума движения в дополнительном нормальной гауссовой функции с нулевым математическим ожиданием. Отсюда, в левой части уравнения (7.6) управление учитывается в виде истинного движения робота. Как мы помним из подраздела 3.3, линеаризация EKF аппроксимирует функцию g с помощью разложения в ряд Тейлора:

$$(7.7)\quad g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

Функция $g(u_t, \mu_{t-1})$ получается простой заменой неизвестного истинного состояния x_{t-1} ожиданием μ_{t-1} (которое известно). Якобиан G_t является производной функции g по отношению к x_{t-1}, оценённой для u_t и μ_{t-1}:

$$(7.8)\quad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} = \begin{pmatrix} \frac{\partial x'}{\partial \mu_{t-1,x}} & \frac{\partial x'}{\partial \mu_{t-1,y}} & \frac{\partial x'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial y'}{\partial \mu_{t-1,x}} & \frac{\partial y'}{\partial \mu_{t-1,y}} & \frac{\partial y'}{\partial \mu_{t-1,\theta}} \\ \frac{\partial \theta'}{\partial \mu_{t-1,x}} & \frac{\partial \theta'}{\partial \mu_{t-1,y}} & \frac{\partial \theta'}{\partial \mu_{t-1,\theta}} \end{pmatrix}$$

Здесь $\mu_{t-1} = (\mu_{t-1,x}\, \mu_{t-1,y}\, \mu_{t-1,\theta})^T$ - средняя оценка, разбитая на три отдельных значения, а $\frac{\partial x'}{\partial \mu_{t-1,x}}$ – короткое обозначение для производной g измерения x', по отношению к x для μ_{t-1}. Вычисление этих производных для выражения (7.6) даёт следующую матрицу:

$$(7.9)\quad G_t = \begin{pmatrix} 1 & 0 & \frac{v_t}{\omega_t}(-\cos\mu_{t-1,\theta} + \cos(\mu_{t-1,\theta} + \omega_t \Delta t)) \\ 0 & 1 & \frac{v_t}{\omega_t}(-\sin\mu_{t-1,\theta} + \sin(\mu_{t-1,\theta} + \omega_t \Delta t)) \\ 0 & 0 & 1 \end{pmatrix}$$

Для вывода ковариации шума движения $\mathcal{N}(0, R_t)$ определим ковариационную матрицу M_t шумов в *пространстве управления*. Вывод напрямую следует из модели движения в выражении (7.5):

$$(7.10)\quad M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2 \omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4 \omega_t^2 \end{pmatrix}$$

Для модели движения (7.6) требуется, чтобы шум движения был спроектирован в *пространство состояний*. Преобразование из пространства управления в пространство состояний выполняется с помощью ещё одной линейной аппроксимации. Требуемый для этого якобиан, обозначенный V_t, является производной функции движения g по параметрам движения, оценённым для u_t и μ_{t-1}:

(7.11)

$$V_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial u_t}$$

$$= \begin{pmatrix} \frac{\partial x'}{\partial v'} & \frac{\partial x'}{\partial \omega'} \\ \frac{\partial y'}{\partial v'} & \frac{\partial y'}{\partial \omega'} \\ \frac{\partial \theta'}{\partial v'} & \frac{\partial \theta'}{\partial \omega'} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{-\sin\theta + \sin(\theta + \omega_t \Delta t)}{\omega_t} & \frac{v_t(\sin\theta - \sin(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t \cos(\theta + \omega_t \Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta - \cos(\theta + \omega_t \Delta t)}{\omega_t} & -\frac{v_t(\cos\theta - \cos(\theta + \omega_t \Delta t))}{\omega_t^2} + \frac{v_t \sin(\theta + \omega_t \Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

Путём перемножения $V_t\, M_t\, V_t^T$ выполняется примерная проекция шумов управления в пространстве управления в шумы движения в пространстве состояний. В таком виде строки 6 и 7 фильтра локализации алгоритма EKF прямо соответствуют обновлениям прогноза общего алгоритма EKF, описанного в Таблице 3.3.

Этап коррекции (Строки 8–20) Для выполнения шага коррекции в локализации EKF также требуется линеаризованная модель измерения с добавочным гауссовым шумом. Модель измерения карт на основе признаков должна быть вариантом выражения (6.40), описанного в подразделе 6.6, а предпосылкой является знание идентификатора ориентира с помощью

переменной соответствия c_t. Пусть $j = c_t^i$ будет обозначением ориентира, соответствующего i-му компоненту вектора измерений. Отсюда,

$$(7.12)\quad \underbrace{\begin{pmatrix} r_t^i \\ \phi_t^i \\ s_t^i \end{pmatrix}}_{z_t^i} = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x}-x)^2+(m_{j,y}-y)^2} \\ \text{atan2}(m_{j,y}-y, m_{j,x}-x)-\theta \\ m_{j,s} \end{pmatrix}}_{h(x_t,j,m)} + \mathcal{N}(0, Q_t),$$

где $(m_{j,x}\, m_{j,y})^T$ - координаты i-го обнаружения ориентира в момент времени t, а $m_{j,s}$ его (корректная) сигнатура. Аппроксимация разложением в ряд Тейлора этой модели измерений

$$(7.13)\quad h(x_t, j, m) \approx h(\bar{\mu}_t, j, m) + H_t^i (x_t - \bar{\mu}_t).$$

H_t^i - это якобиан h расположения робота, вычисленный на основании прогнозируемого математического ожидания $\bar{\mu}_t$:

$$(7.14)\quad H_t^i = \frac{\partial h(\bar{\mu}_t, j, m)}{\partial x_t} = \begin{pmatrix} \frac{\partial r_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial r_t^i}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial \phi_t^i}{\partial \bar{\mu}_{t,\theta}} \\ \frac{\partial s_t^i}{\partial \bar{\mu}_{t,x}} & \frac{\partial s_t^i}{\partial \bar{\mu}_{t,y}} & \frac{\partial s_t^i}{\partial \bar{\mu}_{t,\theta}} \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{m_{j,x}-\bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y}-\bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y}-\bar{\mu}_{t,y}}{q} & -\frac{m_{j,x}-\bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

где q означает $(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$. Заметим, что в последней строке матрицы H_t^i содержатся только нули. Это происходит потому, что сигнатура не зависит от положения робота. Как следствие, наблюдаемая сигнатура s_t^i не оказывает никакого результата на такт обновления EKF. Что неудивительно: знание точного соотношения c_t^i делает наблюдаемую сигнатуру совершенно неинформативной.

Ковариация Q_t дополнительных шумов измерений в выражении (7.12) напрямую следует из (6.40):

$$(7.15)\quad Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

Наконец, заметим, что описанный алгоритм локализации на основе признаков обрабатывает несколько измерений одновременно, в то время, как EKF из подраздела 3.2 способен обрабатывать только единичное показание датчика. Наш алгоритм основан на неявном допущении условной независимости, которую мы уже кратко обсудили в подразделе 6.6, для выражения (6.39). Важно учитывать, что все вероятности измерения признаков считаются независимыми для положения x_t, обозначения ориентира c_t и карты m:

$$(7.16)\quad p(z_t | x_t, c_t, m) = \prod_i p(z_t^i | x_t, c_t^i, m)$$

Обычно это хорошее допущение, особенно для статических сред. Оно позволяет последовательно добавлять в фильтр информацию от нескольких признаков, как указано в строках с 9 по 18 Таблицы 7.2. Следует убедиться, что оценка положения обновляется при каждом проходе по циклу, поскольку в противном случае алгоритм вычисляет неверные гипотезы наблюдений (очевидно, цикл соответствует нескольким обновлениям измерений с нулевым перемещением между ними). Учитывая все вышеперечисленное, легко увидеть, что в строках 8–20, действительно, реализован такт коррекции общего алгоритма EKF. **Правдоподобие измерения (Строка 21).** В строке 21 вычисляется *правдоподобие* $p(z_t|c_{1:t}, m, z_{1:t-1}, u_{1:t})$ измерения z_t. Это правдоподобие не используется для обновления EKF, но полезно для отсечения выбросов или в случае неизвестного соответствия. При условии независимости отдельных векторов признаков можно ограничить вывод z_t^i и вычислять общее правдоподобие аналогично выражению (7.16). Для известной ассоциации данных $c_{1:t}$ правдоподобие может быть вычислено из прогнозируемой оценки $\overline{bel}(x_t) = \mathcal{N}(x_t; \bar{\mu}_t, \bar{\Sigma}_t)$ интегрированием по положению x_t и вычёркиванием не относящихся к делу условных переменных:

ПРАВДОПОДОБИЕ ИЗМЕРЕНИЯ

(7.17)

$$\begin{aligned} & p(z_t^i|c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ & \quad = \int p(z_t^i|x_t, c_{1:t}, m, z_{1:t-1}, u_{1:t})\, p(x_t|c_{1:t}, m, z_{1:t-1}, u_{1:t}) dx_t \\ & \quad = \int p(z_t^i|x_t, c_t^i, m)\, p(x_t|c_{1:t-1}, m, z_{1:t-1}, u_{1:t}) dx_t \\ & \quad = \int p(z_t^i|x_t, c_t^i, m)\, \overline{bel}(x_t) dx_t \end{aligned}$$

Левая часть итогового интеграла представляет собой правдоподобие измерения, считая, что местоположение робота x_t известно. Это правдоподобие задано гауссовой функцией со средним в местоположении x_t. Измерение, обозначаемое $\hat{z}_t^i$, представлено функцией измерения h. Ковариация гауссовой функции задана шумом измерений Q_t.

(7.18)

$$\begin{aligned} p(z_t^i|x_t, c_t^i, m) & \sim \mathcal{N}(z_t^i; h(x_t, c_t^i, m), Q_t) \\ & \approx \mathcal{N}(z_t^i; h(\bar{\mu}_t, c_t^i, m) + H_t(x_t - \bar{\mu}_t), Q_t) \end{aligned}$$

Выражение (7.18) следует из разложения в ряд Тейлора (7.13) функции h. Вставив это выражение обратно в (7.17), и заменив $\overline{bel}(x_t)$ на гауссову функцию, получим следующее правдоподобие измерения:

(7.19)

$$\begin{aligned} & p(z_t^i|c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ & \quad \approx \mathcal{N}(z_t^i; h(\bar{\mu}_t, c_t^i, m) + H_t(x_t - \bar{\mu}_t), Q_t) \otimes \mathcal{N}(x_t; \bar{\mu}_t, \bar{\Sigma}_t) \end{aligned}$$

где $\otimes$ означает уже знакомую свёртку по переменной x_t. Это уравнение показывает, что функция правдоподобия представляет собой свёртку двух гауссовых функций: одной, отображающей шумы измерения и второй, отображающей неопределённость состояния. Мы уже сталкивались с интегралами такого вида в подразделе 3.2 при выводе такта обновления движения

для фильтра Калмана и EKF. Решение этого интеграла в закрытом виде выполняется полностью аналогично. В частности, гауссова функция, определённая выражением (7.19) имеет математическое ожидание $h(\bar{\mu}_t, c_t^i, m)$ и ковариацию $H_t\, \bar{\Sigma}_t\, H_t^T + Q_t$. Отсюда, при линейной аппроксимации следующего выражения для правдоподобия измерения:

(7.20)

$$p(z_t^i | c_{1:t}, m, z_{1:t-1}, u_{1:t}) \sim \mathcal{N}(z_t^i; h(\bar{\mu}_t, c_t^i, m), H_t \bar{\Sigma}_t H_t^T + Q_t)$$

Таким образом,

(7.21)

$$\begin{aligned} &p(z_t^i | c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ &\quad = \eta \, \exp\left\{ -\frac{1}{2} (z_t^i - h(\bar{\mu}_t, c_t^i, m))^T [H_t \bar{\Sigma}_t H_t^T + Q_t]^{-1} (z_t^i - h(\bar{\mu}_t, c_t^i, m)) \right\} \end{aligned}$$

Заменой математического ожидания и ковариации этого выражения на $\hat{z}_t^i$ и S_t, соответственно, получим строку 21 алгоритма EKF из Таблице 7.2.

Сейчас алгоритм локализации EKF можно легко модифицировать для обработки выбросов. Стандартным подходом является использование только тех ориентиров, правдоподобие которых превышает пороговое значение. В общем, это хорошая идея, поскольку гауссова функция экспоненциально убывает и наличие хотя бы одного выброса может сильно повлиять на оценку положения. На практике, использование порога придаёт надёжность алгоритму, и при отсутствии порога EKF локализация будет неустойчива.

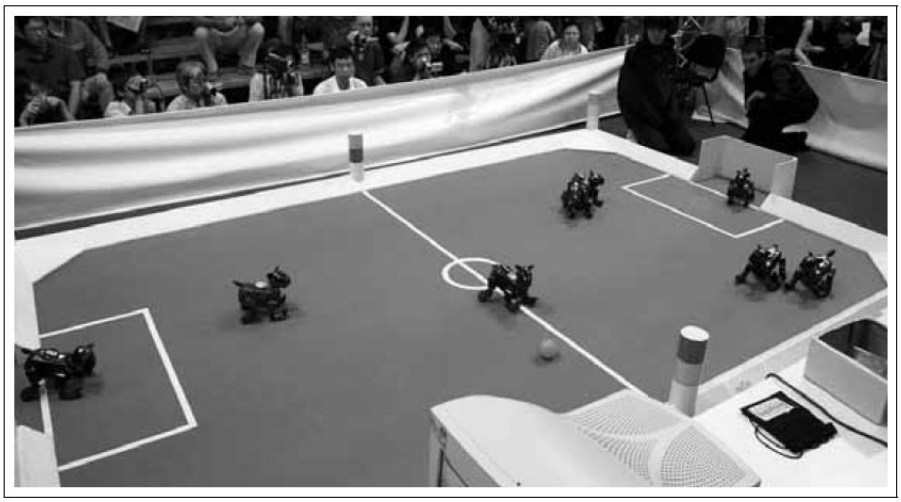


Рис. 7.7 Роботы AIBO на футбольном поле RoboCup. По углам поля и на центральной линии расположены шесть ориентиров.

7.4.4 Физическая реализация

Проиллюстрируем алгоритм EKF на примере локализации четвероногого шагающего робота AIBO на футбольном поле соревнований RoboCup. В

данной ситуации робот выполняет локализацию на основе шести маркеров разных цветов, размещённых вокруг поля (см. Рис. 7.7). Также, как в алгоритме EKF, приведённом в Таблице 7.2, управление перемещением $u_t = (v_t\, \omega_t)^T$ моделируется поступательной и вращательной скоростью, а наблюдения $z_t = (r_t\, \phi_t\, s_t)^T$ содержат значения относительного расстояния и направления на маркер. Для простоты допустим, что робот обнаруживает только один маркер в один момент времени.

Этап экстраполяции (Строки 3–7) На Рис. 7.8 показан этап экстраполяции алгоритма локализации EKF. На нем изображена неопределённость экстраполяции, появившаяся в результате различных параметров шумов движения, $\alpha_1 - \alpha_4$, используемых в строке 5 алгоритма. Параметры α_2 и α_3 установлены в значение 5% во всех визуализациях. Основные параметры зашумления поступательного и вращательного движения α_1 и α_4 варьируются между $\langle 10\%, 10\% \rangle, \langle 30\%, 10\% \rangle, \langle 10\%, 30\% \rangle, \langle 30\%, 30\% \rangle$ (от верхнего левого до нижнего правого на Рис. 7.8). На каждом графике робот выполняет действие управления $u_t = \langle 10\text{см/сек}, 5^\circ/\text{сек} \rangle$ на 9 сек, что приводит к перемещению по дуге длиной 90 см и повороту на 45°. Оценка предыдущего местоположения робота отображается эллипсом с центром в точке математического ожидания $\mu_{t-1} = \langle 80, 100, 0 \rangle$.

Алгоритмом EKF вычисляется прогнозируемое математическое ожидание $\bar{\mu}_t$ сдвигом предыдущей оценки при условии идеального движения (строка 6). Соответствующий эллипс неопределённости $\bar{\Sigma}_t$, состоит из двух компонентов. Одна оценивает неопределённость начального местоположения, другая – неопределённость, вызванную шумами движения (строка 7).

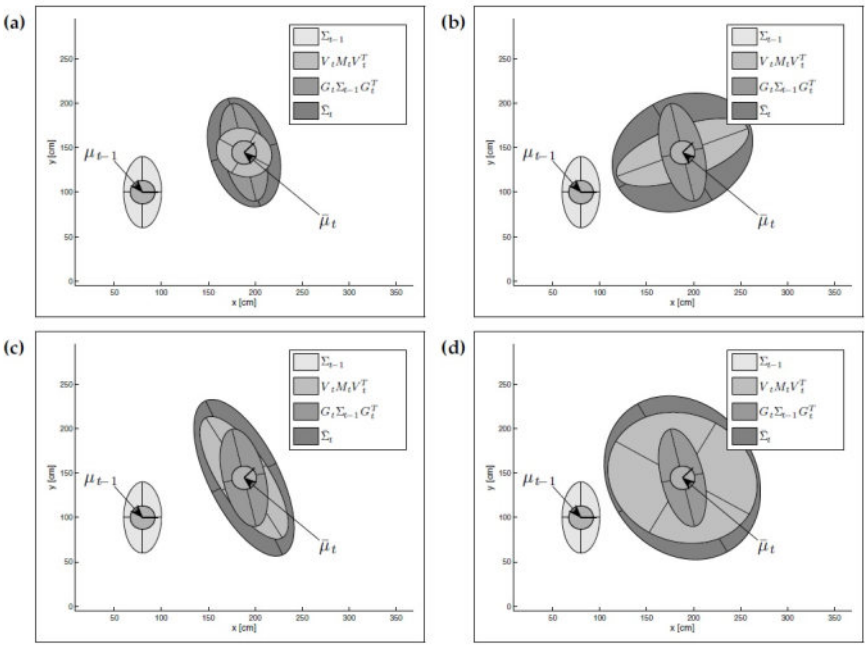


Рис. 7.8 Такт экстраполяции алгоритма EKF. Схемы были сгенерированы с различными параметрами шумов движения. Начальная оценка робота показана эллипсом с центром в μ_{t-1}. После движения по пути длиной 90 см и поворота на 45 градусов влево, прогнозируемое положение имеет центр в $\bar{\mu}_t$. На схеме (a) шум движения относительно мал как для поступательного движения, так и для вращательного. На других схемах показаны: (b) большое зашумление поступательного движения, (c) большое зашумление вращательного движения и (d) большое зашумление поступательного и вращательного движения.

Первый компонент $G_t \Sigma_{t-1} G_t^T$ игнорирует шумы движения и проектирует предыдущую неопределённость Σ_{t-1} с помощью линейной аппроксимации функции движения. Из выражений (7.8) и (7.9) вспомним, что линейная аппроксимация представлена матрицей G_t, которая является якобианом функции движения относительно предыдущего положения робота.

Результирующие эллипсы зашумления идентичны на всех четырёх схемах, поскольку шум движения не учитывается. Неопределённость шума движения моделируется вторым компонентом $\bar{\Sigma}_t$, заданным в виде $V_t\, M_t\, V_t^T$ (строка 7). Матрица M_t представляет шум движения в пространстве сигналов управления (строка 5). Эта матрица шумов движения проектируется в пространство состояний умножением на V_t, который является якобианом функции управления движения (строка 4). Как видно, результирующий эллипс отображает большую ошибку поступательной скорости ($\alpha_1 = 30\%$) в виде большой неопределённости вдоль направления движения (правые графики на Рис. 7.8). Большая ошибка скорости вращения ($\alpha_4 = 30\%$) приводит к большой неопределённости перпендикулярно направлению движения (нижние графики на Рис. 7.8). Общая неопределённость прогноза $\bar{\Sigma}_t$ задана сложением двух компонент.

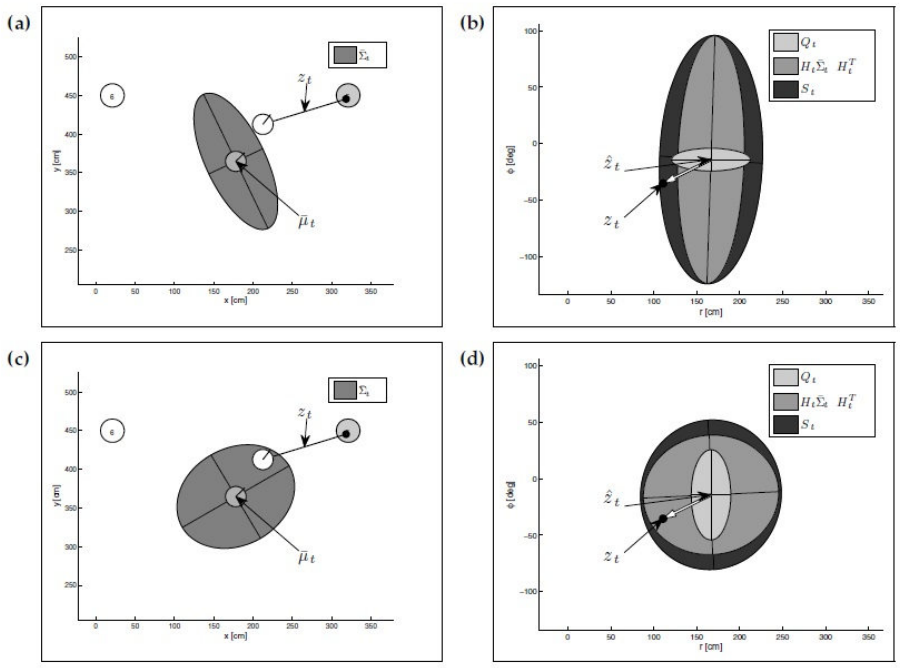


Рис. 7.9 Экстраполяция измерения. На левом графике показаны два прогнозируемых местоположения робота, а также их эллипсы неопределенности. Настоящий робот и его наблюдения показаны белым кругом и жирной линией, соответственно. На схемах справа показаны результирующие прогнозы измерений. Белыми стрелками показан параметр новизны "innovation" - разницей между наблюдаемыми и прогнозируемыми измерениями.

Такт коррекции: Прогноз измерения (Строки 8–14). В первой части такта коррекции алгоритм EKF выполняет прогноз измерения, $\bar{z}_t$, используя прогнозируемое местоположение робота и его неопределённость. На Рис. 7.9 показано прогнозирование измерения. На левых графиках показаны прогнозируемые местоположения робота и их эллипсы неопределённости. Истинное местоположение робота обозначено белым кругом. Теперь допустим, робот наблюдает ориентир впереди справа (показано жирной линией). На схемах справа показаны соответствующие прогнозируемые и текущие измерения в пространстве измерений. Прогнозируемое измерение $\bar{z}_t$ вычисляется из относительного расстояния и угла направления между прогнозируемым математическим ожиданием $\bar{\mu}_t$ и наблюдаемым ориентиром (строка 12). Неопределённость этого прогноза представлена эллипсом S_t. Аналогично прогнозированию состояния, неопределённость возникает из свёртки двух гауссовых функций. Эллипс Q_t представляет неопределённость шума измерений (строка 8), а эллипс $H_t\bar{\Sigma}_t H_t^T$ - неопределённость местоположения робота. Неопределённость местоположения робота $\bar{\Sigma}_t$ проектируется на неопределённость наблюдения путём умножения на якобиан функции измерения H_t, относящийся к местоположению робота (строка 13).

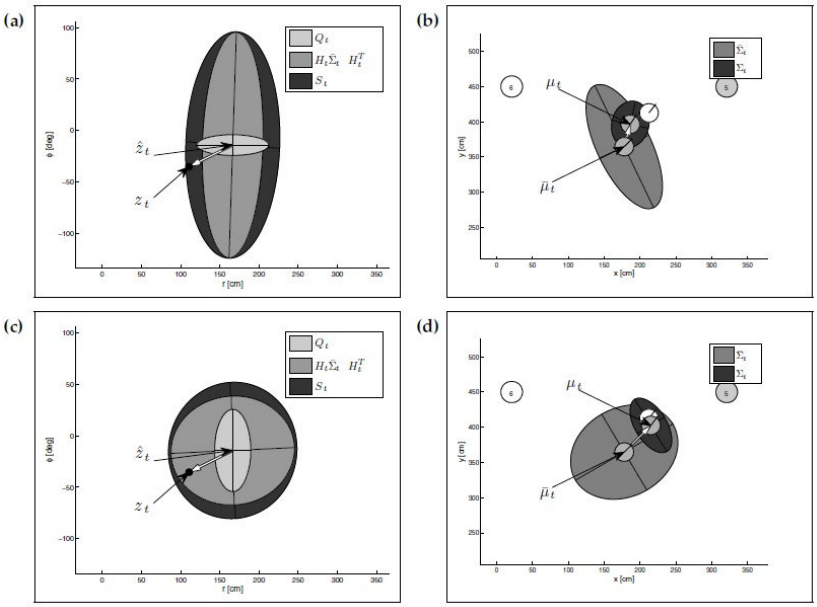


Рис. 7.10 Такт коррекции алгоритма EKF. На графиках слева показаны прогнозы измерения, а на графиках справа – результирующие коррекции, которые обновляют среднюю оценку и уменьшают эллипсы неопределённости положения.

НОВИЗНА ИЗМЕРЕНИЯ

S_t, общая неопределённость прогноза измерения, представляет собой сумму этих двух эллипсов (строка 14).

Белыми стрелками на графиках показаны так называемые *innovation vector - векторы новизны изменения* $z_t - \bar{z}_t$, представляющие собой просто разницу между наблюдаемым и прогнозируемым измерением. Эти векторы играет ключевую роль в последующем такте обновления. Он также предоставляет правдоподобие измерения z_t, которое задано правдоподобием векторов изменений для гауссовой функции с нулевым математическим ожиданием и ковариацией S_t (строка 21). Таким образом, чем короче "короткий" (в смысле расстояния Махаланобиса) вектор изменений, тем более вероятно измерение.

Такт коррекции: Обновление оценки (строки 15–21) Такт коррекции алгоритма локализации EKF показан на Рис. 7.10 и выполняет обновление оценки местоположения на основе вектора изменения и неопределённости прогноза измерения. Для удобства на графиках слева снова показана гипотеза измерения. На графиках справа показаны результирующие коррекции оценок местоположений, что показано белыми стрелками. Эти векторы коррекции вычисляются масштабной проекцией вектора изменений измерений (белые стрелки на левых графиках) в пространство состояний (строка 16) с помощью матрицы усиления Калмана K_t, вычисленной в строке 15. Очевидно, изменения измерения выражают смещение между прогнозируемым и наблюдаемым измерением. Это смещение затем проектируется в пространство состояний и используется для сдвига оценки местоположения в направлении уменьшения изменения. Усиление Калмана дополнительно масштабирует вектор изменений, таким образом, учитывая неопределённость в прогнозе измерения. Чем точнее измерение, тем выше усиление Калмана, и тем сильнее результирующая коррекция местополо-

жения. Эллипс неопределённости оценки местоположения обновляется похожим образом (строка 17).

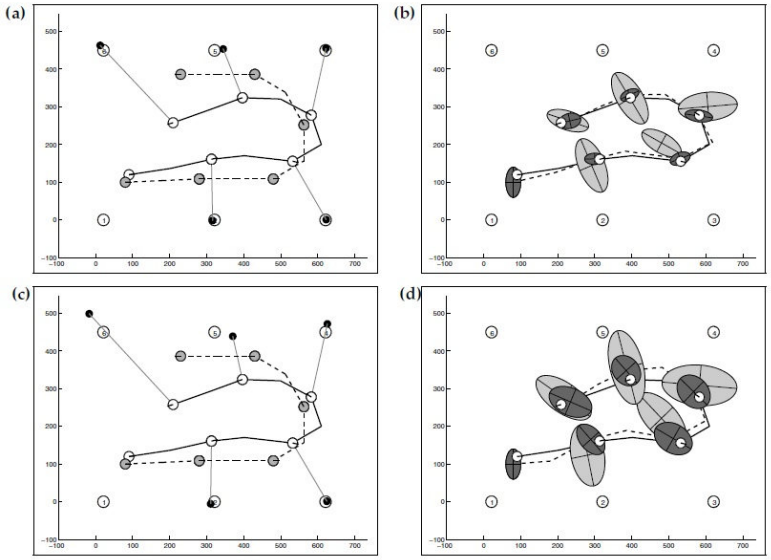


Рис. 7.11 Локализация на основе EKF с точным (верхний ряд) и неточным (нижний ряд) датчиками обнаружения ориентиров. Пунктирными линиями на левом графике показаны траектории робота, оценённые на основе управления движением. Сплошные линии обозначают реальное движение робота на основе этих управляющих воздействий. Обнаруженные в пяти точках ориентиры указаны тонкими линиями. Пунктирные линии на правых графиках демонстрируют скорректированные траектории робота, а также неопределённость до (светло-серый цвет, $\bar{\Sigma}_t$) и после (темно-серый цвет, Σ_t) учёта обнаружения ориентира.

Пример последовательности. На Рис. 7.11 показаны две последовательности обновлений EKF, используя различные неопределённости наблюдения. На левых графиках показаны траектории робота согласно управлению движением (пунктирные линии) и результирующие реальные траектории (жирные линии). Обнаружения ориентиров показаны тонкими линиями, измерения на верхних графиках менее защумлены. Пунктирными линиями на правых графиках показаны пути, вычисленные после оценки алгоритмом локализации EKF. Как и ожидалось, меньшая неопределённость измерения в верхнем ряду ведёт к меньшему размеру эллипсов неопределённости и меньшим ошибкам оценки.

7.5 Оценка соответствия

7.5.1 EKF локализация с неизвестным соответствием

Локализация с помощью EKF, обсуждаемая до текущего момента, применима только в случае, когда соответствия ориентиров могут быть определены с абсолютной точностью. На практике такое происходит редко, поэтому, в большинстве реализаций определение ориентира происходит во время локализации. В ходе изложения мы столкнёмся с рядом стратегий для решения

проблемы соответствия. Самая простая из них известна как максимальное правдоподобие соответствия.

МАКСИМАЛЬНОЕ ПРАВДОПОДОБИЕ СООТВЕТСТВИЯ

Сначала определяется самое вероятное значение переменной соответствия, а затем это значение принимается как эталонное.

Методы максимального правдоподобия неустойчивы, если для переменной соответствия имеется множество равновероятных гипотез. Однако, часто можно спроектировать систему таким образом, чтобы этого избежать. Для предотвращения ошибочной ассоциации данных имеется два метода. Во-первых, можно выбрать настолько сильно различающиеся и далеко расположенные друг от друга ориентиры, чтобы перепутать их было затруднительно. Во-вторых, каким-либо образом гарантировать малую неопределённость положения робота. К сожалению, эти два метода несколько противоречат друг другу, а нахождение верного разбиения ориентиров в окружающей среде сродни искусству.

В любом случае, методы максимального правдоподобия имеют огромное практическое значение. В Таблице 7.3 приведён алгоритм локализации на основе EKF с оценкой максимального правдоподобия для соответствия. Обновление движения в строках со 2 по 7 идентично таковому в Таблице 7.2. Ключевая разница заключается в обновлении измерения: для каждого измерения сначала вычисляется количество элементов, позволяющее определить наиболее вероятное соответствие для всех k ориентиров карты (строки с 10 до 15). Переменная соответствия $j(i)$ выбирается в строке 16 с помощью максимизации правдоподобия измерения z_t^i заданной для каждого возможного ориентира m_k на карте. Заметим, что эта функция правдоподобия идентична функции, используемой алгоритмом EKF для известного соответствия. Обновление EKF в строках 18 и 19 учитывает только наиболее вероятные соответствия.

Заметим, что алгоритм в Таблице 7.3 не слишком эффективен. Его можно улучшить более вдумчивым выбором ориентиров в строке 10. В большинстве ситуаций робот способен одновременно воспринимать только небольшое количество ориентиров в непосредственной близости, но с помощью простых тестов можно отбросить множество ориентиров с малой вероятностью на карте.

7.5.2 Математический вывод ассоциации данных с помощью ML

Оценочная функция максимального правдоподобия определяет соответствие, которое максимизирует правдоподобие данных.

1: **Algorithm EKF_localization** $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m)$:

2: $\theta = \mu_{t-1,\theta}$

3: $$G_t = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

4: $$V_t = \begin{pmatrix} \frac{-\sin\theta+\sin(\theta+\omega_t\Delta t)}{\omega_t} & \frac{v_t(\sin\theta-\sin(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\cos(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta-\cos(\theta+\omega_t\Delta t)}{\omega_t} & -\frac{v_t(\cos\theta-\cos(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\sin(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

5: $$M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2\omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4\omega_t^2 \end{pmatrix}$$

6: $$\bar{\mu}_t = \mu_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

7: $\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^T + V_tM_tV_t^T$

8: $$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

9: *для всех наблюдаемых признаков* $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ *выполнять*

10: *для всех ориентировкна карте твыполнять*

11: $q = (m_{k,x} - \bar{\mu}_{t,x})^2 + (m_{k,y} - \bar{\mu}_{t,y})^2$

12: $$\hat{z}_t^k = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(m_{k,y} - \bar{\mu}_{t,y}, m_{k,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \\ m_{k,s} \end{pmatrix}$$

13: $$H_t^k = \begin{pmatrix} -\frac{m_{k,x}-\bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{k,y}-\bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{k,y}-\bar{\mu}_{t,y}}{q} & -\frac{m_{k,x}-\bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

14: $S_t^k = H_t^k\bar{\Sigma}_t[H_t^k]^T + Q_t$

15: *endfor*

16: $j(i) = \underset{k}{\text{argmax}}\ \det(2\pi S_t^k)^{-\frac{1}{2}}\exp\{-\frac{1}{2}(z_t^i - \hat{z}_t^k)^T[S_t^k]^{-1}(z_t^i - \hat{z}_t^k)\}$

17: $K_t^i = \bar{\Sigma}_t[H_t^{j(i)}]^T[S_t^{j(i)}]^{-1}$

18: $\bar{\mu}_t = \bar{\mu}_t + K_t^i(z_t^i - \hat{z}_t^{j(i)})$

19: $\bar{\Sigma}_t = (I - K_t^i\,H_t^{j(i)})\bar{\Sigma}_t$

20: *endfor*

21: $\mu_t = \bar{\mu}_t$

22: $\Sigma_t = \bar{\Sigma}_t$

23: *return* μ_t, Σ_t

Таблица 7.3 Алгоритм локализации на основе обобщённого фильтра Калмана (EKF) с неизвестным соответствием. Соответствия $j(i)$ оцениваются с помощью алгоритма максимального правдоподобия.

(7.22)

$$\hat{c}_t = \underset{c_t}{\operatorname{argmax}}\, p(z_t | c_{1:t}, m, z_{1:t-1}, u_{1:t})$$

Здесь c_t вектор соответствия в момент времени t. Как и прежде, вектор $z_t = \{z_t^1, z_t^2, ...\}$ – это вектор измерений, содержащий список признаков или ориентиров z_t^i, наблюдаемых в момент времени t.

Оператор argmax в выражении (7.22) выбирает вектор соответствия $\hat{c}_t$, который максимизирует правдоподобие измерения. Заметим, что это выражение основано на предыдущих соответствиях $c_{1:t-1}$. Хотя они были оценены во время предыдущих шагов обновления, в методе максимального правдоподобия считается, что они всегда верны. Это имеет два важных последствия. Во-первых, это даёт возможность инкрементного обновления фильтра. Но, вместе с этим, фильтр становится неустойчивым, его значения отклоняются при ошибочных оценках соответствия.

Даже при условии известного предыдущего соответствия, присутствует экспоненциальное множество членов максимизации (7.22). При большом количестве обнаруженных ориентиров во время одного измерения число возможных соответствий может стать чрезмерно большим для практического использования. Наиболее часто используемый метод, позволяющий избежать экспоненциальной сложности, выполняет максимизацию для каждого индивидуального признака z_t^i в векторе измерений z_t. Мы уже вывели функцию правдоподобия для отдельных признаков в выводе алгоритма локализации на основе EKF с известными соответствиями. Следуя уравнениям с (7.17) до (7.20), соответствие каждого признака вычисляется как:

(7.23)

$$\begin{aligned}\hat{c}_t^i &= \underset{c_t^i}{\operatorname{argmax}}\, p(z_t^i | c_{1:t}, m, z_{1:t-1}, u_{1:t}) \\ &\approx \underset{c_t^i}{\operatorname{argmax}}\, \mathcal{N}(z_t^i; h(\bar{\mu}_t, c_t^i, m), H_t \bar{\Sigma}_t H_t^T + Q_t)\end{aligned}$$

Это вычисление выполняется в строке 16 в Таблице 7.3. Такая покомпонентная оптимизация «оправдана» только если известно, что отдельные векторы признаков условно независимы и обычно делается для удобства вычисления. При таком допущении максимизируемый член в выражении (7.22) выражен произведением членов с несовместными параметрами оптимизации. Максимум произведения достигается, когда каждый множитель максимален, как определено в (7.23). При использовании этой ассоциации данных максимального правдоподобия, правильность алгоритма напрямую следует из правильности алгоритма локализации на основе EKF с известными соответствиями.

7.6 Отслеживание нескольких гипотез

Существует несколько обобщений базового EKF, позволяющих приспособить его для ситуаций, при которых верная ассоциация данных не может быть определена с удовлетворительной надёжностью. Некоторые из этих методов будут обсуждаться позже в книге, поэтому сейчас остановимся на них лишь кратко.

Классический метод, который позволяет обойти ограничения ассоциации данных, это фильтр с*отслеживания нескольких гипотез* или MHT.

Фильтр MHT способен представлять оценку с помощью нескольких гауссовых функций. В нём апостериорное распределение представлено в виде смеси

(7.24)

$$bel(x_t) = \frac{1}{\sum_l \psi_{t,l}} \sum_l \psi_{t,l} \det(2\pi\Sigma_{t,l})^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_t - \mu_{t,l})^T \Sigma_{t,l}^{-1}(x_t - \mu_{t,l})\right\}$$

ВЕС СМЕСИ

Здесь l – индекс компонента смеси. Каждый такой компонент или «след» на жаргоне MHT, представляет собой гауссову функцию с математическим ожиданием $\mu_{t,l}$ и ковариацией $\Sigma_{t,l}$. Скаляр $\psi_{t,l} \geq 0$ это *вес смеси*, определяющий вес l-го компонента смеси в апостериорном распределении. Поскольку апостериорное распределение нормализовано с помощью $\Sigma_l \psi_{t,l}$, каждая величина $\psi_{t,l}$ – это относительный вес, и вклад l-го компонента смеси зависит от величины всех прочих весов смеси.

Как будет показано ниже при описании алгоритма MHT, каждый компонент смеси основан на уникальном решении последовательности ассоциации данных. Поэтому, имеет смысл записать $c_{t,l}$ для вектора ассоциации данных, связанного с l-им следом, а $c_{1:t,l}$ – для всех прошлых и настоящих ассоциаций данных, связанных с l-м компонентом смеси. В такой записи можно считать смесь компонентов в виде содействующих функций локальных оценок, основанных на уникальной последовательности ассоциации данных:

(7.25)

$$bel_l(x_t) = p(x_t | z_{1:t}, u_{1:t}, c_{1:t,l})$$

Здесь $c_{1:t,l} = c_{1,l}, c_{2,l}, ..., c_{t,l}$ определяют последовательность векторов соответствия, связанных с l-м следом.

Прежде чем описывать MHT, обсудим полностью неразрешимый алгоритм, из которого выводится MHT. Это алгоритм полной реализации теоремы Байеса для EKF с неизвестной ассоциацией данных. Его суть удивительно проста: вместо того, чтобы выбирать наиболее вероятный вектор ассоциации данных, наш вымышленный алгоритм сохраняет их все. А именно, пусть для момента времени t каждая смесь разделяется на множество новых смесей, каждая из которых основана на уникальном векторе соответствия c_t. Пусть m будет индексом одного из новых гауссовых распределений, а l – индексом для соответствия $c_{t,l}$, из которого это новое распределение было выведено. Вес этой новой смеси затем устанавливается как

(7.26)

$$\psi_{t,m} = \psi_{t,l}\, p(z_t | c_{1:t-1,l}, c_{t,m}, z_{1:t-1}, u_{1:t})$$

Это произведение веса смеси $\psi_{t,l}$, из которой был выведен новый компонент и правдоподобия измерения z_t при определённом векторе соответствия, приводящему к образованию нового компонента смеси. Другими словами, можно вычислить соответствие латентной переменной и апостериорное правдоподобие того, что компонент смеси верен. Способ вычисления правдоподобия измерения $p(z_t | c_{1:t-1,l}, c_{t,m}, z_{1:t-1}, u_{1:t})$ уже известен из выражения (7.26). Это просто правдоподобие измерения, вычисленное в строке 21 алгоритма локализации на основе EKF для известных ассоциаций данных (Таблица 7.2). Отсюда, можно последовательно вычислить веса смеси каждого нового компонента. Единственным недостатком этого алгоритма является факт экспоненциального увеличения со временем количества компонентов смеси, или следов.

ПРУНИНГ

Алгоритм MHT является приближением описанного метода с сохранением небольшого количества компонентов смеси. Этот процесс называется прунинг (pruning - отсечение). Во время прунинга уничтожается каждый компонент с относительным весом смеси

(7.27)
$$\frac{\psi_{t,l}}{\sum_m \psi_{t,m}}$$

менее порогового значения ψ_{min}. Легко увидеть, что количество компонентов смеси всегда будет максимум ψ_{min}^{-1}. Таким образом, в MHT сохраняется компактный вид апостериорного распределения, которое можно эффективно обновлять. Приближение достигается сохранением очень малого числа гауссовых функций, но на практике и количество вероятных местонахождений робота обычно очень невелико.

Пропустим формальное описание алгоритма MHT, предоставив читателю возможность ознакомиться с большим числом подобных алгоритмов, описанных в книге. Стоит отметить, что при реализации алгоритмов MHT полезно заранее определять стратегии идентификации следов с низким правдоподобием.

7.7 Локализация на основе UKF

UKF локализация – это алгоритм локализации робота на основе признаков, использующий unscented Kalman filter. Как описано в подразделе 3.4, для линеаризации моделей движения и измерения в UKF используется unscented преобразование. Вместо вычисления производных этих моделей unscented преобразование представляет гауссианы с помощью сигма-точек, а затем передаёт их в модели. В Таблице 7.4 приводится алгоритм UKF локализации робота на основе ориентиров. Предполагается, что в наблюдении z_t содержится единственное наблюдение ориентира, а идентификатор этого ориентира известен заранее.

7.7.1 Математический вывод UKF локализации

Главным образом, локализация и общий алгоритм UKF, приведённый в Таблице 3.4, различаются способом прогнозирования и измерения шумов. Напомним, что UKF из Таблице 3.4 основан на допущении аддитивности прогноза и измерения. В силу этого, возможно вычислить зашумление простым суммированием R_t и Q_t и прогнозируемой неопределённости состояния и измерения, соответственно (строки 5 и 9 в Таблице 3.4).

1: **Algorithm UKF_localization** $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, m)$:
Генерировать дополненные ковариацию и ожидание

2: $M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2 \omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4 \omega_t^2 \end{pmatrix}$

3: $Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$

4: $\mu_{t-1}^a = (\mu_{t-1}^T \ (0\ 0)^T \ (0\ 0)^T)^T$

5: $\Sigma_{t-1}^a = \begin{pmatrix} \Sigma_{t-1} & 0 & 0 \\ 0 & M_t & 0 \\ 0 & 0 & Q_t \end{pmatrix}$

Генерация сигма-точек

6: $\mathcal{X}_{t-1}^a = (\mu_{t-1}^a \ \ \mu_{t-1}^a + \gamma\sqrt{\Sigma_{t-1}^a} \ \ \mu_{t-1}^a - \gamma\sqrt{\Sigma_{t-1}^a})$

Пропускание сигма-точек через модель движения и вычисление статистик гауссовых функций

7: $\bar{\mathcal{X}}_t^x = g(u_t + \mathcal{X}_t^u, \mathcal{X}_{t-1}^x)$

8: $\bar{\mu}_t = \sum_{i=0}^{2L} w_i^{(m)} \bar{\mathcal{X}}_{i,t}^x$

9: $\bar{\Sigma}_t = \sum_{i=0}^{2L} w_i^{(c)} (\bar{\mathcal{X}}_{i,t}^x - \bar{\mu}_t)(\bar{\mathcal{X}}_{i,t}^x - \bar{\mu}_t)^T$

Прогнозирование производных в сигма-точках и вычисление статистик гауссовых функций

10: $\bar{Z}_t = h(\bar{\mathcal{X}}_t^x) + \mathcal{X}_t^z$

11: $\hat{z}_t = \sum_{i=0}^{2L} w_i^{(m)} \bar{Z}_{i,t}$

12: $S_t = \sum_{i=0}^{2L} w_i^{(c)} (\bar{Z}_{i,t} - \hat{z}_t)(\bar{Z}_{i,t} - \hat{z}_t)^T$

13: $\Sigma_t^{x,z} = \sum_{i=0}^{2L} w_i^{(c)} (\bar{\mathcal{X}}_{i,t}^x - \bar{\mu}_t)(\bar{Z}_{i,t} - \hat{z}_t)^T$

Обновление значений среднего и ковариации

14: $K_t = \Sigma_t^{x,z} S_t^{-1}$

15: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

16: $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$

17: $p_{z_t} = \det(2\pi S_t)^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_t)^T S_t^{-1}(z_t - \hat{z}_t)\}$

18: *return* μ_t, Σ_t, p_{z_t}

Таблица 7.4 Алгоритм локализации unscented Kalman filter (UKF), сформулированный для карты на основе признаков и робота, оснащённого датчиками измерения расстояния и угла направления. В этой версии обрабатывается единичные наблюдения ориентиров и подразумевается знание точного соответствия. Величина L – размерность дополненного вектора состояний, заданного суммой размерностей состояния, управления и измерения.

Алгоритм UKF_localization реализует другой, более точный метод учёта влияния шумов на процесс оценки. Ключевым действием является дополнение состояния с помощью компонент, отображающих шумы управления и измерения. Размерность L дополненного состояния задана суммой размерности состояния, управления и измерения, которые, в данном случае, равны $3 + 2 + 2 = 7$ (для простоты сигнатура признака измерения игнорируется). Поскольку в расчёт принят гауссовый шум с нулевым математическим ожиданием, математическое ожидание μ_{t-1}^a оценки дополнен-

ного состояния задано средним оценки местоположения μ_{t-1} и нулевыми векторами шумов управления и измерения (строка 4). Ковариация оценки дополненного состояния Σ^a_{t-1}, заданная комбинацией ковариаций местоположения Σ_{t-1}, шумов управления M_t, и шумов измерения Q_t, вычисляется в строке 5.

Отображение оценки дополненного состояния в виде сигма-точек генерируется в строке 6, используя выражение (3.66) для unscented преобразования. В этом примере $\mathcal{X}^a_{t-1}$ содержит $2L+1 = 15$ сигма-точек, в каждой из которых содержатся компоненты в пространствах состояния, управления и измерения.

$$\mathcal{X}^a_{t-1} = \begin{pmatrix} \mathcal{X}^x_{t-1}{}^T \\ \mathcal{X}^{uT}_t \\ \mathcal{X}^{zT}_t \end{pmatrix} \tag{7.28}$$

Выберем смешанное представление индексов времени, чтобы убедиться, что $\mathcal{X}^x_{t-1}$ относится к x_{t-1}, а компоненты управления и измерения - к u_t и z_t, соответственно.

Компоненты местоположения $\mathcal{X}^x_{t-1}$ этих сигма-точек затем передаются в модель движения на основе скорости g, определённую уравнением (5.9). В строке 7 выполняется такт экстраполяции применением модели движения из уравнения (5.13), используя управляющее воздействие u_t с добавочным компонентом зашумления управления $\mathcal{X}^u_{i,t}$ для каждой сигма-точки:

$$\bar{\mathcal{X}}^x_{i,t} = \mathcal{X}^x_{i,t-1} + \begin{pmatrix} -\frac{\upsilon_{i,t}}{\omega_{i,t}} \sin\theta_{i,t-1} + \frac{\upsilon_{i,t}}{\omega_{i,t}} \sin(\theta_{i,t-1} + \omega_{i,t}\Delta t) \\ \frac{\upsilon_{i,t}}{\omega_{i,t}} \cos\theta_{i,t-1} - \frac{\upsilon_{i,t}}{\omega_{i,t}} \cos(\theta_{i,t-1} + \omega_{i,t}\Delta t) \\ \omega_{i,t}\Delta t \end{pmatrix} \tag{7.29}$$

где

$$\upsilon_{i,t} = \upsilon_t + \mathcal{X}^{u[\upsilon]}_{i,t} \tag{7.30}$$

$$\omega_{i,t} = \omega_t + \mathcal{X}^{u[\omega]}_{i,t} \tag{7.31}$$

$$\theta_{i,t-1} = \mathcal{X}^{x[\theta]}_{i,t-1} \tag{7.32}$$

сгенерированной из управления $u_t = (\upsilon_t \ \omega_t)^T$ и отдельных компонент сигма-точек. Допустим, $\mathcal{X}^{u[\upsilon]}_{i,t}$ отображает поступательную скорость υ_t для i-й сигма-точки. Прогнозируемые сигма-точки, $\bar{\mathcal{X}}^x_t$, таким образом, представляют местоположения робота, которые складываются из различных комбинация предыдущих управляющих воздействий и измерений.

В строках 8 и 9 вычисляется математическое ожидание и ковариация прогнозируемого местоположения робота, используя метод unscented преобразования. В строке 9 не требуется добавления компонента шумов движения, который был необходим в алгоритме из Таблицы 3.4. Это происходит потому, что после дополнения состояния прогнозируемые сигма-точки уже содержат шумы движения. Этот факт, вдобавок, делает ненужным копирование сигма-точек из прогнозируемой гауссовой функции (см. строку 6 в Таблице 3.4).

В строке 10, прогнозируемые сигма-точки используются для генерации сигма-точек измерений на основе модели, определённой выражением (6.40) из подраздела 6.6:

$$(7.33)\quad \bar{Z}_{i,t} = \begin{pmatrix} \sqrt{(m_x - \bar{\mathcal{X}}_{i,t}^{x[x]})^2 + (m_y - \bar{\mathcal{X}}_{i,t}^{x[y]})^2} \\ \operatorname{atan2}(m_y - \bar{\mathcal{X}}_{i,t}^{x[y]}, m_x - \bar{\mathcal{X}}_{i,t}^{x[y]} - \bar{\mathcal{X}}_{i,t}^{x[\theta]}) \end{pmatrix} + \begin{pmatrix} \mathcal{X}_{i,t}^{z[r]} \\ \mathcal{X}_{i,t}^{z[\phi]} \end{pmatrix}$$

В данном случае шум измерения считается аддитивным.

Оставшиеся шаги обновления идентичны базовому алгоритму UKF, приведённому в Таблице 3.4. В строках 11 и 12 вычисляются математическое ожидание и ковариация прогнозируемого измерения. Взаимная ковариация между местоположением робота и наблюдением определяется в строке 13. В строках с 14 по 16 обновляется оценка местоположения. Правдоподобие измерения вычисляется из вектора изменения и прогнозируемой неопределённости измерения аналогично алгоритму локализации EKF в Таблице 7.2.

7.7.2 Иллюстрация

Проиллюстрируем алгоритм локализации UKF, используя те же примеры, которые были применены для алгоритма локализации EKF. Читателю предоставляется возможность самостоятельно сравнить следующие рисунки с приведёнными в подразделе 7.4.4.

Такт экстраполяции (Строки 2–9) На Рис. 7.12 показан такт экстраполяции UKF для разных параметров шумов движения. Компоненты местоположения $\mathcal{X}_{t-1}^{x}$ сигма-точек, сгенерированных их предыдущей гипотезы обозначены крестиками, расположенными симметрично вокруг μ_{t-1}. 15 сигма-точек обозначают семь разных местоположений робота, из которых после проекции на плоскость видны только пять.

Две дополнительные точки «сверху» и «снизу» средней сигма-точки отображают различные ориентации направления робота. Дугами показан прогноз движения, выполненный в строке 7. Как можно увидеть, было сгенерировано 11 предположений на основе различных комбинаций предыдущих местоположений и шумов движения. На схемах показано воздействие шумов движения на результат обновления. Среднее $\bar{\mu}_t$ и эллипс неопределённости $\bar{\Sigma}_t$ прогнозируемого местоположения робота генерируется из прогнозируемых сигма-точек.

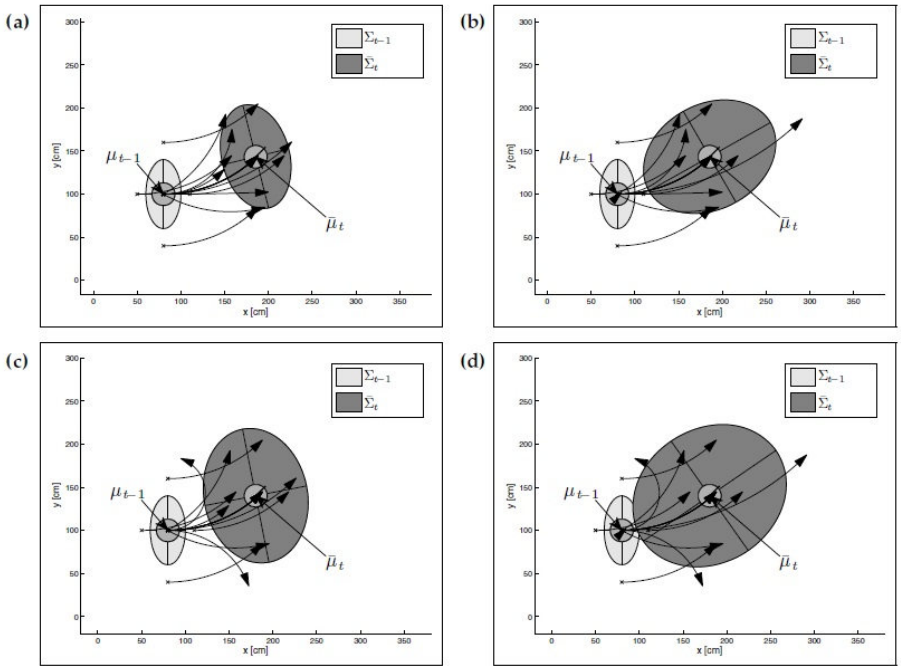


Рис. 7.12 Этап экстраполяции алгоритма UKF. Графики были сгенерированы с различными параметрами шумов движения. Начальная оценка робота изображена в виде эллипса с центром в μ_{t-1}. Робот движется по дуге длиной 90 см, поворачиваясь на 45 градусов влево. На графике (a) шумы движения относительно малы как для поступательной, так и для вращательной скорости. На других графиках показаны: (b) большие шумы поступательного движения, (c) большие шумы вращательного движения (d) высокие шумы как поступательного, так и вращательного движения.

Экстраполяция измерения (Строки 10–12) В такте экстраполяции измерения прогнозируемые местоположения робота $\bar{\mathcal{X}}_t^x$ используются для генерации сигма-точек измерения $\bar{Z}_t$ (строка 10). Чёрными крестиками на левых графиках Рис. 7.13 показаны локальные сигма-точки, а белые крестики на правых графиках обозначают результирующие сигма-точки измерения. Заметим, что 11 разных сигма-точек положения генерируют 15 разных измерений из-за разных компонентов шумов измерений $\mathcal{X}_t^z$, добавленных в строке 10. На графиках также показаны математическое ожидание $\hat{z}_t$ и эллипс неопределённости S_t прогнозируемого измерения, которые были вычислены в строках 11 и 12.

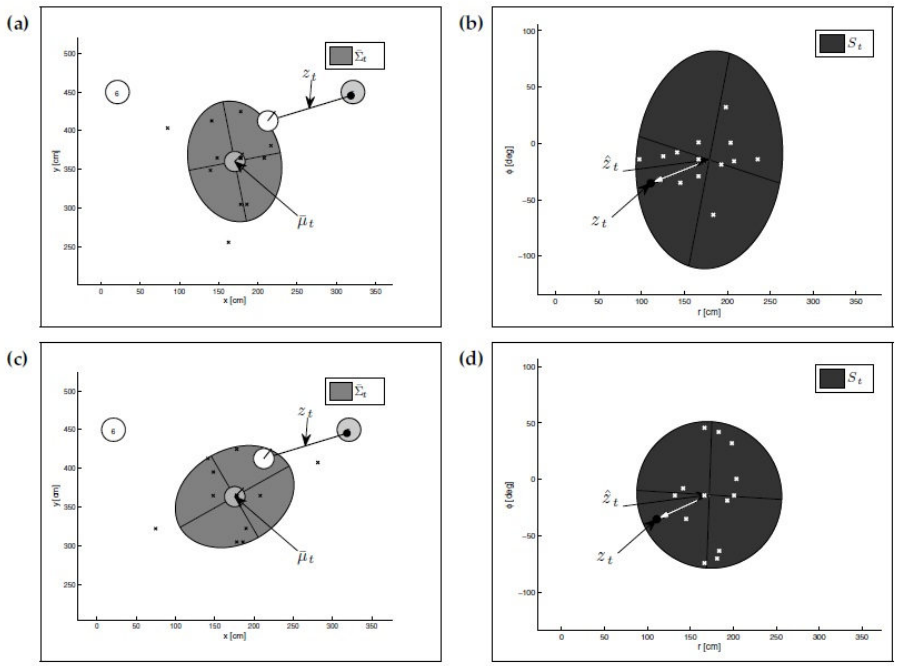


Рис. 7.13 Прогноз измерения. На графиках слева показаны сигма-точки, прогнозируемые на основе двух обновлений движения, и их результирующие эллипсы неопределённости. Истинное положение робота и наблюдение показаны белым кругом и жирной линией, соответственно. На графиках справа показаны результирующие сигма-точки прогноза измерения. Белыми стрелками показаны векторы изменения, разницы между наблюдаемыми и прогнозируемыми измерениями.

Такт коррекции: Обновление оценки (Строки 14–16) Такт коррекции алгоритма локализации UKF практически идентичен такту коррекции EKF. Вектор изменения и неопределённость прогноза измерения используются для обновления оценки, как показано белой стрелкой на Рис. 7.14.

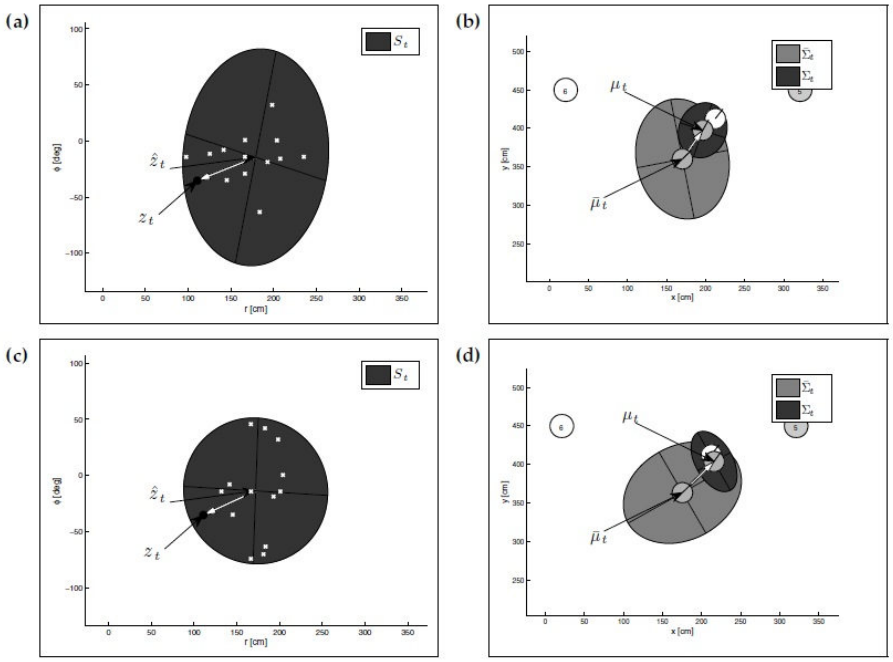


Рис. 7.14 Такт коррекции алгоритма UKF. На схемах с левой стороны показаны прогноз измерения, а на схемах справа – результирующие коррекции, позволяющие после обновления значений средней оценки уменьшить эллипсы неопределённости положения.

Пример На Рис. 7.15 показана последовательность оценок местоположения, сгенерированная многочастичным фильтром (сверху справа), EKF (снизу слева) и UKF (снизу справа). На схеме сверху слева показана траектория робота согласно управлению движением (пунктирная линия) и результирующая истинная траектория (жирная линия). Обнаруженные ориентиры показаны тонкими линиями. Пунктирные линии на других трёх графиках обозначают пути, рассчитанные с помощью различных методов. Ковариации оценок многочастичного фильтра извлекаются из выборок до и после обновления измерения (см. Таблицу 8.2). Оценки многочастичного фильтра приведены для справки, поскольку в нем не выполняется линеаризация приближения. Как можно увидеть, оценки EKF и UKF очень близки к этим эталонным оценкам, но UKF чуть ближе.

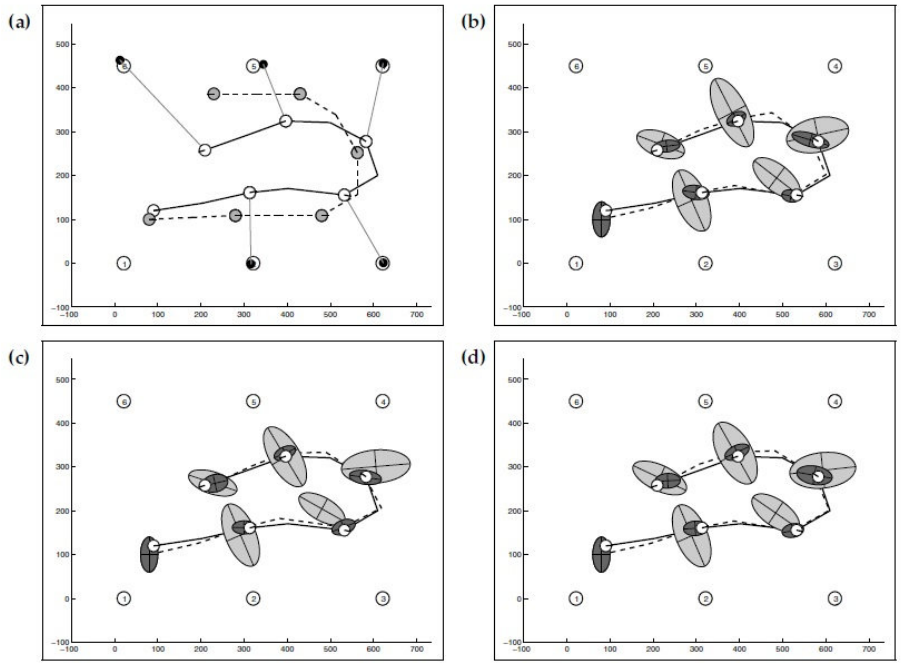


Рис. 7.15 Сравнение оценок UKF и EKF: (a) Траектория робота согласно сигналам управления движением (пунктирные линии) и результирующая истинная траектория (сплошные линии). Обнаружения ориентиров показаны тонкими линиями. (b) Справочные оценки, сгенерированные многочастичным фильтром. Оценки EKF (c) и UKF (d).

Влияние улучшенного алгоритма линеаризации, использованного в UKF, явно видно в примере на Рис. 7.16. Здесь робот выполняет две команды перемещения по кругу, изображённому тонкой линией. На графиках показаны эллипсы неопределённости после двух передвижений (робот не выполняет наблюдений). Ковариации, извлеченные из точных, основанных на выборке, обновлений движения, показаны для сравнения. Эталонные примеры были сгенерированы, используя алгоритм **sample_motion _model_velocity** из Таблицы 5.3. Хотя линеаризация EKF приводит к существенным ошибкам как в местоположении среднего, так и в «форме» ковариации, оценки UKF практически идентичны эталонным. В этом примере также видна небольшая разница между прогнозами EKF и UKF. Математическое ожидание, прогнозируемое EKF, всегда точно обозначает местоположение, прогнозируемое на основании данных управления (строка 6 в таблице 7.2). С другой стороны, математическое ожидание UKF извлекается из сигма-точек и может отличаться от математического ожидания управления (строка 7 в Таблице 7.4).

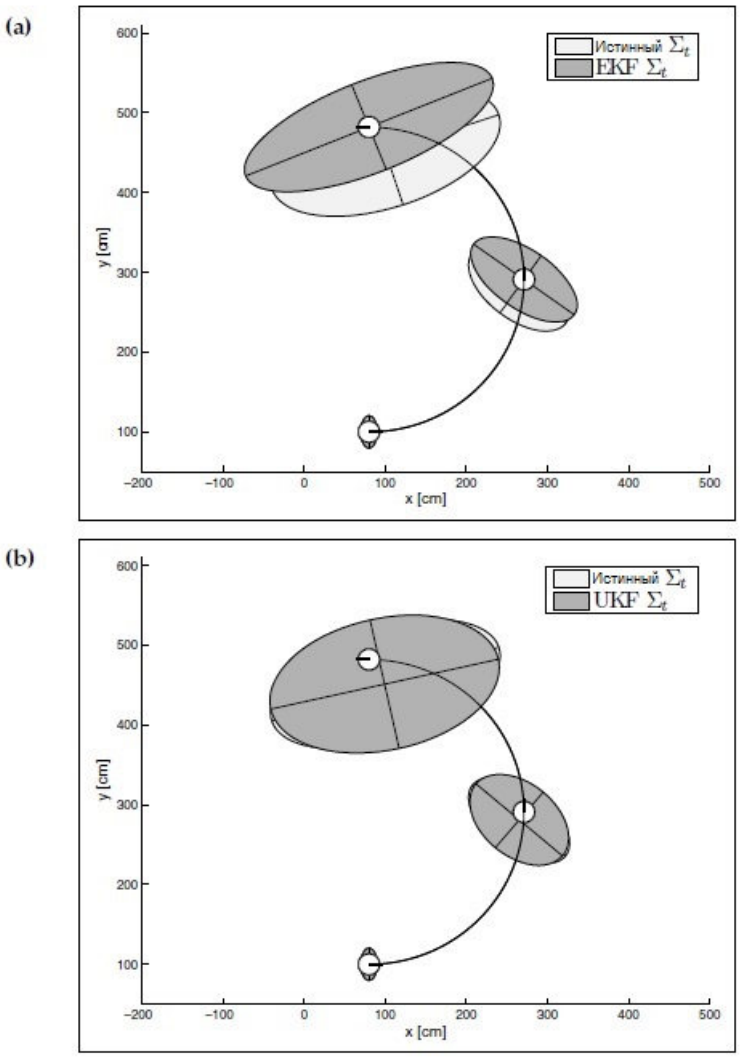


Рис. 7.16 Ошибка аппроксимации, вызванная линеаризацией при движении робота по кругу. Оценки основываются на прогнозе EKF (a) и прогнозе UKF (b). Эталонные ковариации извлечены из точного прогноза на основе выборки.

7.8 Практические соображения

Алгоритм локализации EKF и его близкий вариант, локализация MHT, являются популярными методами отслеживания позиции. Существует большое количество вариантов этих алгоритмов, улучшающих эффективность и надёжность.

• **Эффективный поиск.** Во-первых, перебор в цикле всех ориентиров k на карте, как это выполняется в алгоритме локализации EKF с неизвестным соответствием, часто оказывается непрактичным. Существуют довольно простые способы идентификации подходящих ориентиров (например, можно просто проектировать измерение в пространство $x-y$), позволяющие

отбросить все ориентиры, кроме конечного постоянного числа специально отобранных. Такие алгоритмы могут работать на порядки быстрее своих «наивных» реализаций.

• **Взаимное исключение**. Ключевое ограничение нашей реализации (которое наследуется и в MHT) возникает из-за предполагаемой независимости шумов признаков в EKF. Читатель может вспомнить условие (7.16), позволяющее последовательно обрабатывать отдельные признаки, что предотвращает возникновение операции поиска с экспоненциальной сложностью в пространстве всех векторов соответствия. К сожалению, такой подход позволяет назначить несколько наблюдаемых признаков, скажем z_t^i и z_t^j с $i \neq j$, одному и тому же ориентиру $\hat{c}_t^i = \hat{c}_t^j$ на карте. Для многих датчиков такое назначение признаков по умолчанию неверно. Например, если вектор признаков извлекается из единичного изображения с камеры, точно известно, что два различных региона пространства изображения должны соответствовать разным местоположениям физического мира. Другими словами, обычно известно, что $i \neq j \longrightarrow \hat{c}_t^i \neq \hat{c}_t^j$.

ПРИНЦИП ВЗАИМНОГО ИСКЛЮЧЕНИЯ ПРИ АССОЦИАЦИИ ДАННЫХ

Это (жёсткое!) ограничение называется принципом взаимного исключения при ассоциации данных. Оно позволяет уменьшает пространство всех возможных векторов соответствия и учитывается в хороших реализациях алгоритма. Например, можно сначала раздельно искать каждое соответствие, как в нашей версии локализации EKF, а затем выполнить «ремонтный» проход, в котором будут выявлены и устранены нарушения принципа взаимного исключения с помощью изменения значений соответствия.

• **Вычёркивание выбросов**. Далее, в предложенной реализации не обрабатываются выбросы. Читатель может вспомнить, что в подразделе 6.6 для соответствия было разрешено лишь $c = N + 1$ ориентиров, где N - общее число ориентиров на карте. Такой тест на выбросы довольно легко добавляется к алгоритмам локализации EKF. В частности, если установить π_{N+1} как априорную вероятность выброса, операция argmax в строке 16 локализации EKF (Таблица 7.3) может по умолчанию иметь значение $N + 1$, если наиболее вероятным объяснением вектора измерения является выброс. Вообще-то, выброс не содержит никакой информации о положении робота, поэтому все члены выражения в строках 18 и 19 в Таблице 7.3, имеющие отношение к положению, просто вычёркиваются.

Для задач отслеживания позиции применимы только локализации EKF и UKF, вдобавок, методы линеаризованных гауссовых функций хорошо работают только при малой неопределённости позиции. Это утверждение основано на нескольких взаимодополняющих факторах:

• Одномодальный гауссиан обычно является хорошим представлением неопределённости при отслеживании, если оно не входит в более глобальную задачу локализации.

• Даже при отслеживании позиции одномодальные гауссианы плохо подходят для отображения жёстких пространственных ограничений вида «робот может быть близко к стене, но не может находиться внутри неё». Значимость этих ограничений возрастает с ростом неопределённости местоположения робота.

• Узкий гауссиан уменьшает опасность ошибочного назначения соответствия. Это особенно важно для EKF, поскольку единственное неверное соответствие может дезориентировать трекер, вызвав целый поток ошибок локализации и соответствия.

• Линеаризация обычно хорошо работает только вблизи точки линеаризации. Как правило, если стандартное отклонение ориентации θ превышает ± 20 градусов, эффекты линеаризации, скорее всего, приведут к отказу как EKF, так и UKF алгоритмов.

В алгоритме MHT большинство этих проблем решены, правда, ценою увеличенной вычислительной сложности.

• С его помощью можно решить проблему глобальной локализации, инициализировав начальную гипотезу несколькими гауссовыми гипотезами. Гипотезы могут быть инициализированы по первым измерениям.

• Проблема похищенного робота может быть решена добавлением в смесь новых гипотез.

• Жёсткие пространственные ограничения так же трудно моделировать, и их лучше аппроксимировать несколькими гауссианами.

• Алгоритм MHT более надёжен при ошибках соответствия, хотя может отказать, если верное соответствие среди сохранённых в смеси гауссианов отсутствует.

• Обсуждаемый алгоритм MHT использует ту же самую линеаризацию, что и EKF, страдая от аналогичных негативных эффектов аппроксимации. Алгоритм MHT также возможно реализовать, используя UKF для каждой из гипотез.

Разработка подходящих признаков для алгоритмов гауссовой локализации является, в некотором роде, искусством. Это происходит потому, что необходимо достичь нескольких взаимоисключающих целей. С одной стороны, хочется иметь достаточно много признаков в среде, чтобы сохранить малую степень неопределённости положения робота, которая абсолютно необходима в силу указанных выше причин. С другой стороны, желательно минимизировать шансы перепутать ориентиры друг с другом, или же ошибочно обнаружить признак. Во многих средах не так много точечных ориентиров, которые можно надёжно распознать, поэтому, зачастую, в реализациях предполагаются достаточно свободно распределённые ориентиры. Здесь у алгоритма MHT есть явное преимущество в силу большей надёжности при ошибках ассоциации данных. Как правило, большое количество ориентиров работает лучше даже для EKF и UKF. При плотном расположении ориентиров критически важно использовать взаимное исключение при ассоциации данных.

ОТРИЦАТЕЛЬНАЯ ИНФОРМАЦИЯ

Наконец, заметим, что в процессе локализации в EKF и UKF обрабатывается только часть информации измерения. При переходе от измерений к признакам количество обрабатываемой информации и так сильно уменьшается. Более того, ни EKF, ни UKF методы локализации неспособны обрабатывать отрицательную информацию, указывающую на отсутствие признака. Очевидно, что отсутствие признака там, где он должен быть, тоже

является релевантной информацией. Например, если из какого-то места в Париже не видна Эйфелева башня, очень маловероятно, что это место расположено близко к ней. Проблема отрицательной информации состоит в том, что она порождает негауссовы гипотезы, которые невозможно представить в виде математического ожидания и дисперсии. Именно поэтому в реализациях EKF и UKF отрицательная информация просто игнорируется, а учитывается только информация от обнаруженных признаков. Стандартный алгоритм MHT также не учитывает отрицательной информации. Однако, её можно включить в вес смеси, разложив компоненты, для которых не удалось обнаружить ориентир.

Учитывая все перечисленные ограничения, не означает ли это, что методы локализации на основе гауссианов ненадёжны? Ответ - нет. EKF, UKF, и, особенно, MHT удивительно устойчивы к нарушениям допущений линейных систем. Фактически, успешная локализация зависит только от успешной ассоциации данных. Позже в книге мы столкнёмся с более сложными методами обработки соответствия, нежели только что обсуждаемые. Многие из этих методов применялись (и будут применяться) к нормальным распределениям, и результирующие алгоритмы относятся к лучших из ныне известных.

7.9 Выводы

В этой главе была представлена проблема локализации мобильного робота и выведен первый практический алгоритм ее решения.

• Проблема локализации состоит в оценке положения робота относительно известной карты окружающей среды.

• Отслеживание позиции относится к проблемам обработки локальной неопределённости положения робота, с известным начальным положением. Глобальная локализация - более общая проблема локализации робота «с нуля». Проблема похищенного робота – это проблема локализации в которой робот, имеющий данные о своём местоположении, мгновенно переносится в другое место. Эта проблема является самой трудной.

• Трудность проблемы локализации также является функцией изменчивости окружающей среды со временем. Все обсуждаемые алгоритмы подразумевают статическую окружающую среду.

• Пассивные методы локализации основаны на использовании фильтров. В них обрабатываются данных, полученные роботом, но отсутствует вмешательство в управление. Методы активной локализации управляют роботом при локализации с целью минимизации неопределённости. До настоящего момента изучались только пассивные алгоритмы. Активные алгоритмы будут обсуждаться в Главе 17.

• Марковская локализация является просто другим названием фильтра Байеса, применённого для проблемы локализации мобильного робота.

• Локализация на основе EKF использует обобщенный фильтр Калмана и используется, в основном, для карт на основе признаков.

• Самый часто используемый метод решения проблемы соответствия - метод максимального правдоподобия. В нём подразумевается, что в каждый момент времени истинно соответствие с наибольшей вероятностью.

• Алгоритм отслеживания нескольких гипотез (MHT) для представления апостериорного распределения использует несколько соответствий в виде смеси гауссовых функций. Компоненты смеси создаются динамически и уничтожаются, если их общее правдоподобие падает ниже порогового значения, определённого пользователем.

• MHT более устойчив к проблемам ассоциации данных, чем EKF, но за счёт увеличения вычислительной сложности. MHT также возможно реализовать, используя UKF для отдельных гипотез.

• Локализация UKF использует unscented преобразование для линеаризации моделей движения и измерения в контексте проблемы локализации.

• Все гауссовы фильтры хорошо подходят для задач отслеживания локальной позиции с ограниченной неопределённостью и в средах с явно различимыми признаками. EKF и UKF менее применимы для глобальной локализации или для сред с множеством схожих объектов.

• Отбор признаков для гауссовых фильтров требует определённого навыка. Признаки должны быть достаточно уникальными, чтобы уменьшить шансы их перепутать, но, в то же время, их должно быть довольно много, чтобы робот обнаруживал признаки достаточно часто.

• Производительность гауссовых алгоритмов локализации может быть улучшена различными способами, такими как принудительное взаимное исключение при ассоциации данных.

В следующей главе будут обсуждаться альтернативные методы локализации, направленные на обход ограничений EKF, используя различные представления гипотезы робота.

7.10 Библиографические примечания

Локализация была названа «самой фундаментальной проблемой для создания мобильного робота с возможностями автономности» (Cox, 1991). Использование EKF для оценки состояния роботов, действующих вне помещений впервые было выполнено Дикманнсом и Грэфи (Dickmanns and Graefe, 1988), которые применили EKF для оценки кривизны дороги по изображениям с камеры. Большая часть ранних работ по локализации мобильных роботах в помещениях была описана Боренштейном (Borenstein et al.,1996) (см. также (Feng et al., 1994)). Кокс и Вилфонг (Cox and Wilfong, 1990) представили ранний текст о наиболее современных на тот момент разработках в мобильной робототехнике, включив в него локализацию. Многие ранние методы требовали модификации окружающей среды, например, с помощью искусственных меток. Например, Леонард и Дюран-Уайт (Leonard and Durrant-Whyte, 1991) использовали EKF при сравнении геометрических маркеров, извлекая данные о них из показаний сонара, а также используя бакены, расположенные с учётом геометрической карты окружающей сре-

ды. Практика использования искусственных маркеров сохранилась до сегодняшнего дня (Salichs et al, 1999), поскольку модификация окружающей среды часто оправдана и выгодна. Другие исследователи применяли лазеры для сканирования не модифицированных сред (Hinkel and Knieriemen, 1988).

Уходя от точечных признаков, некоторые исследователи разработали новые геометрические методы локализации. Например, Кокс (Cox, 1991) разработал алгоритм сравнения расстояний, измеренных инфракрасными датчиками, и описания окружающей среды в виде совокупности линейных сегментов. Метод, разработанный Вейсом (Weiss et al., 1994) использовал для локализации корреляцию измерений расстояния. Идея наложения карт, в частности, сравнения локальной карты сетки занятости с глобальной картой среды, принадлежит Моравицу (Moravec, 1988). Метод локализации градиентным спуском на базе этой идеи был описан Труном (Thrun, 1993), и использован в первом соревновании AAAI в 1992 году (Simmons et al. 1992). Шили и Краули (Schiele and Crowley, 1994) выполнили сравнение различных стратегий отслеживания позиции робота с помощью ультразвуковых датчиков на основе карт сеток занятости. Они показали, что наложение локальных карт сеток занятости на глобальную карту с сеткой показывает схожую производительность с наложением на основе признаков, извлечённых из обеих карт. Шаффер (Shaffer et al., 1992) сравнил устойчивость наложения карт и методов на основе признаков и показал, что их комбинация показывает наилучшие эмпирические результаты. Ямаучи и Лэнгли (Yamauchi and Langley, 1997) доказали устойчивость наложения карт к изменениям среды. Идея использования наложения проходов сканирования для локализации в робототехнике восходит к работам Лю и Милиоса (Lu and Milios, 1994, 1998), Гутмана и Шлегеля (Gutmann and Schlegel, 1996), хотя базовый принцип был популярен и в других областях (Besl and McKay 1992). Похожий метод был предложен Аррасом и Вестли (Arras and Vestli, 1998), которые показали, что, используя наложение проходов сканирования, возможно локализовать робота с исключительной точностью. Ортин (Ortin et al., 2004) обнаружил, что использование соечтания камеры с лазерным дальномером увеличивает устойчивость наложения измерений расстояния.

Другая ветвь исследований была посвящена геометрическим методам локализации (Betke and Gurvits 1994). Термин «проблема похищенного робота» принадлежит Энгельсону и МакДермотту (Engelson and McDermott, 1992). Название «марковская локализация» было введено Симмонсом и Кёнигом (Simmons and Koenig, 1995), которые предложили алгоритм локализации с сеткой для отображения апостериорных вероятностей. Однако, корни этой работы восходят к Норбахшу (Nourbakhsh et al., 1995), который разработал идею «коэффициентов уверенности» для локализации мобильного робота. Хотя правила обновления для коэффициентов уверенности не следовали строго теоремам вероятности, в них была отражена важная идея оценки нескольких гипотез. В основополагающей работе Кокса и Леонарда (Cox and Leonard, 1994) эта идея также была исследована в виде динамически поддерживаемых деревьев гипотез локализации робота. Использование нечёткой логики для локализации робота было предложена Саффотти (Saffiotti, 1997). Также можно обратиться к работам Дрянкова и Саффиотти (Driankov and Saffiotti, 2001).

7.11 Упражнения

1. Допустим, робот оснащён датчиком для измерения расстояния и направления на ориентир. Для простоты допустим, что при измерении робот распознаёт и обозначение ориентира (датчик определения ориентира идеальный и не подвержен шумам). Требуется выполнить глобальную локализацию, используя EKF. Обратить внимание, что при обнаружении одиночного ориентира апостериорная вероятность обычно плохо аппроксимируется с помощью гауссовой функции. Однако, при обнаружении двух и более ориентиров одновременно, апостериорное распределение хорошо приближается с помощью гауссиана.

(a) Объяснить, почему это происходит.

(b) Даны k одновременных измерений расстояния и направления для k опознаваемых ориентиров. Разработать процедуру вычисления гауссовой оценки положения робота с равномерным начальным априорным распределением. Следует начать с модели измерения расстояния/направления, представленной в подразделе 6.6.

2. В этом упражнении потребуется создать жёсткие окружающие среды для глобальной локализации. Допустим, необходимо составить плоскую окружающую среду из n непересекающихся прямых отрезков. Свободное пространство среды замкнуто, но внутри карты может быть замкнутый островок занятой территории. Для этого упражнения допустим, что робот оснащён круговым массивом из 360 датчиков расстояния, датчики идеальны и их показания всегда точны.

(a) Каково максимальное количество отдельных модов в функции оценки может встретиться при глобальной локализации? Для $n = 3, ..., 8$ нарисовать среды, вызывающие наибольшие трудности, вместе с применимой гипотезой, максимизирующей количество модов.

(b) Изменится ли алгоритм анализа, если разрешить датчикам расстояния ошибаться? В частности, дать пример для $n = 4$, с большим количеством правдоподобных модов, чем в примере выше. Показать такую среду, обозначив (ложные) показания расстояния и апостериорное распределение.

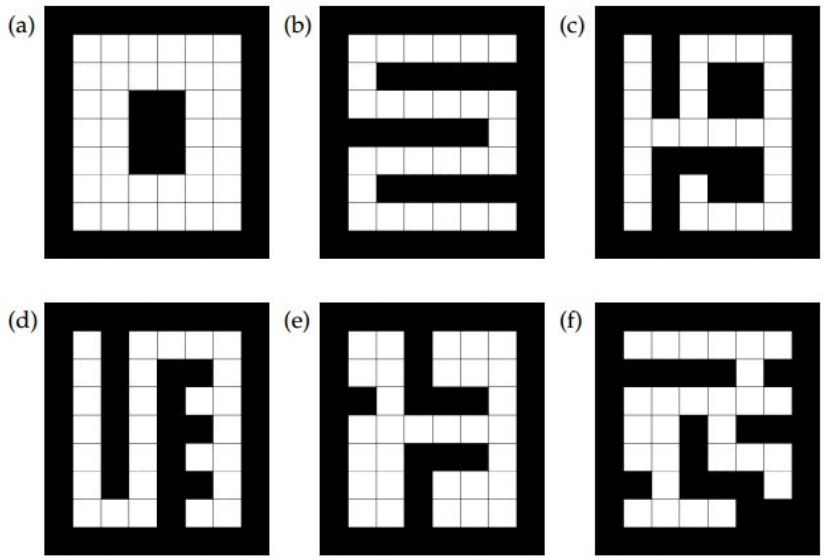
3. Требуется вывести алгоритм локализации EKF для простейшего подводного робота. Робот находится в трехмерном пространстве и оснащён идеальным компасом (то есть ориентация по направлению всегда известна). Для простоты допустим, что робот движется независимо по всем трём перпендикулярным осям (x, y, z), управляя скоростями $\dot{x}, \dot{y}, \dot{z}$. Шумы движения выражены нормальными распределениями и независимы по всем трём направлениям.

Робот окружен некоторым количеством бакенов, которые излучают акустические сигналы. Время момента включения каждого сигнала известно, но робот способен по сигналу определить, каким именно бакеном он был излучён (поэтому проблемы соответствия данных нет). Робот также знает местонахождение всех бакенов и оснащён точным хронометром для замера времени прохождения каждого сигнала. Однако, робот неспособен определить направление, с которого был получен сигнал.

(a) Необходимо вывести алгоритм локализации о робота на основе EKF. Это включает математический вывод моделей движения и измерения, а также аппроксимацию разложением в ряд Тейлора. Также необходимо определить финальный алгоритм на основе EKF с известным соответствием.

(b) Реализовать алгоритм на основе EKF и имитацию среды. Проверить точность и режимы отказов алгоритма локализации EKF в контексте трёх проблем локализации: глобальной локализации, отслеживания позиции и проблемы похищенного робота.

4. Представить упрощённую глобальную локализацию в любой из шести следующих сред с наложенной сеткой:



В каждой среде робот размещается в случайном месте, ориентированный по направлению на север. Требуется вывести локализацию без обратной связи, которая будет содержать последовательность следующих команд:

Действие L: Повернуть налево на 90 градусов.
Действие R: Повернуть направо на 90 градусов.
Действие M: Двигаться вперёд до столкновения с препятствием.

В конце стратегии робот должен оказаться в определимой позиции. Для каждой такой среды представить кратчайшую последовательность (считать только действия “M”). Указать, в каком месте окажется робот после окончания последовательности действий. Если такой последовательности не существует, объяснить, почему.

5. Теперь допустим, что робот способен считать количество шагов после выполнении команды “M” из предыдущего упражнения. Какова будет кратчайшая последовательность для определения местоположения? Объяснить ответ.

Примечание: В этом вопросе может оказаться, что конечное местоположение робота является функцией начального положения. Требуется только

выполнить локализацию робота.

8 Локализация мобильного робота: методы Монте-Карло и построения сеток

8.1 Введение

В этой главе описаны два алгоритма локализации, способных решать проблему глобальной локализации. Обсуждаемые здесь алгоритмы имеют ряд отличий от одномодальных гауссовых методов, обсуждаемых в предыдущей главе.

• Они способны обрабатывать «сырые» показания датчиков. Нет необходимости извлекать признаки из данных показаний датчиков, что напрямую обеспечивает возможность обрабатывать отрицательную информацию.

• Они непараметрические. В частности, они не привязаны к одномодальному распределению, как это было в случае с алгоритмом локализации EKF.

• Они способны решать проблему глобальной навигации, а в некоторых реализациях – и похищенного робота. Алгоритм EKF на такое неспособен, хотя MHT (отслеживание нескольких гипотез) возможно модифицировать для решения задачи глобальной локализации.

Представленные методы показали великолепные результаты в ряде робототехнических систем, действующих в реальных условиях.

Первый метод называется *локализация по сетке*. В нем для отображения апостериорной оценки применен фильтр на основе гистограмм. С применением локализации по сетке связан ряд ограничений: при использовании мелкого разбиения сетки вычислительные затраты, требующиеся для работы наивного алгоритма, неприемлемо велики. Для крупной сетки дополнительная потеря информации в результате дискретизации негативно влияет на фильтр и, если её корректно не обрабатывать, может даже нарушить его работу.

1: **Algorithm Grid_localization**$(\{p_{k,t-1}\}, u_t, z_t, m)$:
2: *for all k do*
3: $\quad \bar{p}_{k,t} = \sum_i p_{i,t-1}$**motion_model**$(\text{mean}(\text{x}_k), u_t, \text{mean}(\text{x}_i))$
4: $\quad p_{k,t} = \eta\, \bar{p}_{k,t}$**measurement_model**$(z_t, \text{mean}(\text{x}_k), m)$
5: *endfor*
6: *return* $\{p_{k,t}\}$

Таблица 8.1 Локализация по сетке, вариант дискретного байесовского фильтра. Функция **motion_model** реализует одну из моделей движения, а **measurement_model** - модель измерения. Функция “mean” возвращает центр масс ячейки сети x_k.

Второй метод - это локализация методом Монте-Карло (Monte Carlo

localization -MCL), наверное, самый популярный алгоритм локализации на сегодняшний день. В нем для получения апостериорных оценок положения робота используются многочастичные фильтры. Мы обсудим ряд недостатков MCL, а также представим варианты использования для задачи похищенного робота и динамических окружающих сред.

8.2 Локализация по сетке

8.2.1 Общий алгоритм

В алгоритме *локализации по сетке* апостериорное распределение аппроксимируется с использованием *фильтра на основе гистограмм* в декомпозиции пространства положений в виде сетки. Дискретный байесовский фильтр уже был подробно объяснён в Главе 4.1 и приведен в Таблице 4.1. Апостериорное распределение хранится в виде набора значений вероятности

(8.1)
$$bel(x_t) = \{p_{k,t}\}$$

где каждая вероятность $p_{k,t}$ определена для ячейки сетки x_k. Набор всех ячеек сетки образует разбиение пространства всех возможных положений:

(8.2)
$$\mathrm{domain}(X_t) = \mathrm{x}_{1,t} \cup \mathrm{x}_{2,t} \cup ... \mathrm{x}_{K,t}$$

В самой простой версии локализации по сетке разбиение пространства всех положений инвариантно по времени, и все ячейки одинакового размера. Для многих помещений популярно разбиение сеткой с ячейкой 15 см по осям x- и y, и 5 градусов для измерения поворота. Более мелкое представление даёт лучшие результаты, но ценой возросшей вычислительной нагрузки.

Локализация по сетке, по большей части, идентична базовому алгоритму фильтра на основе гистограмм, из которого она и произошла. В Таблице 8.1 приводится псевдокод для самой общей реализации. На вход необходимо подать дискретные значения вероятности $\{p_{t-1,k}\}$, а также самые последние измерения, действия управления и карту. Во внутреннем цикле выполняется перебор всех ячеек сетки. В строке 3 выполняется обновление модели движения, а в строке 4 – обновление измерения. Итоговые вероятности нормализуются с помощью нормализующего коэффициента η в строке 4. Функции **motion_model** и **measurement_model** могут быть реализованы с помощью любой модели движения из Главы 5 и модели измерений из Главы 6, соответственно. В алгоритме в Таблице 8.1 допускается, что каждая ячейка имеет одинаковый размер.

На Рис. 8.1 показан пример локализации по сетке для случая одномерного коридора. Эта схема аналогична базовому байесовскому фильтру, за исключением дискретного способа реализации. Как и прежде, робот начинает в ситуации глобальной неопределённости, выраженной равномерной гистограммой. По мере сбора данных в соответствующих ячейках сетки возрастают значения вероятности. Пример явно иллюстрирует возможность представления мультимодальных распределений с помощью локализации по сетке.

8.2.2 Разрешение сетки

Ключевым параметром для алгоритмов локализации по сетке является разрешение сетки. На первый взгляд, это может показаться малозначительной деталью. Но подходящий тип модели датчика, вычислительные мощности, требующиеся для вычисления оценки и тип ожидаемого результата - все они зависят от разрешения сетки.

В конечном счёте, есть два типа отображений, каждый из которых успешно проявил себя в практических робототехнических системах.

ТОПОЛОГИЧЕСКОЕ ОТОБРАЖЕНИЕ

Общепринятым подходом является определение сетки в *топологическом виде*. Результирующие сетки обычно очень грубы, а их разрешение зависит от структуры окружающей среды. Топологические разбиения разбивают пространство всех положений на области, соответствующие *важным местам* в окружающей среде. Такие места могут определяться по наличию (или отсутствию) определённых ориентиров, например, дверей и окон. В средах с коридорами такие места могут соответствовать пересечениям, разветвлениям, тупикам и так далее. Топологические отображения обычно довольно грубы, их разбиение среды зависит от структуры среды. На Рис. 8.5 показано такое разбиение для случая одномерного коридора.

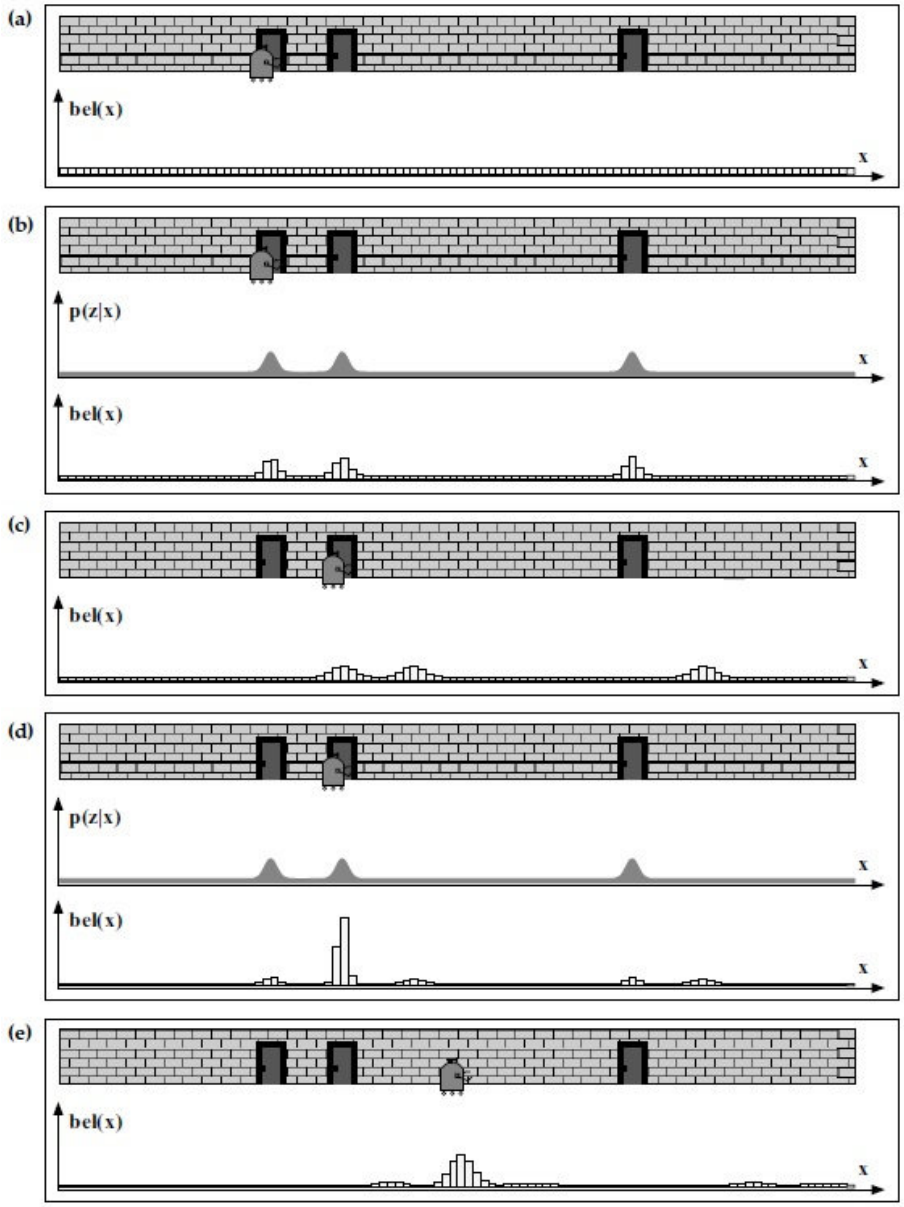


Рис. 8.1 Сеточная локализация, на основе мелкого метрического разбиения. На каждой схеме показана позиция робота в коридоре согласно его оценке $bel(x_t)$ по сетке, отображённая в виде гистограммы.

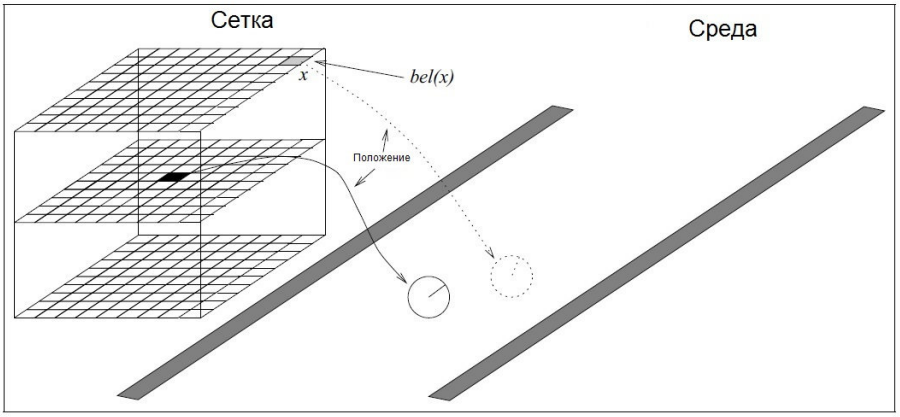


Рис. 8.2 Сетка с фиксированным разрешением по переменным положения робота x, y и θ. Каждая ячейка представляет собой положение робота в окружающей среде. Различные ориентации по углу направления соответствуют различным плоскостям сетки (показано только три ориентации).

МЕТРИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ

Значительно более мелкое разбиение обычно получается в *метрических представлениях*, где пространство состояний разбивается на ячейки одинакового размера. Разрешение таких разбиений обычно гораздо более высокое, по сравнению с топологическими сетями. Например, в некоторых примерах Главы 7 используются разбиения сетки с размером ячейки 15 см или менее, в силу этого они более точны, но ценой возросшей вычислительной сложности. На Рис. 8.2 показана такая сеть с фиксированным разбиением. Мелкое разбиение, наподобие показанного, обычно используется в метрическом представлении пространства.

При реализации локализации по сетке для грубых разбиений важно компенсировать разбиение разрешением моделей датчика и движения. В частности, для датчиков с высоким разрешением, наподобие лазерного датчика расстояния, значение модели измерения $p(z_t|x_t)$ может сильно варьироваться внутри каждой ячейки сетки $\mathrm{x}_{k,t}$. В таком случае простая оценка с помощью центра масс обычно даёт плохие результаты. Аналогично, прогнозирование движения робота на основе данных о центре масс ячейки тоже может дать плохие результаты. Если движение обновляется с интервалами 1 сек для робота, который двигается со скоростью 10 см/сек, и разрешением сетки 1 метр, наивная реализация никогда не выдаст перехода состояния! Это происходит потому, что любое местонахождение, удалённое на 10 см от центра масс ячейки, все ещё попадает внутрь одной и той же ячейки сетки.

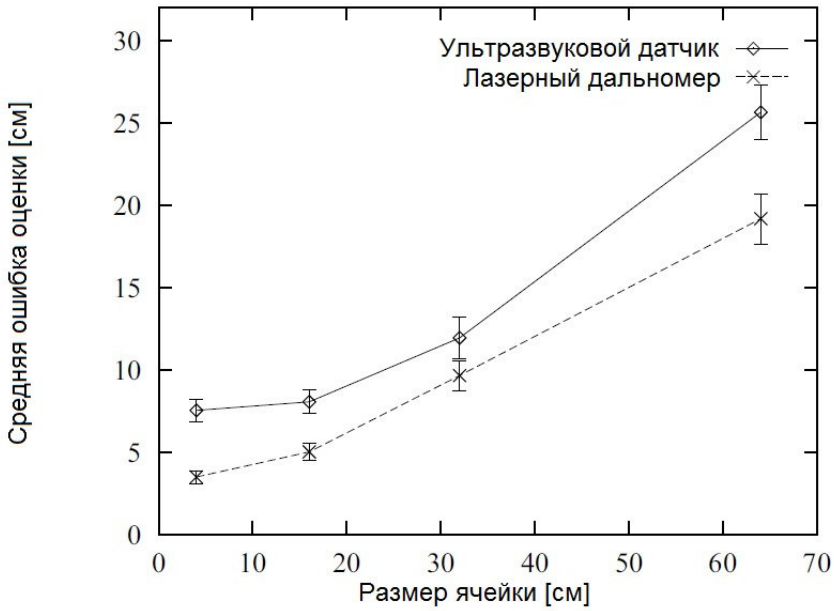


Рис. 8.3 Средняя ошибка локализации ультразвуковых и лазерных датчиков расстояния как функция размера ячейки сети.

Обычным способом компенсации этого эффекта является модификация как модели измерения, так и модели движения путём увеличения параметра зашумления. Например, дисперсия гауссовой модели основного конуса измерений датчика расстояния может быть увеличена на половину диаметра ячейки сети. Таким образом, новая модель станет значительно более гладкой, а ее интерпретация – менее подвержена зависимости от точного местоположения предполагаемой точки относительно истинного местоположения робота. С другой стороны, такая модифицированная модель измерений уменьшает количество информации, извлекаемой из измерений датчика.

Аналогично, и модель движения может выполнять прогнозирование случайного перехода на соседнюю ячейку с вероятностью, пропорциональной длине дуги движения, делённой на диаметр ячейки. Результатом работы такой огрублённой модели движения является возможность робота передвигаться от одной ячейки к другой, даже если его перемещение между последовательными обновлениями относительно размера ячейки сети достаточно мало. Однако, результирующие апостериорные распределения неверны в том, что неоправданно высокая вероятность будет назначена гипотезе, в которой робот переходит в другую ячейку при каждом обновлении движения, а, значит, перемещается гораздо быстрее, чем предполагалось сигналами управления.

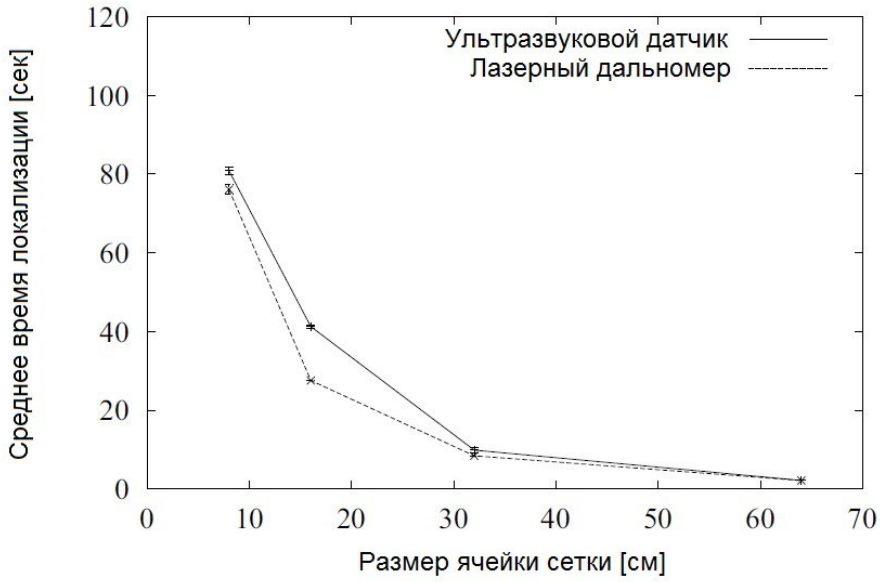


Рис. 8.4 Средние затраты процессорного времени, необходимые для глобальной локализации с помощью ультразвукового и лазерного датчика расстояния как функция разрешения сети.

На Рис. 8.3 и 8.4 показаны графики представления локализации по сетке в виде функции разрешения для разных датчиков расстояния. Как и ожидалось, ошибка локализации возрастает по мере уменьшения разрешения. Общее время, необходимое для локализации робота уменьшается по мере увеличения ячеек сети, как показано на Рис. 8.4.

8.2.3 Вычислительные соображения

При использовании сетки с мелкой ячейкой, наподобие описанных в примере в предыдущем разделе метрических сеток, общий алгоритм не может быть выполнен в реальном времени, поскольку ошибки возникают и при обновлении движения, и при обновлении измерения. Обновление движения требует выполнения свёртки, которая для трёхмерной сети является операцией в шести измерениях. Обновление измерения является операцией в трёх измерениях, но вычисление правдоподобия полного прохода сканирования весьма вычислительно затратно.

ПРЕДВАРИТЕЛЬНОЕ КЭШИРОВАНИЕ МОДЕЛИ

Существует ряд методов уменьшения вычислительной сложности локализации по сетке. *Предварительное кэширование модели* основано на том, что определённые модели измерений очень затратно вычислять. Например, вычисление модели измерения может потребовать бросания лучей, которые возможно предварительно вычислить для любой фиксированной карты. Как было указано в подразделе 6.3.4, общепринятой стратегией является вычисление для каждой ячейки основных статистик, влияющих на обновление измерения. В частности, при использовании модели на основе лучей принято кэшировать верное расстояние до каждой ячейки. Далее, модель измерения может быть вычислена для набора возможных значений расстояния с мелким разбиением. Вычисление модели измерения сводится к операции просмотра двух таблиц, что значительно быстрее.

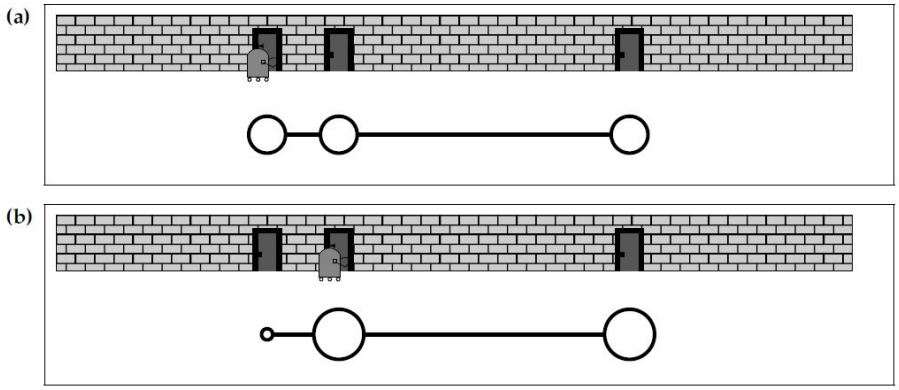


Рис. 8.5 Применение грубого топологического разбиения для локализации мобильного робота. Каждому состоянию соответствует конкретное место в среде (в данном случае, около двери). Гипотеза робота $bel(x_t)$ о нахождении в конкретном состоянии отображена в виде размера кругов. (a) Начальная оценка равномерна по всем положениям (b) Показана оценка после одного перехода состояния и обнаружения двери. В этой точке маловероятно, что робот все ещё находится в крайней левой позиции.

ПОДВЫБОРКА ЗНАЧЕНИЙ ДАТЧИКА

Подвыборка значений датчика даёт ещё больший выигрыш в скорости путём оценки модели измерения только для подмножества всех расстояний. В некоторых из наших систем мы использовали только 8 из 360 измерений расстояния лазером и получали отличные результаты. Подвыборка может выполняться в пространстве и по времени.

ЗАДЕРЖКА ОБНОВЛЕНИЯ ДВИЖЕНИЯ

При обновлении движения с задержкой обновление движения выполняется с меньшей частотой, чем частота управления или измерения робота. Это достигается геометрической интеграцией сигналов управления или показаний за короткий промежуток времени. Хороший метод обновления движения с задержкой легко способен ускорить алгоритм на порядок.

ВЫБОРОЧНОЕ ОБНОВЛЕНИЕ

Выборочное обновление уже было описано в подразделе 4.1.4. В методах выборочного обновления выполняется обновление только части всех ячеек сетки, например, в одной популярной реализации этой идеи обновляются только те ячейки, апостериорная вероятность которых превосходит указанный пользователем порог. Методы выборочного обновления способны уменьшить вычислительные затраты при обновлении гипотез на несколько порядков. Особое внимание следует уделять реактивным ячейкам сетки с малой вероятностью, если решение используется в задаче похищенного робота.

С такими модификациями локализация по сетке становится довольно эффективной. Даже 10 лет назад маломощные персональные компьютеры были достаточно эффективными, чтобы сгенерировать приведённые в данной главе результаты. Однако, предложенные модификации накладывают дополнительную нагрузку на программиста и усложняют итоговую реализацию по сравнению с коротким алгоритмом, предлагаемым в Таблице 8.1.

8.2.4 Иллюстрация

На Рис. 8.6 показан пример марковской локализации на основе метрических сеток с пространственным разрешением 15 сантиметров и угловым разрешением 5 градусов. На рисунке представлен процесс глобальной локализации, где робот, оснащённый двумя лазерными датчиками расстояния, выполняет локализацию «с нуля». Вероятностная модель датчиков расстояния вычислена с использованием модели на основе лучей, описанной в подразделе 6.3 и приведённой в Таблице 8.1.

Вначале вероятность местонахождения робота равномерно распределена по всему пространству состояний. На Рис. 8.6a приведён проход сканирования датчиков расстояния из начальной позиции робота. Здесь отброшены максимальные показания датчиков, а соответствующие части карты закрашены серым. После учёта прохода сканирования плотность вероятность местоположения робота сосредоточена только в нескольких областях в сильно симметричном пространстве, как показано градациями серого на Рис. 8.6b. Заметим, что оценки спроектированы в пространство $x - y$. Истинная оценка определена и по третьему измерению, ориентации робота по направлению θ, которое не указывается на этой и последующих схемах. На Рис. 8.6d показана оценка после того, как робот переместился на 2 м, и учёта второго прохода сканирования, показанного на Рис. 8.6c. Степень определённости оценки позиции возрастает и глобальный максимум оценки уже соответствует истинному местоположению робота. После учёта в оценке ещё одного сканирования робот выполняет проход сканирования, показанный на Рис. 8.6e. Практически вся масса вероятности теперь сосредоточена на истинном положении робота (см. Рис. 8.6f). Очевидно, теперь можно сказать, что локализация роботом была успешно выполнена. Этим примером проиллюстрирована возможность выполнения глобальной локализации с помощью локализации по сетке.

Второй пример показан на Рис. 8.7, его объяснение приведено в описании рисунка. Здесь среда частично симметрична, что вызывает появление симметричных модов в процессе локализации.

Конечно, для выполнения глобальной локализации обычно требуется больше, чем несколько проходов сканирования датчика. Это особенно касается симметричных сред и датчиков с меньшей точностью по сравнению с лазерными. На Рис. с 8.8 до 8.10 показан процесс глобальной локализации мобильного робота, оборудованного только сонарами, в среде со множеством коридоров примерно одинаковой ширины.

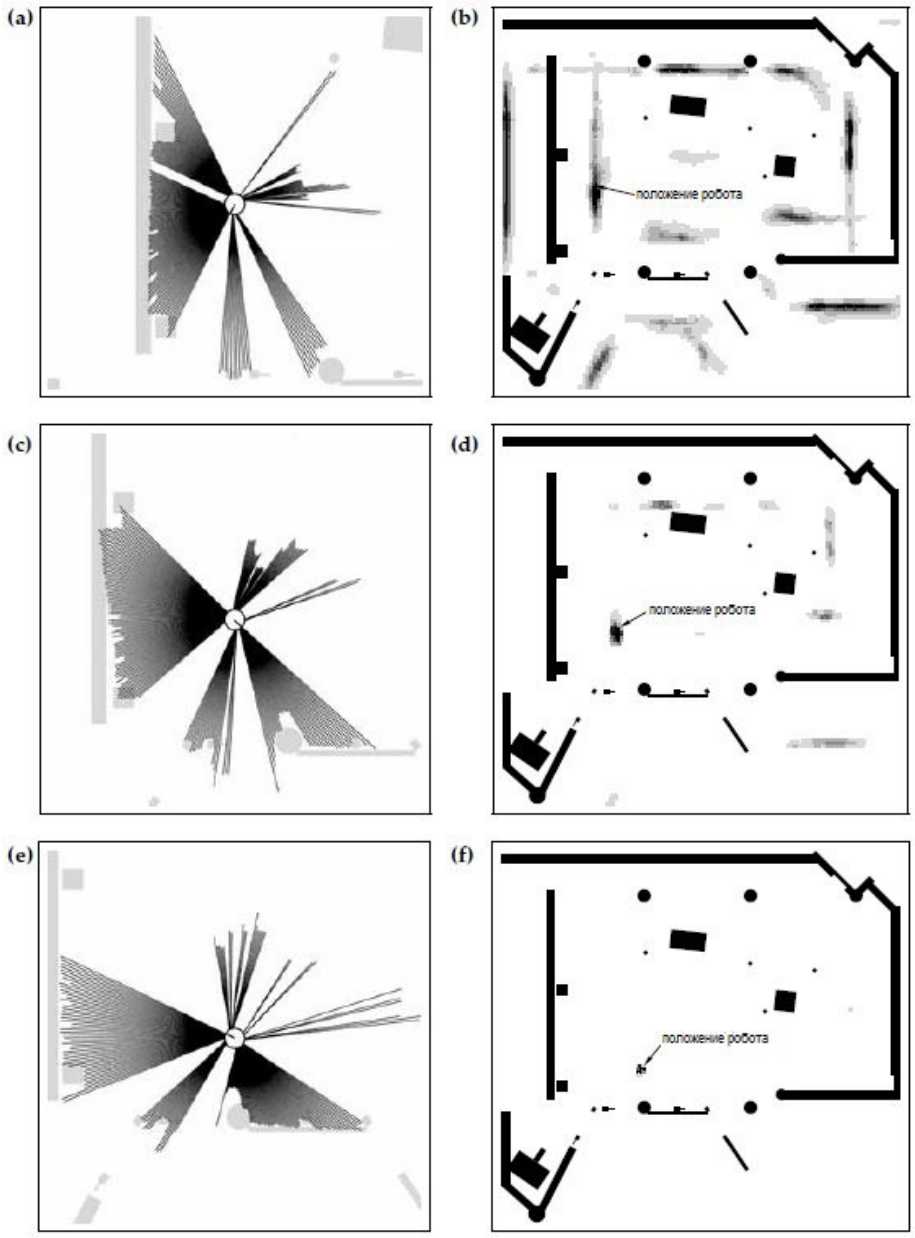


Рис. 8.6 Глобальная локализация на карте с использованием лазерных датчиков расстояния. Проход сканирования лазерных датчиков расстояния из стартовой позиции робота (максимальные показания отброшены)(a). На Рис. (b) показана ситуация после учёта результатов сканирования на равномерном распределении. Второй проход сканирования (c) и результирующая оценка (d). Учёт последнего прохода сканирования показан на Рис. (e), центр оценки у истинного положения (f).

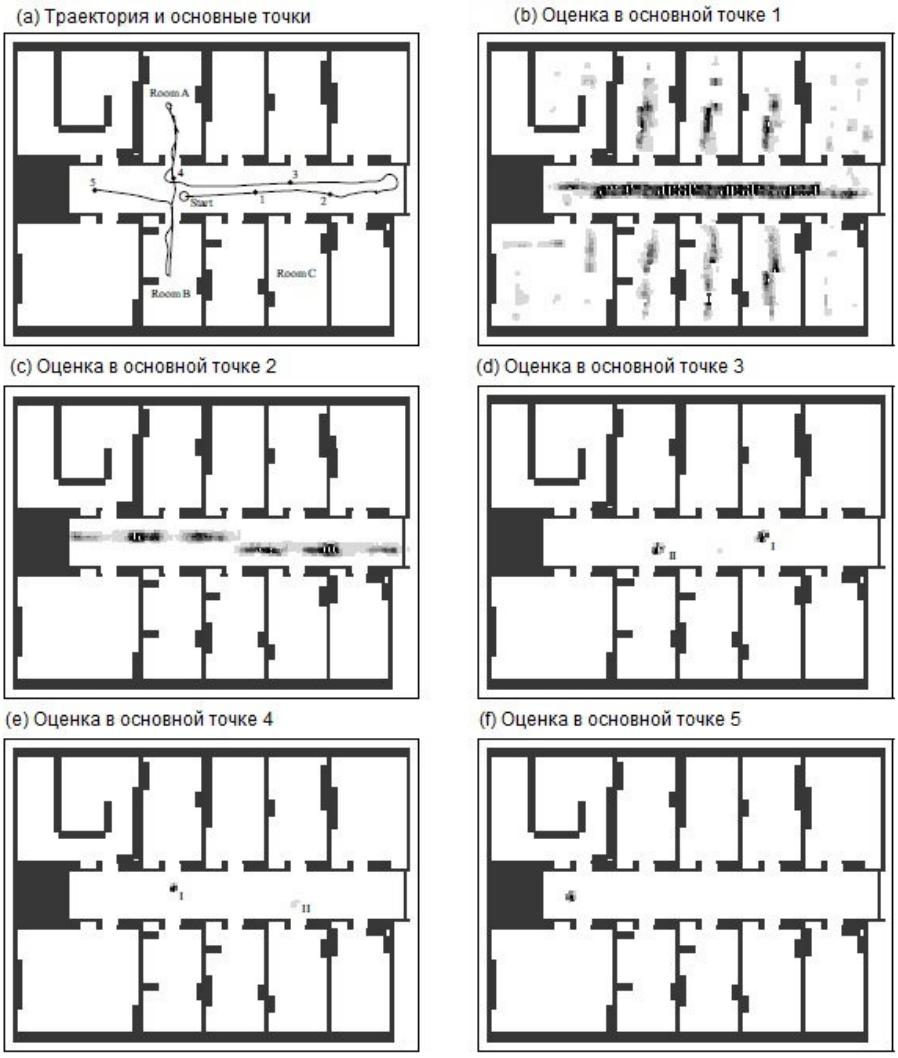


Рис. 8.7 Глобальная локализация в среде офиса, на основе данных сонара. Траектория робота(a). Оценка при прохождении роботом позиции 1 (b). После прохождения нескольких метров робот обнаруживает, что находится в коридоре (c). При достижении позиции 3 сонарами обнаруживается торец коридора и распределение концентрируется на двух локальных максимумах (d). Хотя максимум, отмеченный I отображает истинную позицию робота, из-за симметрии коридора возникает второй максимум (положение II повёрнуто на 180° относительно положения I). После перемещения через Комнату A, вероятность нахождения в верной позиции I выше по сравнению с позицией II (e). Наконец, оценка робота концентрируется у верной позиции (f).

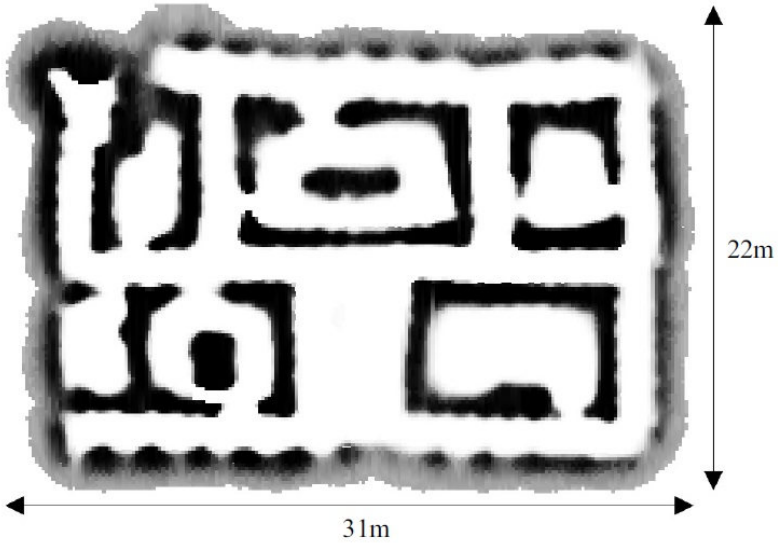


Рис. 8.8 Карта сетки занятости соревновательной зоны для мобильных роботов конкурса AAAI 1994.

Карта сетки занятости показана на Рис. 8.8. На Рис. 8.9a изображён набор данных, полученных при передвижении по одному из коридоров и повороте в другой. Каждый из лучей измерений на Рис. 8.9a соответствует измерению сонара. В данной конкретной среде стены гладкие и значительная часть измерений искажена. В примере снова использована вероятностная модель показаний датчиков на основе лучей, описанная в разделе 6.3. На Рис. 8.9 дополнительно показана оценка для трёх различных моментов времени, обозначенных как “A”, “B”, и “C” (Рис. 8.9a). После перемещения примерно на три метра, в течение которых робот выполнил 5 сканирований сонаром, оценки распределены почти равномерно по всем коридорам сравнительно одинакового размера, как показано на Рис 8.9b. Несколько секунд спустя оценка была разделена на несколько отдельных гипотез, как показано на Рис. 8.9c. Наконец, когда робот свернул за угол и достиг точки с меткой “C”, данных датчиков оказалось достаточно для определения уникальной позиции робота. Оценка, показанная на Рис. 8.9d, сосредоточена около настоящего положения робота. Этот пример показывает, что представление в виде сетки хорошо работает и для сильно зашумлённых данных сонара и симметричных сред, в которых при глобальной локализации необходимо поддерживать несколько гипотез.

На Рис. 8.10 показана возможность сеточных методов корректировать накопившиеся ошибки измерений путём сравнения данных сонара с картами сеток занятости. На Рис. 8.10a показаны «сырые» данные одометрии после прохождения по траектории длиной 240 м. Очевидно, ошибка вращательного движения быстро возрастает. После прохода всего 40 м суммарная ошибка ориентации по направлению (по данным одометрии) составляет около 50 градусов. На Рис. 8.10b показан путь, пройденный роботом, согласно алгоритму локализации.

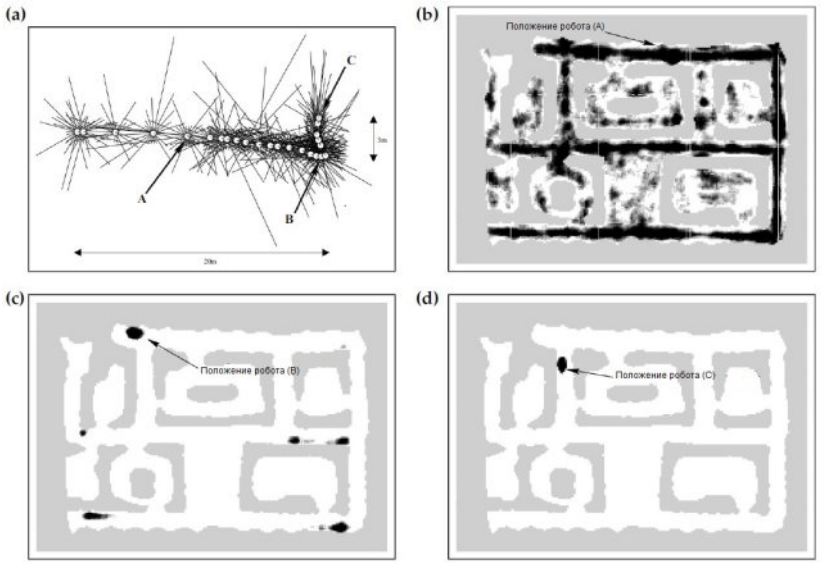


Рис. 8.9 (a) Набор данных (одометрия и данные сканирования расстояний сонаром) собранные в среде, показанной на Рис. 8.8. Этого набора данных достаточно для глобальной локализации методом локализации по сетке. Оценки в точках, обозначенных “A”, “B” и “C” показаны на рис (b), (c) и (d).

Очевидно, разрешение дискретного представления является ключевым параметром для марковской локализации по сетке. При наличии достаточных вычислительных ресурсов и необходимого объёма памяти, подходы с мелким разбиением обычно предпочтительнее. Как уже обсуждалось в подразделе 2.4.4, отображение в виде гистограмм вызывает систематическую ошибку, которая способна нарушить марковское свойство в фильтрах Байеса. Чем мельче разбиение, тем меньше вызываемая ошибка и лучше результаты.

КАТАСТРОФИЧЕСКИЙ СБОЙ

Аппроксимации с мелким разбиением также менее страдают от *катастрофических сбоев* при которых оценка робота существенно отличается от истинной позиции.

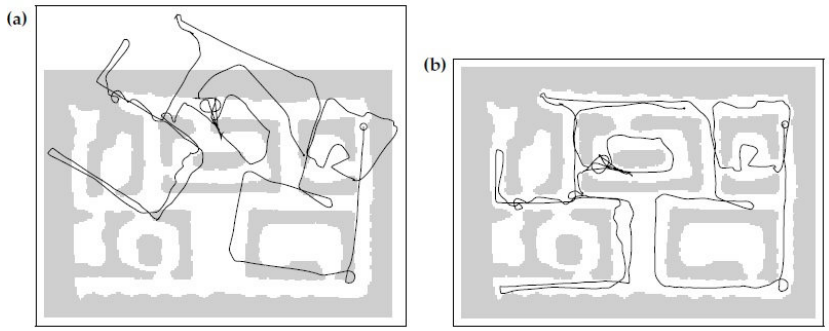


Рис. 8.10 (a) Данные одометрии и (b) скорректированный путь робота.

8.3 Локализация методом Монте-Карло

Теперь рассмотрим популярный алгоритм локализации, в котором оценка $bel(x_t)$ выражена с помощью частиц. Алгоритм называется *локализация методом Монте-Карло* (Monte Carlo Localization – MCL). Так же, как и марковская локализации по сетке, MCL применим как для локальной, так и для глобальной локализации. Несмотря на относительную новизну, MCL уже стал одним из самых популярных алгоритмов локализации в робототехнике. Его легко реализовать, и он хорошо работает для широкого круга проблем локализации.

8.3.1 Иллюстрация

На Рис. 8.11 показан MCL на примере одномерного коридора. Начальная глобальная неопределённость достигается с помощью набора частиц положений, извлекаемых случайно и равномерно из всего пространства положений, как показано на Рис. 8.11a. Когда робот обнаруживает дверь, MCL назначает каждой частице факторы значимости. Результирующий набор частиц показан на Рис. 8.11b. Высотой каждой частицы на рисунке обозначен ее вес значимости. Важно заметить, что этот набор частиц идентичен показанному на Рис. 8.11a, и единственное изменение после обновления измерения - это назначение весов.

На. Рис. 8.11c показан набор частиц после перевыборки и учёта перемещения робота. Результатом является новый набор частиц с равномерными весами значимости, но большим количеством частиц из областей около трёх наиболее вероятных местоположений. Новое измерение назначает набору неравномерные веса значимости, как показано на Рис. 8.11d. В этой точке большинство суммарной массы вероятности сосредоточено около второй двери, которая и является наиболее вероятным местоположением. Дальнейшее перемещение вызывает ещё один такт перевыборки, и генерацию нового набора частиц согласно модели движения (Рис. 8.11e). Как очевидно из примера, выполняется приближение верной апостериорной вероятности по аналогии с точным байесовским фильтром.

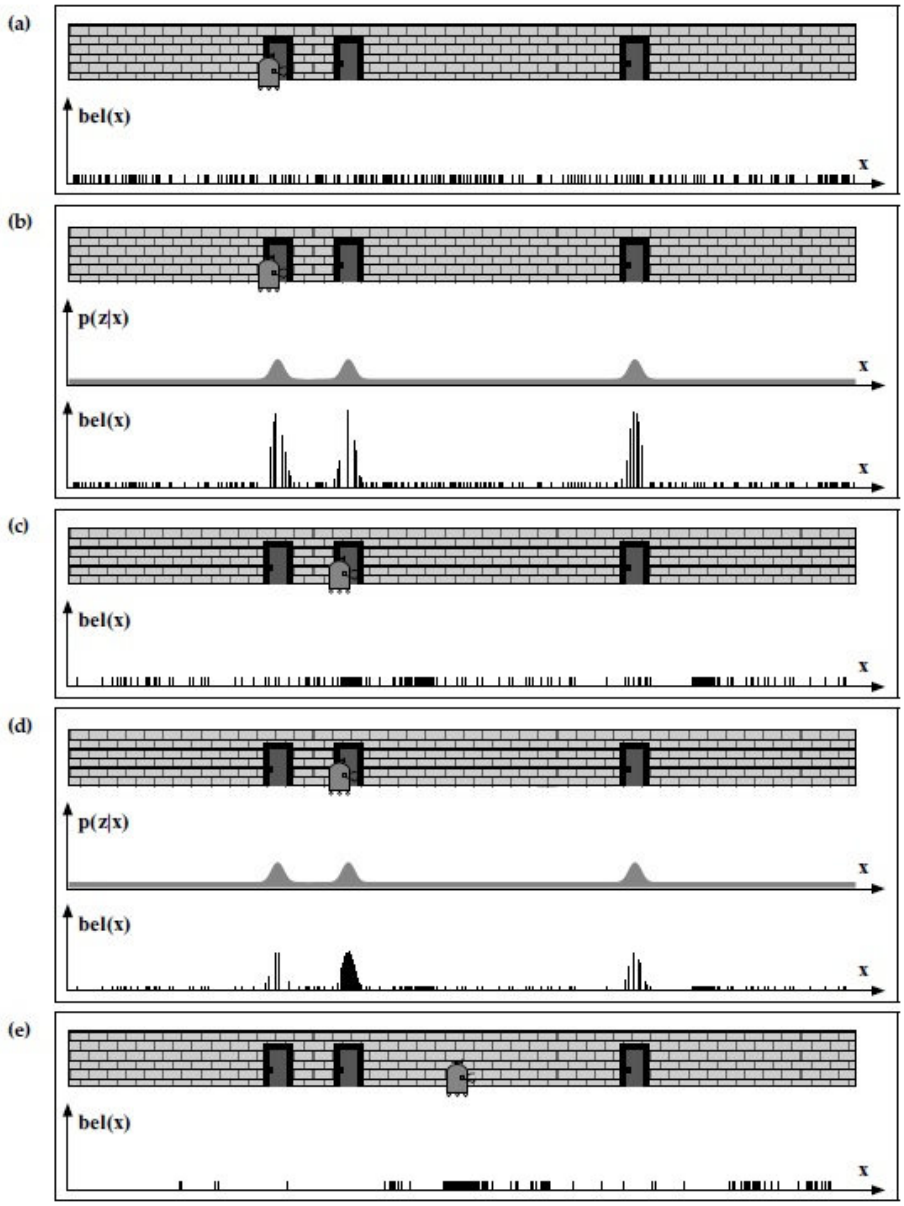


Рис. 8.11 Локализация методом Монте-Карло, многочастичный фильтр в применении к задаче локализации.

1: **Algorithm MCL**$(\mathcal{X}_{t-1}, u_t, z_t, m)$:
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: *для* $m = 1$ *до* M *выполнять*
4: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: *endfor*
8: *для* $m = 1$ *до* M *выполнять*
9: *извлечь* i *с вероятностью* $\propto w_t^{[i]}$
10: *добавить* $x_t^{[i]}$ *к* $\mathcal{X}_t$
11: *endfor*
12: *return* $\mathcal{X}_t$

Таблица 8.2 MCL или локализация методом Монте-Карло. Алгоритм локализации, основанный на многочастичных фильтрах.

8.3.2 Алгоритм MCL

В Таблице 8.2 показан основной алгоритм MCL, полученный заменой соответствующих вероятностных моделей движения и восприятия алгоритмом **particle_filters** (из Таблицы 4.3 на странице 99). Базовый алгоритм MCL выражает оценку $bel(x_t)$ в виде набора M частиц $\mathcal{X}_t = \{x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]}\}$. В строке 4 алгоритма (Таблица 8.2) выполняется выборка значений из модели движения, используя в качестве начальной точки частицы из текущей оценки. Затем для определения весов значимости частицы в строке 5 используется модель движения. Начальная оценка $bel(x_0)$ получается в результате случайного извлечения M таких частиц из априорного распределения $p(x_0)$, и назначения равномерного фактора значимости M^{-1} каждой частице. Как и в локализации по сетке, функции **motion_model** и **measurement_model** могут быть реализованы любой из моделей движения, представленных в Главе 5, и моделей измерения из Главы 6, соответственно.

8.3.3 Физическая реализация

Достаточно простой является реализация алгоритма MCL для сценария локализации на основе ориентиров, представленного в Главе 7. Для этого в строке 4 используется процедура выборки в виде алгоритма **sample_motion_model_velocity** из Таблицы 5.3. Алгоритм **landmark_model_known_correspondence**, приведённый в Таблице 6.4, предоставляет модель правдоподобия для взвешивания прогнозируемых значений, использованную в строке 5.

Эта версия алгоритма MCL показана на Рис. 8.12. Сценарий идентичен показанному на Рис. 7.15. Для удобства иллюстрация пройденного роботом пути и измерений показаны на левой верхней схеме. На нижней схеме показана последовательность наборов значений выборки, сгенерированных алгоритмом MCL. Сплошными линиями показан реальный путь робота, точками – путь на основе данных управления, пунктиром показан средний путь, вычисленный алгоритмом MCL. Прогнозируемые наборы значе-

ний $\bar{\mathcal{X}}_t$ для разных моментов времени показаны тёмным цветом, значения $\mathcal{X}_t$ после шагов перевыборки - светло-серым. Каждый набор частиц определён в трехмерном пространстве положений, хотя показаны только x- и y-координаты каждой частицы. Математическое ожидание и ковариации этих наборов показаны на правой верхней схеме.

На. Рис. 8.13 показан результат использования MCL в настоящей среде офиса роботом с массивом ультразвуковых датчиков расстояния. В этой версии MCL правдоподобие измерений вычисляется на основе алгоритма **beam_range_finder_model**, приведённого в Таблице 6.1. На схеме показаны наборы частиц после передвижения робота на 5, 28 и 55 метров, соответственно. На третьей иллюстрации на Рис. 8.14 используется камера, направленная на потолок и модель измерений, соотносящая яркость в центре изображения с предварительно полученной картой потолка.

8.3.4 Свойства MCL

С помощью MCL возможно аппроксимировать практически любое распределение, имеющее практическую важность. Он не привязан к ограниченному набору параметрических подмножеств распределений, как в случае локализации с помощью EKF. Увеличение общего количества частиц увеличивает и точность аппроксимации. Количество частиц M является параметром, позволяющим пользователю находить компромисс между точностью вычислений и необходимым для запуска MCL объёмом вычислительных ресурсов. Общая стратегия выбора M состоит в выполнении выборки до получения следующей пары значений u_t и z_t. Таким образом, реализация оказывается адаптивной по отношению к вычислительным ресурсам. Чем быстрее имеющийся процессор, тем лучше работает алгоритм локализации. Однако, как будет показано в подразделе 8.3.7, необходимо следить, чтобы количество частиц оставалось достаточно большим для предотвращения нарушения работы фильтра.

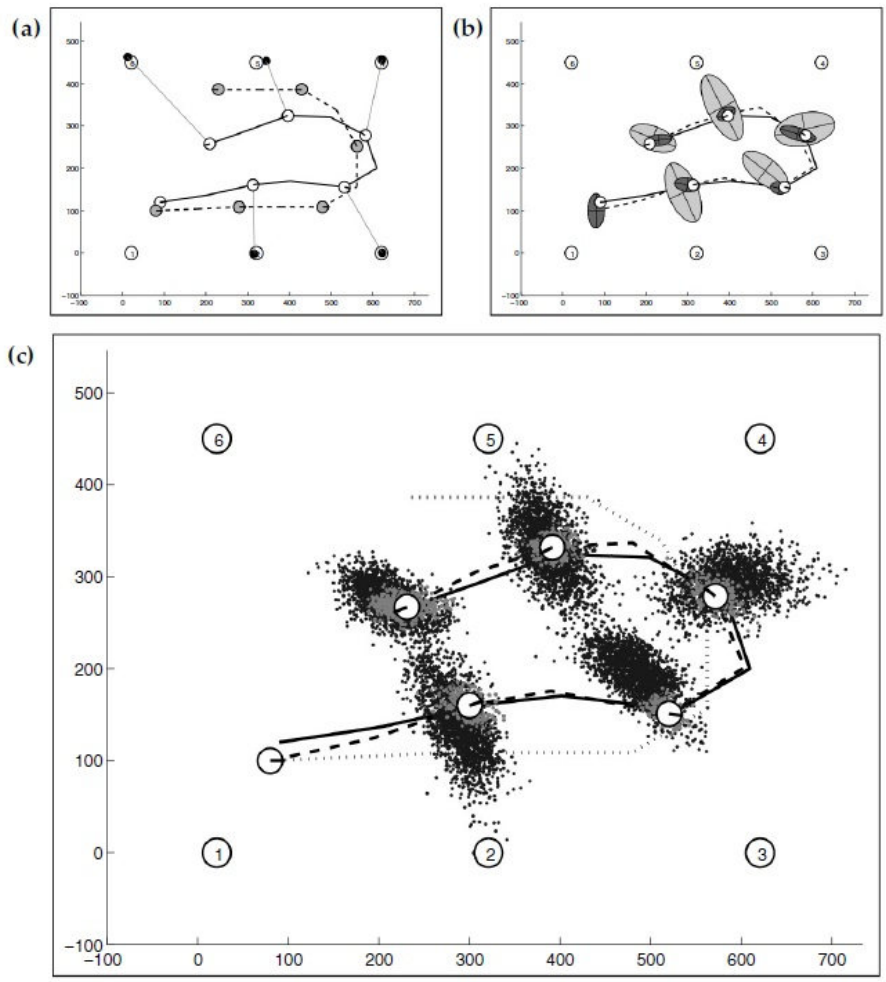


Рис. 8.12 Алгоритм MCL для локализации на основе ориентиров. Траектория робота согласно управлению движением (точки) и результирующая истинная траектория (сплошная линия) (a). Обнаружение ориентиров показано тонкими линиями. Ковариации наборов значений до и после выполнения перевыборки (b). Наборы значений до и после перевыборки (c).

Последним преимуществом MCL является непараметрическая природа аппроксимации. Как следует из иллюстраций, с помощью MCL возможно представить сложные мультимодальные вероятностные распределения и сочетать их с сосредоточенными распределениями наподобие гауссовых функций.

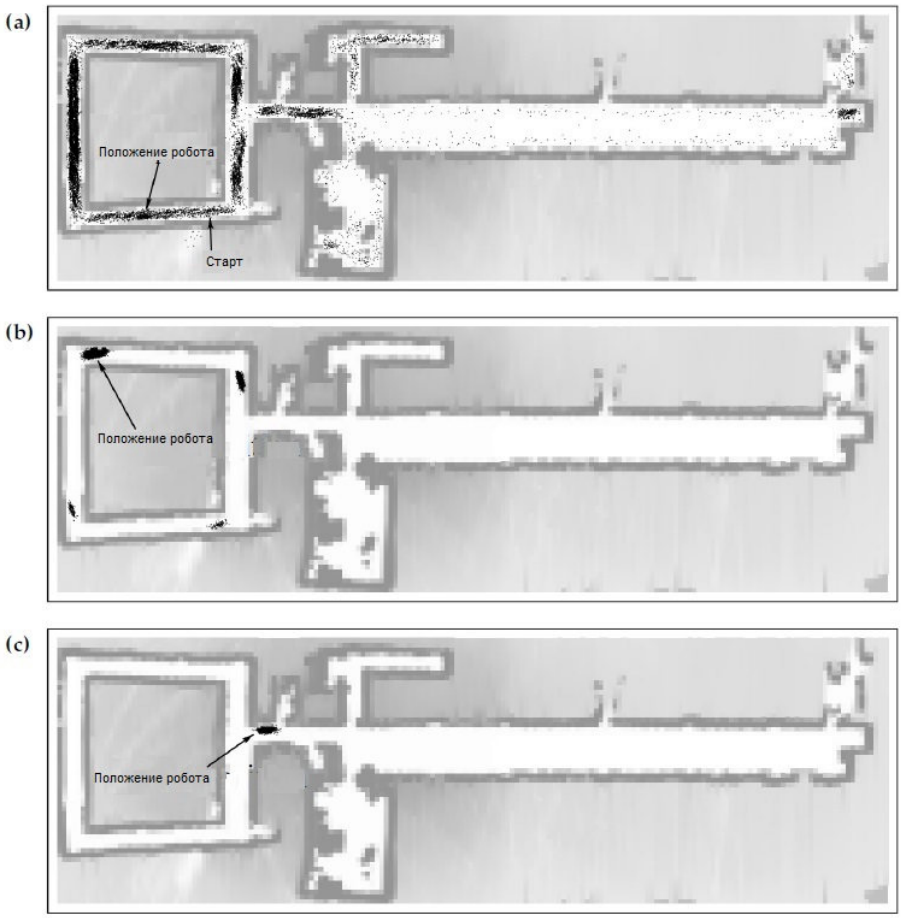


Рис. 8.13 Иллюстрация локализации методом Монте-Карло. Показан робот, действующий в офисной среде размером 54 м ×18 м. После перемещения на 5 м, робот все ещё неспособен определить позицию и частицы распределены на значительной части свободного пространства (a). Даже когда робот достиг левого верхнего угла карты, его оценка все ещё сосредоточена около четырёх возможных местоположений (b). Наконец, после перемещения приблизительно на 55 м, неопределённость разрешилась, и робот обнаружил, где он находится (c). Все вычисления выполнены в реальном времени на персональном компьютере небольшой мощности.

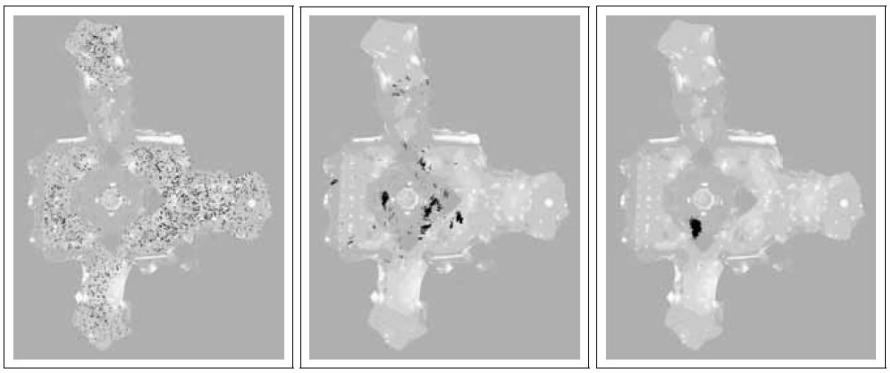


Рис. 8.14 Глобальная локализация с использованием камеры, направленной на потолок.

8.3.5 MCL на случайных частицах: Восстановление после сбоев

Алгоритм MCL в его текущем виде способен решить проблему глобальной локализации, но после сбоя «похищенного робота» или ошибок глобальной локализации восстановиться не может. Это довольно очевидно следует из результатов, показанных на Рис. 8.13. После того, как позиция определена, частицы всех положений, кроме наиболее вероятного, постепенно исчезают. В некоторый момент начинают «выживать» только частицы около единичного положения и, если это положение окажется неверным, алгоритм неспособен восстановить работу.

Это существенная проблема и на практике в любом стохастическом алгоритме, такой как MCL, при выполнении перевыборки возможно случайное отбрасывание всех частиц вблизи верного местоположения. Это может оказаться определяющим фактором при малом количестве частиц (например, M = 50), и если частицы распределены в большом объёме, скажем, при глобальной локализации.

ВНЕСЕНИЕ СЛУЧАЙНЫХ ЧАСТИЦ

К счастью, проблему возможно решить довольно простым эвристическим методом, идея которого состоит во *внесении случайных частиц* в наборы, как уже обсуждалось в подразделе 4.3.4. Такое внесение случайных частиц с математической точки зрения можно считать учётом небольшой вероятности "похищения робота" с помощью внесения в модель движения некоторой доли случайных состояний. Даже если робота не похищают, случайно добавленные частицы добавляют ещё один уровень обеспечения надёжности.

Для метода добавления частиц возникает два вопроса. Во-первых, сколько частиц необходимо добавлять при каждой итерации алгоритма? Во-вторых, из какого распределения следует их генерировать? На первый взгляд, возможно просто добавлять некоторое фиксированное число частиц при каждой итерации, но лучше обосновать выбор некоторой оценкой работы алгоритма локализации.

Одним из способов реализации этой идеи является мониторинг вероятности измерений датчиков

(8.3)
$$p(z_t|z_{1:t-1}, u_{1:t}, m)$$

относительно средней вероятности измерения (которую легко вычислить из данных). В многочастичных фильтрах аппроксимацию этого значения

можно получить из фактора значимости, который, по определению, является стохастической оценкой вероятности. Среднее значение

(8.4)
$$\frac{1}{M}\sum_{m=1}^{M} w_t^{[m]} \approx p(z_t|z_{1:t-1}, u_{1:t}, m)$$

аппроксимирует искомую вероятность. Обычно эту оценку сглаживают усреднением по нескольким тактам времени. Помимо сбоя локализации существует несколько причин низкой вероятности измерения. Так, зашумление датчика может быть чрезмерно высоким, или же на этапе глобальной локализации частицы оказываются разбросаны слишком сильно. По этим причинам предпочтительно вычислять среднее значение правдоподобия измерения в краткосрочной перспективе и соотносить его со средним значением в долгосрочной перспективе при определении числа случайных значений выборки.

Вторую проблему определения типа распределения значений в выборке можно разрешить двумя способами. Конечно, можно извлекать частицы из пространства положений согласно однородному распределению и взвешивать их текущим измерением.

Но для некоторых моделей датчиков возможно напрямую генерировать частицы в соответствии с распределением измерения. Одним из примеров такой модели датчика является модель обнаружения ориентиров, обсуждаемая в разделе 6.6. В этом случае дополнительные частицы могут быть напрямую помещены в местоположениях, распределенных согласно правдоподобности наблюдения (см. Таблица 6.5).

В Таблице 8.3 показан вариант алгоритма MCL с добавлением случайных частиц. Алгоритм адаптивен, поскольку отслеживает средние значения правдоподобия в краткосрочной и долгосрочной перспективе $p(z_t|z_{1:t-1}, u_{1:t}, m)$. Первая его часть идентична алгоритму MCL в Таблице 8.2. Новые положения извлекаются из старого набора частиц, используя модель движения (строка 5), а вес значимости для них устанавливается в соответствии с моделью измерения (строка 6).

В алгоритме **Augmented_MCL** в строке 8 вычисляется эмпирическое правдоподобие измерения и поддерживаются средние значения правдоподобия в краткосрочной и долгосрочной перспективе в строках 10 и 11. Для алгоритма требуется, чтобы $0 \leq \alpha_{slow} \ll \alpha_{fast}$. Параметры α_{slow} и α_{fast}, означают коэффициенты отказов экспоненциальных фильтров, выполняющих оценки средних значений в краткосрочной и долгосрочной перспективе, соответственно. Основная операция алгоритма выполняется в строке 13 - во время процесса перевыборки случайное значение выборки будет добавлено с вероятностью

(8.5)
$$\max\{0.0,\ 1.0 - w_{fast}/w_{slow}\}$$

1: **Algorithm Augmented_MCL**$(\mathcal{X}_{t-1}, u_t, z_t, m)$:
2: статические w_{slow}, w_{fast}
3: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
4: *для* $m = 1$ *до* M *выполнять*
5: $x_t^{[m]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[m]})$
6: $w_t^{[m]} =$ **measurement_model**$(z_t, x_t^{[m]}, m)$
7: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
8: $w_{avg} = w_{avg} + \frac{1}{M} w_t^{[m]}$
9: *endfor*
10: $w_{slow} = w_{slow} + \alpha_{slow}(w_{avg} - w_{slow})$
11: $w_{fast} = w_{fast} + \alpha_{fast}(w_{avg} - w_{fast})$
12: *для* $m = 1$ *до* M *выполнять*
13: *с вероятностью* $\max\{0.0,\ 1.0 - w_{fast}/w_{slow}\}$ *do*
14: *добавить случаное положение к* $\mathcal{X}_t$
15: *else*
16: *извлечь* $i \in \{1, ..., N\}$ *с вероятностью* $\propto w_t^{[i]}$
17: *прибавить* $x_t^{[i]}$ *к* $\mathcal{X}_t$
18: *endwith*
19: *endfor*
20: *return* $\mathcal{X}_t$

Таблица 8.3 Адаптивный вариант алгоритма MCL с добавлением случайных значений выборки. Количество случайных значений определено сравнением правдоподобия измерений датчика в краткосрочной и долгосрочной перспективе.

В остальном, процесс перевыборки протекает обычным образом. В вероятности добавления случайного значения учитывается разница между средними значениями правдоподобия измерения в краткосрочной и долгосрочной перспективе. Если правдоподобие в краткосрочной перспективе лучше или равно правдоподобию в долгосрочной перспективе, случайные значения не добавляются. Если правдоподобие в краткосрочной перспективе хуже, случайные частицы добавляются пропорционально коэффициенту отношения правдоподобия. Таким образом, неожиданное отклонение правдоподобия измерения вызывает увеличение количества случайных значений. Экспоненциальное сглаживание предотвращает опасность ошибочной интерпретации мгновенного всплеска шумов датчика как плохого результата локализации.

На Рис. 8.16 показано практическое использование дополненного алгоритма MCL на примере последовательности наборов частиц для глобальной и повторной локализации шагающего робота, оборудованного цветной камерой и действующего на поле размером 3×2 м, такого же, как используемое на соревнованиях по футболу RoboCup. Измерения датчиков, соответствующие обнаружению и относительной локализации шести визуальных маркеров, размещённых вокруг поля показаны на Рис. 7.7 на странице 195. Алгоритм, описанный в Таблице 6.4, используется для определения правдоподобия обнаружения. Шаг 14 на Рис. 8.3 заменяется алгоритмом выборки по последнему измерению датчика и легко реализуется, используя метод **sample_landmark_model_known_correspondence** в Таблице 6.5.

На схемах с (a) до (d) на Рис. 8.16 показана глобальная локализация. При первом обнаружении маркера практически все частицы извлекаются согласно текущему обнаружению (Рис. 8.16b). Этот этап соответствует ситуации, когда средняя вероятности измерения много хуже соответствующего долгосрочного показателя. После ещё нескольких обнаружений ориентиров частицы концентрируются вокруг истинной позиции робота (Рис. 8.16d), и возрастают как краткосрочное, так и долгосрочное среднее правдоподобие измерения. На этом этапе локализации робот только отслеживает свою позицию, правдоподобия измерений достаточно высоки и лишь иногда добавляется небольшое количество частиц.

Когда рефери физически перемещает робота в другое место, что часто происходит на соревнованиях по футболу для роботов, вероятность измерения падает. Первое обнаружение маркера в новом местоположении пока не добавляет частиц, поскольку сглаженная оценка w_{fast} все ещё высока (см. Рис. 8.16e). После нескольких обнаружений маркера, наблюдаемого из нового местоположения, w_{fast} падает значительно быстрее, чем w_{slow}, поэтому, добавляется больше случайных частиц (Рис 8.16f и 8.16g). Наконец, робот успешно выполняет повторную локализацию, как показано на Рис. 8.16h, подтверждая, тем самым, что наш дополненный алгоритм MCL действительно способен «пережить похищение».

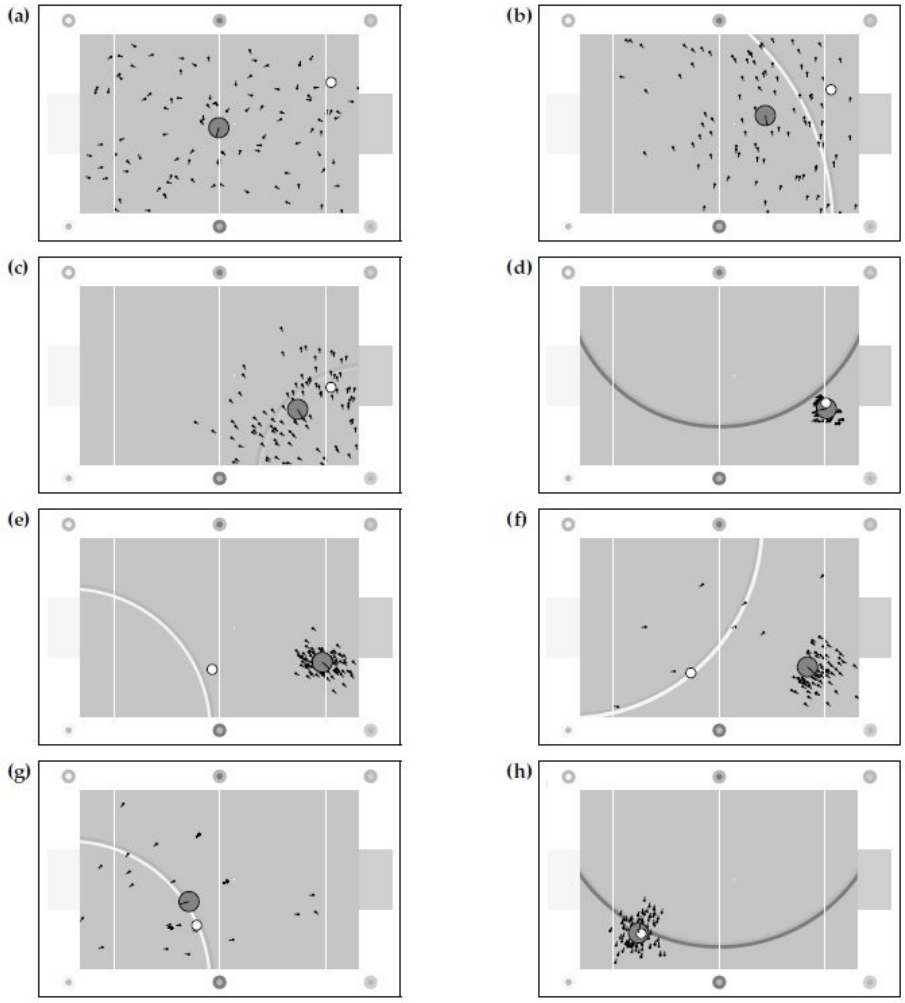


Рис. 8.16 Локализация Монте-Карло со случайными частицами. На каждой картинке показан набор частиц, представляющий оценку роботом своей позиции (короткими отрезками показана ориентацию частиц). Большим кругом показано среднее оценки, истинная позиция робота обозначена маленьким белым кругом. Обнаружения маркера показаны дугами с центром в местоположении маркера. С помощью схем (a)–(d) проиллюстрирована глобальная локализация, (e)–(h) повторная локализация.

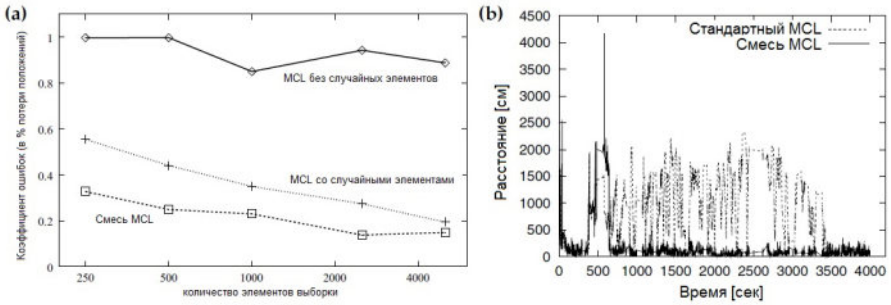


Рис. 8.17 (a) базовый алгоритм MCL (верхняя кривая), MCL со случайной выборкой (центральная кривая), и смесь MCL с распределением в виде смеси (нижняя кривая). Коэффициент ошибки измерен как процент времени, в течение которого робот теряет возможность отслеживать свою позицию, для набора данных, полученных роботом, действующим в людном помещении музея. (b) Ошибка в виде функции времени для стандартного MCL и смеси MCL, при использовании для локализации карты потолка.

8.3.6 Модификация предполагаемого распределения

Механизм предполагаемого распределения в MCL является ещё одним фактором, который может привести к неэффективности алгоритма. Как обсуждалось в подразделе 4.3.4, многочастичный фильтр использует как предполагаемое распределение модель движения, но пытается аппроксимировать произведение этого распределения и правдоподобности восприятия. Чем больше разница между предполагаемым и целевым распределениями, тем больше требуется элементов выборки.

В алгоритме MCL это вызывает неожиданный сбой: если найти идеальный датчик, который всегда, совершенно без шумов будет выдавать роботу его истинное положение, то алгоритм MCL откажет. Это справедливо даже для идеальных датчиков, которые не несут достаточно информации для локализации. Примером может служить одномерный идеальный датчик расстояния. При получении измерения такого датчика пространство пригодных гипотез положения будет двухмерным подпространством трехмерного пространства положений робота. В подразделе 4.3.4 уже подробно обсуждалось, что шансы выборки в это двухмерное подмножество при выборке из модели движения робота равны нулю. Отсюда возникает странная ситуация, когда, при определённых обстоятельствах использования MCL для локализации, менее точный датчик будет предпочтительнее более точного. Это не так в случае локализации на основе EKF, поскольку обновление EKF принимает в расчёт измерения при вычислении нового значения средней, вместо генерации средних только лишь из модели движения. К счастью, есть простое решение. Необходимо использовать модель измерения с искусственно завышенным зашумлением датчика. Такое искусственное увеличение шумов можно воспринимать как учёт не только неопределённости измерения, но и неопределённости, вызванной приближенным характером алгоритма многочастичного фильтра.

В альтернативном, более логичном способе решения используется модификация процесса выборки, которая уже кратко обсуждалась в подразделе 4.3.4. Идея состоит в том, чтобы инвертировать для некоторой малой частиц роли модели измерения и модели движения. Частицы, сгенерированные со-

гласно модели измерения

(8.6)
$$x_t^{[m]} \sim p(z_t|x_t)$$

вес значимости для которых будет вычисляется пропорционально

(8.7)
$$w_t^{[m]} = \int p(x_t^{[m]}|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

Этот новый процесс выборки является адекватной альтернативой простому многочастичному фильтру. Но один лишь этот приём будет неэффективен, поскольку при генерации частиц полностью игнорируются предыдущие наборы. Однако, полностью применимой будет генерировать некоторую долю частиц с помощью одного из двух предложенных механизмов и объединить два набора частиц. Результирующий алгоритм называется *MCL со смесью предполагаемых распределений*, или *Смесь MCL*. На практике обычно достаточно генерировать с помощью нового процесса достаточно небольшую долю частиц (около 5%).

СМЕСЬ MCL

К сожалению, при осуществлении этой идеи имеется ряд трудностей. Два основных шага – выборка из $p(z_t|x_t)$ и вычисление весов значимости $w_t^{[m]}$ — могут быть довольно трудны в реализации. Выборка из модели измерений легко реализуется только если её инвертированная форма имеет решение в закрытом виде, из которого легко выполняется выборка. Обычно это не так: представьте выполнение выборки из пространства всех положений, которые укладываются в сканирование расстояния лазером! Вычисление весов значимости затруднено интегралом в (8.7) и фактом того, что $bel(x_{t-1})$ само по себе представлено в виде набора частиц.

Не вдаваясь в детали, заметим, что оба шага возможно реализовать, но лишь с дополнительными приближениями. На Рис. 8.17 показаны сравнительные результаты для MCL, MCL с дополнениями случайными значениями и смеси MCL для двух реальных наборов данных. Во всех случаях $p(z_t|x_t)$ было обучено на данных и представлено деревом плотности, процедурой детализации, описание которой выходит за рамки книги. Для вычисления весов значимости интеграл был заменён стохастической интеграцией, и априорная оценка была продолжена плотностью заполнения пространства путём свёртки каждой частицы с помощью узкой гауссовой функции. Опуская детали, скажем, что это показывают лучшие результаты для идеи смеси, но может быть затруднительно для реализации.

Заметим также, что смесь MCL также позволяет логичным образом решить проблему похищенного робота. Если начать извлекать частицы, используя только самое последнее измерение, частицы будут генерироваться только в местоположениях, которые являются возможными на основании мгновенных показаний датчика, вне зависимости от предыдущих показателей управления и измерения. В литературе приводятся достаточные доказательства работоспособности таких методов для ситуации полного сбоя локализации (на Рис. 8.17b показан один такой сбой для обычного MCL), обеспечивая лучшую надёжность в практических реализациях.

8.3.7 KLD-выборка: подбор размера наборов значений выборки

Размер набора значений, использованного для представления оценок, является важным параметром, определяющим эффективность многочастичных

фильтров. До этого момента обсуждались только многочастичные фильтры с фиксированным размером набора значений. К сожалению, чтобы избежать отклонений в работе фильтра из-за истощения набора значений в MCL, требуется выбирать большие наборы значений, чтобы позволить мобильному роботу реализовать как отслеживание положения, так и глобальную локализацию. Это может привести к растрате вычислительных ресурсов, как показано на Рис. 8.13. В этом примере все наборы значений содержат по 100000 частиц. Хотя столь большое число частиц может быть необходимым для точного представления оценки на ранних этапах локализации (см. Рис. 8.13a), очевидно, что только небольшой доли этого количества будет достаточно для отслеживания позиции, после того, как робот обнаружил своё местоположение (Рис. 8.13c).

KLD-ВЫБОРКА

KLD-выборка является вариантом MCL, который меняет количество частиц со временем. Математический вывод для KLD-выборки не приводится, только алгоритм и некоторые результаты экспериментов.

ДИВЕРГЕНЦИЯ КУЛЬБАКА-ЛЕЙБНЕРА

Наименование "KLD-выборка" образовано из термина дивергенции Кульбака -Лейбнера, которая является мерой разницы между двумя вероятностными распределениями. Идея KLD-выборки состоит в определении числа частиц по статистическому показателю качества аппроксимации на основе выборки. А именно, при каждой итерации многочастичного фильтра KLD-выборка определяет количество частиц таким образом, что, с вероятностью $1 - \delta$, ошибка между истинным апостериорным распределением и аппроксимацией на основе выборки составляет менее ε. Некоторые допущения, которые здесь не приводятся, делают возможным вывод эффективной реализации.

Алгоритм KLD-выборки показан в Таблице 8.4. На вход принимается предыдущий набор значений, а также карта и самые последние данные управления и измерения. Зеркально по сравнению с MCL, в алгоритме KLD-выборки взвешенное значение принимается на вход. Таким образом, значения $\mathcal{X}_{t-1}$ не подвергаются перевыборке. Вдобавок, для алгоритма требуется границы статистической ошибки ε и δ.

В двух словах, KLD-выборка генерирует частицы, пока удовлетворяется статистическое ограничение в строке 16. Этот порог основан на «объёме» пространства состояний, покрытого частицами. Объем, покрытый частицами, измеряется гистограммированием или сеткой, наложенной на трехмерное пространство состояний. Каждый бин гистограммы H может быть или пуст, или занят, по крайней мере, одной частицей. Изначально, каждый бин установлен пустым (строки с 4 по 6). В строке 8 частица извлекается из предыдущего набора значений. На основе этой частицы прогнозируется новая частица, которая взвешивается и вставляется в новый набор значений (строки 9–11, как и в MCL).

1: **Algorithm KLD_Sampling_MCL**$(\mathcal{X}_{t-1}, u_t, z_t, m, \varepsilon, \delta)$:
2: $\quad \mathcal{X}_t = \emptyset$
3: $\quad M = 0$, $M_\chi = 0$, $k = 0$
4: $\quad$ *для всех* b *в* H *do*
5: $\quad\quad b =$ *пусто*
6: $\quad$ *endfor*
7: $\quad$ *do*
8: $\quad\quad$ *извлечь* i *с вероятностью* $\propto w_{t-1}^{[i]}$
9: $\quad\quad x_t^{[M]} =$ **sample_motion_model**$(u_t, x_{t-1}^{[i]})$
10: $\quad\quad w_t^{[M]} =$ **measurement_model**$(z_t, x_t^{[M]}, m)$
11: $\quad\quad \mathcal{X}_t = \mathcal{X}_t + \langle x_t^{[M]}, w_t^{[M]} \rangle$
12: $\quad\quad$ *if* $x_t^{[M]}$ *попадает в пустой бин* b *then*
13: $\quad\quad\quad k = k + 1$
14: $\quad\quad\quad b =$ *непустой*
15: $\quad\quad\quad$ *if* $k > 1$ *then*
16: $\quad\quad\quad\quad M_\chi := \frac{k-1}{2\varepsilon} \left\{ 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta} \right\}^3$
17: $\quad\quad$ *endif*
18: $\quad\quad M = M + 1$
19: $\quad$ *пока* $M < M_\chi$ *или* $M < M_{\chi_{min}}$
20: $\quad$ *return* $\mathcal{X}_t$

Таблица 8.4 KLD-выборка алгоритма MCL с переменным размером набора значений. Алгоритм генерирует значения до тех пор, пока не будет достигнут статистический порог ошибки аппроксимации.

В строках с 12 по 19 реализуется ключевая идея KLD-выборки. Если вновь сгенерированные частицы попадают в пустой бин гистограммы, то количество k непустых бинов увеличивается на единицу, и бин маркируется непустым. Таким образом, k измеряет число бинов гистограммы, заполненных, по крайней мере, одной частицей. Это число играет важную роль в определении статистического порога в строке 16. Количество M_χ обозначает число частиц, необходимых для достижения этого порога. Заметим, что для данного ε, M_χ, в основном, линейно зависит от числа k непустых бинов. Второй, нелинейный член выражения становится пренебрежимо малым по мере возрастания k. Параметр $z_{1-\delta}$ основан на значении δ и выражает верхний $1-\delta$ квантиль стандартного нормального распределения. Значения $z_{1-\delta}$ для типичных значений δ доступны в стандартных статистических таблицах.

Алгоритм генерирует новые частицы до тех пор, пока их число M не превысит M_χ и определённый пользователем минимум $M_{\chi_{min}}$. Как можно увидеть, порог M_χ служит «движущейся мишенью» для M. Чем больше элементов выборки M генерируется, тем больше бинов k в гистограмме не пусты и тем выше порог M_χ.

На практике алгоритм прерывает работу по следующим причинам. На начальных этапах выполнения выборки значение k возрастает практически с каждым новым элементом, поскольку почти все бины пусты. Это увеличение значения k приводит к увеличению порога M_χ. Однако, со временем заполняется все больше и больше бинов, а M_χ возрастает достаточно редко. Поскольку M увеличивается с каждым новым элементом, M в какой-то

момент достигнет M_χ, и процесс выборки остановится. Момент, когда это произойдет, зависит от оценки. Чем более разбросаны частицы, тем больше бинов заполнены и тем выше порог M_χ. При операции отслеживании KLD-выборка генерирует меньше элементов, поскольку частица сконцентрированы на небольшом числе разных бинов. Следует заметить, что гистограмма не оказывает никакого влияния на само распределение частиц и её единственное назначение – измерять сложность, или объем оценки. В конце каждой итерации многочастичного фильтра сетка отбрасывается.

Рис. 8.18 показаны размеры наборов значений для типичного примера глобальной локализации, используя KLD-выборку. На рисунке приведены схемы использования лазерного датчика расстояния (сплошная линия) и ультразвуковых датчиков (пунктирная линия). В обоих случаях алгоритм отбирает большое количество значений на начальном этапе глобальной локализации. Когда локализация робота завершена, количество частиц уменьшается до гораздо более низкого уровня (менее 1% от начального числа частиц). В какой именно момент и насколько быстро произойдёт переход от глобальной локализации к отслеживанию позиции зависит от типа среды и точности датчиков. В данном примере более высокая точность лазерных датчиков расстояния повлекла более ранний переход на низший уровень задачи.

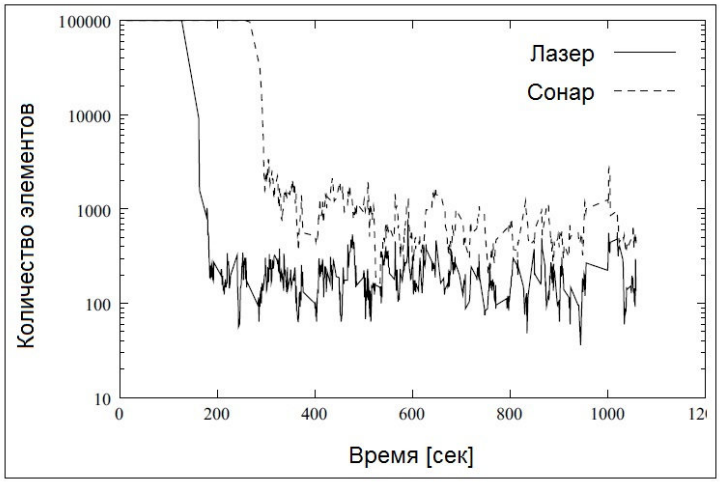


Рис. 8.18 KLD-выборка. Типичное изменение количества элементов для прохода глобальной локализации, на графике по времени (количество элементов показано на логарифмической шкале). Сплошной линией показано количество элементов при использовании лазерного датчика расстояния, пунктиром - данные сонара.

На Рис. 8.19 показана разница между ошибкой аппроксимации KLD-выборки и MCL с фиксированным размером наборов значений. Ошибка аппроксимации измерена с помощью расстояния Кульбака-Лейбнера между оценками (наборами элементов), сгенерированных с различным числом элементов и «оптимальными» оценками. Эти «оптимальные» оценки были сгенерированы запуском MCL с наборами из 200000 элементов, которых более чем достаточно для оценки позиции. Как и ожидалось, чем больше значений использовано, тем меньше ошибка аппроксимации. На пунктирном графике показаны результаты, полученные с помощью MCL с разным

размером наборов значений. Как можно увидеть, для фиксированного метода требуется около 50000 элементов, прежде чем он перейдёт к расстоянию KL менее 0,25. Большие значения обычно указывают на расхождение многочастичного фильтра и неспособность робота выполнить локализацию. Сплошной линией показаны результаты использования KLD-выборки с размерами наборов элементов усреднённых по проходам глобальной локализации. Различные точки данных были получены изменением порога ошибки ε в диапазоне от 0,4 до 0,015, изображённой на графике по убывающей слева направо. KLD-выборка переходит на малый уровень ошибки, используя только около 3000 значений. На графике также показано, что KLD-выборка не гарантирует точного отслеживания оптимальной оценки. Самые левые точки данных на сплошной линии указывает, что KLD-выборка разошлась из-за слишком больших пороговых параметров.

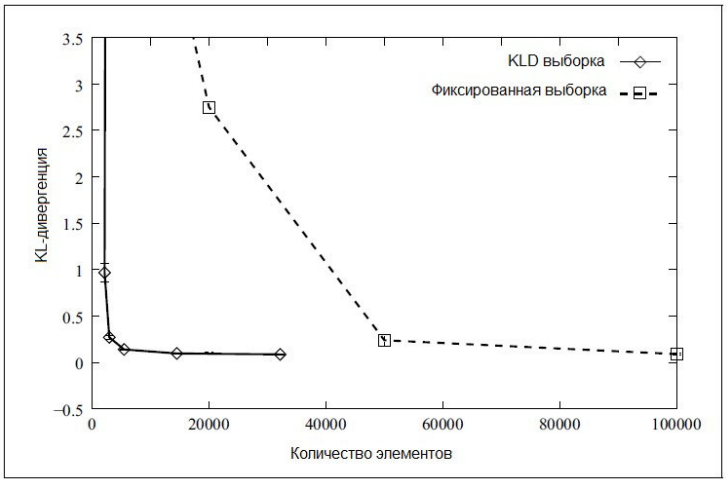


Рис. 8.19 Сравнение KLD-выборки и MCL с фиксированным размером наборов элементов. По оси x показаны средние размеры наборов элементов. По оси y показаны графики KL-расстояния между эталонными оценками и наборами элементов, сгенерированных двумя методами.

KLD-выборка может использоваться в любом многочастичном фильтре, не только в MCL. Гистограмма может быть реализована как в виде фиксированной многомерной сетки, так и в более компактных древовидных структурах. В контексте локализации робота было показано, что KLD-выборка стабильно превосходит MCL с фиксированным размером наборов элементов. Преимущество этого метода наиболее существенно для сочетания глобальной локализации и отслеживания. На практике хорошие результаты достигаются при пороговых значениях ошибки около 0,99 для $(1-\delta)$ и 0,05 для ε в сочетании с размерами бинов гистограммы 50 см$\times$50 см $\times$15 градусов.

8.4 Локализация в динамических средах

Ключевое ограничение всех обсуждаемых алгоритмов локализации возникает в силу условности о статической природе среды или марковского свойства. Наиболее интересные для роботов среды населены людьми и, поэтому, имеют динамику, которую не удаётся моделировать состоянием x_t. До

некоторой степени вероятностные подходы устойчивы к такой несмоделированной динамике из-за способности приспосабливаться к шумам датчиков. Однако, как было указано ранее, шум датчика, который возможно обработать вероятностными алгоритмами фильтров, должен быть независимым на каждом такте времени, в то время, как несмоделированная динамика вызывает воздействие на измерения датчика продолжительностью несколько тактов времени. Когда такие эффекты сохраняются, вероятностные алгоритмы локализации, основанные на допущении о статической среде, могут отказывать.

Хороший пример такой ситуации отказа показан на Рис. 8.20. В этом примере использован мобильный робот-экскурсовод, выполняющий навигацию в музее, полном людей. Люди, а точнее их местоположение, скорости и намерения перемещения являются скрытыми состояниями для алгоритма локализации и не обнаруживаются методами, которые обсуждались ранее. Почему это является проблематичным? Представим, что люди расположились таким образом, что робот ошибочно принял их за стену. С каждым новым измерением датчика оценка робота о том, что он находится у стены, возрастает. Поскольку информация считается независимой, назначается более высокое правдоподобие для положений около стен. Такой эффект возможен и для независимого шума датчика, на правдоподобие в этом случае будет исчезающе мало.

Рис. 8.20 Фотография из *"Deutsches Museum Bonn"*, где мобильный робот «Рино» ("Rhino") часто находился в окружении десятков людей.

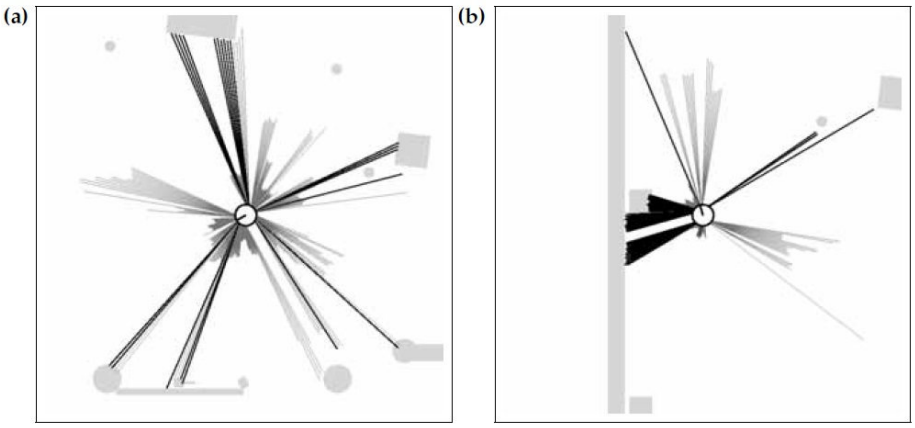


Рис. 8.21 Показания лазерного датчика расстояния часто были сильно повреждены, когда люди находились вокруг робота. Как робот может выполнить точную локализацию в такой ситуации?

ДОПОЛНЕНИЕ СОСТОЯНИЯ

Существуют два фундаментальных метода для работы с динамическими средами. Первый метод, *дополнение состояния*, включает скрытое состояние в состояние, оцениваемое фильтром. Другой, *отбрасывание выбросов* предварительно обрабатывает измерения датчика для удаления искажённых измерений. Вторая технология является математически более обобщённой: вместо простой оценки положения робота, можно определить фильтр для оценки положения людей, их скоростей и так далее. Позже мы обсудим такой подход, как расширение алгоритма картографирования мобильного робота.

Принципиальный недостаток оценивания переменных скрытого состояния состоит в большой вычислительной сложности. Вместо оценки 3 переменных необходимо вычислять апостериорные распределения для значительно большего количества переменных. Фактически, само количество переменных является переменной величиной, поскольку со временем окружающая ситуация изменяется. Поэтому, результирующий алгоритм будет существенно более сложным, чем алгоритмы локализации, которые обсуждались до настоящего момента.

ОТБРАСЫВАНИЕ ВЫБРОСОВ

Альтернативный подход, *отбрасывание выбросов*, хорошо работает в некоторых ограниченных случаях, например, когда присутствие людей может влиять на датчики расстояния или, в меньшей мере, на изображения с камер. Далее будет показан алгоритм для модели датчика расстояния на основе лучей из раздела 6.3.

Идея состоит в том, чтобы найти причину конкретного измерения датчика и отбросить показания, искажённые несмоделированной динамикой окружающей среды. Обсуждаемые модели датчиков используют самые разнообразные методы получения измерения. Если мы сумеем связать конкретные способы появления измерения с присутствием нежелательных эффектов динамики (например, людей), то останется только отбросить измерения с высоким правдоподобием для несмоделированных сущностей.

Идея носит удивительно обобщенный характер и основывается на той же математике, что и в *алгоритме обучения EM* в разделе 6.3, но в онлайновом варианте. В Уравнении (6.12) в разделе 6.3 была определена модель измерения на основе лучей для датчиков расстояния в виде смеси четырех составляющих:

$$(8.8)\quad p(z_t^k|x_t,m) = \begin{pmatrix} z_{hit} \\ z_{short} \\ z_{max} \\ z_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} p_{hit}(z_t^k|x_t,m) \\ p_{short}(z_t^k|x_t,m) \\ p_{max}(z_t^k|x_t,m) \\ p_{rand}(z_t^k|x_t,m) \end{pmatrix}$$

Как видно из данных вывода модели, один из членов выражения, включающий z_{short} и p_{short}, соответствует незапланированным объектам. Для вычисления вероятности того, что измерение z_t^k соответствует незапланированному объекту введём новую переменную соответствия, $\bar{c}_t^k$ которая может принимать одно из четырёх значений {отражение, близко, максимально далеко, случайно}.

Апостериорная вероятность того, что измерение z_t^k соответствует показанию «близко» (нашему обозначению незапланированного препятствия из раздела 6.3) получается применением теоремы Байеса и последовательным отбрасыванием нерелевантных переменных:

$$(8.9)\quad \begin{aligned} & p(\bar{c}_t^k = \text{short}|z_t^k, z_{1:t-1}, u_{1:t}, m) \\ & \quad = \frac{p(z_t^k|\bar{c}_t^k = \text{short}, z_{1:t-1}, u_{1:t}, m)\ p(\bar{c}_t^k = \text{short}|z_{1:t-1}, u_{1:t}, m)}{\sum_c p(z_t^k|\bar{c}_t^k = c, z_{1:t-1}, u_{1:t}, m)\ p(\bar{c}_t^k = c|z_{1:t-1}, u_{1:t}, m)} \\ & \quad = \frac{p(z_t^k|\bar{c}_t^k = \text{short}, z_{1:t-1}, u_{1:t}, m)\ p(\bar{c}_t^k = \text{short})}{\sum_c p(z_t^k|\bar{c}_t^k = c, z_{1:t-1}, u_{1:t}, m)\ p(\bar{c}_t^k = c)} \end{aligned}$$

Здесь переменная c в знаменателе принимает одно из четырёх значений {отражение, близко, далеко, случайно}. Используя нотацию выражения (8.8), априорная вероятность $p(\bar{c}_t^k = c)$ задана переменными z_{hit}, z_{short}, z_{max} и z_{rand} для четырёх указанных значений c. Оставшаяся вероятность в (8.9) получается интегрированием x_t:

$$(8.10)\quad \begin{aligned} & p(z_t^k|\bar{c}_t^k = c, z_{1:t-1}, u_{1:t}, m) \\ & \quad = \int p(z_t^k|x_t, \bar{c}_t^k = c, z_{1:t-1}, u_{1:t}, m)\ p(x_t|\bar{c}_t^k = c, z_{1:t-1}, u_{1:t}, m)dx_t \\ & \quad = \int p(z_t^k|x_t, \bar{c}_t^k = c, m)\ p(x_t|z_{1:t-1}, u_{1:t}, m)dx_t \\ & \quad = \int p(z_t^k|x_t, \bar{c}_t^k = c, m)\ \overline{bel}(x_t)dx_t \end{aligned}$$

Вероятности в форме $p(z_t^k|x_t, \bar{c}_t^k = c, m)$ сокращаются по p_{hit}, p_{short}, p_{max} и p_{rand} аналогично преобразованию в подразделе 6.3. Это даст выражение искомой вероятности (8.9):

$$(8.11)\quad p(\bar{c}_t^k = \text{short}|z_t^k, z_{1:t-1}, u_{1:t}, m) = \frac{\int p_{short}(z_t^k|x_t, m) z_{short} \overline{bel}(x_t)dx_t}{\int \sum_c p_c(z_t^k|x_t, m) z_c \overline{bel}(x_t)dx_t}$$

В общем случае, интегралы в (8.11) не имеют решений в закрытом виде. Для их оценки следует выполнить аппроксимацию репрезентативными значениями апостериорной вероятности $\overline{bel}(x_t)$ по состоянию x_t. Эти значения

могут быть ячейками сетки с высоким правдоподобием в алгоритме локализации по сетке или частиц в алгоритме MCL. Измерение отбрасывается, если его вероятность быть вызванным незапланированным препятствием превышает определённый пользователем порог χ.

1: **Algorithm test_range_measurement** $(z_t^k, \bar{\mathcal{X}}_t, m)$:
2: $p = q = 0$
3: *для* $m = 1$ *до* M *выполнять*
4: $p = p + z_{short} \cdot p_{short}(z_t^k | x_t^{[m]}, m)$
5: $q = q + z_{hit} \cdot p_{hit}(z_t^k | x_t^{[m]}, m) + z_{short} \cdot p_{short}(z_t^k | x_t^{[m]}, m)$
6: $+z_{max} \cdot p_{max}(z_t^k | x_t^{[m]}, m) + z_{rand} \cdot p_{rand}(z_t^k | x_t^{[m]}, m)$
7: *endfor*
8: *if* $p/q \leq \chi$ *then*
9: *return* принять
10: *else*
11: *return* отбросить
12: *endif*

Таблица 8.5 Алгоритм тестирования измерений расстояния в динамической среде.

В Таблице 8.5 приводится реализация этого метода в контексте многочастичных фильтров. На вход принимается набор частиц $\bar{\mathcal{X}}_t$, выражающий оценку $\overline{bel}(x_t)$ измерения расстояния z_t^k и карты. Если вероятность того, что измерение соответствует незапланированному препятствию превышает χ, возвращается значение «отбросить», в противном случае возвращается «принять». Эта процедура предшествует такту интеграции измерения в MCL.

На Рис. 8.22 показан эффект фильтра. На обоих врезках показано сканирование дальности для различных соответствий положениям робота. Выделенные светлым цветом измерения превышают порог и отбрасываются. Ключевым свойством механизма отбрасывания является фильтрация измерений, которые «необычно коротки», но сохранение тех, что «необычно велики». Эта асимметрия отражает факт того, что появление людей вызывает более короткие измерения по сравнению с ожидаемыми. Принимая неожиданно длинные измерения, метод позволяет сохранять возможность восстанавливаться после сбоев глобальной локализации.

На Рис. 8.23 показан эпизод, в котором робот передвигается в среде, где присутствует множество людей (см. Рис. 8.21). Показан также предполагаемый путь робота по всем конечным точкам, учтённым в алгоритме локализации. На рисунке показана эффективность удаления измерений, не соответствующим физическим объектам карты: на правой схеме очень мало оставшихся измерений расстояния в свободном пространстве, поскольку принимаются только измерения, превосходящие пороговое значение.

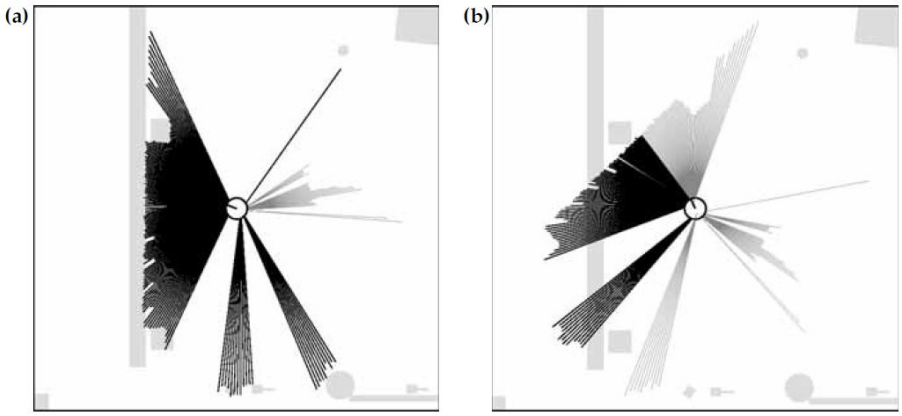


Рис. 8.22 Иллюстрация алгоритма отбрасывания измерений. На обоих схемах показаны измерения расстояния (без максимальных показаний). Выделенные светлым показания отбрасываются.

В общем случае, отбрасывание измерений является хорошей идеей. Статических сред практически не существует, поскольку даже в условиях офиса мебель может передвигаться, двери – открываться и закрываться и так далее. Приведённая реализация использует асимметрию измерений, поскольку присутствие людей укорачивает измерения, а не удлиняет. При использовании того же метода на других данных (например, изображениях) или других типах модификации среды (например, удаление физического препятствия), такая асимметрия может не существовать, но тот же вероятностный анализ все же применим. Недостатком отсутствия симметрии может стать невозможность восстановления после сбоев глобальной локализации, поскольку каждое необычное измерение будет отбрасываться. В таких случаях может иметь смысл добавление новых ограничений, например, предельной доли измерений, которые считаются повреждёнными.

Заметим, что проверка на отбрасывание применима даже в весьма статичных средах по довольно неявным причинам. Модель датчика на основе лучей прерывистая: малые изменения положения могут весьма сильно изменить апостериорную вероятность измерения датчика. Это происходит потому, что результат операции бросания лучей не является непрерывной функцией параметров положения, таких как ориентация робота по направлению. В средах с большим количеством объектов из-за этого увеличивается количество частиц, необходимых для успешной локализации. Путём ручного удаления чрезмерно загромождённых мест с карты и использования фильтра для удаления ошибочных слишком «близких» измерений число частиц можно значительно уменьшить.

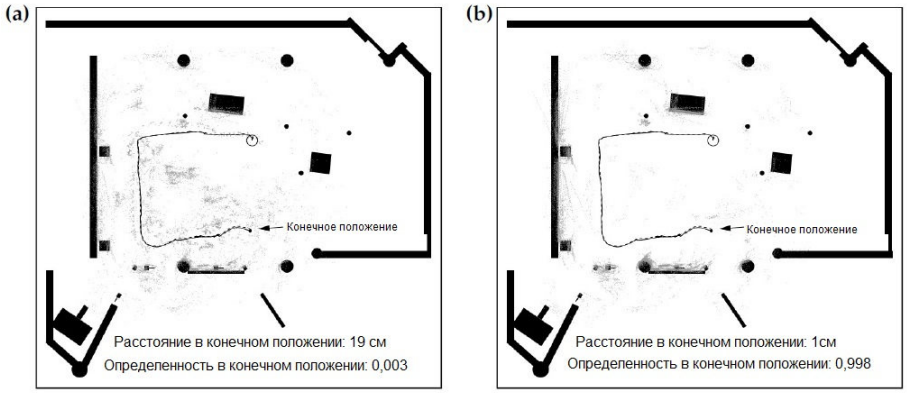


Рис. 8.23 Сравнение стандартного MCL (a) и датчика MCL с удалением измерений, которые, скорее всего, вызваны неожиданными препятствиями (b). На обоих схемах показан путь робота и конечные точки сканирования, используемые для локализации.

Эта модель неприменима для модели полей правдоподобия, поскольку она гладкая относительно параметров положения робота.

8.5 Практические соображения

В Таблице 8.6 приводятся и сравниваются основные методы локализации, обсуждаемые в этой и предыдущих главах. При выборе метода необходимо соблюдать компромисс между различными требованиями. Первым вопросом всегда будет предпочтительность извлечения признаков из измерений датчика. Извлечение признаков может оказаться выигрышным с вычислительной точки зрения, но по цене уменьшенной точности и надёжности.

Хотя в этой главе обсуждались методы для работы в динамических средах в контексте алгоритма MCL, похожие методы могут использоваться и для других методов локализации. Фактически, представленные методы лишь иллюстрируют гораздо более обширное семейство алгоритмов.

При реализации алгоритма локализации стоит настроить дополнительные настройки параметров. Например, условные вероятности часто увеличивают при интеграции близлежащих измерений, для учёта несмоделированных зависимостей, которые всегда присутствуют в робототехнике. Хорошим подходом является сбор эталонных наборов данных, и настойка алгоритма до получения удовлетворительных результатов. Это необходимо потому, что независимо от сложности математической модели всегда будут оставаться не моделированные зависимости и источники систематических шумов, влияющие на общий результат.

	EKF	MHT	грубая (топологическая) сетка	мелкая (метрическая) сетка	MCL
Измерения	ориентиры	ориентиры	ориентиры	исходные измерения	исходные измерения
Шум измерения	гауссовый	гауссовый	любой	любой	любой
Апостериорное распределение	гауссово	смесь гауссианов	гистограмма	гистограмма	частицы
Эффективность (память)	++	++	+	−	+
Эффективность (время)	++	+	+	−	+
Легкость реализации	+	−	+	−	++
Разрешение	++	++	−	+	+
Устойчивость	−	+	+	++	++
Глобальная локализация	нет	да	да	да	да

Таблица 8.6 Сравнение разных реализаций марковских алгоритмов локализации.

8.6 Выводы

В этой главе обсуждались два семейства вероятностных алгоритмов локализации: сеточные методы и методы Монте-Карло (MCL).

• Сеточные методы отображают апостериорные распределения в виде гистограмм.

• Степень разбиения сетки определяется компромиссом между точностью и вычислительной эффективностью. Для грубых сеток обычно необходимо настраивать модели датчика и движения для учёта негативных эффектов разбиения. Для тонких разбиений может понадобиться выборочное обновление ячеек сетки в целях уменьшения общей вычислительной нагрузки.

• Алгоритм локализации Монте-Карло отображает апостериорное распределение, в виде набора частиц. Баланс между вычислительной стоимостью и точностью достигается путём установки размера набора частиц.

• И методы локализации по сетке, и MCL способны выполнять глобальную локализацию робота.

• После добавления случайных частиц MCL также способны решить проблему похищенного робота.

• Смесь MCL - это обобщение, инвертирующее процесс генерации частицы для некоторой части всех частиц. Это позволяет достичь улучшенной производительности для роботов с малошумными датчиками, но ценою более сложной реализации.

• KLD-выборка повышает эффективность многочастичных фильтров с помощью настройки размера наборов элементов. Максимальная эффективность этого метода достигается, если сложность оценок значительно изменяется со временем.

• Несмоделированная динамика среды может быть обработана фильтрацией данных датчиков, отбрасывая измерения, для которых велико правдоподобие, того, что объекту не был моделирован. При использовании датчиков расстояния робот отбрасывает показания датчиков, которые необычно малы.

Популярность MCL, возможно, проистекает из двух факторов. MCL является одним из самых простых в реализации алгоритмов локализации и одним из самых мощных, поскольку способен аппроксимировать практически любое распределение.

8.7 Библиографические примечания

Локализация Монте-Карло на основе сеток была представлена Симмонсом и Кенигом (Simmons and Koenig), на основе связанного метода поддержки факторов определенности Нурбакша (Nourbakhsh et al., 1995). После основополагающей работы Симмонса и Кенига (Simmons and Koenig, 1995) возникло большое количество методов, использующих гистограммы для локализации (Kaelbling et al. 1996). Хотя в первоначальной работе были использованы достаточно грубые разбиения для решения проблемы чрезвычайно высокой вычислительной нагрузки при обновлении сетки, Бургард (Burgard et al, 1996) предложил методы выборочного обновления, для применения сетки значительно более высокого разрешения. Это развитие часто рассматривалось как переход от грубой, топологической марковской локализации к детализированной метрической. Обзорные статьи об этих работах можно найти у Кенига и Симмонса (Koenig and Simmons, 1998), Фокса (Fox et al., 1999c).

Ряд лет методы на основе сеток считались наиболее совершенными для локализации мобильного робота и возможно найти множество различных примеров успешных реализаций марковской локализации. Например, Херцберг и Кирхнер (Hertzberg and Kirchner, 1996) использовали этот метод для роботов, действующих в водосточных трубах, Симмонс (Simmons et al., 2000b) применил его для локализации робота в офисной среде, а Бургард (Burgard et al., 1999a) реализовал алгоритм оценки позиции робота, действующего в музеях. Конолиг и Чоу (Konolige and Chou, 1999) предложили идею *наложения карт* с помощью методов быстрой свертки для вычисления вероятностей положения робота. Расширенный вариант, сочетающий в себе глобальную локализацию и высокоточное отслеживание, был описан Бургардом (Burgard et al., 1998), который назвал его динамической марковской локализацией. Метод машинного обучения для обучения узнаванию мест был предложен Оори (Oore et al., 1997). Трун (Thrun, 1998a) дополнил этот подход обучающимся компонентом для идентификации подходящих ориентиров в среде, основываясь на близкой работе Грейнера и Исукапалли (Greiner and Isukapalli, 1994). Математический аппарат был расширен Махадеваном и Халили (Mahadevan and Khaleeli, 1999) до набора методов, известных как полумарковские процессы принятия решений, которые позволили обосновать точный момент перехода от одной ячейки

НАЛОЖЕНИЕ КАРТ

сетки к другой. Экспериментальное сравнение различных методов на основе сеток и калмановских фильтров было выполнено Гутманном (Gutmann et al., 1998). Активная локализация для парадигмы на основе сеток была представлена в работе Бургарда (Burgard et al., 1997) и расширена до отслеживания нескольких гипотез в работах Остина и Янсфелта (Austin and Jensfelt, 2000), Янсфелта и Кристенсена (Jensfelt and Christensen, 2001a). Фокс (Fox et al., 2000) и Говард (Howard et al., 2003) расширили этот подход для проблемы локализации с несколькими роботами. Отходя от парадигмы на основе сеток, Янсфельт и Кристенсен (Jensfelt and Christensen, 2001a), Раумелиотис и Беки (Roumeliotis and Bekey, 2000) а также Рейтер (Reuter, 2000) показали, что EKF с несколькими гипотезами подходят для проблемы глобальной локализации.

АЛГОРИТМ КОНДЕНСАЦИИ

Основываясь на известном в компьютерном зрении *алгоритме конденсации*, Изард и Блейк (Isard and Blake, 1998), Делаэрт (Dellaert et al., 1999), Фокс (Fox et al.,1999a) впервые использовали многочастичные фильтры для задач локализации мобильного робота. Им также принадлежит термин "локализация методом Монте-Карло", который стал общим названием таких алгоритмов в робототехнике. Идею добавления случайных элементов выборки можно найти в работе Фокса (Fox et al., 1999a). Усовершенствованный метод решения проблемы похищенного робота был предложен Ленсером и Велосо (Lenser and Veloso, 2000) под названием "метода перезагрузки датчика", при котором некоторое количество частиц повторно инициализировались самым последним измерением. На основе этого метода Фокс создал и представил алгоритм "дополненного MCL", определив количество частиц, которое следует добавить (Gutmann and Fox 2002). Алгоритм смеси MCL принадлежит Труну (Thrun et al., 2000c), а также ван дер Мерве (van der Merwe et al., 2001). Это создало математическую основу генерации значений выборки из измерений. KLD-выборка, адаптивная версия многочастичных фильтров, была представлена Фоксом (Fox, 2003). В работах Янсфельта (Jensfelt et al., 2000) и Янсфельта и Кристенсена (Jensfelt and Christensen, 2001b) было описано использование MCL в картах на основе признаков. Квок (Kwok et al., 2004) представил версию MCL, которая была способна изменять число частиц и работающую в реальном времени. Наконец, в ряде работ алгоритмы MCL были реализованы для роботов, оснащённых камерами (Lenser and Veloso 2000; Schulz and Fox 2004; Wolf et al. 2005), включая узконаправленные камеры (Kröse et al. 2002; Vlassis et al. 2002).

Многочастичные фильтры также были использованы в ряде соответствующих задач отслеживания и локализации. Монтемерло (Montemerlo et al., 2002b) изучил проблему одновременной локализации и отслеживания людей, используя каскадный многочастичный фильтр. Многочастичный фильтр для отслеживания переменного количества людей в зоне датчиков описан в работе Шульца (Schulz et al., 2001b). На этом методе было показано, как можно надёжно отслеживать движущихся людей движущимся роботом в неизвестной среде (Schulz et al. 2001a).

8.8 Упражнения

1. Допустим, имеется робот с d переменными состояния. Например, кинематическое состояние свободно летающего робота обычно задано для $d = 6$, а когда в вектор состояний включаются скорости, размерность увеличивается до $d = 12$. Как сложность (время обновления и требуемая память) следующих алгоритмов локализации возрастает с размерностью d: локализация

на основе EKF, локализация по сетке, локализация методом Монте-Карло. Использовать нотацию O( ) и обосновать правильность ответа.

2. Дать математический вывод аддитивной формы интеграции данных по нескольким признакам в строках 14 и 15 в Таблице 7.2.

3. Доказать правильность Выражения (8.4), на странице 241, в пределе $\uparrow \infty$.

4. Как отмечалось в тексте, локализация методом Монте-Карло имеет смещение для любого конечного размера выборки, то есть ожидаемое значение местоположения, вычисленное алгоритмом, отличается от истинного ожидаемого значения. В этом задании необходимо вычислить величину этого смещения.

Для простоты, представим среду с четырьмя возможными местоположениями робота: $X = \{x_1, x_2, x_3, x_4\}$. Вначале извлечём $N \geq 1$ равномерно распределенных значений из этого пространства. Допустимо извлекать более одного значения для любого из местоположений X. Пусть Z будет булевой переменной датчика, характеризующейся следующими условными вероятностями:

$$\begin{aligned} p(z|x_1) &= 0,8 \qquad & p(\neg z|x_1) &= 0,2 \\ p(z|x_2) &= 0,4 \qquad & p(\neg z|x_2) &= 0,6 \\ p(z|x_3) &= 0,1 \qquad & p(\neg z|x_3) &= 0,9 \\ p(z|x_4) &= 0,1 \qquad & p(\neg z|x_4) &= 0,9 \end{aligned}$$

Алгоритм MCL использует эти вероятности для генерации весов частиц, которые последовательно нормализуются и используются в процессе перевыборки. Для простоты будем считать, что в процессе перевыборки генерируется только один новый элемент, независимо от N. Этот элемент может соответствовать любому из четырёх местоположений X. Таким образом, процесс выборки определяет вероятностное распределение по X.

(a) Каково результирующее вероятностное распределение по X для нового элемента? Ответить на этот вопрос для случаев $N = 1, ..., 10$, и для $N = \infty$.

(b) Разница между двумя вероятностными распределениями p и q может быть измерена с помощью расстояния Кульбака-Лейбнера, определяемого как

$$KL(p, q) = \sum_i p(x_i) \log \frac{p(x_i)}{q(x_i)}$$

Каково расстояние Кульбака-Лейбнера между распределением (a) и истинным апостериорным распределением?

(c) Какие изменения условий задачи (но не алгоритма!) смогут гарантировать, что определённый алгоритм оценки из указанных выше не смещён даже при конечных значениях N? Указать, по меньшей мере, два таких изменения (каждого из которых будет достаточно).

5. Имеется робот, оборудованный датчиком расстояния/направления, аналогичный описанному в подразделе 6.6. В этом задании необходимо сформулировать эффективную процедуру выборки значений, которая сможет учесть k одновременных измерений неразличимых между собой ориентиров. Для иллюстрации работоспособности решения предлагается сгенерировать наборы различных ориентиров с $k = 1, ..., 5$ смежными ориентирами. Объяснить, что обеспечивает эффективность алгоритма.

6. В Упражнении 3 Главы 7 на странице 219 описан упрощённый подводный робот, способный воспринимать сигналы акустических маяков для выполнения локализации. Необходимо реализовать алгоритм локализации по сетке для этого робота. Проанализировать точность и наличие критических режимов в контексте трёх проблем локализации: глобальной локализации, отслеживания позиции и проблемы похищенного робота.

Часть III
Картографирование

9 Картографирование с помощью сеток занятости

9.1 Введение

В предыдущих двух главах обсуждалось применение вероятностных методов в задачах восприятия с низкой размерностью, то есть определение положения робота. При этом подразумевалось, что роботу уже дана предварительно созданная карта. Такое допущение справедливо для очень малого количества реальных сценариев, в которых карты априори доступны или же могут быть созданы вручную. В некоторых областях применения такой роскоши, как предварительно доступная карта, не бывает. Удивительно, но для большинства зданий недоступны даже планы помещений, созданные их архитекторами. А если планы имеются в наличии (и они точны), на них, как правило, не обозначены мебель и прочие элементы, которые, с точки зрения робота, определяют конфигурацию среды в той же степени, что окна и двери. Возможность построить карту «с нуля» может сильно уменьшить необходимые для установки мобильного робота усилия и позволит адаптироваться к изменениям среды без участия человека. Фактически, картографирование является одной из базовых возможностей по-настоящему автономных роботов.

Создание карт с помощью мобильных роботов является сложной задачей по целому ряду причин:

• Гипотетическое пространство всех возможных карт невероятно огромно. Поскольку карты определены в непрерывном пространстве, пространство всех карт имеет бесконечную размерность. Даже дискретные приближения, используемые в этой главе, такие, как аппроксимация по сетке, могут быть описаны 10^5 или более переменными. Один только размер такого пространства высокой размерности затрудняет вычисление полных апостериорных распределений для карт. Поэтому подход байесовской филь-

трации, который хорошо работал для локализации, непригоден для задачи изучения карт, по крайней мере, в его наивной интерпретации, которая обсуждалась до этого момента.

• Изучение карт представляет собой проблему «курицы и яйца» поэтому часто именуется одновременной локализацией и картографированием (simultaneous localization and mapping - SLAM) или конкурентной проблемой локализации и картографирования (concurrent mapping and localization problem). Во-первых, имеется проблема локализации. Когда робот перемещается в среде, он накапливает ошибки в одометрии, что постепенно увеличивает степень неопределенности относительно его местонахождения. Как было показано в предыдущей главе, существуют методы определения положения робота при условии наличия карты. Во-вторых, имеется проблема картографирования. Создание карты, когда положение робота известно, относительно простая задача. Это утверждение будет подтверждено доказательствами в этой и последующих главах. При отсутствии как изначально заданной карты, так и точных данных о положении робота, необходимо выполнять обе задачи: составлять карту и выполнить локализацию робота относительно этой карты.

Конечно, не все проблемы картографирования одинаково трудны. Сложность проблемы картографирования является результатом целого набора факторов, самые важные из которых следующие:

• **Размер.** Чем больше размер среды относительно расстояния восприятия робота, тем сложнее задача построения карты.

• **Шумы восприятия и действий.** Если бы датчики и актуаторы робота были идеальны, картографирование превратилось бы в очень простую задачу. Чем выше зашумление, тем сложнее задача.

• **Двойственность восприятия.** Чем более схожи между собой различные места, тем более сложно установить соответствие между различными местоположениями, через которые в разные моменты времени проходит робот.

• **Циклы.** Циклы в среде особенно сложны для картографирования. Если робот просто перемещается вперёд и назад по коридору, он может последовательно корректировать по возвращении. Циклы заставляют робота возвращаться разными путями, и при замыкании цикла накопленная ошибка одометрии просто огромна!

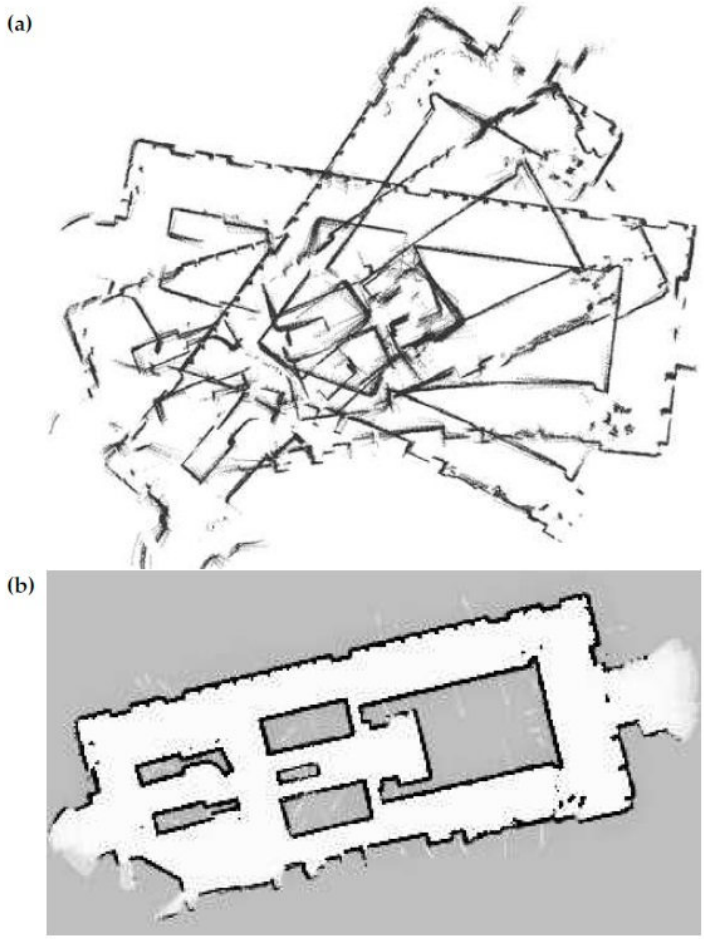


Рис. 9.1 (a) Сырые данные расстояний, положение индексируется одометрией. (b) Карта сетки занятости.

Чтобы в полной мере оценить сложность задачи картографирования, обратимся к Рис. 9.1. На нем показан набор данных, собранных в большом помещении. Рис. 9.1a был сгенерирован, используя исходные данные одометрии. Каждая чёрная точка на этом рисунке соответствует препятствию, обнаруженному лазерным датчиком расстояния робота. На Рис. 9.1b показан результат применения на этом наборе данных алгоритмов картографии, один из которых описывается в данной главе. Этот пример хорошо иллюстрирует проблему.

КАРТОГРАФИРОВАНИЕ С ИЗВЕСТНЫМ ПОЛОЖЕНИЕМ

В этой главе сначала следует изучить проблему при наличии ограничивающего допущения об известном положении робота. Другими словами, пока отложим в сторону трудности задачи SLAM, и представим, что некий оракул предоставляет точную информацию о пути, пройденном роботом при картографировании. На Рис. 9.2 представлена задача, также известная как картографирование с известным положением. Давайте обсудим популярное семейство алгоритмов под общим названием ”картографирование с помощью сеток занятости”, которые решают задачу генерирования непротиворечивых карт из зашумлённых и неопределённых данных измерений, при условии, что положение робота известно. Основная идея карт на основе сеток занятости состоит в представлении карты в виде поля случайных

переменных, распределенных по равномерной сетке. Каждая случайная бинарная переменная отображает состояние занятости местоположения, на котором находится. Алгоритмы картографирования на основе сеток занятости реализуют приблизительную апостериорную оценку для этих случайных переменных.

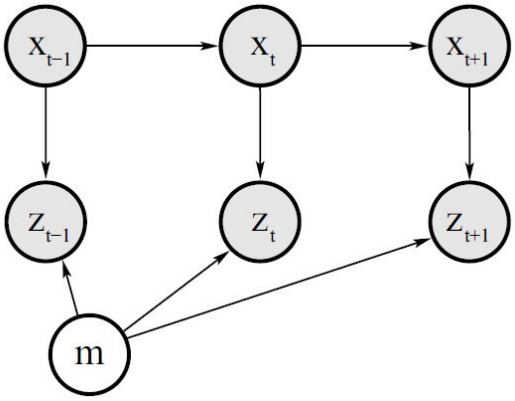


Рис. 9.2 Графическая модель картографирования с известными местоположениями. Переменные, отмеченные цветом (местоположения x и измерения z) известны. Целью картографирования является восстановление карты m.

Читатель может напомнить о важности метода картографирования, требующего точной информации о положении робота. В конце концов, идеальной одометрии для робота не существует! Основным применением метода сеток занятости является постпроцессинг, поскольку многие из техник SLAM, обсуждаемых в следующих главах, неспособны генерировать карты, подходящие для планирования пути и навигации. Карты сеток занятости часто используются после решения задачи SLAM какими-либо другими методами, принимая результирующие оценочные траектории как данность.

9.2 Алгоритм картографирования с помощью сеток занятости

Золотым стандартом любого алгоритма картографирования с помощью сеток занятости является вычисление апостериорного распределения по карту на основе имеющихся данных

(9.1)
$$p(m|z_{1:t}, x_{1:t})$$

Как обычно, m означает карту, $z_{1:t}$ - набор всех измерений до момента времени t, а $x_{1:t}$ траекторию робота, определённую через последовательность всех положений. Сигналы управления $u_{1:t}$ никакой роли в картах сеток занятости не играют, поскольку путь уже известен. Поэтому в этой главе они будут игнорироваться.

Для карт сеток занятости используются мелкоячеистые сетки на непрерывных пространствах местоположений. Наиболее популярной областью

использования карт сеток занятости являются поэтажные двухмерные карты, которые можно описать в виде плоского среза трехмерной среды. Плоские карты обычно наиболее предпочтительны, если робот действует в горизонтальной среде и датчики дают информацию только об узком её срезе. Методы сеток занятости способны к обобщению до трехмерных представлений, но за счёт существенных вычислительных затрат.

Пусть m_i определяет ячейку сетки с индексом i. Карта сеток занятости разбивает пространство на конечное множество ячеек сетки:

(9.2)
$$m = \{\mathbf{m}_i\}$$

Для каждого $\mathbf{m}_i$ имеется бинарное значение занятости, определяющее, свободна ли ячейка. Условимся, что "1" будет обозначать занятые, а "0" - свободные ячейки. Записью $p(\mathbf{m}_i = 1)$ или $p(\mathbf{m}_i)$ обозначим вероятность занятости данной ячейки сетки.

Проблемой апостериорного распределения в выражении (9.1) является размерность, поскольку количество ячеек сетки на картах наподобие показанной на Рис 9.1 составляет порядка десятков тысяч. Для каждой карты с 10000 ячеек количество вариантов карт, которые можно отобразить с её помощью, составляет 2^{10000}. Вычисление апостериорной вероятности для каждого варианта карты в таком случае совершенно нецелесообразно.

Стандартный метод сеток занятости разбивает задачу оценки карты на множество отдельных задач оценки

(9.3)
$$p(\mathbf{m}_i | z_{1:t}, x_{1:t})$$

для каждой ячейки сетки $\mathbf{m}_i$. Каждая из этих задач оценки представляет собой бинарную задачу статичного состояния. Такая декомпозиция удобна, хотя и имеет ряд ограничений. В частности, она не позволяет отображать зависимости между соседними ячейками, вместо этого апостериорное распределение по всем картам аппроксимируется в виде произведения маргиналов:

(9.4)
$$p(m | z_{1:t}, x_{1:t}) = \prod_i p(\mathbf{m}_i | z_{1:t}, x_{1:t})$$

Мы вернёмся к этому вопросу чуть ниже в разделе 9.4 при обсуждении более совершенных методов картографирования. Сейчас, для удобства, оставим эту факторизацию.

Из-за факторизации оценка вероятности занятости для каждой ячейки сетки становится бинарной задачей со статическим состоянием. Фильтр для её решения уже обсуждался в разделе 4.2, *это бинарный фильтр Байеса.* Соответствующий алгоритм приводится в Таблице 4.2 на странице 94.

1: **Algorithm occupancy_grid_mapping**$(\{l_{t-1,i}\}, x_t, z_t)$:
2: *для всех ячеек* $\mathbf{m}_i$ *выполнять*
3: *если* $\mathbf{m}_i$ *в области восприятия* z_t *то*
4: $l_{t,i} = l_{t-1,i} + \textbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) - l_0$
5: *иначе*
6: $l_{t,i} = l_{t-1,i}$
7: *endif*
8: *endfor*
9: *return* $\{l_{t,i}\}$

Таблица 9.1 Алгоритм сетки занятости в версии с бинарным фильтром Байеса из Таблицы 4.2.

В алгоритме в Таблице 9.1 этот фильтр используется в задаче картографирования с помощью сеток занятости. Как и в оригинальном фильтре, в предлагаемом алгоритме картографирования с помощью сеток занятости для отображения занятости используется *логарифм шансов*:

(9.5)
$$l_{t,i} = \log \frac{p(\mathbf{m}_i | z_{1:t}, x_{1:t})}{1 - p(\mathbf{m}_i | z_{1:t} x_{1:t})}$$

Это представление уже знакомо из раздела 4.2. Преимуществом логарифма шансов над представлением в виде вероятности является возможность избежать численной нестабильности для вероятностей вблизи нуля или единицы. Вероятности из отношения логарифма шансов легко восстановить:

(9.6)
$$p(\mathbf{m}_i | z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + \exp\{l_{t,i}\}}$$

Алгоритм **occupancy_grid_mapping** в Таблице 9.1 выполняет последовательный проход по всем ячейкам сетки i, обновляя те, которые попадают в конус измерения датчика z_t. Для таких ячеек значение занятости обновляется с помощью функции **inverse_sensor_model** в строке 4 алгоритма. Для остальных ячеек значение занятости остаётся неизменным, как показано в строке 6. Константа l_0 является априорной вероятностью занятости, выраженной в виде отношения логарифма шансов:

(9.7)
$$l_0 = \log \frac{p(\mathbf{m}_i = 1)}{p(\mathbf{m}_i = 0)} = \log \frac{p(\mathbf{m}_i)}{1 - p(\mathbf{m}_i)}$$

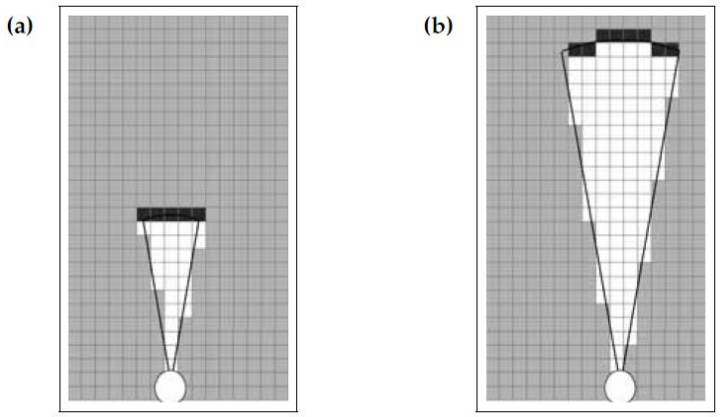


Рис. 9.3 Два примера обратной модели измерения **inverse_range_sensor_model** для двух различных расстояний измерения. Насыщенность цвета каждой ячейки соответствует правдоподобию её занятости. Модель несколько упрощена, и в современных реализациях вероятности занятости по границам конуса измерения обычно устанавливается ниже.

Функция **inverse_sensor_model** реализует обратную модель измерения $p(\mathbf{m}_i|z_t, x_t)$ в виде логарифма шансов:

(9.8)

$$\textbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) = \log \frac{p(\mathbf{m}_i|z_t, x_t)}{1 - p(\mathbf{m}_i|z_t, x_t)}$$

Несколько упрощённый пример такой функции для датчиков расстояния приведён в Таблице 9.2 и показан на Рис. 9.3a и b. Эта модель назначает всем ячейкам в конусе измерений, расстояние до которых близко к данным измерения, значение занятости l_{occ}. В Таблице 9.2 указаны ширина области, управляемой параметром α, и угол раскрытия луча β. Обратим внимание на упрощённый характер модели, поскольку в современных реализациях вероятности занятости на границах конуса измерений обычно устанавливается меньше.

Алгоритм **inverse_sensor_model** вычисляет обратную модель определением индекса луча k и расстояния r для центра масс ячейки $\mathbf{m}_i$. Это вычисление выполняется в строках с 2 до 5 в Таблице 9.2. Как обычно, допустим, что положение робота задано в виде $x_t = (x\ y\ \theta)^T$. В строке 7 возвращается значение априорной вероятности, выраженной в виде логарифма шансов, если ячейка лежит за пределами измеряемого расстояния этого луча датчика или далее, чем $\alpha/2$ обнаруженного расстояния z_t^k. В строке 9 возвращается $l_{occ} > l_0$, если дальность до ячейки более чем на $\pm\alpha/2$ превышает значение обнаруженного расстояния z_t^k. Значение $l_{free} < l_0$ возвращается, если расстояние до до ячейки короче измеренного расстояния больше, чем на $\alpha/2$. На левой и центральной панели на Рис. 9.3 показано вычисления основного конуса луча ультразвукового датчика.

1: **Algorithm inverse_range_sensor_model**$(\mathbf{m}_i, x_t, z_t)$:
2: *Пусть* x_i, y_i *является центром масс* $\mathbf{m}_i$
3: $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
4: $\phi = \text{atan2}(y_i - y, x_i - x) - \theta$
5: $k = \text{argmin}_j |\phi - \theta_{j,sens}|$
6: *если* $r > \min(z_{max}, z_t^k + \alpha/2)$ *или* $|\phi - \theta_{k,sens}| > \beta/2$ *то*
7: *return* l_0
8: *if* $z_t^k < z_{max}$ *and* $|r - z_t^k| < \alpha/2$
9: *return* $l_{занято}$
10: *if* $r \leq z_t^k$
11: *return* $l_{свободно}$
12: *endif*

Таблица 9.2 Простая инвертированная модель измерения для робота с ультразвуковыми датчиками расстояния. Здесь α - толщина препятствий, а β - ширина луча датчика. Значения $l_{занято}$ и $l_{свободно}$ в строках 9 и 11 означает степень уверенности, что показания относятся к двум различным случаям.

Типичное применение обратной модели датчика для ультразвуковых датчиков показано на Рис. 9.4. Начав с первоначальной картой, робот успешно расширяет ее, учитывая локальные карты, созданные обратной моделью. Карта сетки занятости большего размера, созданная этой моделью для той же среды, показана на Рис. 9.5.

На Рис. 9.6 показан пример карты рядом с планом большого открытого выставочного зала и соотнесение ее с картой сетки занятости, полученной роботом. Карта была сгенерирована, используя собранные за несколько минут данные лазерного датчика расстояния. Насыщенность цвета на карте сетки занятости означает апостериорную вероятность занятости на равномерной сетке. Чем темнее ячейка сети, тем более вероятно, что она занята. Хотя карты занятости изначально вероятностные, они быстро сводятся к двум граничным вероятностям, 1 и 0. При сравнении сгенерированной карты и плана видно, что на карте сетки занятости показаны все основные структурные элементы и препятствия, видимые с высоты лазера. При внимательном изучении читатель может заметить некоторые небольшие несоответствия между планами и настоящей конфигурацией среды.

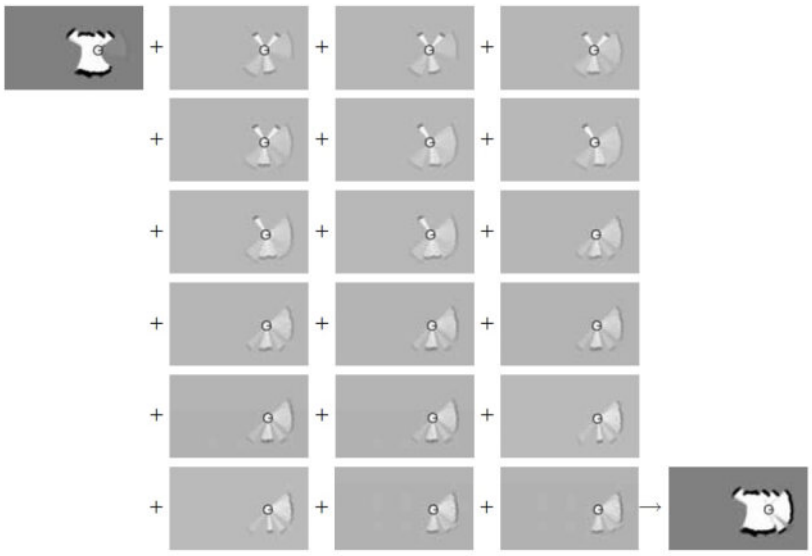


Рис. 9.4 Последовательное построение карты сетки занятости, используя данные ультразвукового датчика в среде в виде коридора. В левом верхнем углу показана начальная карта, а в правом нижнем – результирующая карта. В колонках со 2 по 4 показаны локальные карты, построенные или обратной модели датчика. Измерения за пределами радиуса 2,5 м не учитываются. Угол раскрытия каждого конуса измерения составляет 15 градусов. Изображения принадлежат Сириллу Стачнису из Университета Фрайбурга (Cyrill Stachniss, University of Freiburg).

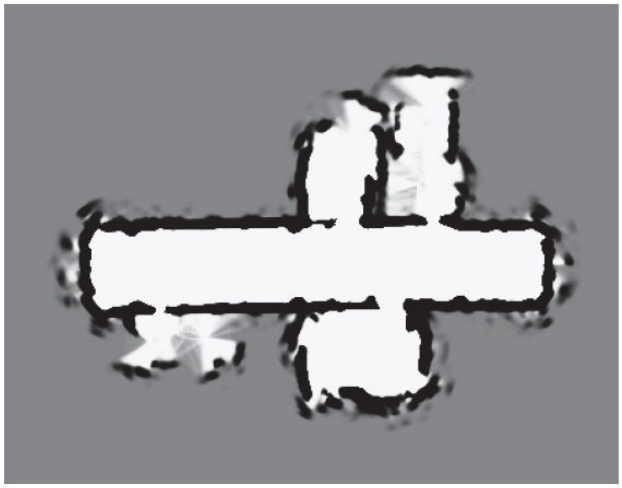


Рис. 9.5 Карта вероятности занятости офисной среды, построенная на основе измерений сонара. Изображение принадлежат Кириллу Стачнису из Университета Фрайбурга (Cyrill Stachniss, University of Freiburg).

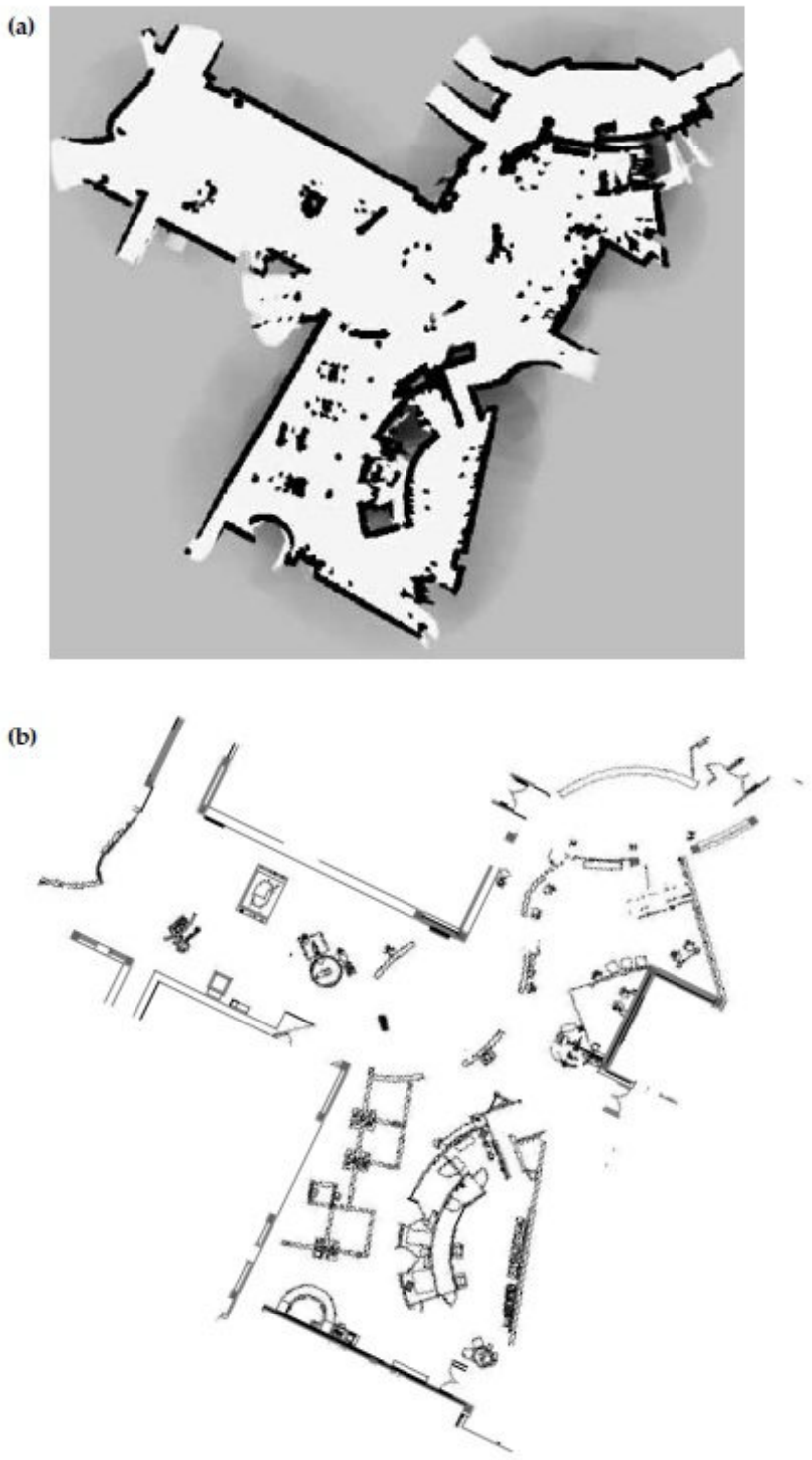


Рис. 9.6 (a) Карта сетки занятости и (b) архитектурный план большого пространства выставки. Заметны мелкие неточности плана.

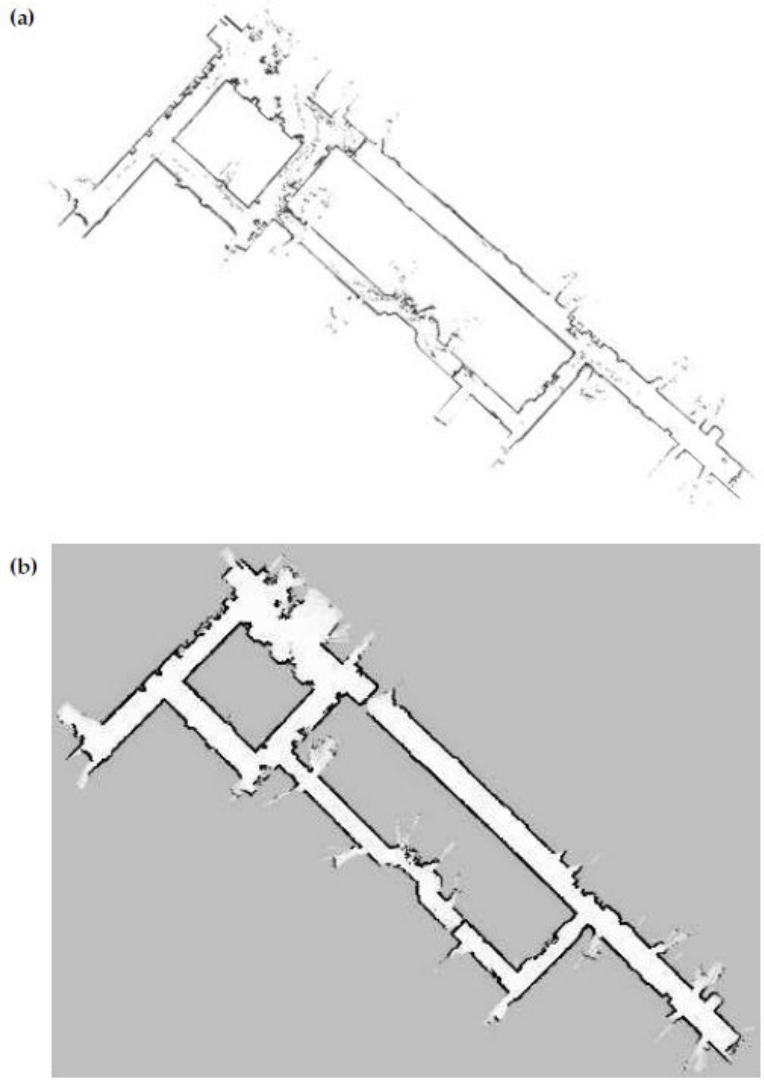


Рис. 9.7 Исходные данные сканирования расстояний лазером для скорректированного положения робота. Каждой точке соответствует обнаружение препятствия. Большинство препятствий статические (например, стены), но имеются и динамические, поскольку во время сбора данных рядом с роботом ходят люди (a). Карта сетки занятости, построенная на этих данных. Интенсивность серого соответствует вероятности: чёрный соответствует высокой вероятности того, что ячейка занята, белый – высокой вероятности того, что она свободна. Серый фон отображает априорное распределение (b). Рис. (a) принадлежит Стефену Гатманну (Steffen Gutmann).

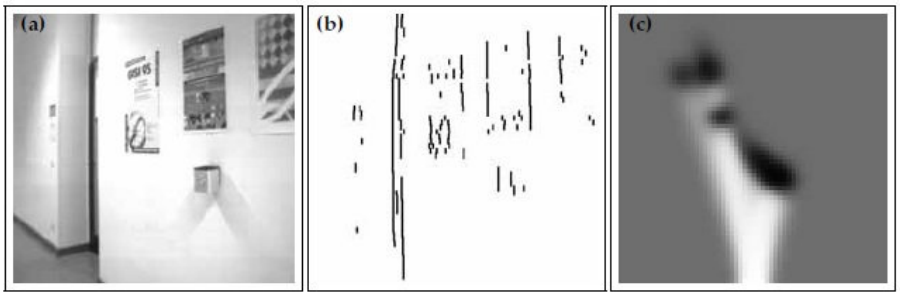


Рис. 9.8 Оценка карт сетки занятости на основе изображения со стереокамеры: изображение с камеры (a), разреженная карта диспаратности (b), карта занятости после проекции изображения диспаратности на плоскость и свёртки результата гауссовой функцией (c). Изображения принадлежат Торстену Фролингаусу (Thorsten Fröhlinghaus)

На Рис. 9.7 сравниваются исходный набор данных и карта сетки занятости, сгенерированная на основе этих данных. Данные на схеме (a) были предварительно обработаны алгоритмом SLAM таким образом, чтобы положения робота совпадали с картой. Часть данных повреждена из-за присутствия людей, но карта сетки занятости достаточно хорошо отфильтровывает людей. Это делает карты сетки занятости значительно более подходящими для навигации робота по сравнению со данными конечных точек сканирования, поскольку планировщик, принимающий на вход сырые данные конечных точек измерений, с большим трудом будет способен найти путь между близко расположенными препятствиями, даже если вероятность того, что ячейка свободна превышает вероятность того, что она занята.

Заметим, что этот алгоритм принимает решения о занятости ячейки на основании данных измерений. Альтернативным источником информации является положение самого робота: при положении робота x_t область вокруг x_t должна быть свободной. Алгоритм обратного измерения в Таблице 9.2 можно легко модифицировать для учёта этой информации путём возвращения большого отрицательного числа для всех ячеек сети, которые занимает робот, находясь в x_t. На практике следует учитывать размеры робота, особенно, если в процессе построения карты в среде присутствуют люди.

9.2.1 Слияние данных нескольких различных типов датчиков

Роботы часто оснащены более чем одним типом датчиков, поэтому естественным решением является интеграция информации на единой карте. Вопрос о наилучшем способе интеграции данных от нескольких датчиков особенно интересен, если датчики имеют разные характеристики. Например, на Рис. 9.8 показаны карты занятости, построены стереосистемой компьютерного зрения, в которой диспаратность проектируется на плоскость и сворачивается с помощью гауссовой функции. Конечно, характеристики стереоизображения отличны от данных расстояния ультразвукового датчика, поскольку разные датчики чувствительны для разных типов препятствий.

К сожалению, слияние данных от различных датчиков с помощью байесовских фильтров – задача крайне нетривиальная. Наивным решением бу-

дет запуск алгоритма **occupancy_grid_mapping** в Таблице 9.1 с разными модальностями датчиков. Такой подход имеет очевидный недостаток. Если разные датчики обнаруживают разные типы препятствий, результат байесовской фильтрации будет неточным. Допустим, препятствие может быть распознано одним типом датчика, но не обнаруживается другим. Тогда два разных типа датчиков будут генерировать противоречивую информацию, а результирующая карта – зависеть от количества подтверждающей информации, полученной от каждой системы. Обычно это нежелательно, поскольку занятость ячейки начнёт зависеть только от относительной частоты опроса датчиков.

Популярным методом интеграции информации от нескольких датчиков является построение раздельных карт для каждого типа датчика с последующей интеграцией карт подходящей комбинирующей функцией. Пусть $m^k = \{\mathbf{m}_i^k\}$ определяет карту, построенную на основе k-го типа датчика. Если измерения датчиков независимы друг от друга, можно напрямую комбинировать их, используя *законы де Моргана*

(9.9)
$$p(\mathbf{m}_i) = 1 - \prod_k (1 - p(\mathbf{m}_i^k))$$

Возможно также вычислить максимум

(9.10)
$$p(\mathbf{m}_i) = \max_k p(\mathbf{m}_i^k)$$

всех карт, отражающий наиболее пессимистичные оценки всех компонентов. Если хотя бы один из датчиков показывает, что ячейка занята, она будет занята на комбинированной карте.

9.3 Обучающиеся обратные модели измерений

9.3.1 Инвертируем модель измерений

Алгоритм картографирования на основе сетки занятости требует маргинализированной *обратной модели измерений* $p(\mathbf{m}_i|x, z)$. Эта вероятность называется "обратной" поскольку показывает зависимость событий от причин, давая информацию об изменении окружающего мира в результате измерения, в свою очередь, вызванном окружающей средой. Результатом является выражение для i-й ячейки сети, а полная инверсия будет давать вероятность $p(m|x, z)$. При изучении основного алгоритма в Таблице 9.2 была представлена специальная процедура для обратной модели. Возникает вопрос, возможно ли получить обратную модель в более точном виде, если имеется обычная модели измерения.

Это возможно, но несколько сложнее, чем кажется на первый взгляд. Теорема Байеса гласит

(9.11)
$$\begin{aligned} p(m|x, z) &= \frac{p(z|x, m)\, p(m|x)}{p(z|x)} \\ &= \eta\, p(z|x, m)\, p(m) \end{aligned}$$

Здесь по умолчанию подразумевается $p(m|x) = p(m)$, поскольку положение робота ничего не говорит о карте (допущение, которое будет сделано в целях удобства). Если нашей целью было вычисление обратной модели для всей карты, этого допущения будет достаточно. Но алгоритм картографирования на основе сеток занятости аппроксимирует апостериорное разделение по картам через их маргиналы, по одному для каждой ячейки сетки m_i. Обратная модель для i-й ячейки сетки получается выбором маргинала соответствующей ячейки:

(9.12)
$$p(\mathbf{m}_i|x,z) = \eta \sum_{m:m(i)=\mathbf{m}_i} p(z|x,m)\, p(m)$$

В этом выражении суммируются все карты m со значением занятости ячейки сетки i равным $\mathbf{m}_i$. Вообще-то, эту сумму нельзя вычислить, поскольку пространство всех карт слишком велико.

Опишем алгоритм аппроксимации этого выражения. Алгоритм включает генерацию значений из модели измерений, и аппроксимацию обратной модель с помощью *алгоритма обучения с учителем*, например, *логистической регрессии* или *искусственной нейронной сети*.

ОБУЧЕНИЕ С УЧИТЕЛЕМ

9.3.2 Выборка элементов из прямой модели

Основная идея проста и достаточна универсальна: если получится сгенерировать случайные триплеты из положений $x_t^{[k]}$, измерений $z_t^{[k]}$, и значений занятости карты $\mathbf{m}_i^{[k]}$ для произвольной ячейки сети $\mathbf{m}_i$, тогда получится обучить функцию, принимающую на вход положение x и измерение z, и возвращающую вероятность занятости ячейки $\mathbf{m}_i$.

Элемент вида $(x_t^{[k]}\ z_t^{[k]}\ \mathbf{m}_i^{[k]})$ может быть сгенерирован следующей процедурой.

1. Выбрать случайную карту $m^{[k]} \sim p(m)$. Например, можно извлечь случайную карту из имеющихся в наличии базы данных карт, которые представляют $p(m)$.

2. Выбрать положение $x_t^{[k]}$ внутри карты. Можно смело считать, что положения распределены равномерно.

3. Выбрать измерение $z_t^{[k]} \sim p(z|x_t^{[k]}, m^{[k]})$. Этот этап выборки напоминает симулятор робота, который случайным образом имитирует измерение датчика.

4. Извлечь искомое «истинное» значение занятости $\mathbf{m}_i$ для целевой ячейки сети карты m.

Результатом является выбранные положение $x_t^{[k]}$, измерение $z_t^{[k]}$, и значение занятости для ячейки сетки $\mathbf{m}_i$. Повторное применение выборки даст набор данных

$$\begin{array}{cccc} x_t^{[1]} & z_t^{[1]} & \longrightarrow & \mathrm{occ}(\mathbf{m}_i)^{[1]} \\ x_t^{[2]} & z_t^{[2]} & \longrightarrow & \mathrm{occ}(\mathbf{m}_i)^{[2]} \end{array}$$

$$x_t^{[3]} \quad z_t^{[3]} \quad \longrightarrow \quad \mathrm{occ}(\mathbf{m}_i)^{[3]}$$
$$\vdots \quad \vdots \qquad\qquad \vdots$$

ОБУЧАЮЩИЕ ПРИМЕРЫ

Эти триплеты могут служить *обучающими примерами* для алгоритма обучения с учителем, который аппроксимирует искомую условную вероятность $p(\mathbf{m}_i|z,x)$. Здесь измерения z и положение x являются входными переменными, а значение занятости $\mathrm{occ}(\mathbf{m}_i)$ - целью для вывода алгоритма.

Такой подход неэффективен, поскольку не способен проявлять ряд следующих свойств, которые, как мы знаем, будут иметь место для обратной модели датчика.

• Измерение не должно содержать информации о ячейках сетки вдали от расстояния восприятия. Это наблюдение влечёт два последствия. Во-первых, можно ограничить процессе генерирования элементов только триплетами, для которых ячейка $\mathbf{m}_i$ находится внутри конуса измерений. Во-вторых, при формулировке гипотезы для ячейки необходимо включить только подмножество данных в измерение z (то есть ближайшие бины) в качестве входа обучающегося алгоритма.

• Характеристики датчика инвариантны по отношению к абсолютным координатам робота или ячейке сетки при выполнении измерения. Имеют значения только относительные координаты. Если обозначить положение робота через $x_t = (x\ y\ \theta)^T$, а координаты ячейки сети как $m_i = (x_{m_i}\ y_{m_i})^T$, то они будут проецироваться на локальную систему координат с помощью следующих операций переноса и поворота:

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{\mathbf{m}_i} - x \\ y_{\mathbf{m}_i} - y \end{pmatrix}$$

В роботах с круговыми массивами датчиков расстояния имеет смысл кодировать относительное местонахождение ячейки сети, используя уже знакомые полярные координаты (расстояние и угол направления).

• Близлежащие ячейки сетки должны иметь схожую интерпретацию для обратной модели датчика. Это свойство гладкость подразумевает, что может оказаться выгодным обучить простую функцию с координатами ячейки сети в качестве входных данных, чем обучать отдельную функцию для каждой ячейки сети.

• Если у робота имеются функционально идентичные датчики, обратная модель датчика должна быть взаимозаменяемой для различных моделей датчиков. Для роботов, оборудованных круговым массивом датчиков расстояния, любой из результирующих лучей датчика характеризуется одной и той же обратной моделью датчика.

Самый простой способ обеспечить такую инвариантность - это ограничить обучающийся алгоритм выбором соответствующих входных переменных. Хорошим решением будет использование информации о положении, таким образом, чтобы обучающийся алгоритм не мог формулировать решение на основе абсолютных координат. Также стоит отбросить измерения датчика, для которых известно, что они иррелеванты по отношению к прогнозам занятости и ограничить прогноз только ячейками внутри поля измерения датчика. Используя это постоянство, размер тренировочного набора

можно существенно уменьшить.

9.3.3 Функция ошибок

Для тренировки обучающегося алгоритма понадобится оценочная функция ошибки. Популярным примером являются искусственные нейронные сети, тренированные с алгоритмом обратного распространения. Обратное распространение обучает *нейронные сети* с помощью *градиентного спуска* в пространстве параметров. Заданная функция ошибки, измеряющая «разницу» между текущим и искомым выводом, обратное распространение вычисляет первую производную целевой функции и параметров нейронной сети, и настраивает параметры в направлении, обратном градиенту, чтобы уменьшить разницу. Поэтому возникает вопрос о том, какую функцию ошибки использовать.

ОБРАТНОЕ РАСПРОСТРАНЕНИЕ

Общепринято обучать алгоритм таким образом, чтобы максимизировать логарифм правдоподобия обучающих данных. Пусть дан обучающий набор данных следующего вида

(9.13)
$$\begin{aligned} \text{input}^{[1]} &\longrightarrow \text{occ}(\mathbf{m}_i)^{[1]} \\ \text{input}^{[2]} &\longrightarrow \text{occ}(\mathbf{m}_i)^{[2]} \\ \text{input}^{[3]} &\longrightarrow \text{occ}(\mathbf{m}_i)^{[3]} \\ \vdots &\qquad\quad \vdots \end{aligned}$$

$\text{occ}(\mathbf{m}_i)^{[k]}$ - это k-й элемент искомой условной вероятности, а $\text{input}^{[k]}$ - соответствующий вход обучающегося алгоритма. Точная форма входных данных может отличаться как результат кодирования известной инвариантности, но как раз точный вид входной функции не играет никакой роли для формы функции ошибки.

Давайте определим параметры обучающегося алгоритма через W. Допустим, что каждый отдельный элемент в обучающих данных был сгенерирован независимо, тогда правдоподобие обучающих данных будет равно

(9.14)
$$\prod_i p(\mathbf{m}_i^{[k]} | \text{input}^{[k]}, W)$$

а дополнение логарифма

(9.15)
$$J(W) = -\sum_i \log p(\mathbf{m}_i^{[k]} | \text{input}^{[k]}, W)$$

Здесь J определяет функцию, которую необходимо минимизировать в процессе обучения.

Давайте определим обучающийся алгоритм $f(\text{input}^{[k]}, W)$. Выход функции – это значение в интервале $[0; 1]$. После обучения обучающийся алгоритм должен выдавать значения занятости:

(9.16)
$$p(\mathbf{m}_i^{[k]} | \text{input}^{[k]}, W) = \begin{cases} f(\text{input}^{[k]}, W) & \text{если } \mathbf{m}_i^{[k]} = 1 \\ 1 - f(\text{input}^{[k]}, W) & \text{если } \mathbf{m}_i^{[k]} = 0 \end{cases}$$

Таким образом, выполняется поиск функции ошибки для подстройки W так, чтобы минимизировать отклонение прогнозируемой вероятности от одного из примеров обучения. Чтобы найти такую функцию ошибки, перепишем (9.16) в следующем виде:

(9.17)

$$p(\mathbf{m}_i^{[k]}|\text{input}^{[k]}, W) = f(\text{input}^{[k]}, W)^{\mathbf{m}_i^{[k]}}(1 - f(\text{input}^{[k]}, W))^{1-\mathbf{m}_i^{[k]}}$$

Легко заметить, что это произведение и выражение (9.16) идентичны. В произведении один из членов всегда равен 1, поскольку экспонента равна нулю. Подстановка произведения в (9.15) и умножение результата на минус один даёт следующую функцию:

(9.18)

$$\begin{aligned} J(W) = & -\sum_i \log\left[f(\text{input}^{[k]}, W)^{\mathbf{m}_i^{[k]}}(1 - f(\text{input}^{[k]}, W))^{1-\mathbf{m}_i^{[k]}}\right] \\ = & -\sum_i \mathbf{m}_i^{[k]} \log f(\text{input}^{[k]}, W) + (1 - \mathbf{m}_i^{[k]}) \log(1 - f(\text{input}^{[k]}, W)) \end{aligned}$$

$J(W)$ – это функция ошибки, которую требуется минимизировать при обучении алгоритма. Она легко укладывается в любой алгоритм, который использует градиентный спуск для настройки параметров.

9.3.4 Примеры и дальнейшие соображения

На Рис. 9.9 показан результат работы искусственной нейронной сети, обученной имитировать обратную модель датчика. В этом примере робот оборудован круговым массивом ультразвуковых датчиков, установленном, приблизительно, на уровне крышки стола. На вход сети подаются относительное расстояние и направление на целевую ячейку, а также набор из пяти связанных измерений расстояния. На выходе будет вероятность занятости: чем темнее ячейка, тем вероятнее она занята. Как показывает пример, метод верно обучается различать свободное и занятое пространство. Равномерная серая окраска за препятствиями соответствует априорной вероятности занятости, поэтому в алгоритме картографирования с помощью сеток занятости ничего не изменяется. На Рис. 9.9b показаны ошибочно короткие показания датчика слева снизу. Здесь одиночного сканирования недостаточно для прогнозирования препятствия с высокой вероятностью.

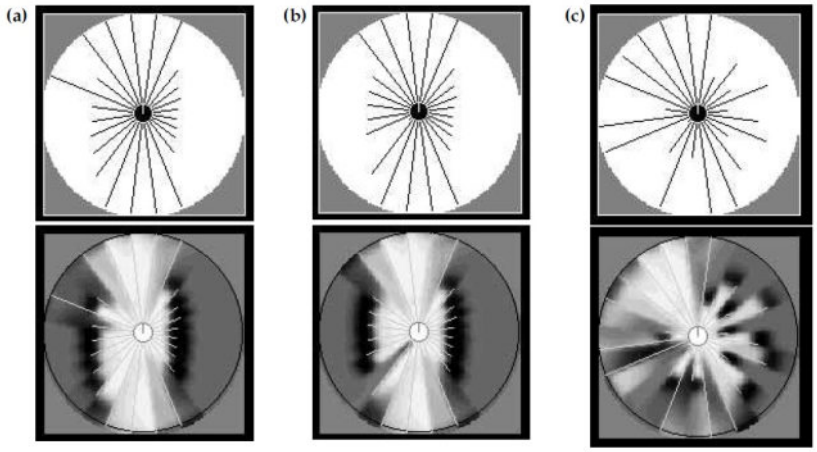


Рис. 9.9 Обратная модель датчика, обученная на данных: три значения сканирования сонара (верхний ряд) и локальные карты занятости (нижний ряд), сгенерированные нейронной сетью. Яркие области означают свободное пространство, а тёмные означают проходы сканирования и препятствия (увеличенные на диаметр робота).

Заметим, что существует множество способов обучить оценивающую функцию на основе данных, собранных роботом, вместо имитационных данных из прямой модели. В общем, это самые точные данные, которые возможно использовать для обучения, поскольку модель измерения всегда является лишь приближением. Одним из таких способов является робот, действующий в известной среде с известной картой. Используя марковскую локализацию, возможно выполнить локализацию, а затем использовать текущие измерения и известную карту занятости в качестве тренировочных примеров. Сначала возможно использовать даже приблизительную карту и обученную модель датчика для того, чтобы сгенерировать лучшую карту, и использовать только что описанную процедуру для улучшения обратной модели измерений.

9.4 Картографирование на основе максимумов апостериорной занятости

9.4.1 Необходимость сохранения зависимостей

В оставшейся части главы вернёмся в одному из самых базовых допущений алгоритма картографирования с помощью сеток занятости. В разделе 9.2 было сделано допущение о том, что можно безопасно разложить задачу получения карты, определённую в пространстве высокой размерности для всех карт, на совокупность задач картографирования единичных ячеек. Это допущение привело к следующей факторизации (9.4):

(9.19)
$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(\mathbf{m}_i|z_{1:t}, x_{1:t})$$

Возникает вопрос, насколько следует доверять результату любого алгоритма на основе полной декомпозиции.

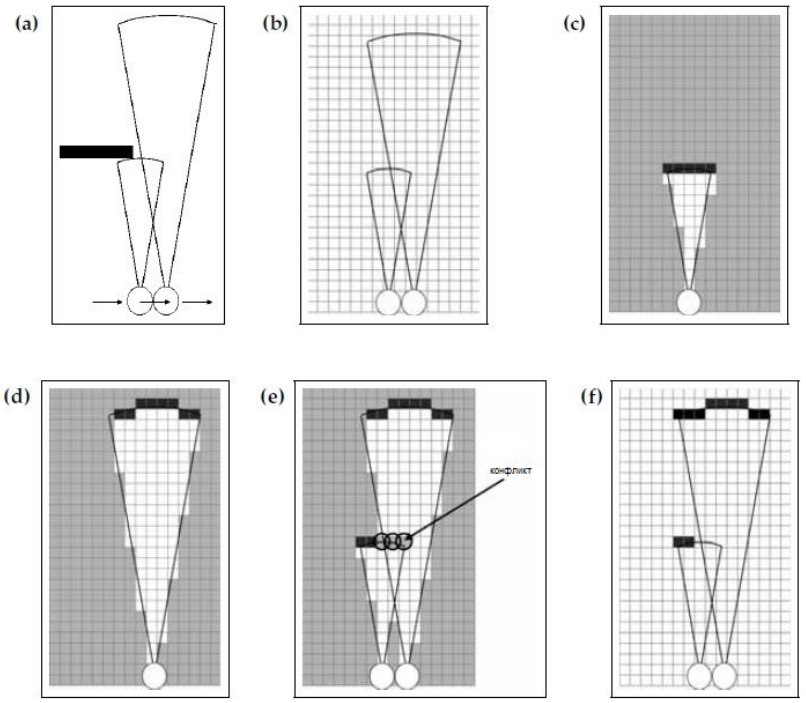


Рис. 9.10 Проблема стандартного алгоритма картографирования с использованием сеток занятости из раздела 9.2. Для среды, показанной на рисунке (a), проходящий робот может получить (идеальное) измерение, показанное на схеме (b). Метод факторизации по отдельности проецирует эти лучи для каждой ячейки сети и каждого луча, как показано на схемах (c) и (d). Результатом комбинирования обоих интерпретаций становится карта, показанная на схеме (e). Очевидно, имеются конфликтующие пересечения, показанные на схеме (e) в виде кругов. Интересным наблюдением является то, что существуют карты, показанные на схеме (f), объясняющие измерения карт безо всяких конфликтов. Для показаний датчиков, которые требуются объяснить, достаточно допустить, что препятствие находится где-то в конусе измерений, а не повсеместно.

На Рис. 9.10 показана задача, являющаяся прямым результатом такой факторизации. В данном случае робот, обращённый в сторону стены, принимает два идеальных измерения расстояния сонаром. Поскольку подход на основе множителей прогнозирует объект по всей дуге измеренного расстояния, значения занятости по всем ячейкам сетки вдоль дуги возрастают. При комбинировании двух разных измерений, показанных на Рис. 9.10c и d, получается конфликт, показанный на Рис. 9.10e. Стандартный алгоритм картографирования на основе сеток занятости «разрешает» этот конфликт суммированием положительных и отрицательных свидетельств занятости ячеек, но результат будет отражать отрицательные частоты двух типов измерений, что нежелательно.

Однако, существуют карты, наподобие показанное на Рис. 9.10f, которые объясняют измерения датчика безо всяких конфликтов. Это происходит потому, что для показаний датчика, которые необходимо объяснить достаточно допустить, что препятствие находится где-то в конусе измерения. Другими словами, факт того, что конус измерения проходит через несколь-

ко ячеек, выявляет важные зависимости между соседними ячейками сети. При декомпозиции картографирования для тысяч задач оценки ячейки сети возможность учёта этих зависимостей будет утеряна.

1: **Algorithm MAP_occupancy_grid_mapping**$(x_{1:t}, z_{1:t})$:
2: *установить* $m = \{0\}$
3: *повторять до сходимости*
4: *Для всех ячеек* $\mathbf{m}_i$ *выполнять*
5: $m_i = \underset{k=0,1}{\operatorname{argmax}}\ k\ l_0 + \sum_t \log$ **measurement_model**$(z_t, x_t, m$ *для* $\mathbf{m}_i = k)$
6: *endfor*
7: *endrepeat*
8: *return* m

Таблица 9.3 Алгоритм максимума апостериорной занятости сетки, использующий стандартные модели измерения вместо обратных моделей.

9.4.2 Картографирование с помощью сеток занятости, используя прямые модели

Эти зависимости учитываются алгоритмом, который выводит моду апостериорного распределения, вместо полного распределения. Мода определяется как максимум логарифма апостериорной вероятности карты:

(9.20)
$$m^* = \underset{m}{\operatorname{argmax}} \log p(m|z_{1:t}, x_{1:t})$$

Апостериорная вероятность карты умножается на априорную и правдоподобие измерения (см. Выражение (9.11)):

(9.21)
$$\log p(m|z_{1:t}, x_{1:t}) = \text{const.} + \log p(z_{1:t}|x_{1:t}, m) + \log p(m)$$

Логарифм правдоподобия $\log p(z_{1:t}|x_{1:t}, m)$ разбивается на сумму логарифмов правдоподобия отдельных измерений:

(9.22)
$$\log p(z_{1:t}|x_{1:t}, m) = \sum \log p(z_t|x_t, m)$$

Более того, логарифм априорной вероятности тоже можно подвергнуть декомпозиции. Заметим, что априорная вероятность любой карты m задана следующим произведением:

(9.23)
$$\begin{aligned} p(m) &= \prod_i p(\mathbf{m})^{\mathbf{m}_i} (1 - p(\mathbf{m}))^{1-\mathbf{m}_i} \\ &= (1 - p(\mathbf{m}))^N \prod_i p(\mathbf{m})^{\mathbf{m}_i} (1 - p(\mathbf{m}))^{-\mathbf{m}_i} \\ &= \eta \prod_i p(\mathbf{m})^{\mathbf{m}_i} (1 - p(\mathbf{m}))^{-\mathbf{m}_i} \end{aligned}$$

Здесь $p(\mathbf{m})$ - априорная вероятность занятости (то есть, $p(\mathbf{m}) = 0.5$), а N - количество ячеек сетки на карте. Выражение $(1 - p(\mathbf{m}))^N$ является просто константой, которая заменяется общим символом η.

Теперь можно получить логарифмическую версию априорной вероятности:

(9.24)

$$\begin{aligned}\log p(m) &= \text{const.} + \sum_i \mathbf{m}_i \log p(\mathbf{m}) - \mathbf{m}_i \log(1 - p(\mathbf{m})) \\ &= \text{const.} + \sum_i \mathbf{m}_i \log \frac{p(\mathbf{m})}{1 - p(\mathbf{m})} \\ &= \text{const.} + \sum_i \mathbf{m}_i \, l_0\end{aligned}$$

Константа l_0 берётся из (9.7). Член $Mlog(1-p(\mathbf{m}_i))$ очевидно, не зависит от карты. Поэтому, достаточно оптимизировать правдоподобие оставшегося выражения и данных:

(9.25)

$$m^* = \underset{m}{argmax} \sum_t \log p(z_t|x_t, m) + l_0 \sum_i \mathbf{m}_i$$

Алгоритм поиска экстремума для логарифма вероятности приводится в Таблице 9.3. Он начинается с полностью пустой карты (строка 2). Он "переворачивает" значение занятости ячейки сети, когда правдоподобие переворота превышает правдоподобие данных (строки 4-6). Для этого алгоритма важно, чтобы априорная вероятность занятости $p(\mathbf{m}_i)$ не была слишком близка к 1, в противном случае, результатом будет карта со всеми занятыми ячейками. Для любого алгоритма поиска экстремума этот метод гарантирует только нахождение локального максимума. На практике локальных максимумов обычно немного, если они вообще есть.

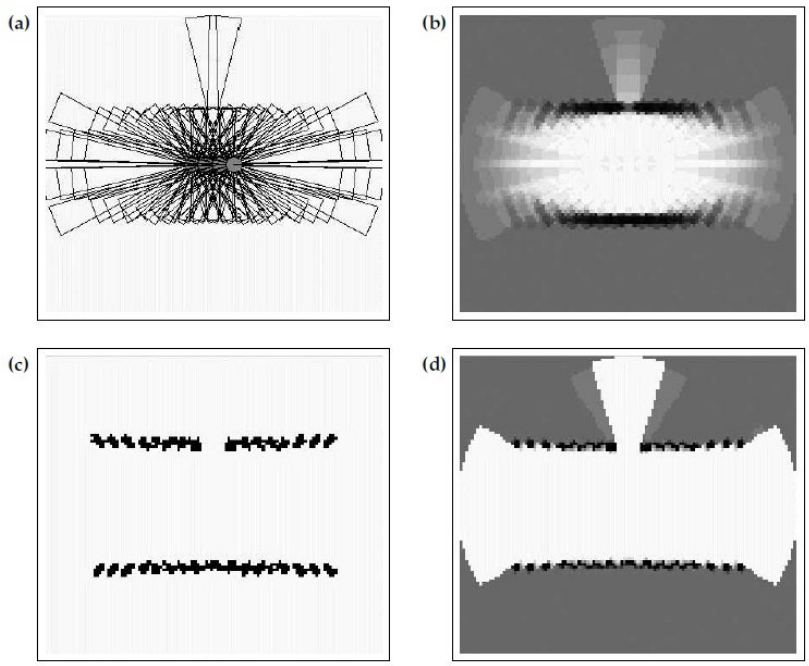


Рис. 9.11 Измерения дальности сонаром из идеальной имитации (a). Результаты стандартного алгоритма классификации на основе занятости, за исключением открытой двери (b). Максимум апостериорной карты (c). Остаточная неопределённость на карте, полученная измерениями чувствительности функции правдоподобия карты по отношению к отдельным ячейкам сетки (d). На этой карте чётко видна дверь, а также более ровные стены с обеих сторон.

На Рис. 9.11 показан эффект алгоритма сетки занятости MAP. На Рис. 9.11a показан набор идеальных данных для робота, проходящего через открытую дверь. Некоторые из измерений сонара обнаруживают открытую дверь, но остальные отражаются от дверного косяка. Стандартный алгоритм картографирования на основе занятости с обратными моделями измерений неспособен обнаружить открытую дверь, как показано на Рис. 9.11b. Мода апостериорного распределения показана на Рис. 9.11c. Эта карта верно моделирует открытую дверь, поэтому она лучше подходит для навигации робота по сравнению со стандартным алгоритмом карты сетки занятости. На. Рис. 9.11d показана остаточная неопределённость карты. Эта схема является результатом анализа чувствительности по каждой ячейке: величина, на которую инверсия состояния ячейки уменьшает функцию логарифма правдоподобия показана насыщенностью серого цвета. Эта схема, похожая на обычную карту сетки занятости, показывает максимум неопределённости для ячеек сети, расположенных за препятствиями.

Существуют определённые ограничения алгоритма **MAP_occupancy_grid_mapping**, и он может быть улучшен различными способами. Алгоритм максимизирует апостериорную вероятность и не возвращает никаких данных о неопределённости карты. Анализ чувствительности аппроксимирует эту неопределённость, но с чрезмерно оптимистической оценкой, поскольку проверяется только локальная мода. Более того, алгоритм является пакетным и не может выполняться последовательно. Фактически, алгоритм

MAP требует хранения всех данных в памяти. С вычислительной стороны, алгоритм можно ускорить инициализацией результатом работы обычного метода картографирования с помощью сетки занятости вместо пустой карты. Наконец, заметим, что только небольшое число измерений подвержены перевороту ячейки сети в строке 5 Таблицы 9.3. Хотя каждая сумма потенциально велика, при вычислении argmax необходимо проверить только небольшое количество элементов. Это свойство можно использовать в базовом алгоритме для увеличения вычислительной эффективности.

9.5 Выводы

В этой главе представлены алгоритмы обучающихся сеток занятости. Во всех представленных алгоритмах требуется точная оценка положения робота, поскольку они не решают проблему глобального построения карт.

• Стандартный алгоритм картографирования на основе определения занятости каждой отдельной ячейки оценивает апостериорную вероятность занятости и представляет собой адаптацию бинарного байесовского фильтра для статических сред.

• Данные с нескольких датчиков могут быть объединены на одной карте двумя способами: поддержания единственной карты с помощью байесовских фильтров и поддержанием нескольких карт, по одной для каждого типа датчика. Предпочтителен обычно второй подход, поскольку разные датчики чувствительны к разным типам препятствий.

• Стандартный алгоритм картографирования на основе занятости основан на обратных моделях измерения, идущих от следствия (измерения) к причинам (занятости). Это отличается от предыдущих реализаций байесовских фильтров в контексте локализации, поскольку они были основаны на обычной модели измерения, связывающей причину со следствием.

• Возможно обучить обратные модели датчиков на основе обычной модели измерений, моделирующей датчик от причины к следствию. Чтобы это сделать, необходимо сгенерировать выборку и обучить обратную модель, используя алгоритм обучения с учителем.

• Стандартный алгоритм картографирования на основе занятости не поддерживает зависимости оценок занятости. Это является результатом декомпозиции задачи апостериорной оценки карты на большое число задач апостериорной оценки единичных ячеек.

• Апостериорное значение для всей карты обычно не вычислимо из-за большого количества карт, которые можно определить на сетке. Однако, его возможно максимизировать, что позволит создавать карты, в большей степени соответствующие данным по сравнению с алгоритмами сетки занятости на основе байесовских фильтров. Но для выполнения максимизации требуется доступность всех данных, а результирующий максимум апостериорной карты не сохраняет остаточную неопределённость карты.

Без сомнения, карты сетки занятости и их различные варианты чрезвычайно популярны в робототехнике в силу их простоты получения и сохра-

нения важных для навигации робота элементов.

9.6 Библиографические примечания

Карты сеток занятости были введены Элфисом (Elfes, 1987), в кандидатской диссертации которого (1989) была определена предметная область. Известная статья Моравица (Moravec, 1988) дала очень доступную формулировку задачи, заложив базовый вероятностный метод, который составляет суть этой главы. В неопубликованной работе Моравица и Мартина (Moravec and Martin,1994) карты сетки занятости были обобщены до 3D со стереокамерой в качестве основного датчика. Слияние данных нескольких датчиков в сетках занятости было впервые описано в работе Труна (Thrun et al., 1998a). Результат обучения обратных моделей датчиков, описанных в этой главе, можно найти в работе Труна (Thrun, 1998b). Метод прямого моделирования, также описанный в этой главе, основан на похожем алгоритме Труна (Thrun, 2003).

Карты занятости были использованы в целом ряде различных областей. Боренштейн и Корин (Borenstein and Koren, 1991) первыми применили карты сетки занятости к задаче предотвращения столкновений. Многие авторы использовали карты сетки занятости для задачи локализации перекрёстным сравнением двух карт. Такие алгоритмы "наложения карт" детально обсуждались в Главе 7. Бисвас (Biswas et al., 2002) использовал карты сеток занятости для обучения модели определения формы подвижных объектов в динамических средах. Этот метод позже был расширен до обучения иерархических классов моделей динамических объектов, представленных в виде карт сетки занятости (Anguelov et al. 2002). Карты сетки занятости также активно использовались в контексте одновременной локализации и картографирования. Эти области применения будут обсуждаться ниже.

Идея представления пространства является только одной из многих идей, исследуемых в литературе по мобильной робототехнике. Классические работы по планированию движения часто полагаются на представление среды в виде полигонов, но не раскрывают, каким образом эти модели были получены из данных (Schwartz et al. 1987). Ранние работы по построению полигональных карт были выполнены Чатила и Лаумондом (Chatila and Laumond, 1985). Первая реализация, использующая калмановские фильтры для соединения прямых, полученных из данных сонара, была выполнена Краули (Crowley, 1989). В более новой работе Ангелов (Anguelov et al., 2004) предложил методы идентификации прямых линий дверей из сырых данных датчиков, а также изучил визуальные атрибуты для улучшения коэффициента обнаружения двери.

Рис. 9.12 Мобильный робот для работы в помещениях типа RWI B21, оснащённый 24 ультразвуковыми датчиками, смонтированными в круговой массив на корпусе робота.

ПРОСТРАНСТВЕННАЯ СЕМАНТИЧЕСКАЯ ИЕРАРХИЯ

Ранние идеи пространственного отображения топологической парадигмы, в которой пространство представлено в виде набора локальных областей, часто соответствует отдельным действиям робота по навигации между соседними местоположениями. Примеры топологического картографирования включают работу Куперса и Левитта (Kuipers and Levitt, 1988) по *пространственной семантической иерархии* (см также Kuipers et al., 2004)), магистерскую работу Матарика (Mataric, 1990), работу Кортенкампа и Веймауса (Kortenkamp and Weymouth, 1994) по топологическим графам, полученным из визуальных данных и данных ультразвуковых датчиков, и метод Шатки и Кэлблинга (Shatkay and Kaelbling, 1997) по использованию пространственных HMM с информацией о длине дуги. Карты сетки занятости используют родственную прадигму метрического отображения, которое прямо описывает среду робота в некоторой абсолютной координатной системе. Вторым примером метрического подхода является алгоритм EKF SLAM который будет обсуждаться в следующей главе.

Было сделано множество попыток создать алгоритмы картографирования, сочетающие лучшее обеих парадигм, топологической и метрической. Томатис (Tomatis et al., 2002) применил топологические выражения к непротиворечивому замыканию циклов, переводя их затем в метрические карты. Трун (Thrun, 1998b) первым построил метрическую карту сетки занятости, а затем предложил топологическую основу для быстрого планирования движения. В Главе 11 будут изучены методы, соединяющие обе парадигмы, метрическую и топологическую.

9.6.1 Упражнения

1. Изменить базовый алгоритм сетки занятости в Таблице 9.1, включив учёт изменения занятости со временем. Для учёта таких изменений, доказательства, собранные Δt тактов времени назад, должны быть уменьшены в $\alpha^{\Delta t}$

ЭКСПОНЕНЦИАЛЬНОЕ ЗАТУХАНИЕ

раз для некоего значения $\alpha < 1$(например, $\alpha = 0,99$). Такое правило называется *экспоненциальным затуханием.* Привести алгоритм построения карт с помощью сетки занятости и экспоненциального затухания в виде логарифма шансов и объяснить его правильность. Если не сможете найти точный алгоритм, привести приблизительный и объяснить, почему применимо предлагаемое приближение. Для простоты можно допустить априорную вероятность занятости $p(\mathbf{m}_i) = 0.5$.

2. Бинарный байесовский фильтр основан на допущениях, что ячейка может быть только занятой или свободной, а датчик даёт зашумлённые свидетельства в пользу верной гипотезы. В этом упражнении требуется построить альтернативный метод оценки ячейки сети: Допустим, датчик способен измерять только "0 = незанято" или "1 = занято", и генерирует последовательность

0, 0, 1, 0, 1, 1, 1, 0, 1, 0.

Какого максимальное правдоподобие вероятности p того, что следующее значение показания датчика будет 1? Привести инкрементную формулу общей оценивающей функции максимального правдоподобия для этой вероятности p. Обсудить разницу этой оценивающей функции и бинарного фильтра Байеса (только для отдельной ячейки).

3. Мы уже изучили общую конфигурацию датчиков роботов для работы внутри помещений. Допустим, робот, находящийся в помещении, использует ультразвуковой датчик с углом раскрытия 15 градусов, установленный на фиксированной высоте горизонтально и параллельно полу. Такой робот показан на Рис. 9.12. Обсудить, что может произойти, если робот встретит препятствие, которое находится чуть ниже датчика (например, на 15 см ниже). В частности, ответить на следующие вопросы:

(a) При каких условиях робот обнаружит препятствие? Когда не сможет обнаружить его? Привести краткий ответ.

(b) Какие следствия это имеет для бинарного байесовского фильтра и лежащего в его основе марковского свойства? Как можно вывести из строя алгоритм занятости сетки?

(c) На основе ответа на предыдущий вопрос можно ли предложить улучшенный алгоритм картографирования с помощью сетки занятости, который будет более надёжно обнаруживать препятствия по сравнению с базовым?

4. В этом упражнении требуется разработать простую модель датчика. Допустим, даны бинарные измерения занятости для следующих четырёх ячеек:

Номер ячейки	Тип	последовательность						
ячейка 1	занята	1	1	0	1	0	1	1
ячейка 2	занята	0	1	1	1	0	0	1
ячейка 3	свободна	0	0	0	0	0	0	0
ячейка 4	свободна	1	0	0	1	0	0	0

Какова модель максимального правдоподобия измерения $p(z|\mathbf{m}_i)$? (Подсказка: $\mathbf{m}_i$ – бинарная переменная занятости, а z – бинарная переменная измерения.)

5. Для Таблицы из Упражнения 4 реализовать базовый алгоритм сетки занятости.

(a) Какова апостериорная вероятность $p(\mathbf{m}_i|z_{1:7})$ для четырёх разных случаев, если априорная $p(\mathbf{m}_i) = 0,5$?

(b) Привести алгоритм настройки для модели датчика приближающий вывод алгоритма картографирования на основе сетки занятости как можно ближе к истинным значениям для четырёх случаев в Упражнении 4. Что удалось найти? (В этом упражнении потребуется найти подходящую меру близости.)

6. Стандартный алгоритм построения карт на основе сетки занятости реализован в виде логарифма шансов, хотя он также может быть реализован, используя вероятности.

(a) Вывести обновлённое правило, напрямую выражающее вероятности занятости, без использования представления в виде логарифма шансов.

(b) Для реализации на популярном языке программирования, например, C++, привести пример, в котором реализация вероятности даёт *разные* результаты с реализацией в виде логарифма шансов, в силу разного числового представления. Объяснить ответ и обсудить, станет ли эта разница проблемой при практическом использовании.

10 Одновременная локализация и картографирование

10.1 Введение

Эта и последующие главы посвящены одной из самых фундаментальных проблем робототехники, *проблеме одновременной локализации и картографирования*. Эта проблема обычно сокращается как SLAM, а также известна как "конкурентная картография и локализация", или CML. Проблемы SLAM возникает, когда у робота нет доступа ни к карте окружения, ни к точной информации о своём положении. Всё, что дано, это измерения $z1 : t$ и сигналы управления $u1 : t$. Термин «одновременная локализация и построение карт» описывает результирующую проблему: в SLAM роботу требуется получить карту окружающей среды, с одновременной локализацией на этой карте. SLAM существенно сложнее всех проблем, которые были рассмотрены до текущего момента. Он более сложен, чем локализация, поскольку карта неизвестна и ее следует строить прямо во время движения. Она более сложна, чем картографирование, поскольку положения неизвестны, и их необходимо определить.

С вероятностной точки зрения, есть две разновидности задачи SLAM, одинаково важных на практике. Первая известна как *проблема онлайн SLAM* и заключается в оценке апостериорного распределения мгновенного значе-

ПРОБЛЕМА ОНЛАЙН SLAM

ния положения по карте:

(10.1)
$$p(x_t, m|z_{1:t}, u_{1:t})$$

Здесь x_t положение в момент времени t, m карта, а $z_{1:t}$ и $u_{1:t}$ сигналы измерения и управления, соответственно. Эта проблема называется онлайн SLAM, поскольку включает оценку переменной, которая сохраняется в момент времени t. Многие алгоритмы онлайн SLAM инкрементны: они отбрасывают прошлые измерения и сигналы управления после обработки. Графическая модель онлайн SLAM показана на Рис. 10.1.

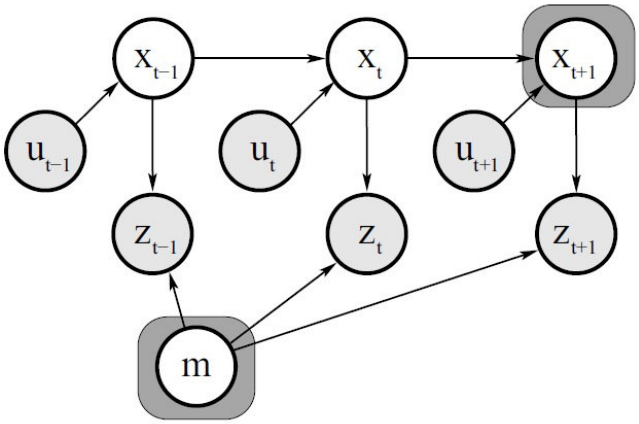


Рис. 10.1 Графическая модель проблемы онлайн SLAM. Целью онлайн SLAM является оценкой апостериорной вероятности по текущему положению робота, а также карту.)

ПОЛНАЯ ПРОБЛЕМА SLAM

Вторая проблема SLAM называется *полная проблема SLAM*. В полной проблеме SLAM требуется вычислить апостериорную вероятность по всему пути $x_{1:t}$ и карту, вместо одного лишь текущего положения x_t (см. также Рис. 10.2):

(10.2)
$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

Эти мелкие различия между онлайн и полной задачей SLAM влияют на алгоритмы, которые требуется сформулировать. В частности, проблема онлайн SLAM является результатом интеграции всех прошлых положений из полной проблемы SLAM:

(10.3)
$$p(x_t, m|z_{1:t}, u_{1:t}) = \int \int \dots \int p(x_{1:t}, m|z_{1:t}, u_{1:t})\, dx_1\, dx_2 \dots dx_{t-1}$$

В онлайн SLAM эта интеграция обычно выполняется по одному положению. Это вызывает интересные изменения структуры зависимостей в SLAM, которые будут в полной мере исследованы в следующей главе.

Второй ключевой характеристикой SLAM является проблема природы оценки. Проблемы SLAM содержат как непрерывный, так и дискретный

вариант. Проблема непрерывной оценки относится к местоположению объектов на карте и собственным переменным положения робота. Объекты могут быть ориентирами в представлении на основе признаков или же частями объектов, обнаруженными датчиками расстояния. Дискретная природа связана с необходимостью иметь дело с соответствием: при обнаружении препятствия алгоритм SLAM должен соотнести этот объект с ранее обнаруженными. Это соотнесение обычно дискретно, то есть объект тот же самый, что был обнаружен ранее, или же нет.

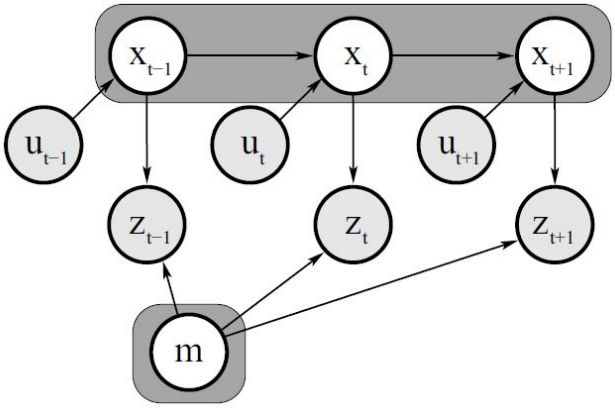


Рис. 10.2 Графическая модель полной проблемы SLAM. Здесь вычисляется полное апостериорное распределение по всему пути робота и карте.)

Мы уже сталкивались с похожими проблемами непрерывной-дискретной оценки в предыдущих главах. Например, локализация на основе EKF в разделе 7.4 выполняет оценку положения робота, которая непрерывна. Но, чтобы это сделать, необходимо установить соответствия измерений и ориентиров на карте, которые дискретны. В этой и последующей главах будут обсуждаться несколько различных методов для работы с непрерывными и дискретными аспектами проблемы SLAM.

Иногда будет полезно явно выделять переменные соответствия, как делалось в Главе 7 по локализации. Апостериорная вероятность онлайн SLAM задана, как

(10.4)
$$p(x_t, m, c_t | z_{1:t} u_{1:t})$$

а полной задачи SLAM

(10.5)
$$p(x_{1:t}, m, c_{1:t} | z_{1:t}, u_{1:t})$$

Онлайновая апостериорная вероятность получается из полной апостериорной интеграцией прошлых положений робота и суммирования по всем прошлым соответствиям:

(10.6)
$$p(x_t, m, c_t | z_{1:t}, u_{1:t}) = \int \int \dots \int \sum_{c_1} \sum_{c_2} \dots \sum_{c_{t-1}} p(x_{1:t}, m, c_{1:t} | z_{1:t}, u_{1:t}) \, dx_1 \, dx_2 \dots dx_{t-1}$$

В обоих версиях проблемы SLAM, и онлайн и полной, оценка полной апостериорной вероятности (10.4) или (10.5) является золотым стандартом SLAM. Полная апостериорная вероятность здесь отображает все, что известно о карте и положении ли пройденном пути.

На практике, вычисление полной апостериорной вероятности обычно невыполнимо. Проблемы возникают из-за двух причин: (1) высокой размерности непрерывного пространства параметров, и (2) большого числа дискретных переменных соответствия. Многие современные алгоритмы SLAM конструируют карты с десятками тысяч признаков, или даже более. Даже при известном соответствии, апостериорная вероятность только по этим картам включает вероятностные распределения с 10^5 или более измерений. Это находится в контрасте с проблемами локализации в котором апостериорная вероятность оценивалась по трехмерным непрерывным пространствам. Далее, в большинстве приложений соответствия неизвестны. Количество возможных назначений к вектору всех переменных соответствия $c_{1:t}$ растет экспоненциально за время t. Поэтому, практические алгоритмы SLAM, которые способны решить проблему соответствия, должны быть основаны на приближениях.

Проблема SLAM будет обсуждаться в большинстве следующих глав. До конца этой главы будет разработан алгоритм EKF для проблемы онлайн SLAM. Большая часть этого материала основана на разделе раздела 3.3, где были описаны EKF, и раздела 7.4, где EKF было применено в задаче локализации мобильного робота. Также будет выведено развитие алгоритмов EKF, которые сначала позволит использовать EKF для SLAM с известным соответствием, а затем обобщить для случая с неизвестным соответствием.

10.2 SLAM с обобщенными фильтрами Калмана

10.2.1 Исходные условия и допущения

Исторически самый ранний и, возможно, имеющий решающее значение, алгоритм SLAM основан на обобщенном фильтре Калмана или EKF. В двух словах, в алгоритме EKF SLAM обобщенный калмановский фильтр применяется для онлайн SLAM, используя ассоциацию данных максимального правдоподобия. В силу этого, EKF SLAM подвержен некоторым аппроксимациям и ограничивающим допущениям:

КАРТЫ НА ОСНОВЕ ПРИЗНАКОВ

Карты в EKF SLAM *основаны на признаках* и состоят из точечных ориентиров. В силу ограничений вычислительной мощности, количество точечных ориентиров достаточно мало (то есть менее 1000). Более того, подход EKF хорошо работает и с менее различимыми ориентирами. По этой причине для EKF SLAM требуется существенная доработка детекторов признаков, иногда используя в качестве ориентиров искусственные маяки.

ГАУССОВСКИЙ ШУМ

Как любой алгоритм EKF, EKF SLAM использует допущение о *гауссовском шуме* для движения и восприятия робота. Величина неопределенности в апостериорной вероятности должна быть относительно малой, поскольку в противном случае линеаризация EKF начинает генерировать неприемлемые ошибки.

ПОЛОЖИТЕЛЬНАЯ ИНФОРМАЦИЯ

Алгоритм EKF SLAM так же как алгоритм локализации EKF, обсуждаемый в разделе 7.4, способен обрабатывать только *положительные* наблюдения ориентиров. Он неспособен обрабатывать отрицательную информацию, возникающую из отсутствия ориентиров в измерениях датчика. Это является прямым следствием использования отображения гауссовых оценок и обсуждалась в разделе 7.4.

10.2.2 SLAM с известным соответствием

Алгоритм SLAM для случая с известным соответствием решает только непрерывную часть проблемы SLAM. Его вывод во многом параллелен выводу алгоритма локализации на основе EKF в разделе 7.4, но с одной ключевой разницей: вдобавок к оценке положения робота x_t, алгоритм EKF SLAM также оценивает координаты всех ориентиров, которые встретились на пути. Это делает необходимым включение координат ориентира в вектор состояния.

КОМБИНИРОВАННЫЙ ВЕКТОР СОСТОЯНИЯ

Для удобства, назовем вектор состояния, включающий положение робота и карту *комбинированным вектором состояния*, и обозначим вектор y_t. Комбинированный вектор задан в виде
(10.7)

$$\begin{aligned} y_t &= \begin{pmatrix} x_t \\ m \end{pmatrix} \\ &= (x\ y\ \theta\ m_{1,x}\ m_{1,y}\ s_1\ m_{2,x}\ m_{2,y}\ s_2 ... m_{N,x}\ m_{N,y}\ s_N)^T \end{aligned}$$

Здесь x, y и θ означают координаты робота в момент времени t (не следует путать с переменными состояния x_t и y_t), $m_{i,x}$, $m_{i,y}$ координаты i-го ориентира, для $i = 1, ..., N$, а s_i - его сигнатура. Размерность этого вектора состояния составляет $3N + 3$, где N означает количество ориентиров на карте.

1: **Algorithm EKF_SLAM_known_correspondences**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

2: $$F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & \underbrace{0...0}_{3N} \end{pmatrix}$$

3: $$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

4: $$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

5: $\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + F_x^T \, R_t \, F_x$

6: $$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

7: *для всех наблюдаемых признаков* $z_t^i = (r_t^i \; \phi_t^i \; s_t^i)^T$ *выполнять*

8: $j = c_t^i$

9: *если ориентир* j *еще не был наблюдаем*

10: $$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

11: *endif*

12: $$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

13: $q = \delta^T \, \delta$

14: $$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

15: $$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0...0}_{3j-3} & 0 & 0 & 0 & \underbrace{0...0}_{3N-3j} \\ 0 & 1 & 0 & & 0 & 0 & 0 & \\ 0 & 0 & 1 & & 0 & 0 & 0 & \\ 0 & 0 & 0 & & 1 & 0 & 0 & \\ 0 & 0 & 0 & & 0 & 1 & 0 & \\ 0 & 0 & 0 & & 0 & 0 & 1 & \end{pmatrix}$$

16: $$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

17: $K_t^i = \bar{\Sigma}_t H_t^{iT} (H_t^i \bar{\Sigma}_t H_t^{iT} + Q_t)^{-1}$

18: $\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$

19: $\bar{\Sigma}_t = (I - K_t^i H_t^i) \bar{\Sigma}_t$

20: *endfor*

21: $\mu_t = \bar{\mu}_t$

22: $\Sigma_t = \bar{\Sigma}_t$

23: *return* μ_t, Σ_t

Таблица 10.1 Алгоритм EKF для проблемы SLAM (с известными соответствиями).

Очевидно, для любого разумного числа N, этот вектор существенно больше вектора положений, который был оценен в разделе 7.4, где был введен алгоритм локализации EKF. EKF SLAM вычисляет онлайновую апостериорную вероятность $p(y_t|z_{1:t}, u_{1:t})$.

Алгоритм EKF SLAM приводится в Таблице 10.1, и явно походит на алгоритм локализации EKF в Таблице 7.2. В строках со 2 до 5 используется обновление движения, а в строках с 6 до 20 учитывается вектор измерения.

В строках с 3 по 5 показатели математического ожидания и ковариации применяются к модели движения. Эта операция затрагивает только те элементы распределения прогноза, которые связаны с положением робота. Она оставляет неизменными все переменные математического ожидания и ковариации для карты, а также ковариации положение-карта. В строках с 7 по 20 выполняется проход по всем измерениям. Проверка в строке 9 возвращает истину только для ориентиров, для которых первичной оценки местоположения нет. Для таких ориентиров в строке 10 выполняется инициализация местоположения проекцией местоположения, полученного из соответствующих измерений расстояния и направления. Как мы обсудим ниже, этот шаг важен для линеаризации в EKF, но не нужен в линейных фильтрах Калмана. Для каждого измерения «ожидаемое» измерение вычисляется в строке 14, а соответствующее усиление Калмана вычисляются в строке 17. Заметим, что усиление Калмана является матрицей размера 3 на $3N + 3$. Эта матрица обычна не разрешена. Информация распространяется через всю оценку состояния. Затем обновление фильтра происходит в строках с 18 по 19, где степень новизны сворачивается обратно в оценку робота.

Тот факт, что усиление Калмана в полной мере распространяется на все переменные состояния, а не только на наблюдаемый ориентир и положение робота, очень важен. В SLAM наблюдение ориентира не просто улучшает оценку позиции этого самого ориентира, а также для других ориентиров. Этот эффект вызывается положением робота. Наблюдение ориентира улучшает оценку положения робота, в результате, устраняет некоторую неопределённость ориентиров, прежде наблюдаемых этим роботом. Положительной стороной здесь является отсутствие явного необходимости моделировать прошлые положения, которое бы привело в область полной проблемы SLAM и сделает EKF алгоритмом не в реальном времени. Вместо этого, зависимость отражается гауссовым апостериорным распределением, а именно, элементами вне главной диагонали матрицы ковариации Σ_t.

На Рис. 10.3 показан алгоритм EKF SLAM для искусственного примера. Робот выполняет навигацию из стартового положения, которое служит началом системы координат. По мере движения неопределённость возрастает, что показано эллипсами неопределённостями возрастающего диаметра. Робот также воспринимает близлежащие ориентиры и определяет для них неопределённость, которая сочетает фиксированную неопределённостью измерения и увеличивающуюся неопределенность положения. В результате, неопределенность определения местоположения ориентира со временем растет. Фактически, это происходит параллельно неопределённости положения в момент наблюдения ориентира. Интересный переход наблюдается на Рис. 10.3d. Здесь робот наблюдает ориентир, который уже наблюдался в самом начале картографирования и местоположение которого относительно хорошо известно.

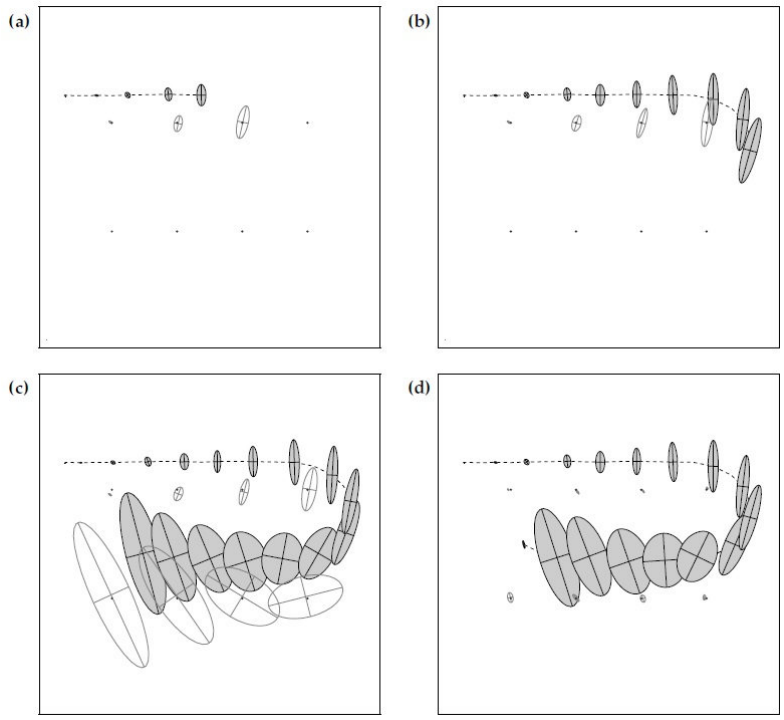


Рис. 10.3 EKF в приложении проблемы онлайн SLAM. Путь робота обозначен пунктирной точечной линией, а оценка позиций показана заштрихованными эллипсами. Восемь различимых между собой ориентиров с неизвестным местоположением показаны маленькими точками, а их оценки местоположения показаны белыми эллипсами. На схемах (a)–(c) неопределенность позиции робота возрастает, как и неопределенность ориентиров, с которыми он сталкивается. На схеме (d) робот снова обнаруживает первый ориентир, и неопределенность всех ориентиров уменьшается, как и неопределенность текущего положения. Изображение принадлежит Майклу Монтемерло из Стенфордского университета (Michael Montemerlo, Stanford University).)

С помощью этого наблюдения ошибка положения робота уменьшается, как показано на Рис. 10.3d. Обратите внимание на очень малый эллипс неопределенности для финального положения робота! Это наблюдение также уменьшает неопределённость для других ориентиров на карте. Это явление возникает из корреляции, выраженной в ковариационной матрице гауссовского апостериорного распределения. Поскольку большая часть неопределённости в ранее наблюдаемых ориентирах вызывается положением робота, и, поскольку эта самая неопределенность сохраняется со временем, оценка местоположения этих ориентиров коррелируются. При сборе информации о положении робота она распространяется на все уже обнаруженные ориентиры. Этот эффект является, возможно, самой важной характеристикой апостериорной вероятности SLAM. Информация, помогающая локализовать робота распространяется по карте и, в результате, улучшает локализацию других ориентиров на карте.

10.2.3 Математический вывод EKF SLAM

Вывод алгоритма EKF SLAM для случая с известным соответствием в значительной степени походит на вывод алгоритма локализации EKF в разделе 7.4. Ключевая разница состоит в дополненном векторе состояния, который сейчас включает все ориентиры вдобавок к положению робота.

В SLAM начальное положение считается началом системы координат. Это определение в чем-то произвольно, поэтому может быть заменено любой координатой. Местоположение ориентиров изначально неизвестно. Оценка гипотезы выражена следующими начальными математическим ожиданием и ковариацией:

$$\mu_0 = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \end{pmatrix}^T \tag{10.8}$$

$$F_{x,j} = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \infty & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \infty \end{pmatrix} \tag{10.9}$$

Ковариационная матрица имеет размер $(3N+3) \times (3N+3)$. Она состоит из маленькой нулевой матрицы 3×3 для переменных положения робота. Все остальные значения ковариации бесконечны.

При движении робота вектор состояния меняется согласно стандартной идеальной модели на основе скоростей (см. выражения (5.13) и (7.4)). В SLAM эта модель движения расширяется до дополненного вектора состояния:

$$y_t = y_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t + \gamma_t \Delta t \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{10.10}$$

Переменные x, y и θ определяют положение робота в y_{t-1}. Поскольку движение влияет только на положение робота и все ориентиры остаются там, где они находятся, только первые три элемента обновления не равны нулю. Это позволяет записать то же равенство более компактно:

$$y_t = y_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t + \gamma_t \Delta t \end{pmatrix} \tag{10.11}$$

Здесь F_x это матрица, проецирующая трехмерный вектор состояния в вектор размерности $3N + 3$.

$$F_x = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \ldots 0}_{3N \text{ columns}} \\ 0 & 1 & 0 & 0 \ldots 0 \\ 0 & 0 & 1 & 0 \ldots 0 \end{pmatrix} \tag{10.12}$$

Полная модель движения с учётом шумов следующая

$$y_t = \underbrace{y_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega \Delta t \end{pmatrix}}_{g(u_t, y_{t-1})} + \mathcal{N}(0, F_x^T \, R_t \, F_x) \tag{10.13}$$

где $F_x^T \, R_t \, F_x$ расширяет ковариационную матрицу до размерности полного вектора состояния в квадрате. Как обычно в EKF, функция движения g аппроксимируется, используя первую степень разложения в ряд Тейлора

$$g(u_t, y_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(y_{t-1} - \mu_{t-1}) \tag{10.14}$$

где якобиан $G_t = g'(u_t, \mu_{t-1})$ является производной g по отношению к y_{t-1} по u_t и μ_{t-1}, как в выражении (7.7).

Очевидно, аддитивная форма в (10.13) позволяет разложить этот якобиан в матрицу тождественного преобразования $(3N + 3) \times (3N + 3)$ (производная y_{t-1}) плюс якобиан низкой размерности g_t, характеризующий изменение положения робота:

$$G_t = I + F_x^T \, g_t \, F_x \tag{10.15}$$

с

$$g_t = \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} \tag{10.16}$$

Вставка этих приближений в стандартный алгоритм EKF дает строки с 2 по 5 Таблицы 10.1.Очевидно, некоторые матрицы, перемножаемые в строке 5, являются разреженными, что следует использовать при реализации алгоритма. Результат обновления даст математическое ожидание $\bar{\mu}_t$ и ковариацию $\bar{\Sigma}_t$ оценки в момент времени t после обновления фильтра управляющим сигналом u_t, но до учёта измерения z_t.

Вывод обновления измерения похож на аналогичный из раздела 7.4. В частности, дана следующая модель измерения

$$z_t^i = \underbrace{\begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \\ m_{j,s} \end{pmatrix}}_{h(y_t, j)} + \mathcal{N}(0, \underbrace{\begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}}_{Q_t}) \tag{10.17}$$

Здесь x, y и θ определяют положение робота, i –индекс отдельного наблюдения ориентира в z_t, а $j = c_t^i$ индекс наблюдаемого ориентира в момент времени t. Переменная r обозначает расстояние до ориентира, ϕ угол направления на ориентир, а s – сигнатуру ориентира. Элементы σ_r, σ_ϕ, и σ_s – это ковариации соответствующих шумов измерений.

Это выражение аппроксимируется линейной функцией

(10.18)
$$h(y_t, j) \approx h(\bar{\mu}_t, j) + H_t^i(y_t - \bar{\mu}_t)$$

Здесь H_t^i производная h по полному вектору состояний y_t. Поскольку h зависит от двух элементов этого вектора состояния – положения робота x_t и местоположения j-го ориентира m_j, переменная умножается на якобиан с низкой размерностью h_t^i и матрицу $F_{x,j}$, которая проектирует h_t^i в матрицу размерности полного вектора состояния:

(10.19)
$$H_t^i = h_t^i \; F_{x,j}$$

В этом выражении h_t^i – якобиан функции $h(y_t, j)$ по $\bar{\mu}_t$, вычисленный по отношению к переменным состояния x_t и m_j:

$$h_t^i = \begin{pmatrix} \frac{\bar{\mu}_{t,x} - \bar{\mu}_{j,x}}{\sqrt{q_t}} & \frac{\bar{\mu}_{t,y} - \bar{\mu}_{j,y}}{\sqrt{q_t}} & 0 & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q_t}} & \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q_t}} & 0 \\ \frac{\bar{\mu}_{j,y} - \bar{\mu}_{t,y}}{q_t} & \frac{\bar{\mu}_{t,x} - \bar{\mu}_{j,x}}{q_t} & -1 & \frac{\bar{\mu}_{t,y} - \bar{\mu}_{j,y}}{q_t} & \frac{\bar{\mu}_{j,x} - \bar{\mu}_{t,x}}{q_t} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{10.20}$$

Скаляр $q_t = (\bar{\mu}_{j,x} - \bar{\mu}_{t,x})^2 + (\bar{\mu}_{j,y} - \bar{\mu}_{t,y})^2$ для $j = c_t^i$, как и раньше, это ориентир, соответствующий измерению z_t^i. Матрица $F_{x,j}$ имеет размерность $6 \times (3N + 3)$. Она отображает матрицу с малым количеством измерений h_t^i в матрицу размерности $3 \times (3N + 3)$:

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \ldots 0 & 0 & 0 & 0 & 0 \ldots 0 \\ 0 & 1 & 0 & 0 \ldots 0 & 0 & 0 & 0 & 0 \ldots 0 \\ 0 & 0 & 1 & 0 \ldots 0 & 0 & 0 & 0 & 0 \ldots 0 \\ 0 & 0 & 0 & 0 \ldots 0 & 1 & 0 & 0 & 0 \ldots 0 \\ 0 & 0 & 0 & 0 \ldots 0 & 0 & 1 & 0 & 0 \ldots 0 \\ 0 & 0 & 0 & \underbrace{0 \ldots 0}_{3j-3} & 0 & 0 & 1 & \underbrace{0 \ldots 0}_{3N-3j} \end{pmatrix} \tag{10.21}$$

Эти выражения составляют суть вычислений усиления Калмана в строках с 8 по 17 в алгоритме EKF SLAM в Таблице 10.1 с одним важным дополнением. Когда ориентир обнаруживается в первый раз, его начальная оценка положения в выражении (10.8) ведёт к плохой линеаризации. Это происходит потому, что инициализация по умолчанию в (10.8), точка, по которой линеаризуется h, составляет $(\hat{\mu}_{j,x} \; \hat{\mu}_{j,y} \; \hat{\mu}_{j,s})^T = (0 \; 0 \; 0)^T$, которая является очень плохой функцией оценки местоположения ориентира. Лучшая функция оценки ориентира дана в строке 10 Таблицы 10.1. Здесь оценка ориентира инициализируется $(\hat{\mu}_{j,x} \; \hat{\mu}_{j,y} \; \hat{\mu}_{j,s})^T$ прогнозируемой позицией. Эта прогнозируемая позиция выводится из прогнозируемого положения робота и переменных измерения для ориентира

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix} \tag{10.22}$$

Заметим, что такая инициализация возможна только потому, что функция измерения h взаимно однозначна. Измерения двухмерны, так же, как местоположения ориентиров. В случае, когда измерение имеет меньшую

размерность, чем координаты ориентира, h является проекцией и вычислить осмысленное значение ожидания для $(\bar{\mu}_{j,x}\ \bar{\mu}_{j,y}\ \bar{\mu}_{j,s})^T$ из единственного измерения невозможно. Такое происходит, например, в случае реализации SLAM в компьютерном зрении, поскольку в камерах часто вычисляется только угол направления для ориентира, но не расстояние до него. Тогда в SLAM обычно выполняется интеграция нескольких наблюдений с последующей триангуляцией для определения подходящей начальной оценки местоположения. В литературе по SLAM такая проблема называется SLAM *только по направлению* и позже будет обсуждаться в одном из упражнений (страница 311).

Наконец, заметим, что алгоритм EKF требует объема памяти, квадратично зависящего от N, количества ориентиров на карте. Время обновления также квадратично к N. Квадратичная сложность обновления возникает из-за перемножения матриц, которое имеет место в различных местах при использовании EKF.

1: **Algorithm EKF_SLAM_known_correspondences**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t)$:

2: $N_t = N_{t-1}$

3: $F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 \end{pmatrix}$

4: $\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$

5: $G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$

6: $\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + F_x^T\, R_t\, F_x$

7: $Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$

8: *для всех наблюдаемых признаков* $z_t^i = (r_t^i\ \phi_t^i\ s_t^i)^T$ *выполнять*

9: $\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$

10: *for* $k = 1$ *to* $N_t + 1$ *do*

11: $\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$

12: $q_k = \delta_k^T\, \delta_k$

продолжение на следующей странице

начало на предыдущей странице

13: $\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \text{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$

14: $F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 & 0 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 1 & 0 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 1 & 0 & 0...0 \\ 0 & 0 & 0 & 0...0 & 0 & 0 & 1 & 0...0 \end{pmatrix}$

15: $H_t^k = \frac{1}{q_k} \begin{pmatrix} -\sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & \sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} & 0 \\ \delta_{k,y} & -\delta_{k,x} & -1 & -\delta_{k,y} & \delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$

16: $\Psi_k = H_t^k \, \bar{\Sigma}_t [H_t^k]^T + Q_t$

17: $\pi_k = (z_t^i - \hat{z}_t^k)^T \Psi_k^{-1} (z_t^i - \hat{z}_t^k)$

18: *endfor*

19: $\pi_{N_t+1} = \alpha$

20: $j(i) = \underset{k}{\text{argmin}} \, \pi_k$

21: $N_t = \max\{N_t, j(i)\}$

22: $K_t^i = \bar{\Sigma}_t [H_t^{j(i)}]^T \Psi_{j(i)}^{-1}$

23: $\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^{j(i)})$

24: $\bar{\Sigma}_t = (I - K_t^i H_t^{j(i)}) \bar{\Sigma}_t$

25: *endfor*

26: $\mu_t = \bar{\mu}_t$

27: $\Sigma_t = \bar{\Sigma}_t$

28: *return* μ_t, Σ_t

Таблица 10.2 Алгоритм EKF SLAM с соответствиями по максимальному правдоподобию, показан с вычёркиванием выбросов.

10.3 EKF SLAM с неизвестными соответствиями

10.3.1 Общий алгоритм EKF SLAM

СООТВЕТСТВИЕ МЕТОДОМ МАКСИМАЛЬНОГО ПРАВДОПОДОБИЯ

Расширим алгоритм EKF SLAM с известными соответствиями до общего алгоритма EKF SLAM использующего инкрементную функцию оценки максимального правдоподобия *(maximum likelihood - ML)* для определения соответствия. В Таблице 10.2 приводится алгоритм для неизвестного соответствия.

Поскольку соответствие неизвестно, на входе алгоритма **EKF_SLAM** отсутствует переменная соответствия c_t и вместо неё включается текущий размер карты N_{t-1}. Обновление движения в строках с 3 по 6 эквивалентен используемому в **EKF_SLAM_known_correspondences** в Таблице 10.1. Но цикл обновления измерения работает по-другому. Начиная со строки 8, сначала вычисляется гипотеза для нового ориентира с индексом N_t+1, на единицу больше, чем для ориентиров на карте в данный момент времени.

Местоположение нового ориентира инициализируется в строке 9 вычислением ожидаемого местоположения на основании местоположения робота, расстояния и угла направления в измерении. В строке 9 новому ориентиру назначается сигнатура посещённого. Далее вычисляются различные параметры обновления в строках с 10 по 18 для всех $N_t + 1$ возможных ориентиров, включая "новые". В строке 19 устанавливается порог для создания нового ориентира. Новый ориентир создаётся, если расстояние Махаланобиса до всех существующих ориентиров на карте превышает значение α. Затем соответствие по максимальному правдоподобию выделяется в строке 20. Ели измерение связано с прежде ненаблюдаемым ориентиром, счётчик ориентиров увеличивается на единицу в строке 21, и, соответствующим образом увеличиваются матрицы и векторы. Этот несколько громоздкий шаг в явном виде в Таблице 10.2 не приводится. Наконец, в строках 23 и 24 выполняется такт обновления EKF. Алгоритм **EKF_SLAM** возвращает новое количество ориентиров N_t, а также математическое ожидание μ_t и ковариацию Σ_t.

Вывод алгоритма EKF SLAM напрямую следует из предыдущих преобразований. В частности, инициализация в строке 9 идентична таковой в строке 10 в алгоритме **EKF_SLAM_known_correspondences** Таблицы 10.1. Строки с 10 по 18 соответствуют строкам с 12 по 17 в алгоритме **EKF_SLAM_known_correspondences** с добавочными переменными π_k необходимыми для вычисления соответствия по максимальному правдоподобию. Выбор соответствия ML в строке 20, и определение расстояния Махаланобиса в строке 17, аналогичен соответствию ML, описанному в разделе 7.5. В частности, в алгоритме **EKF_localization** в Таблице 7.3 на странице 202 было использовано аналогичное выражение для определения наиболее вероятного ориентира (строка 16). Обновление измерения в строках 23 и 24 Таблицы 10.2 также аналогично таковому для алгоритма EKF с известным соответствием, подразумевая, что задействованные векторы и матрицы имеют необходимый размер, в случае, если карта только что была расширена.

Наш пример реализации **EKF_SLAM** может быть сделан более эффективным ограничением учитываемых ориентиров в строках с 10 по 18 только находящимися рядом с роботом. Более того, многие из значений и матриц, вычисляемых во внутреннем цикле, могут быть кэшированы при проходе через более чем один признак вектора измерений z_t^i. На практике, грамотное управление признаками на карте и плотная оптимизация в цикле могут сильно уменьшить время исполнения.

10.3.2 Примеры

На Рис. 10.4 показан алгоритм EKF SLAM с известным соответствием, используемый в имитационном эксперименте. На левой части каждого из трёх графиков показаны апостериорные распределения, маргинализированные для отдельных ориентиров и положения робота. С правой стороны показана матрица корреляции для дополненного вектора состояний y_t, корреляция является нормализованной. Как легко увидеть в результатах, приведённых на Рис. 10.4c, со течением времени все оценки x и y-координат становятся полностью коррелированными. Это означает, что карта в относительных координатах определяется вплоть до неточности глобального местоположения, которая не может быть устранена. Эта особенность является важной характеристикой проблемы SLAM, поскольку абсолютные координаты

карты могут быть лишь приблизительно определены относительно координатной системы, заданной начальным положением робота, в то время, как относительные координаты можно определить асимптотически точно.

На практике EKF SLAM был успешно применён к большому числу проблем навигации, включая воздушные, подводные роботы, роботы для помещений и различные другие. На Рис. 10.5 показан пример результата, полученного с помощью подводного робота «Оберон» (Oberon), разработанного в университете Сиднея, Австралия и показанного на Рис. 10.6. Робот оборудован узкополосным сонаром, способным сканировать пространство с очень высоким разрешением и обнаруживать препятствия на расстоянии до 50 метров. Для решения проблемы картографирования исследователи разместили в воде длинные тонкие вертикальные объекты, которые относительно легко можно обнаружить в показаниях сонара. В этом конкретном эксперименте такие объекты были размещены в ряд с промежутком, примерно, 10 метров. Вдобавок, дополнительные точечные признаки находятся на более дальней гряде, которую можно обнаружить с помощью узкополосного сонара.

В эксперименте, показанном на Рис. 10.5, робот двигается по этим меткам, затем разворачивается и двигается назад. В процессе робот получает измерения и интегрирует их в карту, используя алгоритм EKF SLAM, описанный в этой главе.

Карта, показанная на Рис. 10.5 показывает путь робота, отмеченный треугольниками, соединёнными отрезками. Возле каждого треугольника показан эллипс, соответствующий матрице ковариации оценки фильтра Калмана, спроектированную по координатам x-y робота. Эллипсом показана дисперсия: чем больше его размер, тем менее уверен робот в своём местоположении. Различными точками на Рис. 10.5 показаны наблюдаемые ориентиры, полученные поиском с помощью сонара небольших, сильно отражающих объектов. Большинство таких показаний было отброшено, используя механизм, описанный в следующем разделе. Однако, некоторые связывались с ориентирами и помещались на карту. В конце прохода робот обнаружил 14 таких объектов, определённых как ориентиры, каждый из которых с проектированным эллипсом неопределённости на Рис. 10.5. Эти ориентиры карты включают искусственные маркеры, установленные исследователями, а также различные объекты окружающей среды поблизости от робота. Остаточная неопределённость положения достаточно мала.

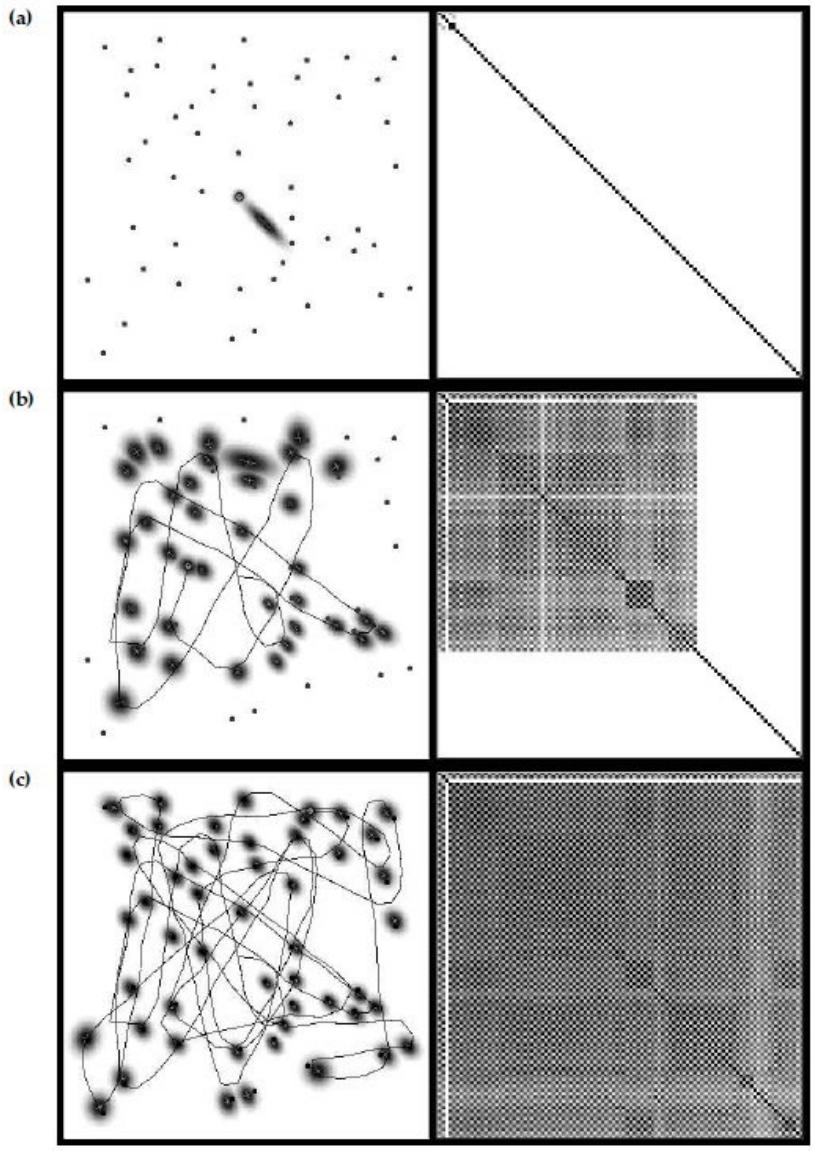


Рис. 10.4 EKF SLAM с известной ассоциацией данных в имитационной среде. Карта показана слева, градации серого соответствуют неопределённости каждого ориентира. Матрица справа – это матрица корреляции, нормализованная ковариационная матрица апостериорной оценки. После некоторого времени все оценки x и y-координат становятся полностью коррелированны.

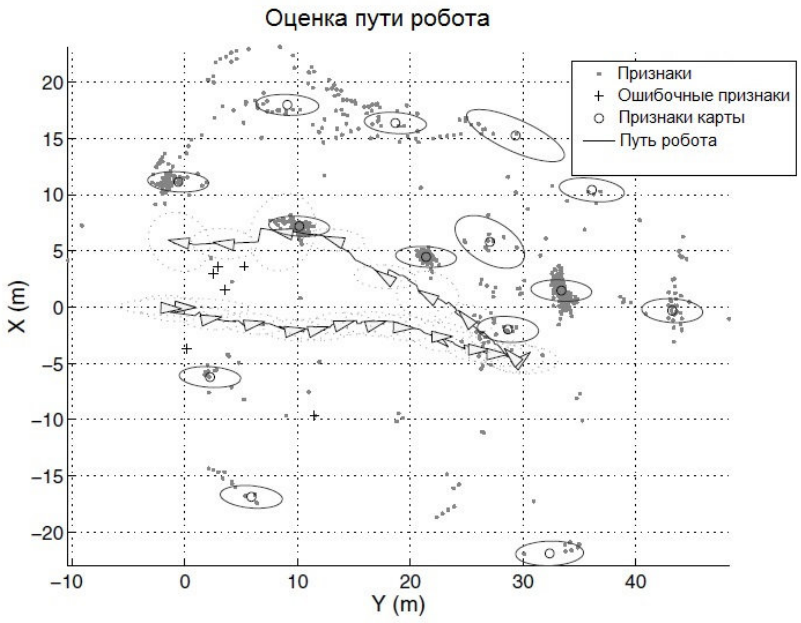


Рис. 10.5 Пример оценки карты и положения робота фильтром Калмана. Изображение принадлежит Стефану Виллиамсу и Хью Дюрран-Уайту, Австралийский центр полевой робототехники (Stefan Williams and Hugh Durrant-Whyte, Australian Centre for Field Robotics).

Рис. 10.6 Подводный аппарат «Оберон», разработанный в университете Сиднея. Изображение принадлежит Стефану Виллиамсу и Хью Дюрран-Уайту, Австралийский центр полевой робототехники (Stefan Williams and Hugh Durrant-Whyte, Australian Centre for Field Robotics)

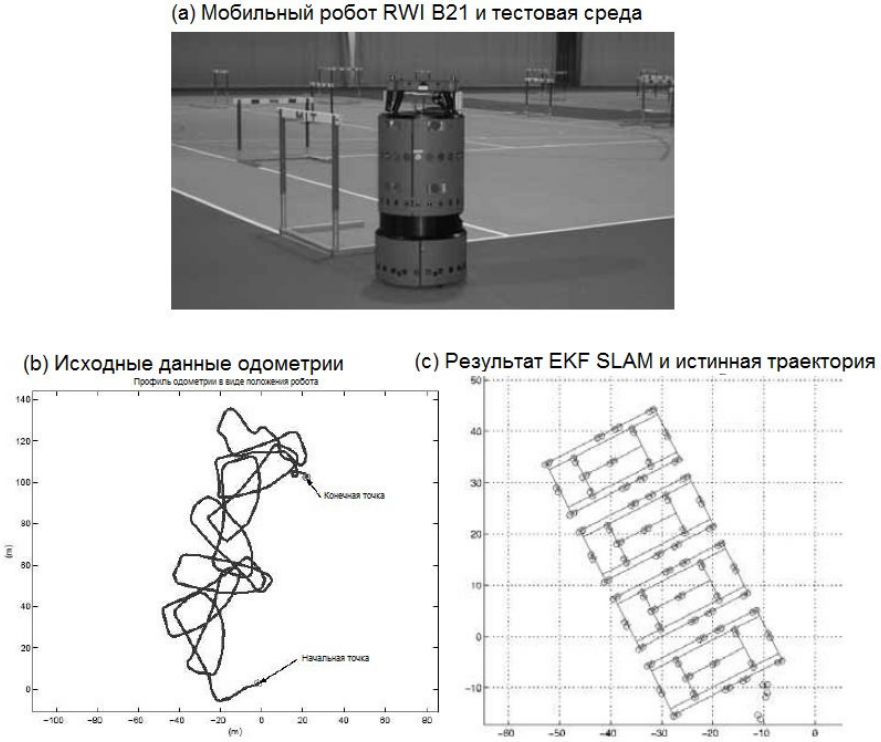


Рис. 10.7 (a) Мобильный робот MIT B21 на размеченной тестовой площадке. (b) Исходная одометрия робота после прохода площадки в ручном режиме. (c) Результатом работы EKF SLAM стала очень точная карта. Показана сгенерированная карта, наложенная поверх сконструированной вручную. Все изображения и результаты принадлежат Джону Леонарду и Мэтью Уолтеру, МИТ (John Leonard and Matthew Walter, MIT).

На Рис. 10.7 показан результат другой реализации EKF SLAM. На врезке (a) показан мобильный робот RWI B21, разработанный в MIT и помещённый в тестовую среду. Тестовой средой служит теннисный корт, переносные барьеры означают препятствия, их положение было вручную определено с точностью порядка сантиметров для сравнения. На врезке (b) Рис. 10.7 показан пройденный путь по исходным данным одометрии. Результат работы EKF SLAM показан на врезке (c), наложенный на карту, сконструированную вручную. Читатель может убедиться, что созданная карта очень точна.

10.3.3 Выбор признаков и управление картой

Обеспечение надёжности EKF SLAM на практике часто требует дополнительных методов управления картой. Многие из них имеют отношение к тому, что допущение о гауссовом характере шума нереалистично, а многие сомнительные измерения принадлежат к большей стороне распределения шумов. Такие сомнительные измерения могут привести к созданию ошибочных ориентиров на карте, которые, в свою очередь, будут отрицательно влиять на локализацию робота.

Многие современные методы имеют механизмы обработки *выбросов* в

ВЫБРОСЫ

пространстве измерений. Такие выбросы определяются как сомнительные наблюдения ориентиров вне пределов диапазона неопределенности любых ориентиров на карте. Самым простым способом удалить такие выбросы является поддержка *списка временных ориентиров*. Вместо дополнения карты новым ориентиром сразу поле того, как измерение покажет его обнаружение, такой ориентир сначала добавляется в список временных ориентиров. Этот список выглядит в точности как как карта, но не используется для уточнения положения робота (соответствующие градиенты в уравнениях измерения установлены в нуль). Если ориентир наблюдается постоянно и размер эллипса неопределенности сильно уменьшается, ориентир переносится на рабочую карту.

СПИСОК ВРЕМЕННЫХ ОРИЕНТИРОВ

В практических реализациях этот механизм способен существенно уменьшить количество ориентиров на карте, при этом сохранив все физические ориентиры, имеющие высокую вероятность. Дальнейшим шагом, который также обычно используется в современных методах, является поддержка *вероятности существования ориентира.* Такая апостериорная вероятность может быть реализована в виде логарифма шансов и обозначаться o_j для j-го ориентира на карте. Когда наблюдается j-ый ориентир m_j, o_j увеличивается на фиксированное значение. Отсутствие наблюдения ориентира m_j, когда он должен быть в радиусе действия датчиков робота ведет к уменьшению o_j. Поскольку никогда нет полной уверенности, находится ли ориентир в радиусе действия датчиков робота, уменьшение можно выполнить в виде коэффициента вероятности такого события. Ориентиры удаляются с карты, когда значение o_j падает ниже установленного порога. Такие методы обычно позволяют создать карты значительно меньшего размера для негауссового шума измерения.

ВЕРОЯТНОСТЬ СУЩЕСТВОВАНИЯ ОРИЕНТИРА

Инициализация оценки для нового ориентира начиная с ковариации с большими элементами, как предлагается в выражении (10.9) может вызвать неустойчивость численного решения. Это происходит потому, что самый первый шаг обновления ковариации будет менять это значение на несколько порядков, возможно слишком много для создания положительно определённой матрицы. Лучшей стратегией будет явное указание шага инициализации для любого признака, который ранее не наблюдался. В частности, такой шаг может инициализировать ковариацию Σ_t напрямую текущей неопределённостью ориентира, вместо выполнения строки 24 в Таблице 10.2 (то же, что и среднее значение в строке 23).

ВЫЧИСЛИТЕЛЬНАЯ НЕУСТОЙЧИВОСТЬ EKF SLAM

Как отмечалось ранее, метод максимального правдоподобия для ассоциации данных имеет одно явное ограничение. Дело в том, что метод максимального правдоподобия отклоняется от идеи полной апостериорной оценки в вероятностной робототехнике. Вместо поддержки общего апостериорного распределения по дополненным векторам и ассоциации данных, он сводит проблему ассоциации данных к детерминированному определению, как если бы ассоциация по методу максимального правдоподобия была бы всегда верна. Это ограничение делает EKF чувствительным к ошибочной идентификации ориентиров, что может привести к неверным результатам. На практике исследователи часто решают эту проблему выбором одного из двух описанных ниже методов, уменьшающих шансы перепутать ориентиры:

- **Распределение в пространстве.** Чем дальше друг от друга расположены ориентиры, тем меньше шанс их спутать. Поэтому общепринято выбирать ориентиры, которые достаточно далеко друг от друга чтобы вероятность их перепутать была мала. Это приводит к интересному компро-

миссу: большое количество ориентиров увеличивает шанс их перепутать. Слишком малое количество ориентиров затрудняет локализацию робота, что также затрудняет локализацию. Об оптимальной плотности ориентиров пока известно мало, и исследователи часто полагаются на интуицию при выборе отдельных значений.

• **Сигнатуры.** При выборе соответствующих ориентиров важно максимизировать их различимость при восприятии. Например, двери бывают разного цвета, а коридоры – могут быть различной ширины. Результирующие сигнатуры важны для работы SLAM.

С такими дополнениями алгоритм EKF SLAM успешно применялся к большому количеству разных практических проблем картографирования, включая воздушные, наземные и подводные дроны.

Ключевое ограничение EKF SLAM лежит в необходимости выбора соответствующих ориентиров. При ограничении потока данных с датчиков наличием и отсутствием ориентиров, большое количество данных датчика отбрасывается. Это ведет к потере данных относительно алгоритма SLAM, который может использовать датчики без интенсивной предварительной фильтрации. Квадратичная зависимость времени обновления EKF ограничивает алгоритм относительно разреженными картами с менее чем 1000 признаков. На практике же часть необходимо стремиться к картам с 10^6 признаков или более, в этом случае EKF неприменим.

Относительно малая размерность карты может усложнять проблему ассоциации данных. Это легко проверить: если открыть глаза и окинуть взглядом всю комнату, скорее всего, проблемы с ориентированием не возникнет! Однако, если иметь информацию только о небольшом количестве ориентиров, например, местоположении всех источников света, решение будет значительно более трудным. В результате, ассоциация данных в EKF SLAM более сложна по сравнению с некоторыми другими алгоритмами SLAM, обсуждаемыми в последующих главах, и способных обрабатывать на порядки больше признаков.

ОСНОВНАЯ ДИЛЕММА EKF SLAM

Это нашло отражение в *основной дилемме EKF SLAM*: хотя инкрементная ассоциация данных по методу максимального правдоподобия может хорошо работать с плотными картами с сотнями миллионов признаков, она неустойчива на разреженных картах. Напротив, EKF требует разреженных карт в силу квадратичной сложности обновления. В последующих главах будут обсуждаться алгоритмы SLAM которые более эффективны и способны обрабатывать большие карты. Также будут обсуждаться более надежные методы ассоциации данных. В силу большого количества ограничений представленный в этой главе алгоритм EKF SLAM представляет, по большей части, исторический интерес.

10.4 Вывод

В данной главе описана общая проблема SLAM и представлен метод EKF.

• Проблема SLAM определена как одновременная локализация и построение карты. Робот предпринимает попытки построить карту окружающей среды, с одновременным поиском своего местоположения на этой карте.

• Проблема SLAM имеет две версии: онлайновую и глобальную. Обе

проблемы включают оценку карты. В проблеме онлайн выполняется поиск текущего местоположения, а в глобальном – определить по всем положениям. Обе проблемы одинаково важны на практике, и одинаково хорошо освещены в литературе.

• Алгоритм EKF SLAM возможно самый легкий из SLAM. Она была применена к проблеме онлайн SLAM. Для известного соответствия результирующий алгоритм инкрементный. Обновление требует времени, квадратичного к числу ориентиров на карте.

• Когда соответствия неизвестны, алгоритм EKF SLAM использует инкрементальную функцию оценки максимального правдоподобия для проблемы соответствия. Результирующий алгоритм хорошо работает для ясно различимых ориентиров.

• Дополнительные методы были обсуждены для управления картами. Две общепринятые стратегии идентификации выбросов включают временный список ориентиров, которые не наблюдаются достаточно часто, и счетчик наблюдения ориентира, вычисляющий апостериорную оценку существования ориентира.

• EKF SLAM применялся со значительным успехом для разнообразных проблем картографии в робототехнике. Основными недостатками являются необходимость удовлетворительного различения ориентиров и вычислительная сложность обновления фильтра.

На практике EKF SLAM использовался с некоторым успехом. Когда ориентиры достаточно различимы, апостериорная оценка вычисляется достаточно хорошо. Преимуществом полной апостериорной оценки является её полнота: она учитывает полную неопределённость и позволяет роботу оценивать воздействие управления, принимая во внимание истинное значение неопределённости. Однако, алгоритм EKF SLAM страдает от чрезмерно высокой вычислительной сложности и ограниченности разреженными картами. Это, в свою очередь, затрудняет проблему ассоциации данных, а EKF SLAM склонен к плохой работе в ситуациях, когда ориентиры неразличимы. Ещё большую неустойчивость вызывает факт того, что алгоритм EKF SLAM основан на методе инкрементной ассоциации данных методом максимального правдоподобия. Этот метод делает невозможным пересмотр прошлой ассоциации данных, и может вызвать сбой при неверной ассоциации данных методом ML.

Алгоритм EKF SLAM применяется не только для проблемы онлайн SLAM но и для полной проблемы SLAM. В полной проблеме SLAM добавление нового положения к вектору состояний на каждом такте времени вызывает бесконтрольный рост как вектора состояний, так и ковариации. Обновление ковариации потребует все возрастающего времени и метод быстро выходит их диапазона вычисления времени, вне зависимости от скорости процессора.

10.5 Библиографические примечания

Проблема SLAM появилась за много столетий до современной робототехники. Проблема моделирования физической структуры с движущейся плат-

формы датчика лежит в основе целого ряда направлений, таких, как науки о Земле, фотограмметрия, и компьютерное зрение. Многие из математических методов, лежащие в основе сегодняшнего SLAM, изначально были разработаны для вычисления планетных орбит. Например, метод наименьших квадратов можно проследить до работ Иоганна Карла Фридриха Гаусса (Johann Carl Friedrich Gauss, 1809). SLAM – это, в основном, проблема географического исследования. Применительно к роботу, это создает проблемы, с которыми исследователи-люди сталкиваются редко, такие, как проблема соответствия и проблема нахождения соответствующих признаков.

В робототехнике проблема EKF в SLAM была впервые представлена в серии основополагающих работ Чизмана и Смита (Cheeseman and Smith, 1986), Смита и Чизмана (Smith and Cheeseman, 1986), Смита (Smith et al., 1990). В этих работах впервые был описан метод EKF, обсуждаемый в данной главе. Также, как и в этой книге, Смит обсуждал EKF в контексте картографирования на основе признаков с точечными ориентирами и известной ассоциацией данных. Первые реализации EKF SLAM были выполнены Маутарлиром и Чатила (Moutarlier and Chatila, 1989a,b) и Леонардом и Дюран-Уайтом (Leonard and Durrant-Whyte, 1991) и в некоторых из них в качестве ориентиров использовались искусственные метки. EKF стал модным, когда многими авторами исследовались альтернативные методы поддержания точной оценки положения при картографировании (Cox 1991). Ранняя работа Дикманса и Грэфи (Dickmanns and Graefe, 1988) по оценке кривизны дороги в автономных автомашинах имеет самое прямое отношение к проблеме, детальная информация приводится в (Dickmanns 2002) для ознакомления.

SLAM очень активно исследуется, что показывают последние работы (Leonard et al. 2002b). Большой объем литература по SLAM (или CML, что значит "concurrent mapping and localization" – конкурентное картографирование и локализация, как его назвали Леонард и Дюран-Уайт (Leonard and Durrant-Whyte,1991)), можно найти в работах Труна (Thrun, 2002). Важность поддержания корреляций на карте была отмечена Цорба (Csorba, 1997), который в кандидатской диссертации также установил некоторые основные результаты конвергенции. После этого усилиями многих авторов основная парадигма была расширена самыми различными методами. Методы управления признаками, описанные в данной главе, были впервые описаны Диссанаяки (Dissanayake et al., 2001, 2002), а также Бейли (Bailey, 2002). Вильямс (Williams et al., 2001) разработал идею временных списков признаков в SLAM для уменьшения эффекта ошибок обнаружения признаков. Инициализация признаков обсуждалась в работе Леонарда (Leonard et al., 2002a), который в явном виде сохранял оценку предыдущих положений для использования датчиков, которые предоставляли неполные данные координат признаков. Представление, которое предотвращало появление специфичности явной факторизацией «отклонений» от апостериорного распределения, было выведено Кастелланосом (Castellanos et al., 1999), показавшего улучшение числовой стабильности базового EKF. Янсфельт (Jensfelt et al., 2002) обнаружил существенные улучшения для SLAM внутри помещений при использовании базовых геометрических ограничений, таких, как факт параллельности и перпендикулярности большинства стен. Ранние работы по SLAM с использованием сонаров восходят к Ренкену (Rencken (1993). Современная система SLAM с ультразвуковыми датчиками была описана Тардос (Tardós et al., 2002). Кастелланос (Castellanos et al., 2004) выполнил критически анализ целостности EKF. Эмпирическое сравнение нескольких алгоритмов можно найти в работах Ваганай (Vaganay et al., 2004). Некото-

рые открытые вопросы обсуждались Салихом и Морено (Salichs and Moreno, 2000). Исследование важной проблемы ассоциации данных будет описано в следующей главе (см. стр. 335).

Как было отмечено, ключевым ограничением решения EKF в проблеме SLAM является квадратичность матрицы ковариации, что не осталось незамеченным. В последние годы множество исследователей предлагали алгоритмы EKF SLAM, которые стали заметно более масштабируемыми используя разбиение карты на вложенные карты с раздельным сохранением ковариаций. Некоторые из оригинальных работ в этой области принадлежат Леонарду и Фидеру (Leonard and Feder, 1999), Гюванту и Неботу (Guivant and Nebot, 2001), а также Вильямсу (Williams, 2001). Леонард и Федер (Leonard and Feder, 1999) ввели алгоритм несвязанного стохастического картографирования с разбиением карты на несколько отдельных, более управляемых карт меньшего размера. Этот подход вычислительно эффективен, но не обеспечивает механизма распространения информациичерез сеть локальных карт (Leonard and Feder 2001). Гювант и Небот (Guivant and Nebot, 2001, 2002), наоборот, выполнили приблизительную факторизацию матрицы ковариации, что позволило существенно уменьшить сложность обновления EKF. Вильямс (Williams (2001, and Williams et al., 2002) предложил ограниченный локальный фильтр вспомогательных карт (constrained local submap filter - CLSF), основанный на создании независимых локальных вспомогательных карт признаков в непосредственной близости от транспортного средства. Вильямс (Williams et al., 2002) привел результаты картографирования для подводной среды (на Рис. 10.5 приводится одна из его ранних работ). Методы последовательного объединения карт (sequential map joining techniques) описанные Тардосом (Tardós et al., 2002) являются относительной декомпозицией. Бейли (Bailey, 2002) вывел похожий метод, представив карты SLAM в иерархическом виде. Фолкессон и Кристенсен (Folkesson and Christensen, 2003) описали метод, в котором частые обновления затрагивали только область вблизи робота, а оставшаяся часть карты обновлялась значительно реже. Все эти методы достигли той же скорости сходимости, как полное решение EKF, за исключением затруднений в виде вычисления операций сложностиO(n2). Тем не менее, они значительно лучше масштабируются для больших проблем с десятками тысяч признаков.

Некоторые исследователи разработали гибридные методы SLAM, комбинирующие подходы EKF SLAM с пространственными, такими как карты сеток занятости. Гибридная метрическая карта (hybrid metric map -HYMM), предложенная Гювантом (Guivant et al., 2004) и Нието (Nieto et al., 2004) разбивает карты на треугольные области (LTR) используя пространственные карты, такие, как карты сеток занятости, в качестве основного отображения этих областей. Эти локальные карты комбинируются, используя EKF. Бургард (Burgard et al., 1999b) также выполнял разбиение карт на локальные карты сеток занятости но использовал алгоритм максимизации ожидания (expectation maximization- EM) (см Dempster et al., 1977) для комбинированя локальных карт в общую глобальную. В работе Бетги-Брезетца (Betgé-Brezetz et al., 1995, 1996) два типа отображений были интегрированы в парадигму SLAM: растровые карты для представления сред в помещениях, и объектное представление для разреженных объектов вне помещений.

Обобщение SLAM для динамических сред можно найти в работах Ванга (Wang et al., 2003), Хунела (Hähnel et al., 2003c), и Вольфа и Сухатми (Wolf and Sukhatme, 2004). Ванг (Wang et al., 2003) разработал алгоритм под названием SLAM с DATMO, сокращение для «SLAM with the detection and

tracking of moving objects» (SLAM с обнаружением и отслеживанием движущихся объектов). Их подход основывался на EKF, но позволял перемещение признаков. Хёнел (Hähnel et al., 2003c) изучил проблему применения SLAM в средах с множеством движущихся объектов. Они успешно применили алгоритм EM для фильтрации измерений, которые, скорее всего, следовало бы отнести к движущимся объектам. Таким образом, им удалось получить карты для сред, где обычные методы SLAM отказывали. В методе Вольфа и Сухатми (Wolf and Sukhatme, 2004) сохранялись две связанные сетки занятости окружающей среды, одна для статической карты, а вторая – для движущихся объектов. Локализация наподобие SLAM алгоритмом с регулярными ориентирами.

SLAM системы привели к появлению многих практических реализаций. Рикоски (Rikoski et al.,2004) применил SLAM к одометрии подводной лодки с помощью сонара, представив новый метод «звуковой одометрии». Использование SLAM в заброшенных шахтах было описано Некером (Nüchter et al., 2004), который обобщил парадигму до полной оценки положения по шести измерениям. Обобщения проблемы SLAM для нескольких роботов также были предложены рядом исследователей. Некоторые из ранних работ принадлежат Неттлетону (Nettleton et al., 2000), разработавшему метод, в кот роботы сохраняли локальные карты EKF, но сливали их с использованием информационного выражения апостериорного распределения. Альтернативный метод принадлежит Реклеитису (Rekleitis et al., 2001a), который использовал группу неподвижных и движущихся роботов для уменьшения ошибки локализации при выполнении SLAM. Фенвик (Fenwick et al., 2002) представл полное теоретическое исследование слияния карт нескольких роботов, в частности для SLAM на основе ориентиров. Методы слияния проходов сканирования были разработаны Конолигом (Konolige et al., 1999) и Труном (Thrun et al. (2000b, 2001). Ряд исследователей предложили SLAM системы для определенных наборов датчиков. Важным датчиком является видеокамера, но она предоставляет только угол направления на признаки. Эта литература хорошо изучена в литературе по компьютерному зрению под названием "структура из движения" (structure from motion- SFM) (Tomasi and Kanade 1992; Soatto and Brockett 1998; Dellaert et al. 2003), и в области фотограмметрии (Konecny 2002). В SLAM фундаментальная работа по SLAM на основе только лишь угла направления принадлежит Динсу и Герберту (Deans and Hebert, 2000, 2002). В их методу рекурсивно оцениваются признаки среды, инвариантные для положения робота в целях отделения ошибки положения от ошибки карты. Большое количество исследователей применяли SLAM с камерами в качестве основного датчика (Neira et al. 1997; Cid et al. 2002; Davison 2003). Дэвисон (Davison, 1998) предложил метод активного зрения в контексте SLAM. Работа Дудека и Ягерссура (Dudek and Jegessur, 2000) основана на распознавании места по внешнему виду, а Хайет (Hayet et al., 2002) и Бугет и Перона (Bouguet and Perona, 1995) использовали визуальные ориентиры. Дибел (Diebel et al., 2004) разработал фильтр для SLAM с активным стереодатчиком, учитывающим нелинейное распределение шумов стерео датчика расстояния. Методы слияния показаний датчиков в SLAM были разработаны Деви и Булата (Devy and Bulata, 1996). Кастелланос (Castellanos et al., 2001) эмпирически обнаружил, что слияние сигналов лазера и камеры дает лучшие результаты по сравнению с их раздельным использованием.

СТРУКТУРА ИЗ ДВИЖЕНИЯ

SLAM также было обобщено для проблемы построения плотных трехмерных моделей. Ранние системы получения трехмерных моделей для мобильных роботов, действующих внутри помещений, можно найти в работах

Рида и Аллена (Reed and Allen, 1997), Иоччи (Iocchi et al., 2000), Трун а (Thrun et al., 2004b). Деви и Парра (Devy and Parra, 1998) получали трехмерные модели, используя параметрические кривые. Жао и Шибасаки (Zhao and Shibasaki, 2001), Теллер (Teller et al., 2001) и Фрах и Захор (Frueh and Zakhor, 2003) разработали впечатляющие системы для построения больших текстурированных трехмерных карт городских сред. Ни одна из этих систем не применялась для проблемы SLAM вне помещений в силу наличия GPS, но они плотно соотносятся с математической основой SLAM. Эти методы хорошо сочетаются с множеством работ по реконструкции городских сред по аэрофотосъемкам (Jung and Lacroix 2003; Thrun et al. 2003).

В последующих главах описаны альтернативы простым EKF. Методы, описанные в них, разделяют многие уже упоминавшиеся догадки относительно обобщений, в результате чего границы между разными типами фильтров становится почти невозможно провести. Литературный обзор будет продолжен в конце следующей главы, при обсуждении алгоритма SLAM, использующего выражение из теории информации.

10.6 Упражнения

1. Какова вычислительная сложность обновления движения в EKF SLAM? Использовать нотацию $O(\ )$. Сравнить с наихудшей сложностью EKF по вектору признаков аналогичного размера.

SLAM ТОЛЬКО ПО НАПРАВЛЕНИЮ

2. SLAM только по направлению относится к проблеме SLAM, когда датчики способны определить только направление на ориентир, но не расстояние до него. Как было отмечено, SLAM только по направлению тесно связан со структурой из движения (SFM) в компьютерном зрении. Одна проблема SLAM только по направлению с EKF касается инициализации оценки местоположения ориентира, даже если соответствия известны. Обсудить, почему это происходит и вывести метод инициализации оценки местоположения ориентира (математическое ожидание и ковариацию), применимый для SLAM только по направлению.

3. На странице 305 было отмечено, что алгоритм EKF в Таблице 10.2 может стать вычислительно нестабильным. Вывести метод явной установки μ_t и Σ_t когда новый признак наблюдается впервые. Такой метод может не потребовать инициализации с очень большим значением ковариации. Показать, что результат математически эквивалентен строкам 23 и 24, где ковариация инициализирована выражением (10.9).

4. В тексте предлагается использовать бинарный байесовский фильтр для вычисления вероятности, что ориентир, выраженный в апостериорном распределении, действительно существует в физическом мире.

(a) В первой части упражнения требуется разработать такой бинарный байесовский фильтр.

(b) Теперь необходимо обобщить фильтр для ситуации, когда ориентиры спорадически исчезают с вероятностью p^*.

(c) Представим ситуацию, когда для точно определённого ориентира долгое время не поступало информации о его существовании (ни положи-

тельной, ни отрицательной). До какого значения сойдется фильтр? Доказать ответ.

5. Представленный в данной главе алгоритм EKF SLAM неспособен решить проблему ассоциации данных в статистически непротиворечивой манере. Предложить алгоритм (и статистический аппарат) для апостериорной оценки с неизвестной ассоциацией данных, представляющей апостериорное распределение в виде смеси гауссиан. Охарактеризовать его преимущества и недостатки. Как сложность апостериорного распределения растет со временем?

6. Основываясь на предыдущей проблеме, разработать приближенный метод апостериорной оценки с неизвестной ассоциацией данных, в котором время, необходимое для каждого инкрементного шага обновления, не увеличивается со временем (допуская фиксированное количество ориентиров).

7. Разработать алгоритм фильтра Калмана, использующего в качестве основных компонентов локальные карты сеток занятости, а не ориентиры. Среди прочего, необходимо решить, как локальные сетки связаны друг с другом и что делать с все возрастающим числом локальных сеток.

11 Алгоритм GraphSLAM

11.1 Введение

Алгоритм EKF SLAM, описанный в предыдущих главах, имеет ряд ограничений. Одним из них является квадратичная сложность обновления, а вторым – метод линеаризации в EKF, который выполняется только раз для каждого нелинейного члена. В этой главе будет представлен альтернативный алгоритм SLAM под названием GraphSLAM. В отличие от EKF, GraphSLAM решает полную задачу SLAM и вычисляет оффлайновое решение для задачи, определённой для всех положений и всех признаков на карте.

РАЗРЕЖЕНЫЙ ГРАФ

Как будет показано в этой главе, апостериорное распределение для *полной задачи SLAM* естественным образом формирует *разреженный граф*. Этот граф сводится к сумме нелинейных квадратичных ограничений, оптимизация которых даст карту максимального правдоподобия и соответствующий набор положений робота. Исторически, эта идея присутствовала в большом числе публикаций по SLAM. Название “GraphSLAM” было отражает основную идею метода.

На Рис. 11.1 показан алгоритм GraphSLAM. Здесь приведён граф, который с помощью GraphSLAM извлекается из пяти положений, отмеченных $x_0, ..., x_4$, и двух признаков карты m_1, m_2. На графике показаны дуги для движения и измерения. Дуги движения связывают два последующих положения робота, а дуги измерения связывают положения с признаками, наблюдаемыми из данных положений. Каждое ребро графа соответствует нелинейному ограничению. Как будет показано далее, эти ограничения выражают отрицательный логарифм правдоподобия моделей измерения и движения, поэтому лучше всего воспринимать их как информационные ограничения. Добавление такого ограничения к графу в SLAM должно быть тривиально и не требовать существенных вычислений. Сумма всех огра-

НАИМЕНЬШИЕ КВАДРАТЫ

ничений даёт нелинейную задачу *наименьших квадратов*, как показано на Рис. 11.1.

Для вычисления апостериорного распределения карты в GraphSLAM выполняется линеаризация набора ограничений. Результатом линеаризации становится информационная матрица и информационный вектор практически того же вида, с которым мы уже сталкивались в Главе 3 при обсуждении информационного фильтра. Однако, информационная матрица наследует свойство разрежённости графа, сконструированного в GraphSLAM. Эта разрежённость позволяет использовать в GraphSLAM алгоритм удаления переменных, и получить граф значительно меньшего размера, определённый только по положению робота. Апостериорная карта пути вычисляется стандартными методами. GraphSLAM также вычисляет карту и определённые маргинальные апостериорные распределения на ней. Полное апостериорное распределение по карте квадратично зависит от размера карты, и восстанавливать его нецелесообразно.

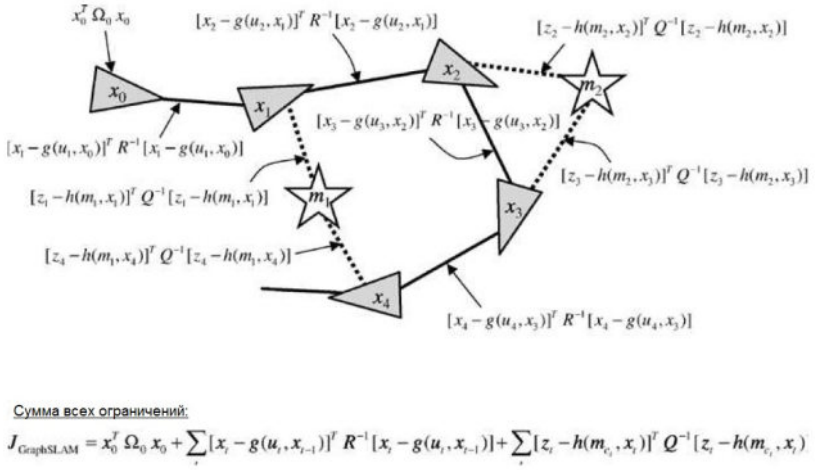


Рис. 11.1 Иллюстрация GraphSLAM для четырёх положений робота и двух признаков карты. Узлами графа обозначаются положения робота и местоположения признаков. На графе имеется два типа рёбер: сплошными рёбрами обозначены последовательные переходы положений, а пунктирные соединяют положения с признаками, которые робот воспринимает из указанного положения. Каждая связь в GraphSLAM представляет собой нелинейное квадратичное ограничение. Ограничения движения интегрируются в модель движения, а ограничения измерения – в модель измерения, соответственно. Целевой функцией GraphSLAM является сумма этих ограничений, а минимизация этой суммы представляет собой наиболее вероятную карту и самый вероятный путь робота.

Во многих отношениях EKF SLAM и GraphSLAM находятся на противоположных концах семейства алгоритмов SLAM. Основным различием между EKF SLAM и GraphSLAM является способ выражения информации. В EKF SLAM информация выражается с помощью матрицы ковариации и вектора средних, а в GraphSLAM информация выражается в виде графа мягких ограничений. Обновление ковариации в EKF вычислительно затратно, в то время, как наращивание графа выполняется крайне просто!

Такая экономия тоже имеет свою цену. В GraphSLAM требуются до-

полнительные преобразования для восстановления карты и пути, а EKF поддерживает наилучшую оценку для карты и положения робота в произвольный момент времени. После построения графа выполняется отдельный такт вычислений, в котором информация преобразуется в оценку состояния. Для EKF SLAM такой шаг не требуется.

ПРОАКТИВНЫЙ SLAM

В силу этого можно воспринимать EKF как *проактивный алгоритм SLAM*, поскольку он немедленно разрешает каждую новую порцию информации в улучшенную оценку состояния окружающего мира.

ЛЕНИВЫЙ SLAM

GraphSLAM, напротив, *ленивый метод SLAM*, который просто аккумулирует информацию в графе, не используя ее. Эта разница существенна, поскольку, в результате, GraphSLAM способен поддерживать карты на много порядков превышающие по размеру те, что способен обработать EKF.

Существуют и другие различия между EKF SLAM и GraphSLAM. Для решения полной задачи SLAM алгоритм GraphSLAM вычисляет апостериорные распределения по траекториям робота, поскольку не является инкрементным алгоритмом. Этот подход отличается от EKF SLAM, который, как и фильтр, сохраняет только апостериорную оценку текущего положения робота. EKF SLAM позволяет роботу произвольно обновлять карту, а GraphSLAM лучше всего подходит для случаев, когда необходимо получить карту из набора данных фиксированного размера. EKF SLAM может сохранять карту в течение всего жизненного цикла робота, без необходимости учитывать общее число тактов времени, прошедших с момента начала получения данных.

Поскольку GraphSLAM при построении карты имеет доступ ко всем данным, в нем можно использовать улучшенные методы ассоциации данных и линеаризации. В EKF SLAM, линеаризация и соответствие в момент времени t вычисляются на основе карты, полученной вплоть до предыдущего момента. В GraphSLAM для линеаризации и вычисления соответствия используются все имеющиеся данные. Другими словами, GraphSLAM способен пересмотреть уже выполненную в прошлом ассоциацию данных и выполнить линеаризацию несколько раз. Фактически, в GraphSLAM последовательно выполняется три критических шага картографирования: создание карты, вычисление переменных соответствия и линеаризацию моделей измерения и движения, что позволяет получить наиболее точные оценки их значений. В результате этого GraphSLAM часто создаёт карты, превосходящие по точности EKF.

Однако, GraphSLAM имеет ряд ограничений по сравнению с EKF. Одно из них уже обсуждалось: размер графа линейно растёт со временем, а в EKF нет такой зависимости количества занимаемой оценками памяти от времени. Другой случай относится к ассоциации данных. Если в EKF SLAM вероятности ассоциации данных легко получить из матрицы ковариации апостериорного распределения, вычисление тех же вероятностей в GraphSLAM требует дополнительных выводов. Эта разница будет описана ниже, в разделе определении алгоритма вычисления соответствия в GraphSLAM. Поэтому, вопрос наилучшей применимости метода не имеет чёткого ответа и зависит от области применения, поскольку ни один метод не будет превосходить остальные по всем критическим показателям.

В этой главе сначала описывается основная идея GraphSLAM и основные такты обновления. Затем будет представлен математический вывод различных тактов обновления и доказана их правильность относительно конкретных линейных аппроксимаций. Также будет выведен метод ассоциации данных, а затем – приведено обсуждение конкретных реализаций алгоритма GraphSLAM.

11.2 Общее описание

Основная идея, лежащая в основе GraphSLAM, достаточно проста: из данных извлекается набор мягких ограничений, который отображается в виде разреженного графа. Карта и траектория робота получаются в результате разрешения этих ограничений в виде глобально непротиворечивой оценки. Ограничения, в общем случае, имеют нелинейный вид, но в процессе разрешения линеаризуются и преобразуются в информационную матрицу. В силу этого, GraphSLAM представляет собой, в основном, информационно-теоретический метод. Мы опишем GraphSLAM как в виде метода построения разреженного графа нелинейных ограничений, так и в виде метода распространения разреженной информационной матрицы линеаризованных ограничений.

11.2.1 Построение графа

Допустим, дан набор измерений $z_{1:t}$ со связанными переменными соответствия $c_{1:t}$ и набором управляющих воздействий $u_{1:t}$. GraphSLAM превращает эти данные в граф, узлы которого представляют положения робота $x_{1:t}$ и признаки карты $m = \{m_j\}$. Каждое ребро графа соответствует некоторому событию: событие движения создаёт ребро между двумя положениями робота, а событие измерения связывает положение робота и признак на карте. Ребра выражают мягкие ограничения между положениями робота и признаками в GraphSLAM.

Для линейной системы эти ограничения эквивалентны элементам информационной матрицы и информационного вектора большой системы уравнений. Как обычно, обозначим информационную матрицу через Ω, а информационный вектор - через ξ. Как будет показано ниже, каждое измерение и каждое управляющее воздействие приводят к локальному обновлению Ω и ξ, что соответствует локальному добавлению ребра графа в GraphSLAM. Фактически, правило учёта управляющего воздействия или измерения в Ω и ξ – это локальное добавление, в силу важного факта аддитивности информации.

На. Рис. 11.2 показан процесс создания графа, а также соответствующая информационная матрица. Сначала учитывается измерение z_t^i. Это измерение предоставляет информацию о связи между местоположением признака $j = c_t^i$ и положением робота x_t в момент времени t. В GraphSLAM эта информация проектируется в виде ограничения между x_t и m_j. Можно описать ребро в виде некой «пружины» по аналогии с кинематической моделью масс, соединённых пружинами. Как будет показано ниже, ограничение имеет вид:

$$(z_t^i - h(x_t, m_j))^T \, Q_t^{-1} \, (z_t^i - h(x_t, m_j)) \tag{11.1}$$

Здесь h - уже знакомая функция измерения, а Q_t - ковариация шумов измерения. На Рис. 11.2а показано добавление такой связи в граф, сохраняемый в GraphSLAM.

Теперь необходимо учесть движение робота. Управляющее воздействие u_t даёт информацию об относительном значении положения робота в момент времени $t-1$ и положении в момент времени t. И снова, интегрирование

этой информации приводит к появлению нового ребра в графе вида

(11.2)
$$(x_t - g(u_t, x_{t-1}))^T R_t^{-1} (x_t - g(u_t, x_{t-1}))$$

Здесь g – уже знакомая кинематическая модель движения робота, а R_t - ковариация шумов движения.

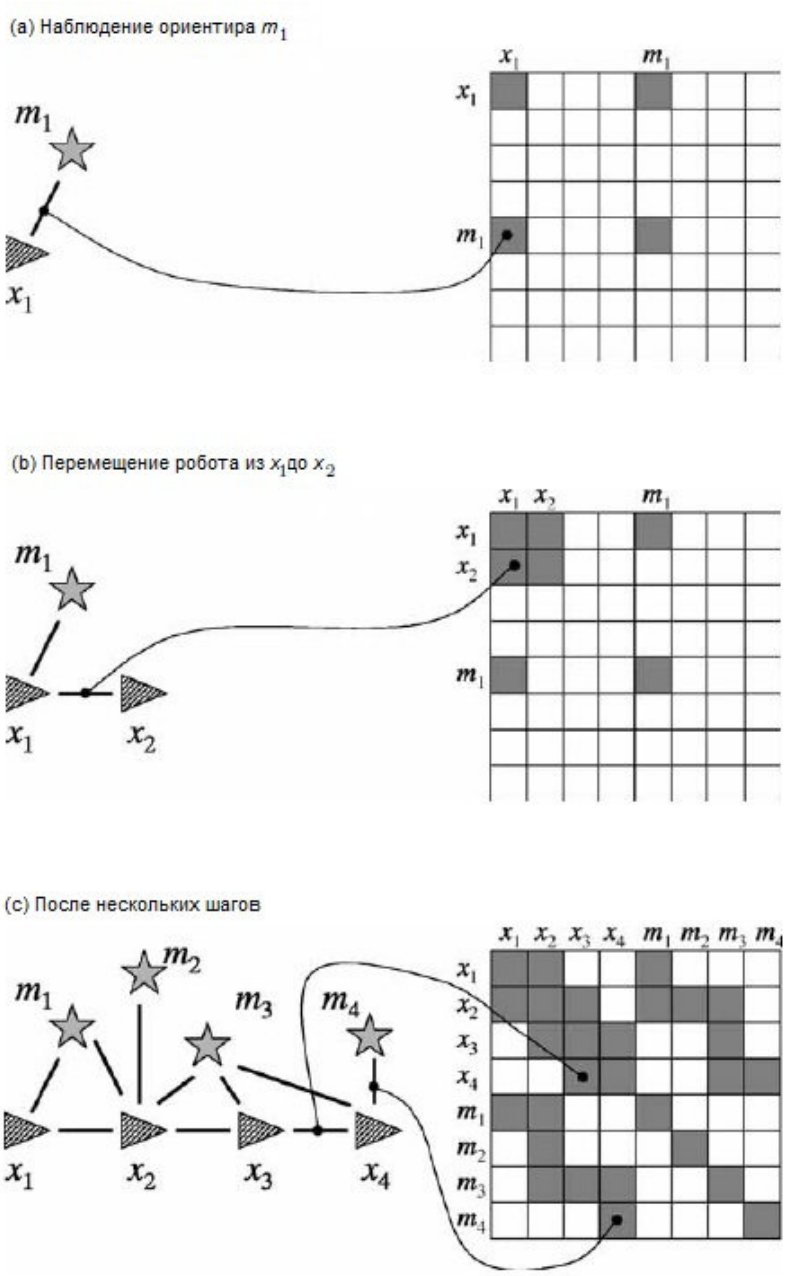


Рис. 11.2 Получение информационной матрицы в GraphSLAM. На левой схеме показан граф зависимостей, на правой – информационная матрица.

На Рис. 11.2b показано добавление такой связи к графу. Также изображено добавление нового элемента связи между положением x_t и измере-

нием z_t^i в информационную матрицу. Эта процедура тоже аддитивна. Как и раньше, величина этих значений отражает остаточную неопределённость R_t, вызванную шумами измерений. Чем точнее датчик, тем большее значение будет добавлено к Ω и ξ.

После учёта всех измерений $z_{1:t}$ и управляющих воздействий $u_{1:t}$ получим разреженный граф мягких ограничений. Число ограничений в графе линейно зависит от времени, следовательно, граф разрежен. Сумма всех ограничений графа должна иметь вид

(11.3)

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t (x_t - g(u_t, x_{t-1}))^T R_t^{-1} (x_t - g(u_t, x_{t-1})) + \sum_t \sum_i (z_t^i - h(y_t, c_t^i))^T Q_t^{-1} (z_t^i - h(y_t, c_t^i))$$

и представляет собой функцию, определённую по переменным положения $x_{1:t}$ и всем расположениям признаков на карте m. Заметим, что это выражение также характеризует фиксирующее ограничение вида $x_0^T \ \Omega_0 \ x_0$. Это ограничение выполняет привязку абсолютных координат карты путём инициализации первого положения робота в точке $(0\ 0\ 0)^T$.

ФИКСИРУЮЩЕЕ ОГРАНИЧЕНИЕ

В связанной информационной матрице Ω все элементы вне главной диагонали равны нулю, за исключением двух. Между двумя последовательными положениями x_{t-1} и x_t будет находиться ненулевое значение, выражающее информационную связь, внесённую управляющим воздействием u_t. Также ненулевым будет любой элемент между признаком карты m_j и положением x_t, для которого m_j можно наблюдать из положения робота x_t. Все элементы между прочими парами признаков остаются нулевыми. Это отражает факт отсутствия информации об относительном местоположении, поскольку всё, что поступает на вход SLAM, это измерения, ограничивающие расположение признака относительно положения робота. Поэтому информационная матрица также разрежена, а все элементы, за исключением некоторого линейного количества, равны нулю.

11.2.2 Допущения

Конечно, ни отображение в виде графа, ни информационная матрица не дают желаемого результата, то есть карты и траектории движения. В Graph SLAM карта и траектория получаются из линеаризованной информационной матрицы с помощью $\mu = \Omega^{-1}\xi$ (см. равенство (3.73) на странице 74). Для этой операции требуется решить систему линейных уравнений и возникает вопрос о том, насколько эффективно возможно восстановить оценку карты μ и ковариацию Σ.

Ответ на вопрос о сложности зависит от топологии среды. Если каждый признак наблюдается только локально по времени, то и граф, выраженный в виде ограничений, будет линеен. Поэтому Ω можно переопределить в виде ленточной матрицы, где все ненулевые элементы расположены вдоль главной диагонали, а уравнение $\mu = \Omega^{-1}\xi$ можно вычислить за линейное время. Это соображение соответствует неповторяющейся окружающей среде, через которую проходит робот, так что все признаки различимы лишь в течение короткого периода времени, последовательно, один за другим.

Более общий случай, однако, включает признаки, наблюдаемые несколько раз с большими промежутками времени между наблюдениями. Это мо-

ЦИКЛЫ

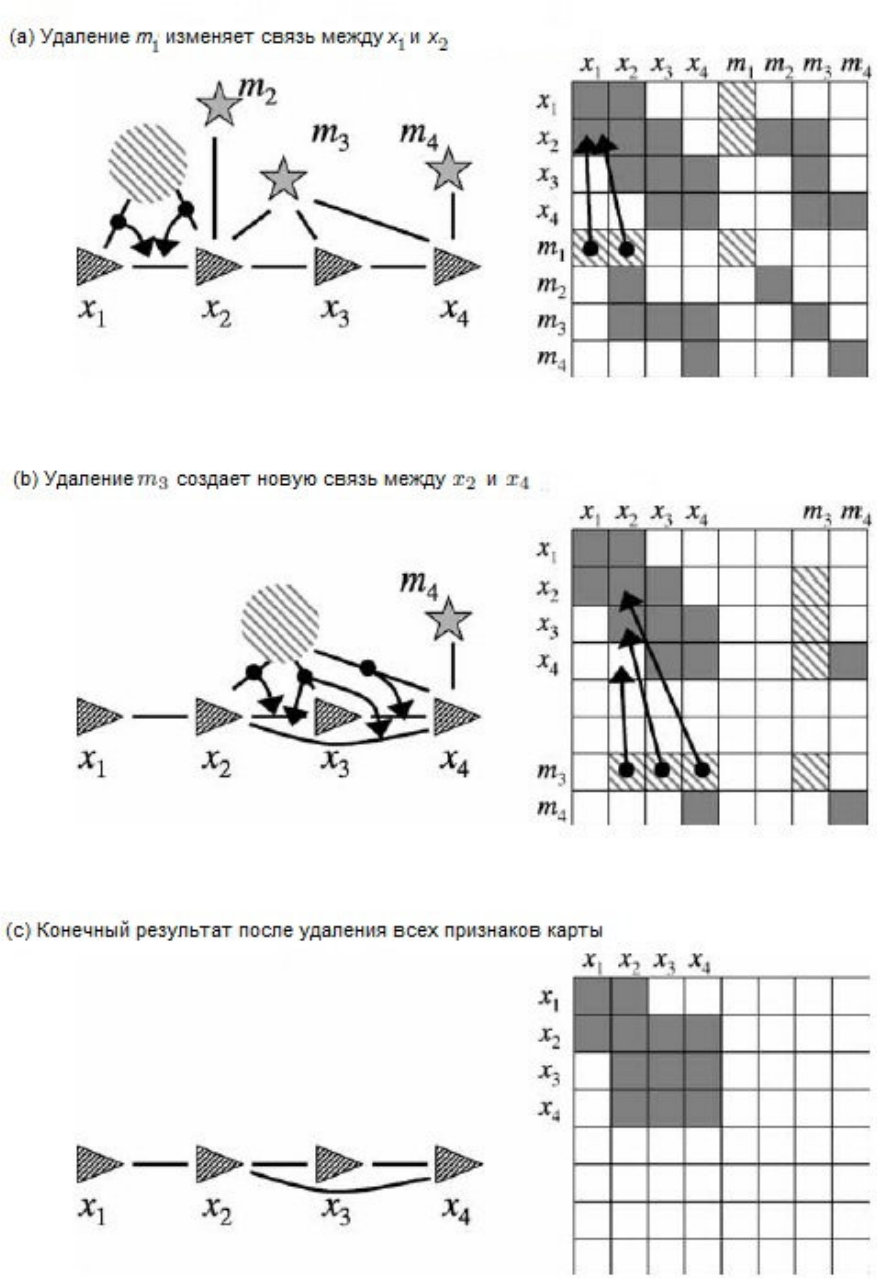
жет произойти, например, если робот перемещается вперёд и назад по коридору, или в силу наличия в окружающей среде *циклов*. В любом случае, будут существовать признаки m_j, наблюдаемые в сильно отличающиеся моменты времени x_{t_1} и x_{t_2}, с $t_2 \gg t_1$. На графе ограничений это вызывает циклическую зависимость: x_{t_1} и x_{t_2} соединены через последовательность управляющих действий $u_{t_1+1}, u_{t_1+2}, ..., u_{t_2}$ и через последовательность наблюдений между x_{t_1} и m_j, в промежутке между x_{t_2} и m_j, соответственно. Такие связи не позволяют выполнить переопределение переменных, что усложняем восстановление карты. Фактически, поскольку инвертированная Ω перемножается с вектором, результат возможно вычислить, используя методы оптимизации, например, сопряжённых градиентов, без необходимости явно вычислять полную обратную матрицу. Поскольку большая часть сред имеет циклы, этот случай представляет интерес.

ФАКТОРИЗАЦИЯ

В алгоритме GraphSLAM теперь можно выполнить *факторизацию*, который можно считать распространением информации через информационную матрицу (это обобщение известного алгоритма *удаления переменных* для инверсии матрицы). Допустим, хотелось бы удалить признак m_j из информационной матрицы Ω и информационного состояния ξ. В кинематической модели масс на пружинах это эквивалентно удалению узла массы и всех соединённых с ним пружин. Как мы увидим ниже, это возможно с помощью довольно простой операции: можно «удалить пружины» между m_j и положениями, из которых наблюдался m_j, поставив «новые пружины» для каждой пары таких положений.

Этот процесс показан на Рис. 11.3, где показано удаление двух признаков карты, m_1 и m_3 (удаление m_2 и m_4 в этом примере тривиально). В обоих случаях, удаление признаков изменяет связь в каждой паре положений, из которых наблюдался признак. Как показано на Рис. 11.3b, эта операция может привести к возникновению новых связей в графе. В показанном примере удаление m_3 приводит к появлению новой связи между x_2 и x_4.

В формальном виде, пусть $\tau(j)$ будет набором положений, из которых наблюдался m_j (так, что: $x_t \in \tau(j) \iff \exists i \,:\, c_t^i = j$. Уже известно, что m_j связан только с положением x_t в $\tau(j)$. На графе m_j не связан с другим положением или признаком карты. Теперь можно установить значение всех связей между m_j и положениями $\tau(j)$ в нуль, создав новую связь между двумя положениями $x_t, x_{t'} \in \tau(j)$. Похожим образом, значения информационного вектора для всех положений $\tau(j)$ также обновляется. Важной характеристикой этой операции является ее локальность, поскольку она затрагивает лишь небольшое число ограничений. После удаления всех связей с m_j можно безопасно удалить m_j из информационной матрицы и вектора. Размер результирующей информационной матрицы уменьшается, поскольку в ней отсутствует m_j. Но все остальные элементы остаются неизменными, и апостериорное распределение, определённое этой информационной матрицей, математически эквивалентно начальному распределению до удаления m_j. Этот вывод достаточно интуитивен: необходимо просто заменить «пружины», соединяющие m_j с различными положениями в нашей модели масс, набором пружин, напрямую соединяющими эти положения. В результате, общее усилие на пружинах останется неизменным, за исключением того, что исключается m_j.



(Рис. 11.3 Уменьшение графа в GraphSLAM: дуги, соединяющих только положения робота, удалены, чтобы выделить связи.)

Красота такого подхода состоит в возможности постепенно разбивать задачу связности на более мелкие. Удаляя каждый признак m_j из Ω и ξ, постепенно можно прийти к значительно меньшему информационному представлению $\bar{\Omega}$ и $\bar{\xi}$, определённому только по пути робота. Это уменьшение может быть выполнено за время, линейно зависящее от размера карты. Тем самым выполняется обобщение метода уничтожения переменной для инверсии матрицы информационного представления, в котором также поддерживается информационное состояние. Апостериорное распределение по пути робота теперь можно восстановить, как $\bar{\Sigma} = \bar{\Omega}^{-1}$ и $\bar{\mu} = \bar{\Sigma}\xi$. К сожа-

лению, такт уменьшения размера не уничтожает циклы в апостериорном распределении. Оставшаяся задача математического вывода может потребовать дополнительных нелинейных шагов.

На последнем шаге GraphSLAM восстанавливает расположение признаков. Теоретически, это достигается построением новой информационной матрицы Ω_j и информационного вектора ξ_j для каждого признака m_j. Они определены для переменной признака m_j и положений $\tau(j)$, в которых он наблюдался. Матрица содержит начальные связи между m_j и $\tau(j)$, но положения $\tau(j)$ установлены в значения $\bar{\mu}$, без учёта неопределённости. С информационной точки зрения, достаточно просто вычислить местоположение m_j, используя общий метод обращения матриц. Ясно, что Ω_j содержит только элементы, соединяющие признак m_j, поэтому обращение займёт время, линейно зависящее от $\tau(j)$.

Должно быть очевидно, почему выражение в виде графа столь естественно. Полная задача SLAM решается локальным добавлением информации в большой информационный граф по одному ребру за такт для каждого измерения z_t^i и каждого управляющего воздействия u_t. Чтобы превратить такую информацию в оценку карты и пути робота, она сначала подвергается линеаризации, а затем информация между положениями и признаками последовательно смещается в информацию между парами положений. Результирующая структура состоит только из положений робота, которые вычисляются, используя инверсию матриц. После восстановления положений местоположения признаков вычисляются одно за другим, на основе начальной информации отношения признака к положению.

11.3 Алгоритм GraphSLAM

Давайте выполним точное вычисление различных тактов алгоритма Graph SLAM. *Алгоритм полного GraphSLAM* будет описан в виде последовательности шагов. Главной трудностью реализации простого аддитивного информационного алгоритма является преобразование условной вероятности вида $p(z_t^i|x_t, m)$ и $p(x_t|u_t, x_{t-1})$ в связь в информационной матрице. Все элементы информационной матрицы линейны, поэтому этот шаг включает линеаризацию $p(z_t^i|x_t, m)$ и $p(x_t|u_t, x_{t-1})$. В EKF SLAM эта линеаризация была вычислена с помощью якобиана для средних оценок положений $\mu_{0:t}$. Для построения начальной информационной матрицы Ω и ξ требуется начальная оценка $\mu_{0:t}$ для всех положений $x_{0:t}$.

1: **Algorithm GraphSLAM_initialize**$(u_{1:t})$:

2: $$\begin{pmatrix} \mu_{0,x} \\ \mu_{0,y} \\ \mu_{0,\theta} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

3: *для всех управляющих действий* $u_t = (v_t\ \omega_t)^T$ *do*

4: $$\begin{pmatrix} \mu_{t,x} \\ \mu_{t,y} \\ \mu_{t,\theta} \end{pmatrix} = \begin{pmatrix} \mu_{t-1,x} \\ \mu_{t-1,y} \\ \mu_{t-1,\theta} \end{pmatrix}$$

4: $$+ \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

5: *endfor*

6: *return* $\mu_{0:t}$

Таблица 11.1 Инициализация вектора среднего положения $\mu_{1:t}$ в алгоритме GraphSLAM.

1: **Algorithm GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$:

2: *Установить* $\Omega = 0$, $\xi = \overline{0}$

3: *прибавить* $\begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix}$ *к* Ω *при* x_0

4: *для всех управляющих действий* $u_t = (v_t\ \omega_t)^T$ *выполнить*

5: $$\hat{x}_t = \mu_{t-1} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

6: $$G_t = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

продолжение на следующей странице

начало на предыдущей странице

7: *прибавить* $\begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1} \begin{pmatrix} -G_t & 1 \end{pmatrix}$ *to* Ω *к* x_t *и* x_{t-1}

8: *прибавить* $\begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1} [\hat{x}_t - G_t \mu_{t-1}]$ *к* ξ *при* x_t *и* x_{t-1}

9: *endfor*

10: *для всех измерений* z_t *do*

11: $Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$

12: *для всех наблюдаемых признаков* $z_t^i = (r_t^i \ \phi_t^i \ s_t^i)^T$ *do*

13: $j = c_t^i$

14: $\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \mu_{j,x} - \mu_{t,x} \\ \mu_{j,y} - \mu_{t,y} \end{pmatrix}$

15: $q = \delta^T \delta$

16: $\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \mu_{t,\theta} \\ s_j \end{pmatrix}$

17: $H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix}$

18: *добавить* $H_t^{iT} Q_t^{-1} H_t^i$ *к* Ω *в* x_t *и* m_j

19: *добавить* $H_t^{iT} Q_t^{-1} [z_t^i - \hat{z}_t^i + H_t^i \begin{pmatrix} \mu_{t,x} \\ \mu_{t,y} \\ \mu_{t,\theta} \\ \mu_{j,x} \\ \mu_{j,y} \\ \mu_{j,s} \end{pmatrix}]$ *к* ξ *в* x_t *и* m_j

20: *endfor*

21: *endfor*

22: *return* Ω, ξ

Таблица 11.2 Вычисление Ω и ξ в GraphSLAM.

1: **Algorithm GraphSLAM_reduce**(Ω, ξ) :

2: $\bar{\Omega} = \Omega$
3: $\bar{\xi} = \xi$
4: *для каждого признака* j *выполнить*
5: *пусть* $\tau(j)$ *набор всех положений* x_t *в которых наблюдался* j
6: *вычесть* $\bar{\Omega}_{\tau(j),j}\bar{\Omega}_{j,j}^{-1}\xi_j$ *из* $\bar{\xi}$ *в* $x_{\tau(j)}$ *и* m_j
7: *вычесть* $\bar{\Omega}_{\tau(j),j}\bar{\Omega}_{j,j}^{-1}\bar{\Omega}_{j,\tau(j)}$ *из* $\bar{\Omega}$ *в* $x_{\tau(j)}$ *и* m_j
8: *удалить из* $\bar{\Omega}$ *и* $\bar{\xi}$ *все строки/столбцы, соответсвующие* j
9: *endfor*
10: *return* $\bar{\Omega}, \bar{\xi}$

Таблица 11.3 Алгоритм уменьшения размера информационного представления апостериорного распределения в GraphSLAM.

1: **Algorithm GraphSLAM_solve**$(\bar{\Omega}, \bar{\xi}, \Omega, \xi)$:

2: $\Sigma_{0:t} = \bar{\Omega}^{-1}$
3: $\mu_{0:t} = \Sigma_{0:t}\bar{\xi}$
4: *для каждого признака* j *выполнять*
5: *set* $\tau(j)$ *к набору всех положений* x_t *в которых наблюдался* j
6: $\mu_j = \Omega_{j,j}^{-1}(\xi_j + \Omega_{j,\tau(j)}\bar{\mu}_{\tau(j)})$
7: *endfor*
8: *return* $\mu, \Sigma_{0:t}$

Таблица 11.4 Алгоритм обновления апостериорного распределения μ.

1: **Algorithm GraphSLAM_known_correspondence**$(u_{1:t}, z_{1:t}, c_{1:t})$:

2: $\mu_{0:t} =$ **GraphSLAM_initialize** $(u_{1:t})$
3: *повторять*
4: $\Omega, \xi =$ **GraphSLAM_linearize** $(u_{1:t}, z_{1:t}, c_{1:t}\mu_{0:t})$
5: $\bar{\Omega}, \bar{\xi} =$ **GraphSLAM_reduce** (Ω, ξ)
6: $\mu, \Sigma_{0:t} =$ **GraphSLAM_solve** $(\bar{\Omega}, \bar{\xi}, \Omega, \xi)$
7: *до сходимости*
8: *return* μ

Таблица 11.5 Алгоритм GraphSLAM для полной задачи SLAM с известным соответствием.

Существует большое количество решений задачи нахождения начального среднего μ, подходящего для линеаризации. Например, можно запустить алгоритм EKF SLAM и использовать его оценку для линеаризации. В этой

главе будет использован ещё более простой приём. Начальная оценка будет получена простым связываем воедино модели движения $p(x_t|u_t, x_{t-1})$.Такой алгоритм приводится в Таблице 11.1, и называется **GraphSLAM_initialize**. Этот алгоритм принимает на вход сигналы управления $u_{1:t}$ и выдаёт на выход последовательность оценок положений $\mu_{0:t}$. В нем первое положение инициализируется нулём, а затем последующие положения вычисляются путём рекурсивного применения модели движения на основе скорости. Поскольку нас интересует только средний вектор положений $\mu_{0:t}$, в алгоритме **GraphSLAM_initialize** используется только детерминированная часть модели движения. Измерения в оценке также не учитываются.

Когда начальное значение $\mu_{0:t}$ доступно, алгоритм GraphSLAM создаёт полную информационную матрицу SLAM Ω и соответствующий информационный вектор ξ, выполняя линеаризацию всех связей на графе. Алгоритм **GraphSLAM_linearize** приведён в Таблице 11.2. В нём присутствует множество математических выражений, смысл которых будет раскрыт ниже в разделе математического вывода алгоритма. **GraphSLAM_linearize** принимает на вход набор управляющих сигналов, $u_{1:t}$, измерения $z_{1:t}$, связанные переменные соответствия $c_{1:t}$ и средние оценки положения $\mu_{0:t}$. Затем с помощью линеаризации постепенно создаётся информационная матрица Ω и информационный вектор ξ, локально добавляя субматрицы в соответствии с информацией, полученной из каждого измерения и каждого управляющего воздействия.

В частности, в строке 2 в **GraphSLAM_linearize** выполняется инициализация информационных элементов. "Бесконечный" информационный элемент в строке 3 фиксирует начальное положение x_0 по координатам $(0\ 0\ 0)^T$. Это необходимо, поскольку в противном случае результирующая матрица становится вырожденной, отражая факт невозможности восстановления абсолютных оценок на основании только относительной информации.

Управление интегрируется в строках с 4 по 9 алгоритма **GraphSLAM_linearize**. Положение $\hat{x}$ и якобиан G_t, вычисленные в строках 5 и 6, отображают линейную аппроксимацию нелинейной функции управления g. Как очевидно следует из этих уравнений, такт линеаризации использует оценки положения $\mu_{0:t-1}$ при $\mu_0 = (0\ 0\ 0)^T$. Это ведёт к обновлениям Ω и ξ, вычисленных в строках 7 и 8, соответственно. Оба члена добавляются в соответствующие строки и столбцы Ω и ξ. Это добавление реализует включение нового ограничения в апостериорную оценку SLAM, что очень близко к интуитивному описанию в предыдущем разделе.

Измерения интегрируются в строках с 10 по 21 алгоритма **GraphSLAM_linearize**. Матрица Q_t, вычисленная в строке 11, представляет собой уже знакомую ковариацию шумов измерений. В строках с 13 по 17 выполняется разложение в ряд Тейлора функции измерений, определённой здесь для модели измерения на основе признаков, которая была описана в разделе 6.6. Следует обратить внимание на реализацию в строке 16, поскольку угловые величины могут быть произвольно смещены на 2π. Это вычисление даёт операцию обновления измерения в строках 18 и 19. Матрица, добавляемая к Ω в строке 18, имеет размер 6×6. Для добавления она разбивается на матрицу размером 3×3 для положения x_t, матрицу размером 3×3 для признака m_j, и две матрицы размером 3×3 каждая для связей между x_t и m_j. Они добавляются к Ω в соответствующих строках и столбцах. Аналогично, вектор добавляется к информационному вектору ξ размером 5 по вертикали. Он также разбивается на векторы размером 3 и 2 и добавляется к элементам, соответствующим x_t и m_j. Результатом работы **GraphSLAM_linearize** являются информационный вектор ξ и мат-

рица Ω. Уже было отмечено, что Ω является разреженной матрицей. Она содержит ненулевые субматрицы только вдоль главной диагонали, между последовательными положениями, и между положениями и признаками на карте. Время работы алгоритма линейно зависит от t, количества тактов времени, когда были получены данные.

Следующим шагом алгоритма GraphSLAM является уменьшение размерности информационной матрицы и вектора. Это достигается с помощью алгоритма **GraphSLAM_reduce** в Таблице 11.3. Этот алгоритм принимает на вход Ω и ξ, определённые по всему пространству признаков карты и положений, а на выход поступает уменьшенная матрица $\bar{\Omega}$ и векторы $\bar{\xi}$, определённые в пространстве всех положений, но без карты! Это преобразование выполняется последовательным удалением признаков m_j, в строках с 4 по 9 **GraphSLAM_reduce**. Учёт точных индексов каждого элемента $\bar{\Omega}$ и $\bar{\xi}$ достаточно утомителен, поэтому в Таблице 11.3 приводится лишь общее описание.

В строке 5 вычисляется набор положений $\tau(j)$, в которых робот наблюдает признак j. Затем из существующих матриц $\bar{\Omega}$: $\bar{\Omega}_{j,j}$ и $\bar{\Omega}_{\tau(j),j}$ извлекаются две субматрицы. $\bar{\Omega}_{j,j}$ - это квадратная субматрица между m_j и m_j, а $\bar{\Omega}_{\tau(j),j}$ составлена из недиагональных элементов между m_j и переменными положения $\tau(j)$. Затем из информационного вектора состояний $\bar{\xi}$ извлекаются элементы, соответствующие j-му признаку, обозначаемому здесь как ξ_j. Затем из $\bar{\Omega}$ и $\bar{\xi}$ вычитается информация, как показано в строках 6 и 7. После этой операции строки и столбцы для признака m_j становятся равны нулю. Эти строки и столбцы удаляются, что уменьшает размер $\bar{\Omega}$ и $\bar{\xi}$, соответственно. Процесс повторяется до удаления всех признаков, пока в $\bar{\Omega}$ и $\bar{\xi}$ не останутся только положения. Сложность **GraphSLAM_reduce** снова линейно зависит от t.

Последним шагом в алгоритме GraphSLAM является вычисление математического ожидания и ковариации для всех положений на пути робота, и средней оценки местоположения для всех признаков на карте. Это достигается с помощью алгоритма **GraphSLAM_solve**, приведённого в Таблице 11.4. В строках 2 и 3 вычисляются оценки пути $\mu_{0:t}$ с помощью инверсии уменьшенной информационной матрицы $\bar{\Omega}$ и умножения результирующей ковариации на информационный вектор. Затем в **GraphSLAM_solve** в строках с 4 по 7 вычисляется местоположение каждого признака. Возвращаемое значение **GraphSLAM_solve** содержит средний путь робота и все признаки карты, но только ковариацию для пути робота. Было замечено, что существуют другие, более эффективные способы вычисления $\mu_{0:t}$, не требующие обращения матрицы. Они будут обсуждаться ближе к концу главы, в описании применения стандартных методов оптимизации для GraphSLAM.

Качество решения, вычисленного алгоритмом GraphSLAM, зависит от качества начальных средних оценок, вычисленных **GraphSLAM_initialize**. Компоненты x и y координат этих оценок линейно воздействуют на соответствующие модели, поэтому линеаризация от них никак не зависит. Но для переменных ориентации $\mu_{0:t}$ это не так. Ошибки в начальных оценках влияют на точность приближения разложением в ряд Тейлора, что, в свою очередь, влияет на результат.

Для уменьшения потенциальных ошибок разложения в ряд Тейлора при линеаризации, процедуры **GraphSLAM_linearize**, **GraphSLAM_reduce**, и **GraphSLAM_solve** могут запускаться несколько раз подряд на одном и том же наборе данных. Каждая итерация принимает на вход оценочный вектор $\mu_{0:t}$ предыдущей итерации, выдавая новую, уточнённую оценку. По-

вторение итераций GraphSLAM необходимо только когда начальные оценки положения имеют большую ошибку (более 20 градусов по углу ориентации). Небольшого количества итераций (обычно, трёх) достаточно.

В Таблице 11.5 приводится общий вид алгоритма. В нем инициализируются средние значения, затем повторяется такты создания матрицы, уменьшения размерности, и решения. Обычно, для сходимости достаточно двухтрёх итераций. Результирующее среднее μ является наилучшей гипотезой для пути робота и карты.

11.4 Математический вывод GraphSLAM

Вывод алгоритма GraphSLAM начинается с вывода рекурсивной формулы для вычисления полного апостериорного распределения SLAM, выраженного в информационном виде. Затем будет подробно описан каждый член этого распределения, а затем выведен аддитивный шаг обновления SLAM с помощью разложения в ряд Тейлора. После этого будут выведены необходимые уравнения для восстановления пройденного пути и карты.

11.4.1 Полное апостериорное распределение SLAM

Также, как и при обсуждении EKF SLAM, будет выгодно ввести переменную для дополненного состояния в полной проблеме SLAM. Будем использовать y для обозначения переменных состояния, сочетающих одно или более положений x на карте m. В частности, определим $y_{0:t}$ в виде вектора, состоящего из пути $x_{0:t}$ и карты m, где y_t состоит из моментального положения в момент времени t и карты m:

$$y_{0:t} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \\ m \end{pmatrix} \quad \text{и} \quad y_t = \begin{pmatrix} x_t \\ m \end{pmatrix} \tag{11.4}$$

Апостериорное распределение для полной задачи SLAM $p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t})$, где $z_{1:t}$ - измерения с соответствиями $c_{1:t}$, а $u_{1:t}$ – управляющие воздействия. Теорема Байеса позволяет выполнить факторизацию этого распределение:

$$\begin{aligned} &p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \\ &\quad = \eta\, p(z_t|y_{0:t}, z_{1:t-1}, u_{1:t}, c_{1:t}) p(y_{0:t}|z_{1:t-1}, u_{1:t}, c_{1:t}) \end{aligned} \tag{11.5}$$

где η – уже знакомый нормализующий член. Первое значение вероятности с правой стороны может быть упрощено отбрасыванием нерелевантных переменных:

$$p(z_t|y_{0:t}, z_{1:t-1}, u_{1:t}, c_{1:t}) = p(z_t|y_t, c_t) \tag{11.6}$$

Похожим образом можно факторизовать вторую вероятность, разбив $y_{0:t}$ по x_t и $y_{0:t-1}$, что даёт

(11.7)

$$\begin{aligned} & p(y_{0:t}|z_{1:t-1}, u_{1:t}, c_{1:t}) \\ & \quad = p(x_t|y_{0:t-1}, z_{1:t-1}, u_{1:t}, c_{1:t}) p(y_{0:t-1}|z_{1:t-1}, u_{1:t}, c_{1:t}) \\ & \quad = p(x_t|x_{t-1}, u_t) p(y_{0:t-1}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \end{aligned}$$

Вставка этих выражений обратно в (11.5) даст рекурсивное определение полного апостериорного распределения SLAM:

(11.8)

$$\begin{aligned} & p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \\ & \quad = \eta\, p(z_t|y_t, c_t) p(x_t|x_{t-1}, u_t) p(y_{0:t-1}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1}) \end{aligned}$$

Выражение в закрытом виде получается с помощью индукции по t. Здесь $p(y_0)$ - априорная вероятность по карте m и начальному положению x_0.

(11.9)

$$\begin{aligned} p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) &= \eta\, p(y_0) \prod_t p(x_t|x_{t-1}, u_t) p(z_t|y_t, c_t) \\ &= \eta\, p(y_0) \prod_t \left[p(x_t|x_{t-1}, u_t) \prod_i p(z_t^i|y_t, c_t^i) \right] \end{aligned}$$

Здесь, как и прежде, z_t^i это i-е измерение в векторе измерения z_t в момент времени t. Априорная вероятность $p(y_0)$ разбивается на две независимые априорные вероятности, $p(x_0)$ и $p(m)$. В SLAM обычно нет никаких предварительных данных о карте m, поэтому $p(y_0)$ просто заменяется $p(x_0)$, а множитель $p(m)$ выносится в нормализующий член η.

11.4.2 Отрицательный логарифм апостериорной вероятности

В информационном виде вероятности представлены логарифмическими выражениями. Форма апостериорной вероятности log-SLAM напрямую следует из предыдущего равенства:

(11.10)

$$\begin{aligned} & \log\, p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \\ & \qquad = \text{const.} + \log p(x_0) + \sum_t \left[\log p(x_t|x_{t-1}, u_t) + \sum_i \log p(z_t^i|y_t, c_t^i) \right] \end{aligned}$$

Так же, как в Главе 10, допустим, что результат движения робота нормально распределён по $\mathcal{N}(g(u_t, x_{t-1}), R_t)$, где g – детерминированная функция движения, а R_t – ковариация ошибки движения. Аналогично, измерения z_t^i генерируются согласно распределению $\mathcal{N}(h(y_t, c_t^i), Q_t)$, где h уже знакомая функция измерения, а Q_t – ковариация ошибки измерения. В виде уравнений получается

(11.11)

$$p(x_t|x_{t-1}, u_t) = \eta\, \exp\left\{ -\frac{1}{2} (x_t - g(u_t, x_{t-1}))^T R_t^{-1} (x_t - g(u_t, x_{t-1})) \right\}$$

(11.12)

$$p(z_t^i|y_t, c_t^i) = \eta \exp\left\{-\frac{1}{2}(z_t^i - h(y_t, c_t^i))^T Q_t^{-1}(z_t^i - h(y_t, c_t^i))\right\}$$

Априорная вероятность $p(x_0)$ в (11.10) также выражена распределением, похожим на гауссовское. Она связывает начальное положение x_0 и началу глобальной системы координат: $x_0 = (0\ 0\ 0)^T$:

(11.13)

$$p(x_0) = \eta \exp\left\{-\frac{1}{2}x_0^T\, \Omega_0\, x_0\right\}$$

с

(11.14)

$$\Omega_0 = \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix}$$

Сейчас не нужно беспокоиться о том, что значение ∞ невозможно представить, поскольку его можно легко заменить достаточно большим положительным числом. Это даст следующую квадратичную форму отрицательной log-SLAM апостериорной вероятности в (11.10):

(11.15)

$$\begin{aligned} &-\log p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \\ &\quad = \text{const.} + \frac{1}{2}[x_0^T \Omega_0 x_0 + \sum_t (x_t - g(u_t, x_{t-1}))^T R_t^{-1}(x_t - g(u_t, x_{t-1})) \\ &\qquad\qquad + \sum_t \sum_i (z_t^i - h(y_t, c_t^i))^T Q_t^{-1}(z_t^i - h(y_t, c_t^i))] \end{aligned}$$

В основном, это то же самое, что $J_{GraphSLAM}$ в выражении (11.3), с некоторыми различиями в силу отбрасывания нормализующих постоянных (включая умножение на -1). Выражение (11.15) показывает важную характеристику полной апостериорной вероятности SLAM в информационном виде: она состоит из определённого числа квадратичных членов, одного для априорной вероятности, и по одному для каждого управляющего воздействия и измерения.

11.4.3 Разложение в ряд Тейлора

Различные члены в Выражении (11.15) квадратичны для функций g и h, но не для переменных, которые следует оценить (положения и карта). GraphSLAM решает проблему линеаризации g и h разложением в ряд Тейлора, то есть полностью аналогично выражениям (10.14) и (10.18) в выводе EKF. В частности, получается:

ЛИНЕАРИЗАЦИЯ

(11.16)

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1})$$

(11.17)

$$h(y_t, c_t^i) \approx h(\mu_t, c_t^i) + H_t^i(y_t - \mu_t)$$

Здесь μ_t - текущая оценка вектора состояний y_t, а $H_t^i = h_t^i F_{x,j}$ как было определено в выражении (10.19).

Это линейное приближение превращает логарифм правдоподобия (11.15) в функцию, квадратичную по $y_{0:t}$. В частности, получается

(11.18)

$$\begin{aligned}\log p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) = \text{const.} - \frac{1}{2} \\ \{x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T \\ R_t^{-1}[x_t - g(u_t, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})] \\ + \sum_i [z_t^i - h(\mu_t, c_t^i) - H_t^i(y_t - \mu_t)]^T Q_t^{-1}[z_t^i - h(\mu_t, c_t^i) - H_t^i(y_t - \mu_t)]\}\end{aligned}$$

Эта функция действительно квадратична по $y_{0:t}$, и необходимо перегруппировать члены, отбросив несколько констант.

(11.19)

$$\begin{aligned}\log p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) = \text{const.} \\ -\frac{1}{2} \underbrace{x_0^T \Omega_0 x_0}_{\text{квадратична по } x_0} - \frac{1}{2} \sum_t \underbrace{x_{t-1:t}^T \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1} (-G_t \ 1) x_{t-1:t}}_{\text{квадратична по } x_{t-1:t}} \\ \underbrace{+ x_{t-1:t}^T \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1}[g(u_t, \mu_{t-1}) - G_t \mu_{t-1}]}_{\text{линейна в } x_{t-1:t}} \\ -\frac{1}{2} \sum_i \underbrace{y_t^T H_t^{iT} Q_t^{-1} H_t^i y_t}_{\text{квадратична по } y_t} + \underbrace{y_t^T H_t^{iT} Q_t^{-1}[z_t^i - h(\mu_t, c_t^i) + H_t^i \mu_t]}_{\text{линейна в } y_t}\end{aligned}$$

Здесь $x_{t-1:t}$ определяет вектор состояния, присоединяя x_{t-1} и x_t. Отсюда можно записать $(x_t - G_t \ x_{t-1})^T = x_{t-1:t}^T (-G_t \ 1)^T = x_{t-1:t}^T \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix}$.

Если собрать все квадратичные члены в матрицу Ω, а все линейные – в вектор ξ, можно увидеть, что выражение (11.19) имеет вид

(11.20)

$$\log p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) = \text{const.} - \frac{1}{2} y_{0:t}^T \Omega y_{0:t} + y_{0:t}^T \xi$$

11.4.4 Создание информационного представления

Эти элементы можно напрямую взять из (11.19), и убедиться, что они действительно реализуют алгоритм **GraphSLAM_linearize** в Таблице 11.2:

• Априорная вероятность. Начальное положение показано квадратичным членом Ω_0 по переменной начального положения x_0 в информационной матрице. Допуская соответствующее расширение матрицы Ω_0 до размерности $y_{0:t}$, получим

(11.21)

$$\Omega \longleftarrow \Omega_0$$

Эта инициализация выполняется в строках 2 и 3 алгоритма **GraphSLAM _linearize**.

• **Управление.** Из (11.19), можно увидеть, что каждое управляющее воздействие u_t добавляет к Ω и ξ следующие члены, при условии, что матрицы расширены до одинаковых размеров:

(11.22)
$$\Omega \longleftarrow \Omega + \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1}(-G_t \; 1)$$

(11.23)
$$\xi \longleftarrow \xi + \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R_t^{-1}[g(u_t, \mu_{t-1}) - G_t\mu_{t-1}]$$

Это выполняется в строках с 4 по 9 в **GraphSLAM_linearize**.

• **Измерения.** Согласно Выражению (11.19), каждое измерение z_t^i преобразует Ω и ξ добавлением следующих членов, снова подразумевая расширение матриц до одинаковых размеров:

(11.24)
$$\Omega \longleftarrow \Omega + H_t^{iT} Q_t^{-1} H_t^i$$

(11.25)
$$\xi \longleftarrow \xi + H_t^{iT} Q_t^{-1}[z_t^i - h(\mu_t, c_t^i) + H_t^i \mu_t]$$

Это обновление выполняется в строках с 10 по 21 в **GraphSLAM_linearize** и доказывает правильность алгоритма конструирования **GraphSLAM_linearize**, относительно приближения разложением в ряд Тейлора.

Также заметим, что операции, указанные выше, затрагивают только элементы вне главной диагонали, соответствующие, по крайней мере, одному положению. Поэтому, в результирующей матрице все элементы, лежащие между признаками, равны нулю.

Изолированные многомерные гауссианы. Пусть распределение вероятности $p(x, y)$ по случайным векторам x и y будет гауссовым и выраженным в информационном виде

$$\Omega = \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \quad \text{и} \quad \xi = \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}$$

Если Ω_{yy} обратима, изолированный гауссиан $p(x)$, имеет следующий информационный вид

$$\bar{\Omega}_{xx} = \Omega_{xx} - \Omega_{xy}\Omega_{yy}^{-1}\Omega_{yx} \quad \text{и} \quad \bar{\xi}_x = \xi_x - \Omega_{xy}\Omega_{yy}^{-1}\xi_y$$

Доказательство. Изолированный гауссиан, выраженный моментами

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix} \quad \text{и} \quad \mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$$

равен $\mathcal{N}(\mu_x, \Sigma_{xx})$. По определению, информационная матрица гауссиана равна Σ_{xx}^{-1}, а информационный вектор $\Sigma_{xx}^{-1}\mu_x$. Покажем, что $\Sigma_{xx}^{-1} = \bar{\Omega}_{xx}$ с помощью формулы Вудбери из Таблицы 3.2 на странице 52. Пусть $P = (0\ 1)^T$, и $[\infty]$ матрицы такого же размера, как и Ω_{yy} но со всеми элементами, равными бесконечности (при $[\infty]^{-1} = 0$). Это даёт

$$(\Omega + P[\infty]P^T)^{-1} = \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & [\infty] \end{pmatrix}^{-1} \overset{(*)}{=} \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

То же выражение можно раскрыть с помощью формулы Вудбери следующим образом:

$$\begin{aligned}
&(\Omega + P[\infty]P^T)^{-1} \\
&= \Omega - \Omega P([\infty]^{-1} + P^T\Omega P)^{-1}P^T\Omega \\
&= \Omega - \Omega P(0 + P^T\Omega P)^{-1}P^T\Omega \\
&= \Omega - \Omega P(\Omega_{yy})^{-1}P^T\Omega \\
&= \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & \Omega_{yy}^{-1} \end{pmatrix} \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \\
&\overset{(*)}{=} \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} 0 & \Omega_{xy}\Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \\
&= \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} - \begin{pmatrix} \Omega_{xy}\Omega_{yy}^{-1}\Omega_{yx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} = \begin{pmatrix} \bar{\Omega}_{xx} & 0 \\ 0 & 0 \end{pmatrix}
\end{aligned}$$

Оставшееся выражение, $\Sigma_{xx}^{-1}\mu_x = \bar{\xi}_x$, получается аналогично, используя факт того, что $\mu = \Omega^{-1}\xi$ (см. Выражение (3.73)) и равенство двух выражений выше, обозначенных "$(*)$":

$$\begin{aligned}
\begin{pmatrix} \Sigma_{xx}^{-1}\mu_x \\ 0 \end{pmatrix} &= \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} = \begin{pmatrix} \Sigma_{xx}^{-1} & 0 \\ 0 & 0 \end{pmatrix} \Omega^{-1} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\
&\overset{(*)}{=} \left[\Omega - \begin{pmatrix} 0 & \Omega_{xy}\Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \Omega\right] \Omega^{-1} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\
&= \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} - \begin{pmatrix} 0 & \Omega_{xy}\Omega_{yy}^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} = \begin{pmatrix} \bar{\xi}_x \\ 0 \end{pmatrix}
\end{aligned}$$

Таблица 11.6 Лемма изоляции гауссианов в информационном виде. Форма ковариации Ω_{xx} также известна как *дополнение Шура*.

Условные вероятности многомерного гауссиана. Пусть распределение вероятности $p(x, y)$ по случайным векторам x и y выражено гауссианом в информационном виде

$\Omega = \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix}$ и $\xi = \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}$

Условная вероятность $p(x|y)$ - это гауссиан с информационной матрицей Ω_{xx} и информационным вектором $\xi_x - \Omega_{xy}y$.

Доказательство. Результат очевидно следует из определения гауссиана в информационном виде:

$$
\begin{aligned}
& p(x \,|\, y) \\
& = \eta \exp\left\{ -\tfrac{1}{2} \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \right\} \\
& = \eta \exp\{ -\tfrac{1}{2} x^T \Omega_{xx} x - x^T \Omega_{xy} y - \tfrac{1}{2} y^T \Omega_{yy} y + x^T \xi_x + y^T \xi_y \} \\
& = \eta \exp\{ -\tfrac{1}{2} x^T \Omega_{xx} x + x^T (\xi_x - \Omega_{xy} y) \underbrace{- \frac{1}{2} y^T \Omega_{yy} y + y^T \xi_y}_{\text{const.}} \} \\
& = \eta \exp\{ -\tfrac{1}{2} x^T \Omega_{xx} x + x^T (\xi_x - \Omega_{xy} y) \}
\end{aligned}
$$

Таблица 11.7 Лемма приведения гауссианов к информационному виду.

11.4.5 Уменьшение размерности информационного представления

Шаг уменьшения размерности в **GraphSLAM_reduce** основан на факторизации полной апостериорной вероятности SLAM.

(11.26)

$$p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) p(m|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$$

Здесь $p(x_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \sim \mathcal{N}(\xi, \Omega)$ апостериорная вероятность только по траекториям, с вычитанием карты:

(11.27)

$$p(x_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) = \int p(y_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) dm$$

как вкратце будет показано, эта вероятность действительно вычисляется алгоритмом **GraphSLAM_reduce** в Таблице 11.3, поскольку

(11.28)

$$p(x_{0:t}|z_{1:t}, u_{1:t}, c_{1:t}) \sim \mathcal{N}(\bar{\xi}, \bar{\Omega})$$

В общем случае, интеграция в (11.27) будет затруднительна из-за большого количества переменных карты m. Для гауссианов этот интеграл может быть вычислен в закрытом виде. Ключевая идея приведена в Таблице 11.6, где приводится и доказывается *лемма изоляции* для гауссовых функций.

Разделим матрицу Ω и вектор ξ на субматрицы, для пути робота $x_{0:t}$ и карты m:

(11.29)

$$\Omega = \begin{pmatrix} \Omega_{x_{0:t}, x_{0:t}} & \Omega_{x_{0:t}, m} \\ \Omega_{m, x_{0:t}} & \Omega_{m,m} \end{pmatrix}$$

(11.30)
$$\xi = \begin{pmatrix} \xi_{x_{0:t}} \\ \xi_m \end{pmatrix}$$

Согласно лемме о изоляции, вероятность (11.28) получается в виде

(11.31)
$$\bar{\Omega} = \Omega_{x_{0:t},x_{0:t}} - \Omega_{x_{0:t},m}\Omega_{m,m}^{-1}\Omega_{m,x_{0:t}}$$

(11.32)
$$\bar{\xi} = \xi_{x_{0:t}} - \Omega_{x_{0:t},m}\Omega_{m,m}^{-1}\xi_m$$

Матрица $\Omega_{m,m}$ блочно-диагональная. Это следует из способа создания матрицы Ω, в частности, отсутствия любых связей между парами признаков. Это делает инверсию эффективной:

(11.33)
$$\Omega_{m,m}^{-1} = \sum_j F_j^T \, \Omega_{j,j}^{-1} \, F_j$$

где $\Omega_{j,j} = f_j \Omega F_j^T$ субматрица Ω, соответствующая j-му признаку на карте:

(11.34)
$$F_j = \begin{pmatrix} 0...0 & 1\ 0\ 0 & 0...0 \\ 0...0 & 0\ 1\ 0 & 0...0 \\ 0...0 & \underbrace{0\ 0\ 1}_{\text{j-й признак}} & 0...0 \end{pmatrix}$$

Это наблюдение делает возможным разбиение уравнений реализации (11.31) и (11.32) с последовательным обновлением:

(11.35)
$$\bar{\Omega} = \Omega_{x_{0:t},x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\Omega_{j,j}^{-1}\Omega_{j,x_{0:t}}$$

(11.36)
$$\bar{\xi} = \xi_{x_{0:t}} - \sum_j \Omega_{x_{0:t},j}\Omega_{j,j}^{-1}\xi_j$$

Матрица $\Omega_{x_{0:t},j}$ ненулевая только по элементам в $\tau(j)$, набору положений, из которых наблюдается признак j. Это, в целом, доказывает верность алгоритма уменьшения размерности **GraphSLAM_reduce** в Таблице 11.3. Операцию, выполняемую над Ω в этом алгоритме, можно рассматривать как алгоритм вычёркивания переменных для инверсии матриц, применяемых только для переменных признаков, не затрагивая переменные положения.

11.4.6 Восстановление пути и карты

Алгоритм **GraphSLAM_solve** в Таблице 11.4 математическое ожидание и дисперсию гауссиана $\mathcal{N}(\bar{\xi}, \bar{\Omega})$, используя стандартные уравнения (3.72) и (3.73), приведённые на странице 74:

(11.37)
$$\bar{\Sigma} = \bar{\Omega}^{-1}$$

(11.38)
$$\bar{\mu} = \bar{\Sigma}\,\bar{\xi}$$

В частности, эта операция даёт математическое ожидание апостериорной вероятности по пути робота, но не даёт местоположений признаков на карте.

Остаётся восстановить второй множитель в выражении (11.26):

(11.39)
$$p(m|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$$

Лемма приведения, описанная и доказанная в Таблице 11.7, показывает, что вероятностное распределение является гауссовым с параметрами

(11.40)
$$\Sigma_m = \Omega^{-1}_{m,m}$$

(11.41)
$$\mu_m = \Sigma_m(\xi_m + \Omega_{m,x_{0:t}}\bar{\xi})$$

Здесь ξ_m и $\Omega_{m,m}$ - субвекторы ξ и субматрица Ω, соответственно, ограниченные переменными карты. Матрица $\Omega_{m,x_{0:t}}$ - это внедиагональная субматрица Ω соединяющая путь робота с картой. Как отмечалось выше, $\Omega_{m,m}$ является блочно-диагональной, поэтому ее можно разложить в виде

(11.42)
$$p(m|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t}) = \prod_j p(m_j|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$$

где каждая вероятность $p(m_j|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$ распределена в соответствии с

(11.43)
$$\Sigma_j = \Omega^{-1}_{j,j}$$

(11.44)
$$\mu_j = \Sigma_j(\xi_j + \Omega_{j,x_{0:t}}\bar{\mu}) = \Sigma_j(\xi_j + \Omega_{j,\tau(j)}\bar{\mu}_{\tau(j)})$$

Последнее преобразование основано на том факте, что субматрица $\Omega_{j,x_{0:t}}$ нулевая, за исключением переменных положения $\tau(j)$, из которых наблюдается j-ый признак.

Важно заметить, что эта вероятность задана гауссианом $p(m|x_{0:t}, z_{1:t}, u_{1:t}, c_{1:t})$, приведённым к истинной траектории $x_{0:t}$. На практике траектория пути неизвестна, поэтому хотелось бы знать апостериорную вероятность $p(m|z_{1:t}, u_{1:t}, c_{1:t})$ без учёта пути в наборе переменных для преобразования. Этот гауссиан невозможно факторизовать в представлении в виде моментов, поскольку местоположения различных признаков коррелируются через величину неопределённости по положению робота. По этой причине **GraphSLAM_solve** возвращает среднюю оценку апостериорной вероятности, но только ковариацию по пути робота. К счастью, для представления в виде моментов никогда не требуется полный гауссиан, вычисление которого может создать полностью заполненную матрицу ковариации огромных размеров, поскольку на все основные вопросы задачи SLAM можно приближённо ответить, не зная точного значения Σ.

11.5 Ассоциация данных в GraphSLAM

Ассоциация данных в GraphSLAM реализуется с помощью переменных соответствия, так же как в EKF SLAM. GraphSLAM ищет единственный вектор лучшего соответствия, вместо вычисления всего распределения по соответствиям. Нахождение вектора соответствия является задачей поиска. Однако, будет удобно определить соответствия в GraphSLAM несколько иначе: соответствия определяются по парам признаков на карте, а не по ассоциациям измерений и признаков. В частности, $c(j,k) = 1$, если m_j и m_k относятся к одному и тому же физическому признаку окружающего мира. В противном случае, $c(j,k) = 0$. Это соответствие между признаками, логически эквивалентно соответствию, определённому в предыдущем разделе, но проще для решения в виде алгоритма.

Используется жадный метод поиска в пространстве зависимостей, так же как в EKF. Каждый шаг поиска значения наилучшего соответствия приводит к улучшению измеренному подходящей функцией логарифма правдоподобия. Однако, поскольку GraphSLAM имеет доступ ко всем данным одновременно, возможно использовать значительно более мощные методы соответствия, чем инкрементальный подход в EKF. В частности:

1. В любой точке поиска GraphSLAM может учитывать соответствие любого набора признаков. Требования последовательной обработки наблюдаемых признаков нет.

2. Поиск соответствия можно комбинировать с вычислением карты. Допущение о том, что два наблюдаемых признака соответствуют одному и тому же физическому признаку в окружающем мире влияет на результирующую карту. Внедряя такую гипотезу соответствия в карту, другие гипотезы соответствия будут более или менее вероятны.

3. Решения ассоциации данных в GraphSLAM могут быть отменены. Качество ассоциации данных зависит от значения других решений по ассоциации данных, поэтому решение, казавшееся хорошим на ранних этапах поиска, на более поздних этапах может оказаться неудачным. Чтобы разрешить такую ситуацию, в GraphSLAM можно эффективно отменять предыдущие решения ассоциации данных.

Опишем один из алгоритмов поиска соответствия, который использует первые два свойства, но не использует третье. Алгоритм ассоциации данных все ещё останется жадным и будет выполнять последовательный поиск в пространстве возможных соответствий, чтобы найти непротиворечивую карту. Однако, как все жадные алгоритмы, он подвержен проблеме локальных максимумов, поскольку полное пространство соответствий экспоненциально зависит от числа признаков на карте. В любом случае, пока ограничимся алгоритмом поиска экстремумов, а полное изложение отложим до следующей главы.

11.5.1 Алгоритм GraphSLAM для неизвестного соответствия

ПРОВЕРКА ПРАВДОПОДОБИЯ СООТВЕТСТВИЯ

Ключевым компонентом алгоритма является проверка правдоподобия соответствия. Алгоритм GraphSLAM проверяет соответствия очень простым способом. Какова вероятность того, что два разных признака на карте, m_j и m_k, соответствуют одному и тому же физическому признаку окружающего мира? Если эта вероятность выше порогового значения, необходимо

соединить вместе эти признаки на единой карте.

1: **Algorithm GraphSLAM_correspondence_test**$(\Omega, \xi, \mu, \Sigma_{0:t}, j, k)$:
2: $\quad \Omega_{[j,k]} = \Omega_{jk,jk} - \Omega_{jk,\tau(j,k)} \Sigma_{\tau(j,k),\tau(j,k)} \Omega_{\tau(j,k),jk}$
3: $\quad \xi_{[j,k]} = \Omega_{[j,k]} \mu_{j,k}$
4: $\quad \Omega_{\Delta j,k} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$
5: $\quad \xi_{\Delta j,k} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \xi_{[j,k]}$
6: $\quad \mu_{\Delta j,k} = \Omega^{-1}_{\Delta j,k} \xi_{\Delta j,k}$
7: $\quad$ *return* $|2\pi \Omega^{-1}_{\Delta j,k}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} \mu^T_{\Delta j,k} \Omega^{-1}_{\Delta j,k} \mu_{\Delta j,k}\right\}$

Таблица 11.8 Проверка соответствия в GraphSLAM: на вход поступает информационное представление апостериорной вероятности SLAM вместе с результатом такта **GraphSLAM_solve**. На выходе апостериорная вероятность соответствия m_j и m_k.

Алгоритм проверки соответствия приводится в Таблице 11.8. Проверку проходят два признака с индексами j и k, для которых вычисляются вероятность того, что они соответствуют одному и тому же признаку физического мира. Для вычисления этой вероятности в алгоритме используется ряд значений: информационное представление апостериорной вероятности SLAM в виде Ω, ξ и результата выполнения процедуры **GraphSLAM_solve**, который будет выражен математическим ожиданием μ и ковариацией пути $\Sigma_{0:t}$.

Проверка соответствия происходит следующим образом: во-первых, вычисляется изолированная апостериорная вероятность по двум целевым признакам. Она выражена информационной матрицей $\Omega_{[j,k]}$ и вектором $\xi_{[j,k]}$, вычисленными в строках 2 и 3 в Таблице 11.8. На этом этапе вычислений используются различные субэлементы информационного представления Ω, ξ, средние местоположения признаков, определённые через μ, и ковариация пути $\Sigma_{0:t}$. Далее, вычисляются параметры новой случайной гауссовой переменной, значение которой выражает разницу между m_j и m_k. Обозначив переменную разницы как $\Delta_{j,k} = m_j - m_k$, информационные параметры $\Omega_{\Delta j,k}$, $\xi_{\Delta j,k}$ вычисляются в строках 4 и 5, а соответствующие ожидания разницы вычисляются в строке 6. В строке 7 возвращается вероятность нулевой разницы между m_j и m_k.

1: **Algorithm GraphSLAM**$(u_{1:t}, z_{1:t})$:
2: *инициализировать все* c_t^i *уникальным значением*
3: $\mu_{0:t} =$ **GraphSLAM_initialize**$(u_{1:t})$
4: $\Omega, \xi =$ **GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$
5: $\bar{\Omega}, \bar{\xi} =$ **GraphSLAM_reduce**(Ω, ξ)
6: $\mu, \Sigma_{0:t} =$ **GraphSLAM_solve**$(\bar{\Omega}, \bar{\xi}, \Omega, \xi)$
7: *повторять*
8: *для каждой пары несвязанных признаков* m_j, m_k *выполнить*
9: $\pi_{j=k} =$ **GraphSLAM_correspondence_test**$(\Omega, \xi, \mu, \Sigma_{0:t}, j, k)$
10: *if* $\pi_{j=k} > \mathcal{X}$ *then*
11: *for all* $c_t^i = k$ *set* $c_t^i = j$
12: $\Omega, \xi =$ **GraphSLAM_linearize**$(u_{1:t}, z_{1:t}, c_{1:t}, \mu_{0:t})$
13: $\bar{\Omega}, \bar{\xi} =$ **GraphSLAM_reduce**(Ω, ξ)
14: $\mu, \Sigma_{0:t} =$ **GraphSLAM_solve**$(\bar{\Omega}, \bar{\xi}, \Omega, \xi)$
15: *endif*
16: *endfor*
17: *пока не останется пар* m_j, m_k *для которых* $\pi_{j=k} < \mathcal{X}$
18: *return* μ

Таблица 11.9 Алгоритм GraphSLAM для полной проблемы SLAM с неизвестным соответствием. Эффективность внутреннего цикла алгоритма можно увеличить выборочной проверкой пар признаков m_j, m_k, и сохранением нескольких соответствий перед решением результирующего свёрнутого набора уравнений.

Проверка соответствия даёт алгоритм выполнения поиска ассоциации данных в GraphSLAM, который показан в Таблице 11.9. В нём переменные соответствия инициализированы уникальными значениями. Четыре последующих шага (в строках 3-7) одинаковы с алгоритмом GraphSLAM с известным соответствием, приведённом в Таблице 11.5. Однако в этом общем алгоритме SLAM выполняется и поиск ассоциации данных. Для каждой пары различных признаков на карте вычисляется вероятность соответствия (строка 9 в Таблице 11.9). Если вероятность превышает порог $\mathcal{X}$, значение векторов соответствия устанавливается одинаковым (строка 11).

Алгоритм GraphSLAM последовательно выполняет такты создания матриц, уменьшение размерности, и нахождения решения для апостериорной вероятности SLAM (строки с 12 по 14). В результате, последующие проверки соответствия влияют на предыдущие решения о соответствии на заново создаваемой карте. Создание карты прерывается, когда во внутреннем цикле больше не находится признаков.

Строго говоря, алгоритм **GraphSLAM** не особенно эффективен. В частности, проверяются на соответствие все пары признаков, не только близлежащие. Далее, карта восстанавливается при нахождении каждого соответствия, вместо пакетной обработки наборов данных. Такие модификации довольно очевидны. Хорошая реализация GraphSLAM будет намного лучше по сравнению с обсуждаемым общим алгоритмом.

11.5.2 Математический вывод проверки соответствия

Ограничим вывод показом правильности проверки соответствия в Таблице 11.8. Первой целью будет определение апостериорного вероятностного распределения по переменной $\Delta_{j,k} = m_j - m_k$, разницы между местоположением признака m_j и признака m_k. Два признака m_j и m_k эквивалентны тогда и только тогда, когда их местоположение одинаково. Поэтому, вычисляя апостериорную вероятность $\Delta_{j,k}$, можно получить искомую вероятность соответствия.

Получим апостериорную вероятность $\Delta_{j,k}$ вычислением общей вероятности по m_j и m_k:

(11.45)

$$p(m_j, m_k | z_{1:t}, u_{1:t}, c_{1:t}) = \int p(m_j, m_k | x_{1:t}, z_{1:t}, c_{1:t}) p(x_{1:t} | z_{1:t}, u_{1:t}, c_{1:t}) dx_{1:t}$$

Определим информационный вид изолированной апостериорной вероятности в виде $\xi_{[j,k]}$ и $\Omega_{[j,k]}$. Обратите внимание на использование квадратных скобок, разделяющих значения и субматрицы объединённого информационного вида.

Распределение (11.45) получается из объединённой апостериорной вероятности $y_{0:t}$ путём применения леммы об изоляции. Пусть Ω и ξ отображают общую апостериорную вероятность по всему вектору состояний $y_{0:t}$ в информационном виде, а $\tau(j)$ и $\tau(k)$ определяют наборы положений, из которых робот наблюдает признаки j и k, соответственно. GraphSLAM дает средний вектор положений $\bar{\mu}$. Для использования леммы об изоляции (Таблица 11.6), необходимо оптимизировать результат алгоритма **GraphSLAM_solve**. В частности, **GraphSLAM_solve** уже предоставляет среднее для признаков m_j и m_k. Переопределим вычисления для объединённой пары признаков:

(11.46)

$$\mu_{[j,k]} = \Omega^{-1}_{jk,jk}(\xi_{jk} + \Omega_{jk,\tau(j,k)}\mu_{\tau(j,k)})$$

Здесь $\tau(j,k) = \tau(j) \cup \tau(k)$ определяет набор положений, в котором робот наблюдает m_j или m_k.

Для объединённой апостериорной вероятности необходимо знать ковариацию. Эта ковариация не вычисляется в **GraphSLAM_solve**, просто потому, что объединённая ковариация по нескольким признакам требует пространства, квадратичному по числу признаков. Однако, для пар признаков объединённая ковариация легко восстанавливается.

Пусть $\Sigma_{\tau(j,k),\tau(j,k)}$ будет субматрицей ковариации $\Sigma_{0:t}$ ограниченной для всех положений $\tau_{j,k}$. Здесь ковариация $\Sigma_{0:t}$ вычисляется в строке 2 алгоритма **GraphSLAM_solve**. Затем, лемма об изоляции даст изолированную информационную матрицу апостериорной вероятности по $(m_j \; m_k)^T$:

(11.47)

$$\Omega_{[j,k]} = \Omega_{jk,jk} - \Omega_{jk,\tau(j,k)}\Sigma_{\tau(j,k),\tau(j,k)}\Omega_{\tau(j,k),jk}$$

Отображение в информационном виде для искомой апостериорной вероятности дополненным следующим информационным вектором:

(11.48)

$$\xi_{[j,k]} = \Omega_{[j,k]}\mu_{[j,k]}$$

отсюда получаем объединённую вероятность

(11.49)

$$p(m_j, m_k|z_{1:t}, u_{1:t}, c_{1:t}) = \eta \exp\left\{-\frac{1}{2}\begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} m_j \\ m_k \end{pmatrix} + \begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \xi_{[j,k]}\right\}$$

Эти уравнения идентичны строкам 2 и 3 в Таблице 11.8.

Преимуществом такого выражения является возможность немедленного определения искомой вероятности соответствия. Для этого введём случайную переменную

(11.50)

$$\begin{aligned} \Delta_{j,k} &= m_j - m_k \\ &= \begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \begin{pmatrix} m_j \\ m_k \end{pmatrix} \\ &= \begin{pmatrix} m_j \\ m_k \end{pmatrix}^T \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{aligned}$$

Вставляя это выражение в гауссиан в информационном виде, получаем:

(11.51)

$$\begin{aligned} &p(\Delta_{j,k}|z_{1:t}, u_{1:t}, c_{1:t}) \\ &= \eta \exp\left\{-\frac{1}{2}\Delta_{j,k}^T \underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \Omega_{[j,k]} \begin{pmatrix} 1 \\ -1 \end{pmatrix}}_{=:\Omega_{\Delta j,k}} \Delta_{j,k} + \Delta_{j,k}^T \underbrace{\begin{pmatrix} 1 \\ -1 \end{pmatrix}^T \xi_{[j,k]}}_{=:\xi_{\Delta j.k}}\right\} \\ &= \eta \exp\{-\frac{1}{2}\Delta_{j,k}^T \Omega_{\Delta j,k} + \Delta_{j,k}^T \xi_{\Delta j,k}\}^T \end{aligned}$$

То есть гауссову функцию в виде информационной матрицы $\Omega_{\Delta j,k}$ и информационного вектора $\xi_{\Delta j,k}$ как определено выше. Для вычисления приближенного значения $\Delta_{j,k} = 0$ гауссианом будет полезно переписать его в виде моментов:

(11.52)

$$p(\Delta_{j,k}|z_{1:t}, u_{1:t}, c_{1:t}) = |2\pi\Omega_{\Delta j,k}^{-1}|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(\Delta_{j,k} - \mu_{\Delta j,k})^T \Omega_{\Delta j,k}^{-1}(\Delta_{j,k} - \mu_{\Delta j,k})\}$$

где математическое ожидание задано очевидным выражением:

(11.53)

$$\mu_{\Delta j,k} = \Omega_{\Delta j,k}^{-1}\xi_{\Delta j,k}$$

Эти шаги приведены в строках с 4 по 6 в Таблице 11.8.

Искомая вероятность для $\Delta_{j,k} = 0$ является результатом ставки нулевых значений в распределение и считывания результирующей вероятности:

(11.54)

$$p(\Delta_{j,k} = 0 | z_{1:t}, u_{1:t}, c_{1:t}) = |2\pi\Omega^{-1}_{\Delta j,k}|^{-\frac{1}{2}} \exp\{-\frac{1}{2}\mu^T_{\Delta j,k}\Omega^{-1}_{\Delta j,k}\mu_{\Delta j,k}\}$$

Этой вероятностью выражается возможность того, что m_j и m_k соответствуют одним и тем же признакам на карте. Это вычисление реализовано в строке 7 в Таблице 11.8.

11.6 Соображения эффективности

Практические реализации GraphSLAM основаны на некоторых дополнительных идеях и методах увеличения эффективности. Возможно, самым большим недостатком GraphSLAM, как уже обсуждалось, является начальное допущение о том, что разным наблюдаемым признакам соответствуют разные физические признаки. Предложенный алгоритм объединяет их по одному, и для любого разумного количества признака он будет недопустимо медленным. Вдобавок, в нем не учитывается важное ограничение, что в любой момент времени, один и тот же признак может наблюдаться лишь единожды.

В существующих реализациях идеи GraphSLAM используются такие возможности.

ЛОКАЛЬНЫЕ КАРТЫ

Признаки, которые с высокой вероятностью определяются как соответствующие, часто объединяются с самого начала, ещё до запуска полного решения GraphSLAM. Например, довольно распространено объединение коротких сегментов в *локальные карты*, то есть локальные карты сетки занятости. Вывод GraphSLAM выполняется только между локальными картами сетки занятости, где схожесть двух карт используется в качестве вероятностного ограничения между относительными положениями этих карт. Такой иерархический метод уменьшает сложность SLAM на порядки, при этом сохраняя некоторые ключевые элементы GraphSLAM, в частности, возможность выполнять ассоциацию данных на больших наборах данных.

Часто роботы оборудованы датчиками, которые обнаруживают множество признаков за один момент времени. Например, лазерные датчики расстояния воспринимают десятки признаков на одном проходе сканирования. Для любого такого прохода принято считать, что различные измерения действительно соответствуют разным признакам в среде, в силу того, что проходы сканирования направлены в различные точки. Это известно под названием принципа взаимного исключения, который уже обсуждался в Главе 7. Из него следует $i \neq j \longrightarrow c^i_t \neq c^j_t$, поэтому никакие два измерения, полученные в течение одного прохода сканирования, не могут соответствовать одному признаку окружающего мира.

Метод попарной ассоциации данных, описанный выше, неспособен учитывать это ограничение, поскольку может, скажем, назначить два измерения (z^i_t и z^j_t) некоторому признаку z^k_s для некоторого $s \neq t$. Чтобы избежать этой проблемы для одного и то же момента времени обычно назначают целые векторы измерений z_t и z_s. Это позволяет вычисление вероятности по всем признакам в z_t и z_s. Такой способ обобщает метод попарного вычисления и математически прямолинеен.

GraphSLAM, описанный в этой главе, не использует возможность отмены ассоциации данных. Как только выбор ассоциации данных сделан,

в процессе дальнейшего поиска его невозможно изменить. Математически, отмена решения об ассоциации данных в информационной структуре относительно очевидна. В приведённом выше алгоритме можно изменить значение любых двух переменных соответствия произвольным образом. Однако, труднее проверить, следует ли отметить ассоциацию данных, поскольку очевидного способа проверить различимость двух ранее связанных признаков нет. Простая реализация состоит из отмены проверяемой ассоциации данных, перестройки карты с последующей проверкой критерия соответствия. Такой метод может быть вычислительно сложен, так как не позволяет определить, какие именно ассоциации данных необходимо проверять. Механизмы обнаружения маловероятных связей данных выходят за рамки изложения книги, но будут вскользь упомянуты при описании реализации подхода.

Рис. 11.4 Робор «Groundhog» («Сурок») - 640 килограммовый самоходный робот, оборудованный бортовым компьютером, лазерными датчиками расстояния, датчиками газа и воды, а также системой видеофиксации. Робот был построен для картографирования заброшенных шахт)

ОТРИЦАТЕЛЬНАЯ ИНФОРМАЦИЯ

Наконец, алгоритм GraphSLAM не принимает в расчёт отрицательную информацию. На практике отсутствие признака может быть столь же информативным, как и его наблюдение. Однако, такая простая формулировка не включает необходимых геометрических вычислений.

В практических задачах возможность использовать отрицательную информацию зависит от модели датчика и модель признаков среды. Например, может появиться необходимость вычислить вероятность сцепления признаков, что будет затруднительно для определённых типов датчиков, например, расстояния и направления. Современные реализации, конечно же, учитывают отрицательную информацию, но часто лишь путём замещения чистых вероятностных вычислений приближениями. Один такой пример будет приведён в следующем разделе.

11.7 Практическая реализация

Обсудим практические реализации GraphSLAM. Мобильный робот, используемый для экспериментов и показанный на Рис. 11.4, предназначен для составления карт заброшенных шахт.

Типовая карта, полученная роботом, показана на Рис. 11.5. Это карта сетки занятости, эффективно использующая попарное сравнение сканирований для восстановления положений робота. Попарное сравнение сканиро-

ваний можно воспринимать в виде GraphSLAM, но соответствие устанавливается только между двумя последовательными проходами сканирования. Результат использования очевидно показывает недостатки на карте, показанные на Рис. 11.5.

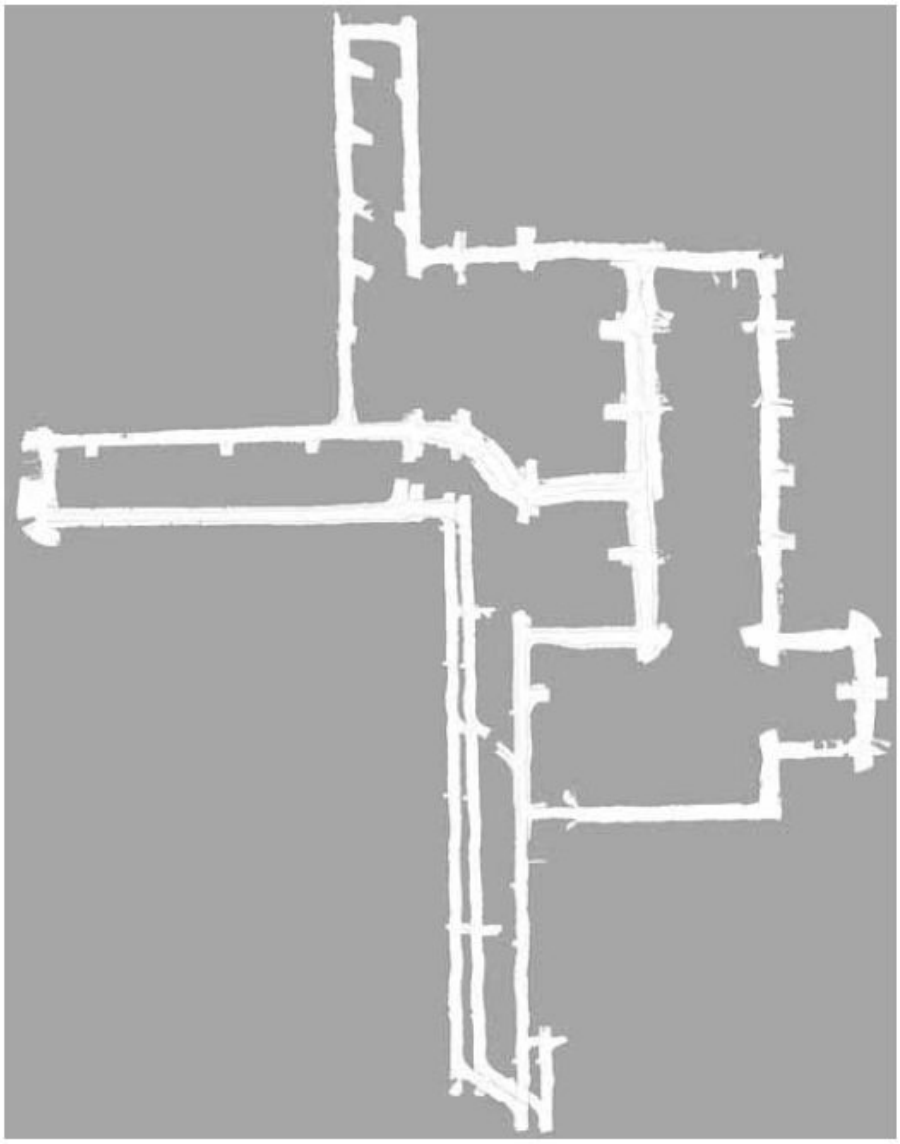


Рис. 11.5 Карта шахты, полученная попарным сравнением проходов сканирования. Поперечный размер составляет, приблизительно, 250 метров. Карта не целостная, некоторые проходы показаны несколько раз. Изображение принадлежит Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

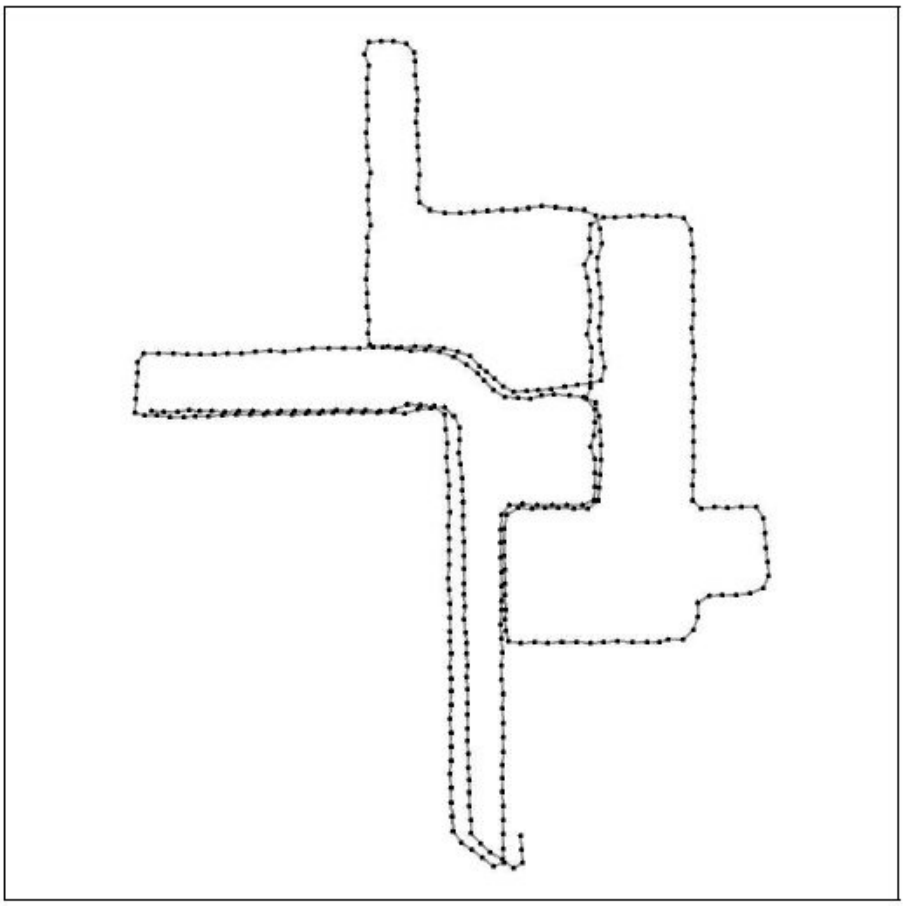


Рис. 11.6 Основа карты шахты для визуализации локальных карт.

Для использования алгоритма GraphSLAM в нашем программном обеспечении выполняется разбиение карты на мелкие локальные карты, по одной на каждые пять метров движения робота. На протяжении этих пяти метров карты имеют удовлетворительную точность, поскольку общий дрейф показаний мал, и сравнение проходов сканирования происходит, по большей части, гладко. Координаты каждой вспомогательной карты становятся узлом в GraphSLAM. Соседние вспомогательные карты соединяются с помощью относительных ограничений движения между ними. Результирующая структура показана на Рис. 11.6.

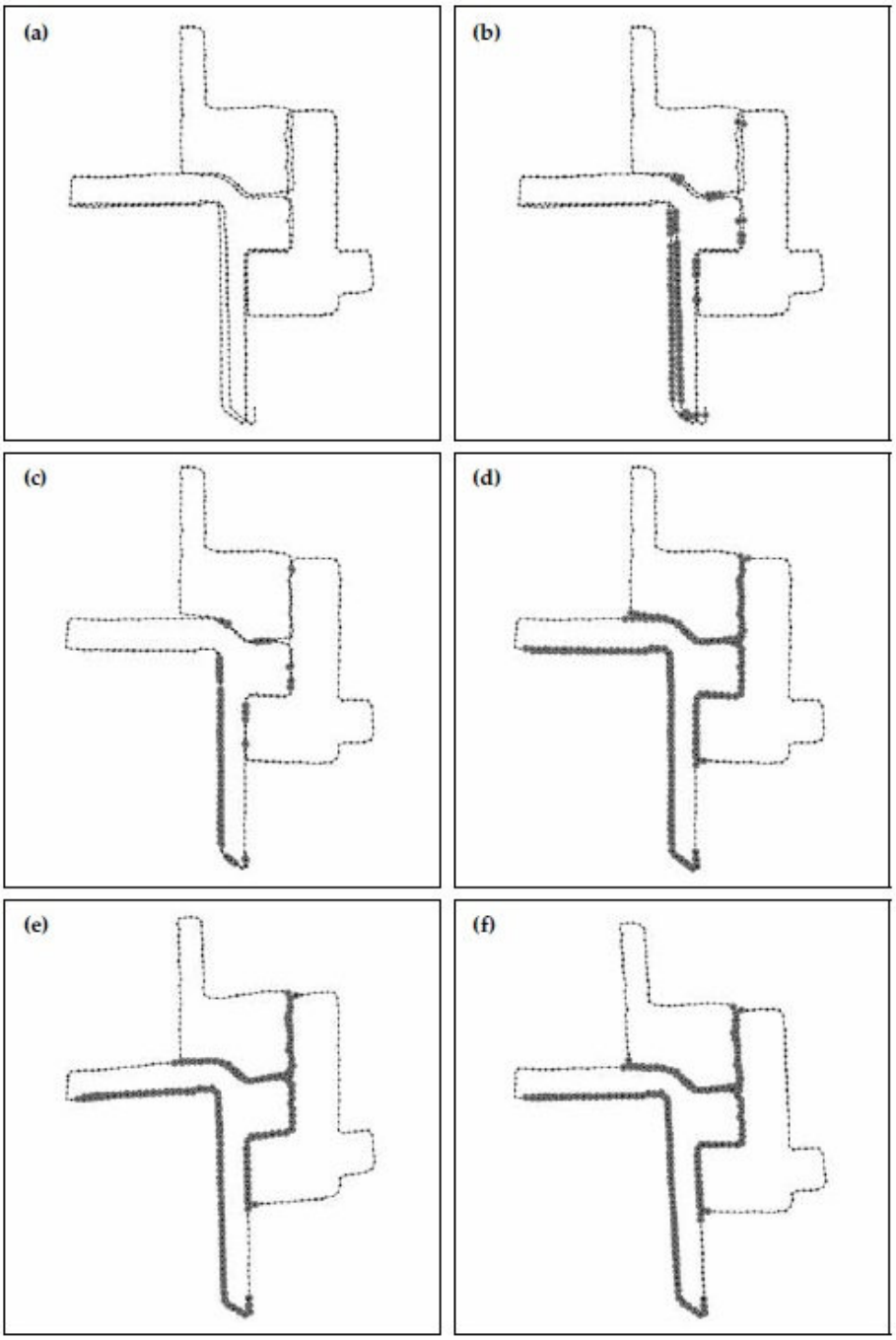


Рис. 11.7 Поиск ассоциации данных. Пояснения в тексте.

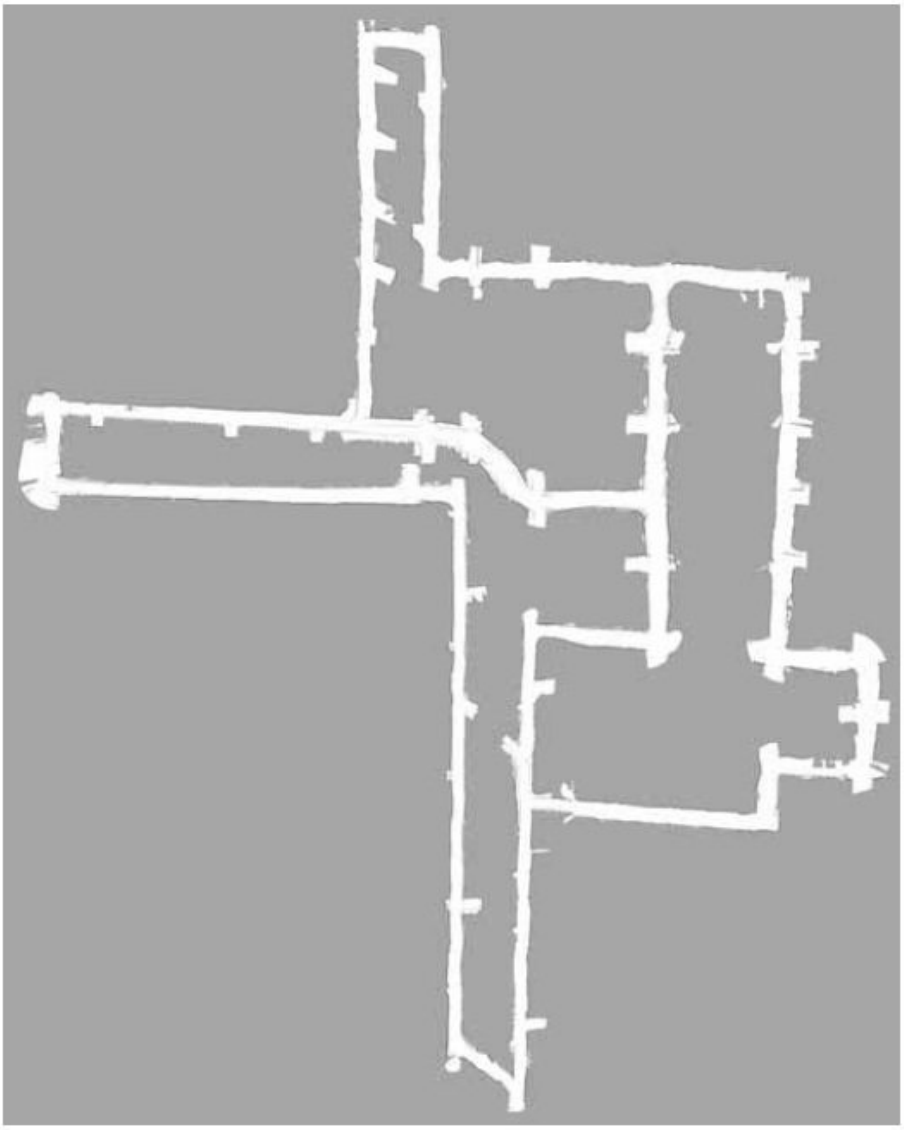


Рис. 11.8 Итоговая карта после оптимизации ассоциации данных. Изображение принадлежит Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

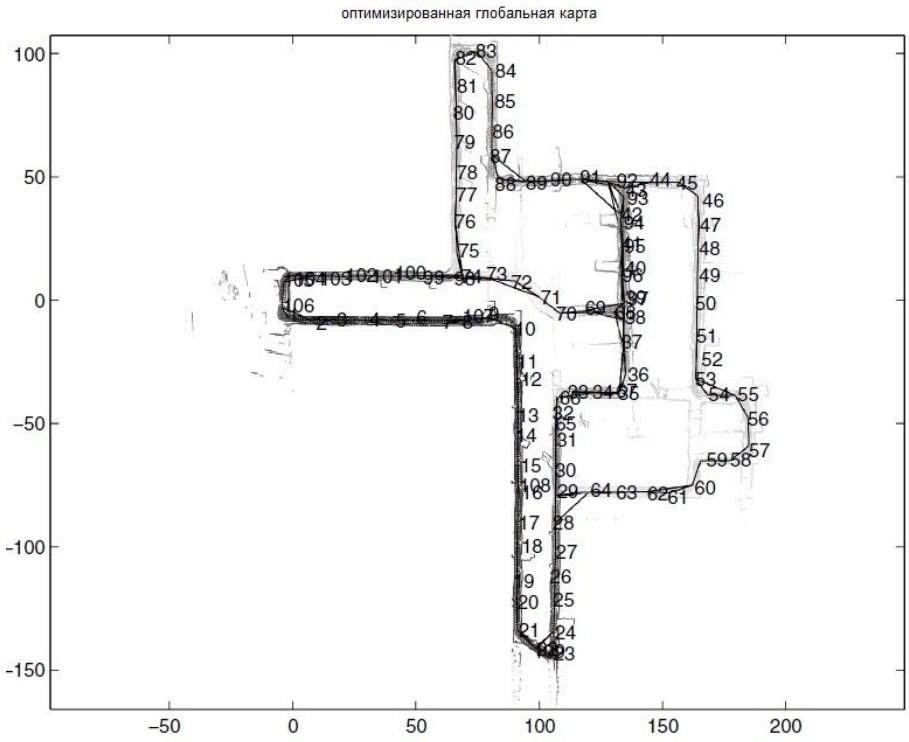


Рис. 11.9 Карта шахты, сгенерированная алгоритмом Atlas SLAM, созданным Боссе (Bosse et al., 2004). Изображение принадлежит Майклу Боссе, Полу Ньюману, Джону Леонарду и Сету Теллеру, МИТ (Michael Bosse, Paul Newman, John Leonard, Seth Teller, MIT).

Теперь применим рекурсивный поиск ассоциации данных. Проверка соответствия сейчас будет выполняться путём корреляционного анализа двух наложенных друг на друга карт и гауссовых ограничений совпадения, восстановленных аппроксимацией функции совпадения с помощью гауссовых функций. На Рис. 11.7 показан процесс ассоциации данных. Каждый круг соответствует новому ограничению, которое вызывается при конструировании информационного вида с помощью GraphSLAM. На этом рисунке хорошо видна последовательность поиска: некоторые соответствия обнаруживаются только после установления других, а некоторые – удаляются в процессе поиска. Конечная модель стабильна, когда при выполнении поиске новых ассоциаций данных не вносится дальнейших изменений. Двухмерная карта в виде сетки представлена на Рис. 11.8. Хотя она далека от идеальной, в основном из-за грубой реализации ограничений наложения локальных карт, результат намного лучше варианта, созданного инкрементным наложением карт.

Читатель может заметить,что другие информационно-теоретические методы SLAM дают сходные результаты. На Рис. 11.9 показана карта на основе того же набора данных, созданного Боссе (Bosse et al., 2004) с помощью алгоритма под названием Atlas. Этот алгоритм разбивает карты на субкарты, отношение между которыми сохраняется в виде относительных информационно-теоретических связей. Более детальная информация приводится в библиографических примечаниях.

11.8 Альтернативные методы оптимизации

Как читатель может припомнить, центральная целевая функция $J_{GraphSLAM}$ в алгоритме GraphSLAM - это нелинейная квадратичная функция из выражения (11.3). GraphSLAM минимизирует эту функцию путём последовательных линеаризаций, вычёркивания переменных и оптимизации. Метод предположений в **GraphSLAM_solve** в Таблице 11.4, в общем, не слишком эффективен. Если необходимо найти только карту и траекторию пути, без ковариаций, вычисления инверсии в строке 2 в Таблице 11.4 можно избежать. Получившаяся реализация будет намного более вычислительно эффективной.

Ключ к эффективной формулировке - в функции $J_{GraphSLAM}$. Эта функция в общем, представляет собой метод наименьших квадратов, а значит, может быть минимизирована с помощью различных алгоритмов, описанных в литературе. Примеры включают *методы градиентного спуска, Левенберга-Марквардта, и сопряжённых градиентов.*

СОПРЯЖЕННЫЕ ГРАДИЕНТЫ

На Рис. 11.10a показаны результаты, полученные с использованием сопряжённого градиента для минимизации $J_{GraphSLAM}$. Данные карты открытого пространства размером примерно 600 на 800 метров были собраны в кампусе Стенфордского университета роботом, показанным на Рис. 11.10b. На Рис. 11.11 показан процесс выравнивания набора данных, основанного только на информации о положениях робота, до построения полной карты и пути робота. В этом наборе данных содержится приблизительно 10^8 признаков и 10^5 положений. Запуск EKF на таком большом наборе данных вряд ли будет разумным, как и инверсия матрицы Ω в Таблице 11.4. Методу сопряжённых градиентов потребовалось всего несколько секунд для выполнения минимизации в $J_{GraphSLAM}$. Именно поэтому во многих современных реализациях используются различные способы оптимизации вместо относительно сложных алгоритмов, обсуждаемых в книге. Заинтересованный читатель может обратиться к библиографическим примечаниям с указанием источников, описывающих альтернативные методы оптимизации.

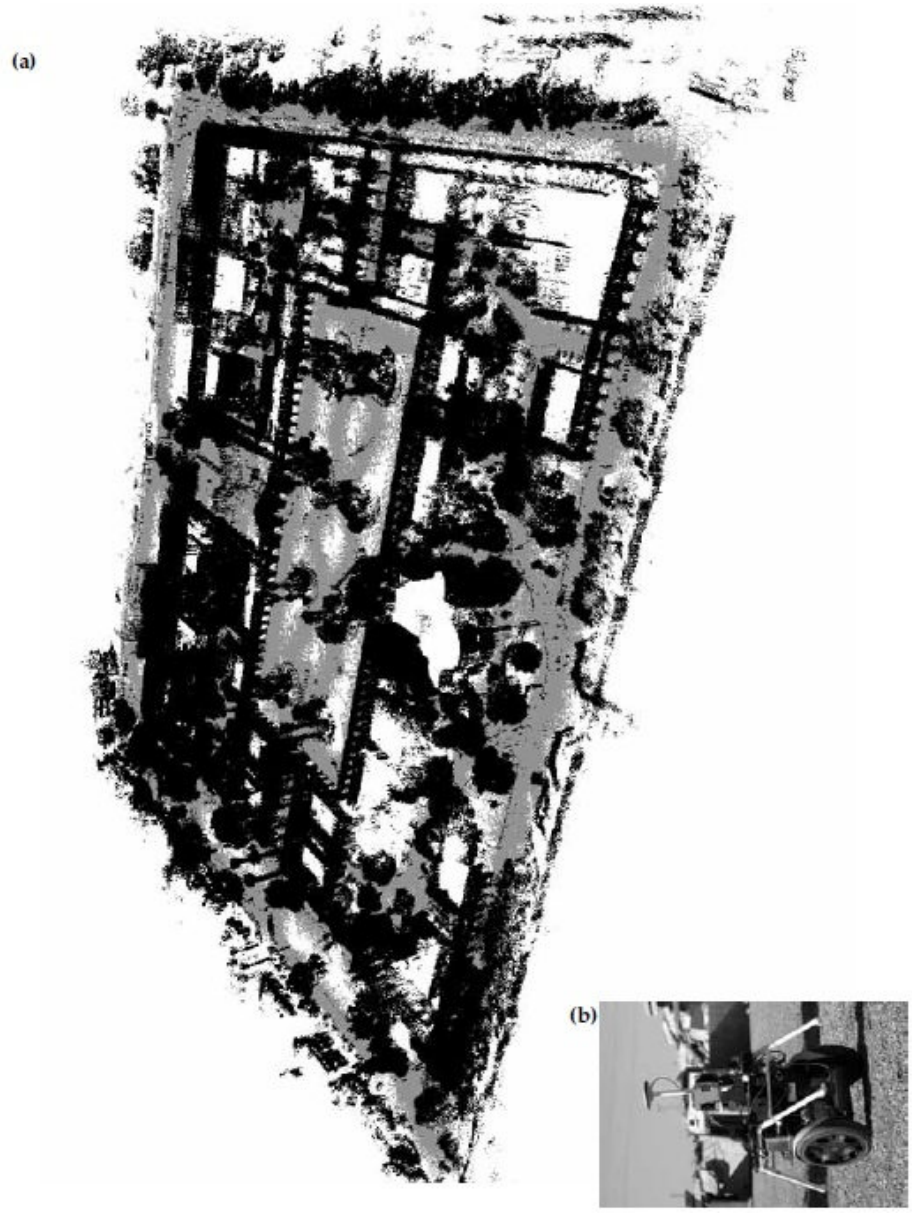


Рис. 11.10 Трёхмерная карта кампуса Стэнфорда(a). Робот, используемый для получения этих данных на шасси сегвея RMP, разработка была финансово поддержана программой DARPA MARS(b). Изображение принадлежит Майклу Монтемерло из университета Стэнфорда (Michael Montemerlo, Stanford University). (Страницу следует повернуть вправо.)

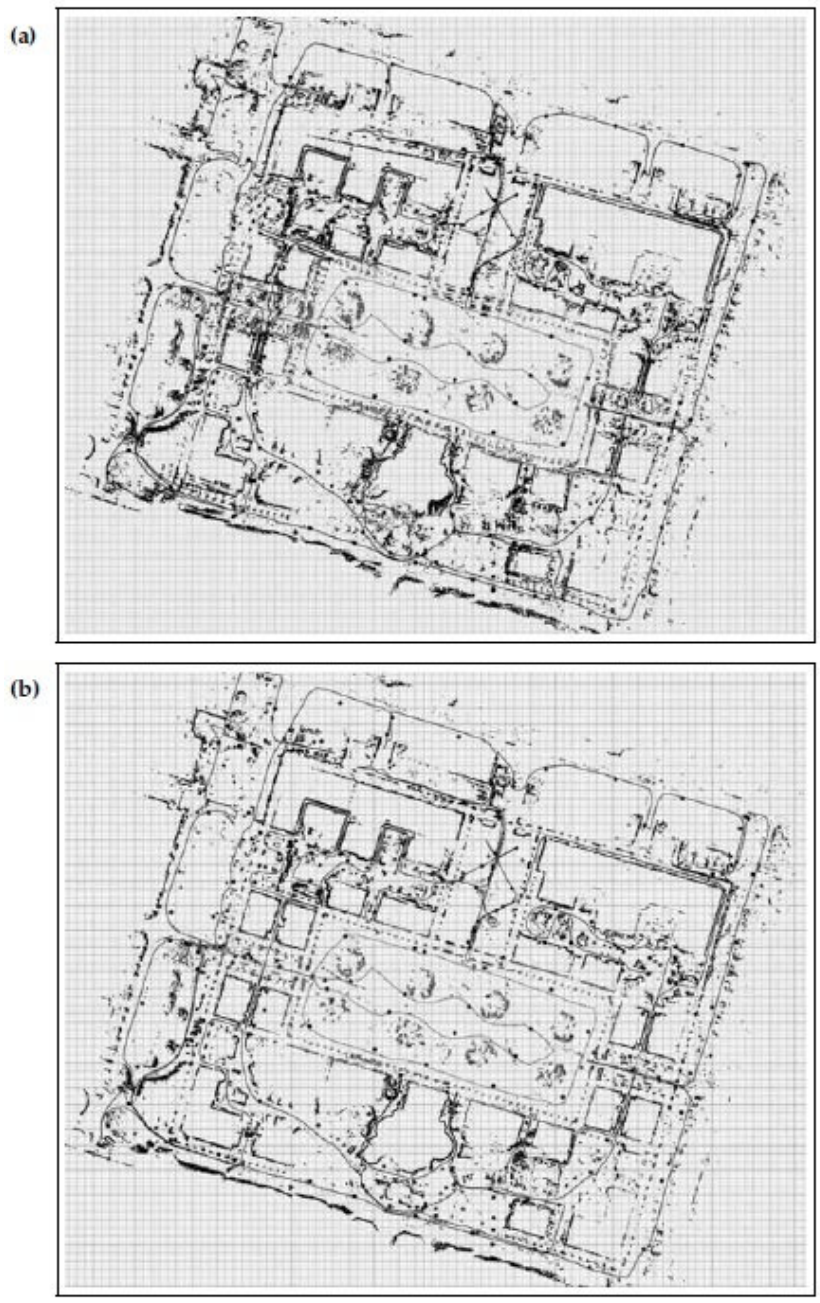


Рис. 11.11 Двухмерный срез карты кампуса Стэнфорда до (a) и после выравнивания с помощью метода сопряжённых градиентов (b). Такая оптимизация методом сопряжённых градиентов требует всего нескольких секунд, применительно к формулировке GraphSLAM методом наименьших квадратов. Изображение принадлежат Майклу Монтемерло, университет Стэнфорда (Michael Montemerlo, Stanford University)

11.9 Выводы

В этой главе описывается алгоритм GraphSLAM для полной задачи SLAM.

• Алгоритм GraphSLAM предназначен для решения полной задачи SLAM. Он вычисляет апостериорные вероятности по всему пути робота и по всей карте, поэтому, GraphSLAM является пакетным алгоритмом, а не онлайновым, как EKF SLAM.

• GraphSLAM строит граф мягких ограничений на основе набора данных. В частности, измерения проектируются в виде рёбер, представляющих собой нелинейные ограничения между положениями и обнаруженными признаками, а команды на движение представлены в виде мягких ограничений между соседними положениями. Граф изначально разрежен. Количество рёбер является линейной функцией количества узлов, и каждый узел соединён с конечным количеством других узлов независимо от размера графа.

GraphSLAM просто сохраняет всю информацию в графе с помощью связей, определённых между положениями и признаками, а также между парами последовательных положений. Однако, это информационное выражение не даст оценок карты или пути робота.

• Сумма всех ограничений задана функцией $J_{GraphSLAM}$. Оценки максимального правдоподобия для пути робота и карты могут быть получены минимизацией функции $J_{GraphSLAM}$.

• GraphSLAM выполняет проекцию графа в виде изоморфной информационной матрицы и информационного вектора, определённых по всем положениям пути и всей карте. Ключевой идеей алгоритма GraphSLAM является разрежённость структуры представления информации. Измерения дают информацию о признаке относительно положения робота в момент измерения. В информационном пространстве они образуют ограничение между парами переменных. Похожим образом, движение несёт информацию о взаимосвязи двух последовательных положений. В информационном пространстве каждая команда на движение образует ограничение между последовательными положениями. Эта разрежённость наследуется из разреженного графа.

• Чистый алгоритм GraphSLAM восстанавливает карты с помощью итеративной процедуры из трёх шагов: построение линейного информационного вида с помощью разложения в ряд Тейлора, уменьшения размера представления для извлечения карты, и решения результирующей проблемы оптимизации по положениям робота. Эти три шага эффективно разрешают информацию и создают целостное вероятностное апостериорное распределение по пути робота и карте. Поскольку GraphSLAM выполняется пакетно, шаг линеаризации можно повторить для улучшения результата.

• В альтернативных реализациях вывод выполняется с помощью оптимизации методом наименьших квадратов функции JGraphSLAM. Однако, такие методы позволяют найти только моду апостериорной вероятности, но не её ковариацию.

• Ассоциация данных в GraphSLAM выполняется вычислением вероятности того, что два признака имеют идентичные координаты в среде. Поскольку GraphSLAM – пакетный алгоритм, его можно выполнять для любой пары признаков и в любое время. Это приводит к итеративному жадному алгоритму поиска по всем переменным ассоциации данных, рекурсивно

определяющим пары признаков на карте с наибольшим значением соответствия.

• В практических реализациях GraphSLAM часто используются дополнительные приёмы уменьшения количества вычислений и предотвращения ошибочных ассоциаций данных. В частности, на практике стремятся уменьшить сложность данных, извлекая локальные карты и используя каждую карту в качестве базового элемента картографирования. В таких методах сравниваются несколько признаков за проход, и есть возможность, учитывается отрицательную информацию при выполнении поиска ассоциации данных.

• Были кратко представлены результаты GraphSLAM на основе декомпозиции, но использованием карты сеток занятости для представления наборов данных расстояния. Несмотря на эти приближения, было обнаружено, что методы ассоциации данных и вывода карт демонстрируют лучшие результаты в масштабных задачах картографирования.

• Также были представлены результаты для реализации метода сопряжённого градиента в задаче наименьших квадратов. Было замечено, что общая целевая функция GraphSLAM может быть оптимизирована методом наименьших квадратов. Некоторые методы, например, сопряжённый градиент, существенно быстрее по сравнению с базовым методом оптимизации в GraphSLAM.

Как было замечено во вводной части, EKF SLAM и GraphSLAM расположены на противоположных сторонах диапазона SLAM алгоритмов. Алгоритмы, которые попадают между этими крайними значениями, будут обсуждаться в следующих двух главах. Ссылки на такие методы находятся в библиографических примечаниях нескольких следующих глав.

11.10 Библиографические примечания

Методы графического представления хорошо известны в машинном зрении и фотограмметрии и относятся к структуре из движения и блочного выравнивания (Hartley and Zisserman 2000; B et al. 2000; Mikhail et al. 2001). Первые упоминания об относительных ограничениях в виде графа в литературе по SLAM восходят к работам Чизмана и Смита (Cheeseman and Smith, 1986) и Дюран-Уайта (Durrant-Whyte, 1988), но в этих подходах не было оптимизации. Представленный в этой главе алгоритм произвольно основан на основополагающей работе Лю и Милиоса (Lu and Milios, 1997). Исторически, они первые выразили априорную вероятность SLAM в виде набора связей между положениями робота, сформулировав алгоритм глобальной оптимизации для генерации карты на основе набора ограничений. Их оригинальный алгоритм для глобального выравнивания целостных сканирований расстояния использовал переменные положения робота как систему отсчёта, что отличалось от стандартной точки зрения EKF, в которой положения интегрировались. Используя анализ данных одометрии и сканирования расстояния лазером, их метод создавал относительные ограничения между положениями, которые можно рассматривать аналогично рёбрам в GraphSLAM. Однако, они не сформулировали метод в информационном представлением. Алгоритм Лю и Милиоса (1997) был впервые успешно реализован Гутманом и Небелем (Gutmann and Nebel, 1997), которые отметили

вычислительную неустойчивость, возможно, из-за излишнего использования инверсии матриц. Голфарелли (Golfarelli et al., 1998) первым установил связь задач SLAM и кинематических моделей на основе масс и пружин, а Дакитт (Duckett et al., 2000, 2002) представил первый эффективный метод решения таких задач. Отношение между ковариацией и информационной матрицей обсуждалось в работе Фризи и Хирзингера (Frese and Hirzinger, 2001). Аранеда (Araneda, 2003) разработал более детальную и точную графическую модель.

Алгоритм Лю и Милиоса положил начало разработке оффлайновых алгоритмов SLAM, которые, до настоящего времени, развиваются почти параллельно работам по EKF. Гутманн и Конолиг (Gutmann and Konolige) объединили эту реализацию с шагом марковской локализации для установки соответствия при замыкании петель в циклической среде. Босси (Bosse et al., 2003, 2004) разработал алгоритм Atlas, иерархическую картографическую систему на основе парадигмы разделённого стохастического картографирования, сохраняющего относительную взаимосвязь между вспомогательными картами. В нем используется метод оптимизации, похожий на предложенный Дакиттом (Duckett et al.,2000) и на GraphSLAM с выравниванием нескольких карт. Фолькессон и Кристенсен (Folkesson and Christensen, 2004a,b) раскрыли перспективу оптимизации SLAM, применив версию градиентного спуска к логарифму правдоподобия апостериорного распределения. Их алгоритм Graphical SLAM уменьшал число переменных для пути, так же как GraphSLAM, при замыкании петли. Это уменьшение (математически являющееся приближением, поскольку выполнялось отбрасывание карты) существенно ускорило выполнение градиентного спуска. Конолиг (Konolige, 2004) и Монтемерло и Трун (Montemerlo and Thrun, 2004) представили сопряжённый градиент в области SLAM, который показал как большую эффективность по сравнению с градиентным спуском. В обеих работах количество переменных при закрытии больших циклов уменьшалось, что позволяли выравнивать карты с 10^8 признаков всего за несколько секунд. Метод Левенберга-Маркардта, упомянутый в тексте, принадлежит Левенбергу (Levenberg, 1944) и Маркардту (Marquardt, 1963), которые вывели его в контексте оптимизации вычисления методом наименьших квадратов. Фризи (Frese et al., 2005) проанализировал эффективность SLAM в информационном виде, и разработал высокоэффективные методы оптимизации, используя оптимизацию нескольких сеток. Он отметил увеличение скорости на несколько порядков, и эти методы оптимизации в настоящее время являются наиболее совершенными. Делаэрт (Dellaert, 2005) разработал эффективные методы факторизации для графа ограничений GraphSLAM, направленные на преобразование графа ограничений в более компактные версии, сохраняя разрежённость.

Следует отметить, что идея сохранения относительных связей между локальными элементами лежит в основе многих методов на основе вспомогательных карт, описанных в предыдущем разделе, хотя и редко в явном виде. Рядом автором, среди которых Гювант и Небот (Guivant and Nebot, 2001), Вильямс (Williams, 2001), Тардос (Tardós et al., 2002), Бейли (Bailey, 2002) были описаны структуры данных для относительных смещений между вспомогательными картами, которые легко укладываются в информационные теоретические концепции. Хотя многие из этих алгоритмов являются фильтрами, они разделяют многие общие идеи информационного представления, обсуждаемого в этой главе.

Насколько нам известно, алгоритм GraphSLAM, представленный здесь, никогда не публиковался в указанном виде (в раннем черновике этот алго-

ритм называется "расширенным информационным видом"). Однако, он тесно связан с публикациями, указанными ранее, и основан на фундаментальном алгоритме Лю и Милиоса (1997). Название GraphSLAM напоминает название Graphical SLAM, использованное Фолкенссоном и Кристенсеном (2004a). Оно было выбрано для этой главы, поскольку графы ограничений являются основой всего направления исследований SLAM. Множество авторов разработали фильтры в информационном виде, которые решают онлайн задачу SLAM вместо полной задачи SLAM. Эти алгоритмы будут обсуждаться в следующей главе, полностью посвящённой проблеме фильтрации.

Формулировка GraphSLAM проблемы SLAM основана на десятилетнем обсуждении выражения пространственных карт, которые уже упоминались в библиографических примечаниях к Главе 9. Информационные представления сводят вместе две разные парадигмы представления карт: топологическую и метрическую. Дебаты о представлении пространства для людей и роботов остались позади (Chown et al. 1995). Топологические подходы уже обсуждались в библиографических примечаниях к Главе 9. Ключевым свойством топологического представления является факт отображения только относительной информации между элементами карты. Поэтому в них отсутствует проблема нахождения целостного метрического обоснования этой относительной информации. В наиболее современных топологических методах связи между элементами дополняются метрической информацией, такой как расстояние между двумя местоположениями.

Так же как в топологических представлениях, информационно- теоретические методы накапливают относительную информацию между соседними объектами (ориентирами и роботами). Но относительная линейная информация карты «переводится» в метрическое обоснование. В случае использования линейной гауссовой функции, шаг вывода не теряет информацию и, поэтому, обратим. Вычисление полной апостериорной вероятности, включая ковариацию, требует обращения матриц. Вторая операция обращения матриц ведёт к начальной форме относительных ограничений. Поэтому, топологический и метрический вид дополняют друг друга, так же как информационно-теоретическое и вероятностное представления (или EKF и GraphSLAM). Так может, стоит объединить математический и подход, чтобы соединить вместе топологические и метрические карты? Читателю следует знать, что такая точка зрения ещё не так распространена среди большинства исследователей.

11.11 Упражнения

1. В предыдущей главе в упражнениях уже использовалось *SLAM только по направлению*, как разновидность SLAM, в которой датчики способны измерять только направление на ориентир, но не расстояние до него. Мы предполагаем, что GraphSLAM лучше подходит для этой задачи, чем EKF. Почему?

2. В этом вопросе, требуется доказать сходимость результатов для особого класса проблем SLAM: *SLAM в виде линейных гауссианов*. В SLAM в виде линейных гауссианов, уравнение движения имеет простой аддитивный вид

SLAM В ВИДЕ ЛИНЕЙНЫХ ГАУССИАНОВ

$$x_t \sim \mathcal{N}(x_{t-1} + u_t, R)$$

а уравнение измерений - вид

$$z_t = \mathcal{N}(m_j - x_t Q)$$

где R и Q - диагональные матрицы ковариации, а m_j признак, наблюдаемый в момент времени t. Можно допустить, что количество ориентиров конечно, все ориентиры наблюдаются бесконечно часто и без определённого порядка, а соответствия - известны.

(a) Доказать, что для GraphSLAM расстояние между любыми двумя ориентирами сводится к верному с вероятностью 1.

(b) Как это доказательство относится к EKF SLAM?

(c) Подходит ли GraphSLAM для общей задачи SLAM с известным соответствием? Если да, объяснить, почему, если нет – тоже объяснить (без доказательства).

3. Алгоритм **GraphSLAM_reduce** уменьшает набор ограничений, интегрируя переменные карты, и сохраняя систему ограничений только по положениям робота. Возможно ли вместо этого интегрировать положения робота, так, чтобы результирующая сеть ограничений была определена исключительно по переменным карты? Если да, будет ли результирующая задача вывода разреженной? Как циклы на пути робота повлияют на новый набор ограничений?

4. Алгоритм GraphSLAM в этой главе игнорирует сигнатуры ориентиров. Расширить базовый алгоритм GraphSLAM для использования сигнатур в измерениях и на карте.

12 Разреженный обобщенный информационный фильтр

12.1 Введение

В предыдущих двух главах были описаны противоположные реализации в спектре алгоритмов SLAM. Уже было замечено, что EKF SLAM построен как *проактивный*. При каждом получении новой порции информации он реализует ее в виде вероятностного распределения, что вычислительно весьма затратно. Алгоритм GraphSLAM выполняется по другому и просто накапливает информацию. Принято считать такое накопление *ленивым*: в момент получения данных GraphSLAM просто запоминает полученную информацию. Для превращения собранной информации в карту GraphSLAM выполняет вывод, который может быть осуществлён только после сбора всех данных, что делает GraphSLAM оффлайновым алгоритмом.

Возникает вопрос о возможности вывода *онлайнового* алгоритма фильтра, который унаследовал бы эффективность информационного отображения. Это возможно, но лишь с рядом приближений. Разреженный обобщенный информационный фильтр (sparse extended information filter или SEIF),

реализует информационное решение для проблемы онлайн SLAM. Так же, как и в EKF, SEIF интегрирует и удаляет прошлые положения робота, сохраняя лишь апостериорное распределение по текущему положению робота и карте. Но, так же, как GraphSLAM и, в отличие от EKF SLAM, SEIF поддерживает информационное выражение всей информации. В результате, такт обновления в SEIF становится ленивой операцией информационного сдвига, превосходящей проективное вероятностное обновление EKF. Поэтому SEIF можно считать наилучшим выбором, поскольку он запускается онлайн и вычислительно эффективен.

Будучи онлайн алгоритмом, SEIF сохраняет оценку того же самого информационного вектора, что и EKF:

(12.1)

$$y_t = \begin{pmatrix} x_t \\ m \end{pmatrix}$$

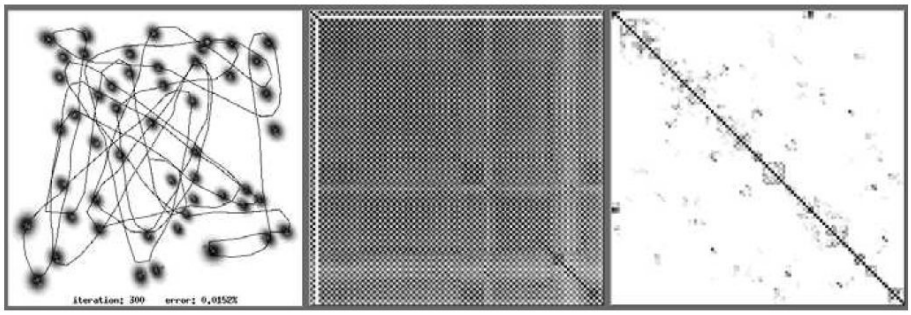


Рис. 12.1 Причины использования информационного фильтра в онлайн SLAM. Слева: проход робота в имитационной среде с 50 ориентирами. В центре: матрица корреляции EKF, демонстрирующая сильную корреляцию между двумя любыми координатами ориентиров. Справа: нормализованная информационная матрица EKF изначально разрежена. Эта разрежённость позволяет использовать алгоритм SLAM, который позволяет более эффективное обновление.

Здесь x_t – положение робота, а m - карта. Апостериорная вероятность с известным соответствием задана в виде $p(y_t|z_{1:t}, u_{1:t}, c_{1:t})$.

Ключевая идея превращения GraphSLAM в алгоритм онлайн SLAM показана на Рис. 12.1. На рисунке показан результат работы алгоритма EKF SLAM в имитационной среде, где размещены 50 ориентиров. На левой врезке показан движущийся робот, а также вероятностная оценка местоположения для всех 50 точечных признаков. Главная информация, которая сохраняется в EKF SLAM - это ковариационная матрица различных оценок. Корреляция, которая является нормализованной ковариацией, показана на центральной врезке рисунка. По каждой из двух осей показано положение робота (местоположение и ориентация по направлению), а также двухмерное местоположение всех 50 ориентиров. Тёмные элементы сильно коррелированны. В главе, посвящённой EKF SLAM, уже обсуждалось, что в пределе координаты всех признаков становятся полностью коррелированными – отсюда и схожесть матрицы корреляции с шахматной доской.

На правой врезке Рис. 12.1 показана информационная матрица Ω_t, нормализованная так же, как матрица корреляции. Так же, как и в предыдущей главе, элементы этой нормализованной информационной матрицы

можно воспринимать в виде ограничения или связи, которые ограничивают относительные положения пар признаков на карте. Чем темнее показанный элемент, тем сильнее связь. Как видно из рисунка, нормализованная информационная матрица выглядит разреженной. В ней преобладает небольшое количество сильных связей и имеется множество связей, значения которых при нормализации становятся практически равным нулю.

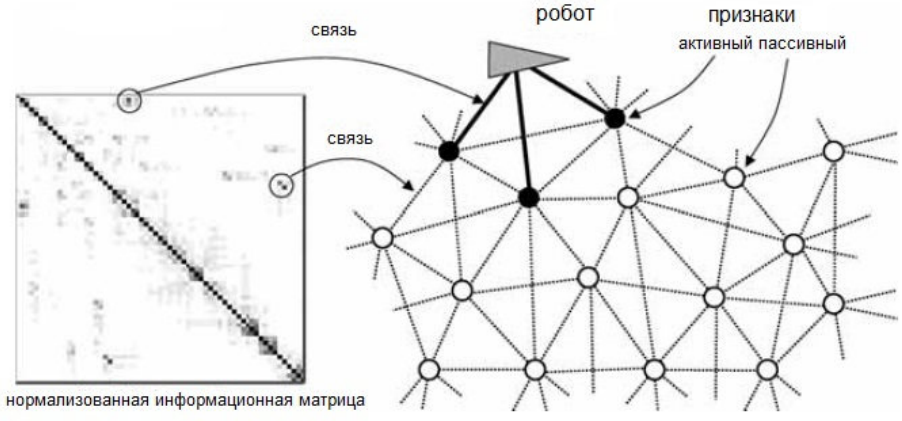


Рис. 12.2 Иллюстрация сети признаков, сгенерированной нашим методом. Слева показана разреженная информационная матрица, а справа – карта, значения в которой связаны только ненулевые элементы информационной матрицы. Как упоминалось в тексте, тот факт, что не все признаки связаны, является ключевым структурным элементом задачи SLAM и основой нашего решения с постоянным временем.

Сила каждой связи связана с расстоянием между соответствующими признаками, то есть сильные связи расположены только между близлежащими признаками. Чем дальше признаки друг от друга, тем слабее связь.

Эта концепция разреженности сильно отличается от описанной в предыдущей главе. Во-первых, существуют связи между парами признаков. В предыдущей главе такие связи не могли существовать. Во-вторых, сама разреженность лишь приблизительная. Фактически, хотя элементы нормализованной информационной матрицы ненулевые, но, практически все близки к нулю.

РАЗРЕЖЕННАЯ ИНФОРМАЦИОННАЯ МАТРИЦА

Алгоритм SEIF SLAM использует эту идею, сохраняя разреженную информационную матрицу, в которой только близлежащие признаки соединены ненулевым элементом. Результирующая сетевая структура показана справа на Рис. 12.2, где кружки означают точечные признаки, а пунктирные дуги – связи, изображённые в информационной матрице слева. На этой схеме также показан робот, который связан лишь с небольшим числом признаков.

АКТИВНЫЕ ПРИЗНАКИ

Эти признаки называются активными и изображены чёрным цветом. Хранение разреженной информационной матрицы требует объёма памяти, линейно зависящего от количества признаков на карте. Более важным является возможность выполнить все важные обновления SEIF SLAM за одинаковое время, независимо от количества признаков на карте. Этот результат достаточно нетривиален, поскольку наивная реализация обновления движения в информационных фильтрах, как указано в Таблице 3.6 на странице 79, требует обращения всей информационной матрицы.

SEIF - это онлайн алгоритм SLAM, сохраняющий такую разреженную

информационную матрицу, время выполнения всех тактов обновления которой не зависит от размера карты для случая известной ассоциации данных, и зависит логарифмически, если необходимо выполнять поиск ассоциаций. Это делает SEIF первым эффективным онлайновым SLAM алгоритмом, приведённом в книге.

12.2 Интуитивное описание

Начнём с интуитивного описания обновления SEIF, используя графические иллюстрации. Обновление SEIF состоит из 4 действий: обновление движения, обновление измерения, разрежение и обновление состояния.

Алгоритм начинается с шага обновления измерения, показанного на Рис. 12.3. На обеих врезках показана информационная матрица, сохраняемая SEIF, а также граф, определённый информационными связями. Так же, как и в GraphSLAM, обнаружение признака m_1 заставляет алгоритм SEIF обновить недиагональный элемент информационной матрицы, соединяющий оценку положения робота x_t с наблюдаемым признаком m_1. Это показано на схеме слева на Рис. 12.3a.

Обнаружение признака m_2 приводит к обновлению элементов информационной матрицы, соединяющих положение робота x_t и признак m_2, как показано на Рис. 12.3b. Как будет показано далее, каждому из этих обновлений соответствует локальное добавление в информационную матрицу и информационный вектор. В обоих случаях (и для информационной матрицы, и для вектора), эти добавления затрагивают только связи между переменной положения робота и наблюдаемым признаком. Так же, как в GraphSLAM, вычислительная сложность учёта измерения в SEIF не зависит от размера карты.

Обновление движение, однако, отличается от такового в GraphSLAM, поскольку SEIF является фильтром и как показано на Рис. 12.4, отсекает оценки прошлых положений. Здесь происходит смена положения робота – на Рис. 12.4a показано информационное состояние до перемещения, а на Рис. 12.4b – после передвижения, соответственно. Движение влияет на информационное состояние в нескольких аспектах. Во-первых, ослабляется связь между положением робота и признаками m_1, m_2. Это является результатом привнесения движением робота дополнительной неопределённости, поскольку теряется информация о местонахождении робота относительно карты. Однако, потеря этой информации неполная и её часть проецирована в виде связи между признаками. Это информационный сдвиг происходит потому, что, несмотря на потерю информации о положении робота, информация об относительном местоположении признаков на карте сохранилась. Там, где эти признаки были связаны опосредовано через положение робота, после шага обновления они становятся связаны напрямую.

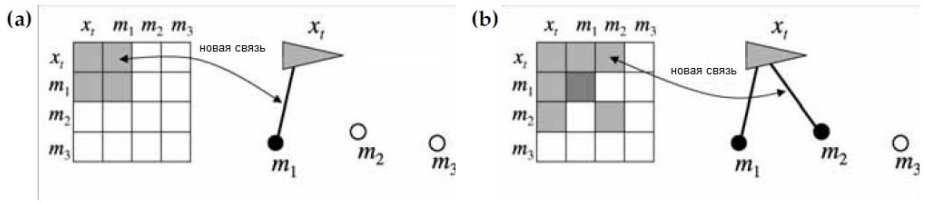


Рис. 12.3 Воздействие измерений на информационную матрицу и связанную с ней сеть признаков: наблюдение m_1 вызывает изменение элементов Ω_{x_t,m_1} в информационной матрице (a). Аналогично, наблюдение m_2 влияет на Ω_{x_t,m_2} (b).

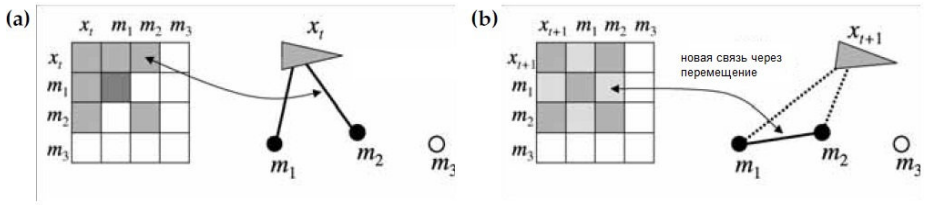


Рис. 12.4 Воздействие движения на информационную матрицу и связанную с ней сеть признаков: до перемещения (a) и после движения (b). Если движение не детерминировано, обновление движения образует новые связи (или усиливает существующие) между любыми двумя активными признаками, при этом связи между роботом и этими признаками ослабляются. На этом шаге образуются связи между парами признаков.

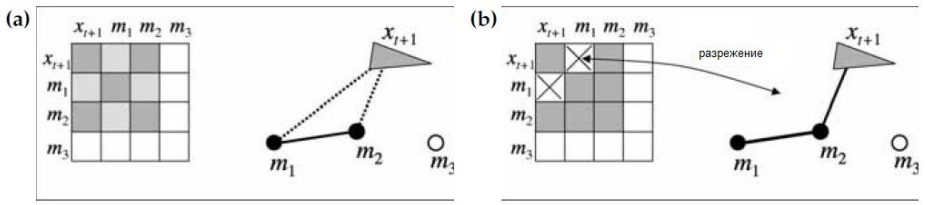


Рис. 12.5 Разрежение: признак деактивируется уничтожением его связи с роботом. Для компенсации потери связи в информационном состоянии обновляются связи между активными признаками и/или роботом. Время выполнения всей операции постоянно.

Смещение информации от связи с роботом к связи между признаками является ключевым элементом SEIF. Это прямое следствие использования информационного представления в виде фильтра для задачи онлайн SLAM. При вычёркивании переменные прошлых положений робота связи с ними теряются, но заново проецируются на связи между признаками в информационной матрице. В этом состоит отличие от алгоритма GraphSLAM, обсуждаемого в предыдущей главе, в котором новые связи между парами признаков на карте не образуются никогда.

Чтобы между парой признаков образовалась новая связь, перед обновлением они оба должны быть активными, поскольку элементы соответствия,

связывающие их с положением робота в информационной матрице, должны быть ненулевыми. Это показано на Рис. 12.4: Связь между признаками образуется только между m_1 и m_2. Неактивный признак m_3 остаётся незатронутым. Это подразумевает, что, управляя количеством активных ориентиров в произвольный момент времени, возможно управлять вычислительной сложностью обновления движения и количеством связей в информационной матрице. Если количество активных связей остаётся достаточно малым, вычислительная сложность обновления движения тоже будет небольшой, как и количество ненулевых элементов связей между ориентирами в информационной матрице.

РАЗРЕЖЕНИЕ

На этом шаге алгоритм SEIF выполняет *разрежение*, как показано на Рис. 12.5. Разрежение включает удаление связи между роботом и активным признаком, эффективно превращая активный признак в пассивный. В SEIF это удаление дуги ведёт к перераспределению информации на соседние связи, в частности, между активными признаками и положением робота. Требуемое для разрежения время не зависит от размера карты. Однако, поскольку это операция приближения, происходит потеря информации в апостериорной вероятности. Преимуществом такого приближения является вносимая разреженность, что позволяет эффективно выполнять обновление фильтра.

Существует ещё один, последний шаг выполнения алгоритма SEIF, не показанный на рисунках. На этом шаге выполняется распространение средней оценки по графу. Как уже обсуждалось в Главе 3, в обобщённом информационном фильтре для выполнения линеаризации моделей движения и измерения требуется оценка состояния μ_t. В SEIF для шага разрежения также требуется оценка состояния.

Строго говоря, можно восстановить оценку состояния через равенство $\mu = \Omega^{-1}\xi$, где Ω - информационный вектор, а ξ - информационное состояние. Однако, это потребует решения задачи вывода, слишком большой для обработки ее на каждом такте времени. В SEIF этот шаг выполняется с

АЛГОРИТМ РЕЛАКСАЦИИ

помощью алгоритма релаксации, распространяющего оценки состояния в информационном графе. Каждая локальная оценка состояния обновляется на основании наилучших оценок соседних узлов информационного графа. Этот алгоритм релаксации сходится к точному значению среднего μ. Поскольку информационный вид в SEIF разрежённый, каждое такое обновление выполняется за постоянное время, хотя и с допущением необходимости большого количества таких обновлений для получения хороших результатов. Для поддержания вычислительной сложности, независимой от размера пространства состояний, в SEIF при каждой итерации выполняется фиксированное количество таких обновлений. Результирующий вектор состояния является приближенным и используется вместо правильной оценки среднего на всех шагах обновления.

12.3 Алгоритм SEIF SLAM

Внешний цикл обновления SEIF показан в Таблице 12.1. Алгоритм принимает на вход информационную матрицу Ω_{t-1}, информационный вектор ξ_{t-1} и оценку состояния μ_{t-1}. Также принимаются измерение z_t, управляющее воздействие u_t и вектор соответствия c_t. Выход алгоритма **SEIF_SLAM _known_correspondences** даёт новую оценку состояния, выраженную информационной матрицей Ω_t и информационным вектором ξ_t, а также улучшенную оценку μ_t.

Как указано в Таблице 12.1, обновление SEIF происходит за четыре основных шага. Обновление движения в Таблице 12.2 учитывает управляющее воздействие u_t в оценке фильтра с помощью ряда вычислительно эффективных операций. Например, единственными модифицируемыми при этом обновлении компонентами информационного вектора/матрицы являются положение робота и активные признаки. Обновление измерения в Таблице 12.3 учитывает вектор измерения z_t с известным соответствием c_t. Этот шаг выполняется локально, так же, как обновление движения, то есть обновляются только информационные значения положения робота и наблюдаемых признаков на карте. Шаг разрежения, показанный в Таблице 12.4, является приближением: он удаляет активные признаки, преобразуя соответствующим образом информационную матрицу и информационный вектор. Этот шаг также вычислительно эффективен, поскольку затрагивает только связи между роботом и активными ориентирами. Наконец, обновление оценки состояния в Таблице 12.5, использует смягченный метод покоординатного спуска для восстановления оценки состояния μ_t. На этом шаге вновь используется разреженность SEIF, благодаря которой при каждом инкрементном обновлении необходимо обратиться лишь к небольшому числу других элементов вектора состояний.

Весь совокупный цикл обновления SEIF выполняется за постоянное время, которое не зависит от размера карты. Это является прямой противоположностью другого обсуждаемого алгоритма SLAM—EKF, в котором для каждого обновления требуется время, квадратично зависящее от размера карты.

1: **Algorithm SEIF_SLAM_known_correspondences**$(\xi_{t-1}, \Omega_{t-1}, \mu_{t-1}, u_t, z_t, c_t)$:
2: $\quad \bar{\xi}_t, \bar{\Omega}_t, \bar{\mu}_t = \textbf{SEIF\_motion\_update}(\xi_{t-1}, \Omega_{t-1}, \mu_{t-1}, u_t)$
3: $\quad \mu_t = \textbf{SEIF\_update\_state\_estimate}(\bar{\xi}_t, \bar{\Omega}_t, \bar{\mu}_t)$
4: $\quad \xi_t, \Omega_t = \textbf{SEIF\_measurement\_update}(\bar{\xi}_t, \bar{\Omega}_t, \mu_t, z_t, c_t)$
5: $\quad \bar{\xi}_t, \bar{\Omega}_t = \textbf{SEIF\_sparsification}(\xi_t, \Omega_t)$
6: $\quad$ *return* $\bar{\xi}_t, \bar{\Omega}_t, \mu_t$

Таблица 12.1 Алгоритм разреженного обобщённого информационного фильтра для задачи SLAM с известной ассоциацией данных.

1: **Algorithm SEIF_motion_update**$(\xi_{t-1}, \Omega_{t-1}, \mu_{t-1}, u_t)$:

2: $F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & \underbrace{0...0}_{3N} \end{pmatrix}$

3: $\delta = \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$

4: $\Delta = \begin{pmatrix} 0 & 0 & \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & \frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix}$

5: $\Psi_t = F_x^T[(I + \Delta)^{-1} - I]F_x$

6: $\lambda_t = \Psi_t^T\Omega_{t-1} + \Omega_{t-1}\Psi_t + \Psi_t^T\Omega_{t-1}\Psi_t$

7: $\Phi_t = \Omega_{t-1} + \lambda_t$

8: $\kappa_t = \Phi_t F_x^T(R_t^{-1} + F_x\Phi_t F_x^T)^{-1}F_x\Phi_t$

9: $\bar{\Omega}_t = \Phi_t - \kappa_t$

10: $\bar{\xi}_t = \xi_{t-1} + (\lambda_t - \kappa_t)\mu_{t-1} + \bar{\Omega}_t F_x^T\delta_t$

11: $\bar{\mu}_t = \mu_{t-1} + F_x^T\delta$

12: *return* $\bar{\xi}_t, \bar{\Omega}_t, \bar{\mu}_t$

Таблица 12.2 Обновление движения в SEIF.

1: **Algorithm SEIF_measurement_update**$(\bar{\xi}_t, \bar{\Omega}_t, \mu_t, z_t, c_t)$:

2: $Q_t = \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix}$

3: *для всех наблюдаемых признаков* $z_t^i = (r_t^i \ \phi_t^i \ s_t^i)^T$ *выполнять*

4: $j = c_t^i$

5: *если ориентир* j *никогда не наблюдался*

6: $\begin{pmatrix} \mu_{j,x} \\ \mu_{j,y} \\ \mu_{j,s} \end{pmatrix} = \begin{pmatrix} \mu_{t,x} \\ \mu_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \mu_{t,\theta}) \\ \sin(\phi_t^i + \mu_{t,\theta}) \\ 0 \end{pmatrix}$

7: *endif*

8: $\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \mu_{j,x} - \mu_{t,x} \\ \mu_{j,y} - \mu_{t,y} \end{pmatrix}$

9: $q = \delta^T \delta$

10: $\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \mu_{t,\theta} \\ \mu_{j,s} \end{pmatrix}$

11: $H_t^i = \frac{1}{q} \begin{pmatrix} \sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & 0...0 & -\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 & 0...0 \\ \delta_y & \delta_x & -1 & 0...0 & -\delta_y & -\delta_x & 0 & 0...0 \\ 0 & 0 & 0 & \underbrace{0...0}_{3j-3} & 0 & 0 & 1 & \underbrace{0...0}_{3j} \end{pmatrix}$

12: *endfor*

13: $\xi_t = \bar{\xi}_t + \Sigma_i H_t^{iT} Q_t^{-1} [z_t^i - \hat{z}_t^i - H_t^i \mu_t]$

14: $\Omega_t = \bar{\Omega}_t + \Sigma_i H_t^{iT} Q_t^{-1} H_t^i$

15: *return* ξ_t, Ω_t

Таблица 12.3 Обновление измерения в SEIF.

1: **Algorithm SEIF_sparsification**(ξ_t, Ω_t) :

2: *определить* F_{m_0}, F_{x,m_0}, F_x *как проекционные матрицы от* y_t *до* $m_0, \{x, m_0\}$ *и* x, *соответственно*

3: $\bar{\Omega}_t = \Omega_t - \Omega_t^0 F_{m_0} (F_{m_0}^T \Omega_t^0 F_{m_0})^{-1} F_{m_0}^T \Omega_t^0$
$+\Omega_t^0 F_{x,m_0} (F_{x,m_0}^T \Omega_t^0 F_{x,m_0})^{-1} F_{x,m_0}^T \Omega_t^0$

2 : $-\Omega_t F_x (F_x^T \Omega_t F_x)^{-1} F_x^T \Omega_t$

4: $\bar{\xi}_t = \xi_t + \mu_t (\bar{\Omega}_t - \Omega_t)$

5: *return* $\bar{\xi}_t, \bar{\Omega}_t$

Таблица 12.4 Шаг разрежения в SEIF.

1: **Algorithm SEIF_update_state_estimate**$(\bar{\xi}_t, \bar{\Omega}_t, \bar{\mu}_t)$:
2: *для малого набора признаков карт* m_i *выполнять*
3: $F_i = \begin{pmatrix} 0...0 & 1 & 0 & 0...0 \\ \underbrace{0...0}_{2(N-i)} & 0 & 1 & \underbrace{0...0}_{2(i-1)x} \end{pmatrix}$
4: $\mu_{i,t} = (F_i \Omega_t F_i^T)^{-1} F_i [\xi_t - \Omega_t \bar{\mu}_t + \Omega_t F_i^T F_i \bar{\mu}_t]$
5: *endfor*
6: *для всех остальных признаков карты* m_i *do*
7: $\mu_{i,t} = \bar{\mu}_{i,t}$
8: *endfor*
9: $F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & \underbrace{0...0}_{3N} \end{pmatrix}$
10: $\mu_{x,t} = (F_x \Omega_t F_x^T)^{-1} F_x [\xi_t - \Omega_t \bar{\mu}_t + \Omega_t F_x^T F_x \bar{\mu}_t]$
11: *return* μ_t

Таблица 12.5 Шаг смягчённого обновления состояния в SEIF затрагивает небольшое количество оценок состояния.

SLAM С ПОСТОЯННЫМ ВРЕМЕНЕМ

Однако, такое предположение о "*SLAM с постоянным временем*" следует воспринимать с некоторым скепсисом, поскольку восстановление оценок состояния в средах с большими циклами является вычислительной проблемой, для которой пока не существует решений, линейно зависящих от времени.

12.4 Математический вывод SEIF

12.4.1 Обновление движения

Обновление движения в SEIF обрабатывает управляющее воздействие u_t путём преобразования информационной матрицы Ω_{t-1} и информационного вектора ξ_{t-1} в новую матрицу $\bar{\Omega}_t$ и вектор $\bar{\xi}_t$. Как обычно в нашей записи, черта указывает, что в этом прогнозе учитывается лишь управление, а измерение в расчёт не принимается.

Обновление движения в SEIF использует разреженность информационной матрицы, что позволяет выполнять его за время, не зависящее от размера карты n. Этот вывод лучше всего начать с соответствующей формулы для EKF. Начнём с алгоритма **EKF_SLAM_known_correspondences** в Таблице 10.1 на странице 292. Обновление движение происходит в строках 3 и 5, приведённых ниже для удобства:

(12.2)
$$\bar{\mu}_t = \mu_{t-1} + F_x^T \delta$$

(12.3)
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + F_x^T R_t F_x$$

Ключевые элементы обновления определены следующим образом:

(12.4)

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0...0 \\ 0 & 1 & 0 & 0...0 \\ 0 & 0 & 1 & 0...0 \end{pmatrix}$$

(12.5)

$$\delta = \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

(12.6)

$$\Delta = \begin{pmatrix} 0 & 0 & \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & \frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\sin(\mu_{t-1,\theta} + \omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix}$$

(12.7)

$$G_t = I + F_x^T \Delta F_x$$

В SEIF необходимо определить обновление движения через информационный вектор ξ и информационную матрицу Ω. Из выражения (12.3), определение G_t в (12.7), и определения информационной матрицы $\Omega = \Sigma^{-1}$ следует, что

(12.8)

$$\begin{aligned} \bar{\Omega}_t &= [G_t\Omega_{t-1}^{-1}G_t^T + F_x^T R_t F_x]^{-1} \\ &= [(I + F_x^T\Delta F_x)\Omega_{t-1}^{-1}(I + F_x^T\Delta F_x)^T + F_x^T R_t F_x]^{-1} \end{aligned}$$

Ключевым фактором такого метода обновления является возможность выполнить его за постоянное время вне зависимости от размерности Ω. Факт, что это возможно и для разреженных матриц Ω_{t-1} несколько нетривиален, поскольку в выражении (12.8), похоже, требуется две вложенных инверсии матриц размером $(3N+3)\times(3N+3)$. Как мы увидим, если Ω_{t-1} разрежена, это может быть эффективно выполнено. Определим

(12.9)

$$\Phi_t = [G_t\Omega_{t-1}^{-1}G_t^T]^{-1} = [G_t^T]^{-1}\Omega_{t-1}G_t^{-1}$$

и перепишем выражение (12.8) в виде

(12.10)

$$\bar{\Omega}_t = [\Phi_t^{-1} + F_x^T R_t F_x]^{-1}$$

применив лемму об обращении матриц, получим:

(12.11)

$$\begin{aligned} \bar{\Omega}_t &= [\Phi_t^{-1} + F_x^T R_t F_x]^{-1} \\ &= \Phi_t - \underbrace{\Phi_t F_x^T (R_t^{-1} + F_x\Phi_t F_x^T)^{-1} F_x\Phi_t}_{\kappa_t} \\ &= \Phi_t - \kappa_t \end{aligned}$$

Здесь κ_t определено следующим образом. Это выражение можно вычислить за постоянное время, если удастся вычислить за постоянное время Φ_t из матрицы Ω_{t-1}. Чтобы доказать, что это возможно, заметим, что

аргумент внутри обращения, $R_t^{-1} + F_x \Phi_t F_x^T$ трёхмерный. Перемножение с обращением F_x^T и F_x даст матрицу такого же размера, как и Ω, в которой ненулевые только субматрицы размером 3×3, соответствующей положению робота. Перемножение этой матрицы и разреженной матрицы Ω_{t-1} (левой и правой) затрагивает только ненулевые элементы вне главной диагонали Ω_{t-1} между признаком карты и положением робота. Другими словами, результат этой операции затрагивает только строки и столбцы, соответствующие активным признакам на карте. Поскольку разреженность подразумевает, независимость количества активных признаков Ω_{t-1} от размера Ω_{t-1}, общее количество ненулевых элементов κ_t также составляет $O(1)$. Следовательно, вычитание требует времени $O(1)$.

Остаётся показать, что можно вычислить Φ_t и Ω_{t-1} за постоянное время. Начнём с того, что учтём обращение G_t, которое можно эффективно вычислить следующим образом:

(12.12)

$$\begin{aligned} G_t^{-1} &= (I + F_x^T \Delta F_x)^{-1} \\ &= (I \underbrace{- F_x^T I F_x + F_x^T I F_x}_{=0} + F_x^T \Delta F_x)^{-1} \\ &= (I - F_x^T I F_x + F_x^T (I + \Delta) F_x)^{-1} \\ &= I - F_x^T I F_x + F_x^T (I + \Delta)^{-1} F_x \\ &= I + \underbrace{F_x^T [(I + \Delta)^{-1} - I] F_x}_{\Psi_t} \\ &= I + \Psi_t \end{aligned}$$

По аналогии получаем перестановку $[G_t^T]^{-1} = (I + F_x^T \Delta^T F_x)^{-1} = I + \Psi_t^T$. Здесь матрица Ψ_t ненулевая только для элементов, соответствующих положению робота и нулевая для всех признаков карты, поэтому может быть вычислена за постоянное время. Это даст следующее выражение для искомой матрицы Φ_t:

(12.13)

$$\begin{aligned} \Phi_t &= (I + \Psi_t^T) \Omega_{t-1} (I + \Psi_t) \\ &= \Omega_{t-1} + \underbrace{\Psi_t^T \Omega_{t-1} + \Omega_{t-1} \Psi_t + \Psi_t^T \Omega_{t-1} \Psi_t}_{\lambda_t} \\ &= \Omega_{t-1} + \lambda_t \end{aligned}$$

где Ψ_t нулевое, за исключением субматрицы положения робота. Поскольку Ω_{t-1} разрежена, матрица λ_t ненулевая, за исключением конечного числа элементов, которые соответствуют активным признакам карты и положению робота.

В силу этого, Φ_t можно вычислить из Ω_{t-1} за постоянное время, считая Ω_{t-1} разреженой. Выражения с (12.11) по (12.13) эквивалентны строкам с 5 по 9 в Таблице 12.2, что доказывает правильность обновления информационной матрицы в **SEIF_motion_update**.

Наконец, покажем аналогичный результат для информационного вектора. Из (12.2) получаем

(12.14)

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \delta_t$$

Для информационного вектора это даст:

(12.15)

$$\begin{aligned}
\bar{\xi}_t &= \bar{\Omega}_t(\Omega_{t-1}^{-1}\xi_{t-1} + F_x^T\delta_t) \\
&= \bar{\Omega}_t\Omega_{t-1}^{-1}\xi_{t-1} + \bar{\Omega}_t F_x^T\delta_t \\
&= (\bar{\Omega}_t + \Omega_{t-1} - \Omega_{t-1} + \Phi_t - \Phi_t)\Omega_{t-1}^{-1}\xi_{t-1} + \bar{\Omega}_t F_x^T\delta_t \\
&= (\bar{\Omega}_t \underbrace{-\Phi_t + \Phi_t}_{=0} \underbrace{-\Omega_{t-1} + \Omega_{t-1}}_{=0})\Omega_{t-1}^{-1}\xi_{t-1} + \bar{\Omega}_t F_x^T\delta_t \\
&= (\underbrace{\bar{\Omega}_t - \Phi_t}_{=-\kappa_t} + \underbrace{\Phi_t - \Omega_{t-1}}_{=\lambda_t})\underbrace{\Omega_{t-1}^{-1}\xi_{t-1}}_{=\mu_{t-1}} + \underbrace{\Omega_{t-1}\Omega_{t-1}^{-1}}_{=I}\xi_{t-1} + \bar{\Omega}_t F_x^T\delta_t \\
&= \xi_{t-1} + (\lambda_t - \kappa_t)\mu_{t-1} + \bar{\Omega}_t F_x^T\delta_t
\end{aligned}$$

Поскольку и λ_t и κ_t разрежены, перемножение $(\lambda_t - \kappa_t)\mu_{t-1}$ содержит только конечное множество ненулевых элементов и может быть вычислено за постоянное время. Более того, $F_x^T\delta_t$ - разреженная матрица. Разрежённость произведения $\bar{\Omega}_t F_x^T\delta_t$ напрямую следует из того, что $\bar{\Omega}_t$ также разрежена.

12.4.2 Обновления измерений

Вторым важным шагом в SLAM является обновление фильтра в соответствии с движением робота. Обновление измерений в SEIF напрямую реализовано с помощью обновления обобщённого информационного фильтра, как показано в строках 6 и 7 Таблицы 3.6 на странице 79:

(12.16)
$$\Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t$$

(12.17)
$$\xi_t = \bar{\xi}_t + H_t^T Q_t^{-1}[z_t - h(\bar{\mu}_t) - H_t\mu_t]$$

Запись прогноза $\hat{z}_t = h(\bar{\mu}_t)$ и суммирование по всем элементам в векторе измерений даст форму, приведённую в строках 13 и 14 Таблицы 12.3:

(12.18)
$$\Omega_t = \bar{\Omega}_t + \sum_i H_t^{iT} Q_t^{-1} H_t^i$$

(12.19)
$$\xi_t = \bar{\xi}_t + \sum_i H_t^{iT} Q_t^{-1}[z_t^i - \hat{z}_t^i - H_t^i\mu_t]$$

Здесь Q_t, δ, q и H_t^i определены, как и прежде (см. Таблицу 11.2 на странице 322).

12.5 Разрежение

12.5.1 Общий принцип

Ключевым шагом SEIF является разрежение информационной матрицы Ω_t. Поскольку разрежение столь важно для SEIF, сначала обсудим общие положения а затем - применение их к информационному фильтру. Разрежение

- это приближение, при котором апостериорное распределение аппроксимируется двумя предельными значениями. Допустим, a, b, и c - наборы случайных переменных (не путать с другими аналогичными обозначениями, используемыми в книге!), и задано совместное распределение $p(a, b, c)$ по этим переменным. Для разрежения этого распределения необходимо удалить прямую связь между переменными a и b. Другими словами, хотелось бы аппроксимировать p распределением $\bar{p}$, для которого выполняются следующие свойства: $\bar{p}(a|b, c) = p(a|c)$ и $\bar{p}(b|a, c) = p(b|c)$. Для многомерных гауссовых функций легко показать, что эта условная независимость эквивалентна отсутствию прямой связи между a и b. Соответствующий элемент информационной матрицы равен нулю.

Хорошая аппроксимация $\bar{p}$ получается с помощью коэффициента, пропорционального произведению предельных $p(a, c)$ и $p(b, c)$. Ни один из множителей не сохраняет зависимость между переменными a и b, поскольку они содержат только по одной из указанных переменных. Поэтому, произведение $p(a, c)p(b, c)$ не содержит прямых зависимостей между a и b. Наоборот, a и b условно независимы при данном c. Однако, $p(a, c)p(b, c)$ ещё не является верным вероятностным распределением по a, b и c. Это происходит, поскольку c дважды встречается в выражении. Соответствующая нормализация по $p(c)$ даст вероятностное распределение (считая $p(c) > 0$):

(12.20)
$$\bar{p}(a, b, c) = \frac{p(a, c)p(b, c)}{p(c)}$$

для понимания эффекта этого приближения, применим следующее преобразование:

(12.21)
$$\begin{aligned}\bar{p}(a, b, c) &= \frac{p(a, b, c)}{p(a, b, c)} \frac{p(a, c)p(b, c)}{p(c)} \\ &= p(a, b, c) \frac{p(a, c)}{p(c)} \frac{p(b, c)}{p(a, b, c)} \\ &= p(a, b, c) \frac{p(a|c)}{p(a|b, c)}\end{aligned}$$

Другими словами, удаление прямой зависимости между a и b эквивалентно аппроксимации условной вероятности $p(a|b, c)$ через условную вероятность $p(a|c)$. Также заметим (без доказательства) что среди всех аппроксимаций q по p, где a и b условно независимы при данном c, описанное распределение «наиболее близко» к p, где близость определяется расстоянием Кульбака-Лейблера, общепринятой ассиметричный мерой «близости» двух вероятностных распределений.

Важным наблюдением является факт того, что $p(a|b, c)$ является по меньшей мере, столь же информативным, как и $p(a|c)$, условная вероятность, заменяющая $p(a|b, c)$ в $\bar{p}$. Это происходит потому, что $p(a|b, c)$ обусловлена суперсетом переменных на основе условных переменных в $p(a|c)$. Для гауссовых функций это означает, что дисперсия аппроксимации $p(a|c)$ равна или превышает дисперсию начальной условной вероятности $p(a|b, c)$. Более того, дисперсия пределов $\bar{p}(a)$, $\bar{p}(b)$, и $\bar{p}(c)$ также больше или равна соответствующим дисперсиям $p(a)$, $p(b)$ и $p(c)$. Другими словами, невозможно уменьшить дисперсию путём приближения.

12.5.2 Разрежение в SEIF

В SEIF используется разрежение апостериорного распределения $p(y_t|z_{1:t}, u_{1:t}, c_{1:t})$, для сохранения информационной матрицы Ω_t разреженной. Чтобы это сделать, необходимо деактивировать связи между положением робота и отдельными признаками на карте. При правильном выполнении это также ограничит количество связей между парами признаков.

Чтобы продемонстрировать это, давайте кратко опишем два обстоятельства, при которых может появиться новая связь. Во-первых, наблюдение пассивного признака активирует его и создаёт новую связь между положением робота и этим признаком. Во-вторых, перемещение робота создаёт связи между двумя активными признаками. Это предполагает, что управление количеством активных признаков может предотвратить нарушение границ разрежённости, поэтому она достигается поддержанием небольшого количества активных признаков в каждый момент времени.

Для определения шага разрежения будет полезно разбить набор всех признаков на три несвязных подмножества:

(12.22)
$$m = m^+ + m^0 + m^-$$

где m^+ множество всех активных признаков, которые должны оставаться активными. Набор m^0 состоит из активных признаков, которые необходимо деактивировать. Другими словами, следует удалить связи между m^0 и положением робота. И, наконец, m^- - подмножество всех пассивных признаков, которые остаются пассивными в процессе разрежения. Поскольку $m^+ \cup m^0$ содержат все активные в текущий момент признаки, апостериорное распределение можно выразить следующим образом:

(12.23)
$$\begin{aligned}
&p(y_t|z_{1:t}, u_{1:t}, c_{1:t}) \\
&\quad = p(x_t, m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t}) \\
&\quad = p(x_t|m^0, m^+, m^-, z_{1:t}, u_{1:t}, c_{1:t}) p(m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t}) \\
&\quad = p(x_t|m^0, m^+, m^- = 0, z_{1:t}, u_{1:t}, c_{1:t}) p(m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t})
\end{aligned}$$

На последнем шаге используется факт того, что, если известны активные признаки m^0 и m^+, переменная x_t не зависит от пассивных признаков m^-. Также можно установить m^- в произвольное значение без изменения условного апостериорного распределения по x_t, $p(x_t|m^0, m^+, m^-, z_{1:t}, u_{1:t}, c_{1:t})$. Здесь просто происходит установка $m^- = 0$.

Согласно идее разрежения, обсуждаемой в общих словах в предыдущем разделе, заменим $p(x_t|m^0, m^+, m^- = 0)$ на $p(x_t|m^+, m^- = 0)$ отбросив зависимость по m^0.

(12.24)
$$\begin{aligned}
&\bar{p}(x_t, m|z_{1:t}, u_{1:t}, c_{1:t}) \\
&\quad = p(x_t|m^+, m^- = 0, z_{1:t}, u_{1:t}, c_{1:t}) p(m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t})
\end{aligned}$$

Очевидно, эта зависимость эквивалентна выражению:

(12.25)

$$\begin{aligned}\bar{p}(x_t, m|z_{1:t}, u_{1:t}, c_{1:t}) \\ = \frac{p(x_t, m^+|m^- = 0, z_{1:t}, u_{1:t}, c_{1:t})}{p(m^+|m^- = 0, z_{1:t}, u_{1:t}, c_{1:t})} p(m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t})\end{aligned}$$

12.5.3 Математический вывод метода разрежения

В оставшейся части раздела будет показано, что алгоритм **SEIF _ sparsification** в Таблице 12.4 реализует это вероятностное вычисление, и делает это за постоянное время. Начнём с вычисления информационной матрицы для распределения $p(x_t, m^0, m^+|m^- = 0)$ всех переменных, кроме m^-, обусловленных по $m^- = 0$. Это выполняется извлечением субматрицы всех переменных состояния, кроме m^-:

(12.26)

$$\Omega_t^0 = F_{x,m^+,m_0} F_{x,m^+,m^0}^T \Omega_t F_{x,m^+,m_0} F_{x,m^+,m_0}^T$$

Использования леммы об обращении матриц (Таблица 3.2 на странице 52) даст следующие информационные матрицы для $p(x_t, m^+|m^- = 0, z_{1:t}, u_{1:t}, c_{1:t})$ и $p(m^+|m^- = 0, z_{1:t}, u_{1:t}, c_{1:t})$, обозначенные Ω_t^1 и Ω_t^2, соответственно:

(12.27)

$$\Omega_t^1 = \Omega_t^0 - \Omega_t^0 F_{m_0} (F_{m_0}^T \Omega_t^0 F_{m_0})^{-1} F_{m_0}^T \Omega_t^0$$

(12.28)

$$\Omega_t^2 = \Omega_t^0 - \Omega_t^0 F_{x,m_0} (F_{x,m_0}^T \Omega_t^0 F_{x,m_0})^{-1} F_{x,m_0}^T \Omega_t^0$$

Здесь различные F–матрицы являются матрицами проекции полного состояния y_t в соответствующее субсостояние, содержащее только подмножество всех переменных, по аналогии с матрицей F_x, используемой в различных предыдущих алгоритмах. Финальный вид аппроксимации (12.25), $p(m^0, m^+, m^-|z_{1:t}, u_{1:t}, c_{1:t})$, даст следующую информационную матрицу:

(12.29)

$$\Omega_t^3 = \Omega_t - \Omega_t F_x (F_x^T \Omega_t F_x)^{-1} F_x^T \Omega_t$$

Сводя эти выражения вместе, согласно уравнению (12.25), получаем следующую информационную матрицу, в которой признак m^0 деактивирован:

(12.30)

$$\begin{aligned}\bar{\Omega}_t &= \Omega_t^1 - \Omega_t^2 + \Omega_t^3 \\ &= \Omega_t - \Omega_t^0 F_{m_0} (F_{m_0}^T \Omega_t^0 F_{m_0})^{-1} F_{m_0}^T \Omega_t^0 \\ &+ \Omega_t^0 F_{x,m_0} (F_{x,m_0}^T \Omega_t^0 F_{x,m_0})^{-1} F_{x,m_0}^T \Omega_t^0 \\ &- \Omega_t F_x (F_x^T \Omega_t F_x)^{-1} F_x^T \Omega_t\end{aligned}$$

Результирующий информационный вектор получается простым допущением:

(12.31)

$$\begin{aligned}\bar{\xi}_t &= \bar{\Omega}_t \mu_t \\ &= (\Omega_t - \Omega_t + \bar{\Omega}_t)\mu_t \\ &= \Omega_t \mu_t + (\bar{\Omega}_t - \Omega_t)\mu_t \\ &= \xi_t + (\bar{\Omega}_t - \Omega_t)\mu_t\end{aligned}$$

На этом вывод строк 3 и 4 в Таблице 12.4 завершён.

12.6 Смягчённое приближенное восстановление карты

Последний шаг обновления SEIF связан с вычислением среднего μ. В этом разделе индекс времени будет отбрасываться, поскольку в обсуждаемом методе он никакой роли не играет и будет использована запись μ вместо μ_t.

Перед выводом алгоритма для восстановления оценки состояния μ из информационного вида, кратко перечислим части μ, которые потребуются в SEIF, и укажем, когда именно. Для SEIF потребуются оценка состояния μ, положения робота и активных признаков на карте. Эти оценки потребуются в трёх различных случаях:

1. Среднее используется для линеаризации модели движения, которая имеет место в строках 3, 4 и 10 Таблице 12.2.

2. Оно также используется для линеаризации при обновлении измерения, см. строки 6, 8, 10, 13 Таблице 12.3.

3. Наконец, оно используется в шаге разрежения, а именно, в строке 4 Таблице 12.4.

Однако, полный вектор μ не нужен никогда. Понадобится только оценка положения робота, а также оценки местоположений всех активных признаков, что является лишь небольшим подмножеством всех переменных состояния в μ. Тем не менее, эффективное вычисление этих оценок требует привлечения дополнительной математики, поскольку точный метод восстановления среднего с помощью $\mu = \Omega^{-1}\xi$ требует обращения матриц или использования неких методов оптимизации, даже если восстанавливается только подмножество переменных.

И снова, ключевая идея выводится из разрежённости матрицы Ω. Разрежённость позволяет определить итеративный онлайн алгоритм восстановления переменных состояния, поскольку были собраны данные и сгенерированы ξ и Ω. Чтобы это сделать, для удобства переформулируем $\mu = \Omega^{-1}\xi$ в виде задачи оптимизации. Как будет показано чуть ниже, μ является модой

(12.32)
$$\hat{\mu} = \underset{\mu}{\operatorname{argmax}}\ p(\mu)$$

следующего гауссового распределения, определённого по переменной μ:

(12.33)
$$p(\mu) = \eta\ \exp\{-\frac{1}{2}\mu^T\Omega\mu + \xi^T\mu\}$$

Здесь μ – вектор того же вида и размерности, что и μ. Чтобы увидеть, что это действительно так, заметим, что переменная $p(\mu)$ сводится к нулю при $\mu = \Omega^{-1}\xi$:

(12.34)
$$\frac{\partial p(\mu)}{\partial \mu} = \eta\,(-\Omega\mu + \xi)\exp\{-\frac{1}{2}\mu^T\Omega\mu + \xi^T\mu\} \stackrel{!}{=} 0$$

что влечёт $\Omega\mu = \xi$ или же $\mu = \Omega^{-1}\xi$.

В этом преобразовании подразумевается, что восстановление вектора состояний μ эквивалентно нахождению моды (12.33), что и стало задачей оптимизации. Для задачи оптимизации опишем итеративный алгоритм поиска экстремума, который может быть реализован в силу разрежённости информационной матрицы.

ПОКООРДИНАТНЫЙ СПУСК

Наш метод представляет собой экземпляр метода *покоординатного спуска*. Для простоты сформулируем его только для одной координаты. В нашей реализации алгоритм проходит постоянное число K таких оптимизаций после каждого этапа обновления измерения. Мода $\hat{\mu}$ в (12.33) имеет вид:

(12.35)

$$\begin{aligned}\hat{\mu} &= \underset{\mu}{\operatorname{argmax}}\ \exp\{-\frac{1}{2}\mu^T\Omega\mu + \xi^T\mu\} \\ &= \underset{\mu}{\operatorname{argmax}}\ \frac{1}{2}\mu^T\Omega\mu - \xi^T\mu\end{aligned}$$

Заметим, что аргумент в операторе минимизации (12.35) может быть записан таким образом, чтобы явно указать отдельные переменные координат μ_i (для i-й координаты μ_t):

(12.36)

$$\frac{1}{2}\mu^T\Omega\mu - \xi^T\mu = \frac{1}{2}\sum_i\sum_j \mu_i^T\Omega_{i,j}\mu_j - \sum_i \xi_i^T\mu_i$$

где $\Omega_{i,j}$ – элемент с координатами (i,j) в матрице Ω, а ξ_i - i-ый компонент вектора ξ. Взяв производную этого выражения по отношению к произвольной координатной переменной μ_i,получим

(12.37)

$$\frac{\partial}{\partial\mu_i}\left\{\frac{1}{2}\sum_i\sum_j \mu_i^T\Omega_{i,j}\mu_j - \sum_i \xi_i^T\mu_i\right\} = \sum_j \Omega_{i,j}\mu_j - \xi_i$$

Приравнивание производной к нулю даст оптимум i-й координатной переменной μ_i, если заданы все остальные оценки μ_j:

(12.38)

$$\mu_i = \Omega_{i,i}^{-1}\left[\xi_i - \sum_{j\neq i}\Omega_{i,j}\mu_j\right]$$

То же самое выражение удобно переписать в виде матриц. Определим $F_i = (0...0\ 1\ 0...0)$ как матрицу проекции для извлечения i-го компонента из матрицы Ω:

(12.39)

$$\mu_i = (F_i\Omega F_i^T)^{-1}F_i[\xi - \Omega\mu + \Omega F_i^T F_i\mu]$$

На этом соображении построен алгоритм инкрементного обновления. Несколько раз выполняя обновление

(12.40)

$$\mu_i \longleftarrow (F_i\Omega F_i^T)^{-1}F_i[\xi - \Omega\mu + \Omega F_i^T F_i\mu]$$

Для некоторого элемента вектора состояния μ_i возможно уменьшить ошибку между левой и правой стороной уравнения (12.39). Бесконечное выполнение обновления для всех элементов вектора состояния сводит его к корректному среднему (приводится без доказательства).

Легко заметить, что количество элементов при суммировании в (12.38), а, значит, и перемножении вектора в правиле обновления (12.40), остаётся постоянным, если матрица Ω разрежена. Поэтому каждое обновление требует одинакового времени. Для сохранения свойства постоянного времени обработки алгоритма SLAM можно определить постоянное количество обновлений K на такт времени. Это обычно приводит к сходимости после множества обновлений.

Однако, следует соблюдать осторожность. Качество этой аппроксимации зависит от множества факторов, среди которых размер наибольшей циклической структуры на карте. В общем случае, фиксированного количества K обновлений на такт времени может быть недостаточно для получения хороших результатов. Кроме этого, существует множество методов оптимизации, которые более эффективны описанного здесь покординатного спуска. Классическим примером является метод сопряженных градиентов в контексте GraphSLAM. В практических реализациях рекомендуется полагаться на эффективные методы оптимизации для восстановления значения μ.

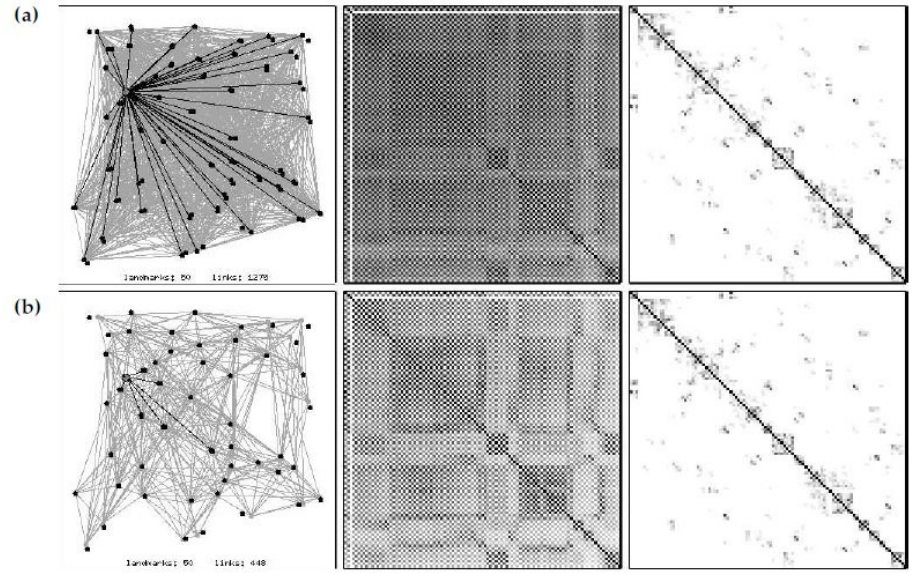


Рис. 12.6 Сравнение SEIF без разрежения (a) и SEIF с разрежением (b) по 4 активным ориентирам. Сравнение выполнено в имитационной среде с 50 ориентирами. В каждом ряду слева показан набор связей в фильтре, в центре – матрица корреляции, а справа – нормализованная информационная матрица. Очевидно, разреженный SEIF сохраняет намного меньше связей, но результат имеет значительно меньшую степень определённости, что видно по менее выраженной матрице корреляции.

12.7 Насколько разреженными должны быть SEIF?

Ключевым вопросом является степень разрежённости, которую следует поддерживать в SEIF. В частности, степень разрежённости определяет количество активных признаков в SEIF. Разрежённость является компромиссом двух факторов: вычислительной эффективности SEIF и точности результата. При практической реализации алгоритма SEIF рекомендуется найти подходящий компромисс.

«Золотым стандартом» SEIF является EKF, в котором нет разрежения и который не полагается на методы релаксации для восстановления оценок состояния. В следующем сравнении характеризуются три ключевых показателя производительности, которые отделяют разреженные SEIF от EKF. Сравнение основано на имитационной среде робота, в которой робот воспринимает расстояние, направление, и идентификаторы близлежащих ориентиров.

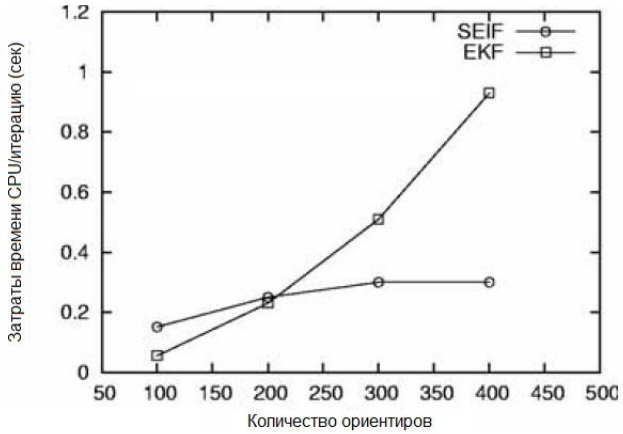


Рис. 12.7 Сравнение средних затрат процессорного времени для SEIF и EKF.

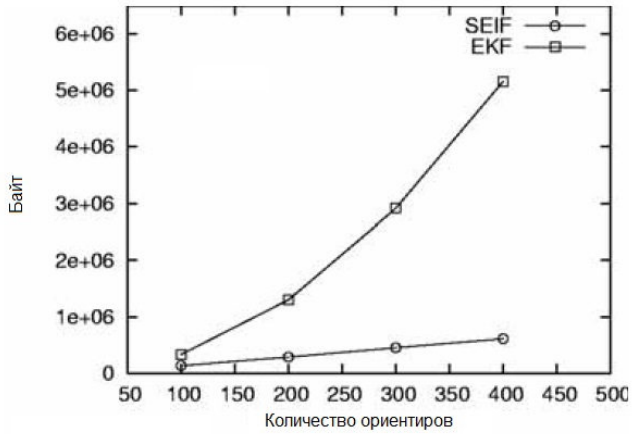


Рис. 12.8 Сравнение среднего использования памяти между SEIF и EKF.

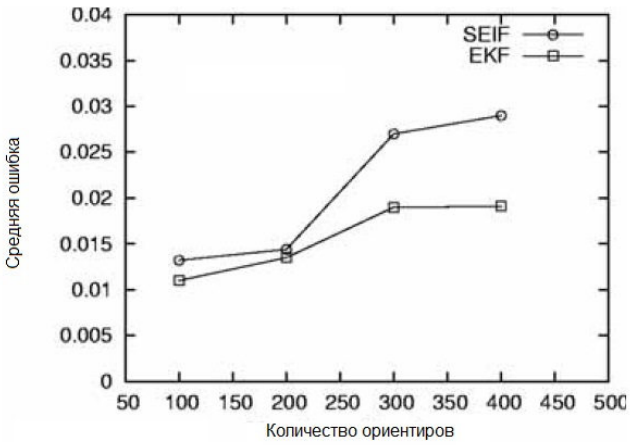


Рис. 12.9 Сравнение квадратного корня среднеквадратичной ошибки между SEIF и EKF.

1. **Вычисление.** На Рис. 12.7 показаны вычислительные затраты на обновление SEIF по сравнению с EKF. В обоих случаях реализация оптимизирована. На графике показано основное отличие вероятностного представления в фильтре от информационного. Хотя EKF действительно требуют времени, квадратично зависящего от размера карты, SEIF выполняет выравнивание, что требуют постоянного времени.

2. **Память.** На Рис. 12.8 приводится сравнение использования памяти EKF и SEIF. И снова, EKF квадратично возрастает, а SEIF возрастёт линейно, в силу разрежённости информационного представления.

3. **Точность.** Здесь EKF превосходит SEIF потому, что SEIFS требует аппроксимации для поддержания разреженности, а также при восстановлении оценки состояния μ_t. Это показано на Рис. 12.9, где приведены графики ошибки обоих методов в зависимости от размера карты.

Один из способов оценки эффекта разреженности – это имитационный эксперимент. На Рис. 12.10 показаны время обновления и ошибка аппроксимации как функция количества активных ориентиров в обновлении SEIF для карты, состоящей из 50 ориентиров. Время обновления монотонно убывает с увеличением количества ориентиров. На Рис. 12.11 показан соответствующий график ошибки, в сравнении EKF с SEIF с различной степенью разреженностью. Сплошной линией показан алгоритм SEIF, а пунктирная соответствует SEIF с явным восстановлением μ_t. Как показано на графике, для 6 активных признаках результаты сходные, при значительной более высокой вычислительной эффективности по сравнению с EKF. Для меньшего количества активных признаков ошибка драматически возрастает. Тщательная реализация SEIF потребует от экспериментатора настройки этого важного параметра, и отображения эффекта влияния на ключевые показатели, как показано на графике.

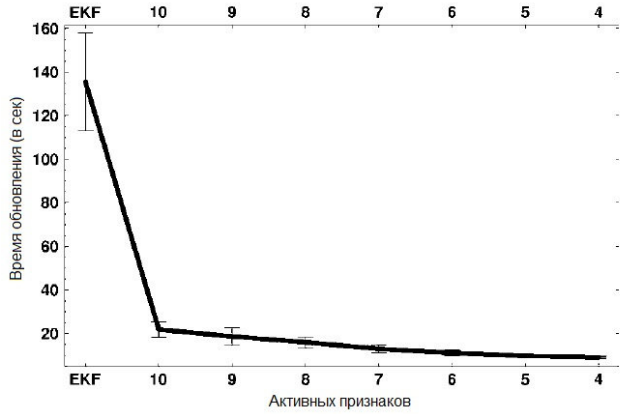


Рис. 12.10 Время обновления EKF (самая левая точка) и SEIF для различных степеней разреженности на основании количества активных признаков.

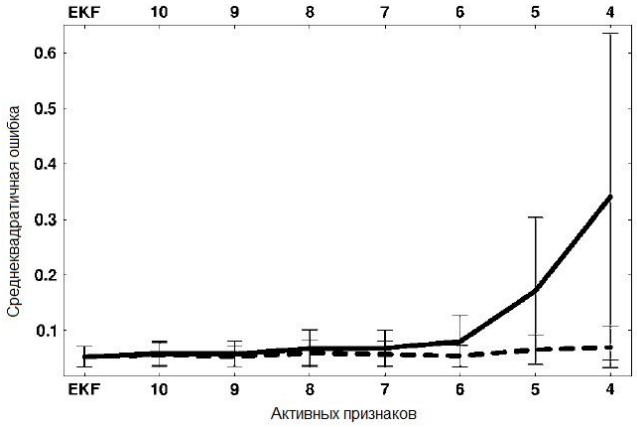


Рис. 12.11 Ошибка аппроксимации EKF (слева) и SEIF для разных степеней разреженности. На обеих схемах карта состоит из 50 ориентиров.

12.8 Инкрементная ассоциация данных

Обратим внимание на проблему ассоциации данных в SEIF. Первым методом будет уже знаковый инкрементный подход, жадно идентифицирующий наиболее вероятное соответствие, а затем принимающем это значение как истину. Мы уже встречали экземпляр такого жадного метода ассоциации данных в разделе 10.3 при обсуждении ассоциации данных в EKF. Фактически, единственная разница между жадной инкрементной ассоциацией данных в SEIF и EKF лежит в способе вычисления вероятности ассоциации данных. В общем, вычисление вероятности более сложно в информационном фильтре, чем в вероятностном фильтре, таком как EKF, поскольку информационный фильтр не отслеживает ковариаций.

12.8.1 Вычисление инкрементных вероятностей ассоциации данных

Как и прежде, вектор ассоциации данных в момент времени t будет обозначаться c_t. Жадный инкрементный метод сохраняет набор гипотез ассоциации данных, обозначенных $\hat{c}_{1:t}$. В инкрементном режиме заданы оценки соответствия $\hat{c}_{1:t-1}$ из предыдущих обновлений при вычислении $\hat{c}_t$. Шаг ассоциации данных затем переходит в оценку наиболее вероятного значения для переменной ассоциации данных $\hat{c}_t$ в момент времени t. Это достигается следующей функцией оценки максимального правдоподобия:

(12.41)

$$\begin{aligned}\hat{c}_t &= \operatorname*{argmax}_{c_t} p(z_t|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}, c_t) \\ &= \operatorname*{argmax}_{c_t} \int p(z_t|y_t, c_t) \underbrace{p(y_t|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1})}_{\bar{\Omega}_t, \bar{\xi}_t} dy_t \\ &= \operatorname*{argmax}_{c_t} \int\int p(z_t|x_t, y_{c_t}, c_t) p(x_t, y_{c_t}|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) dx_t \, dy_{c_t}\end{aligned}$$

Наша запись $p(z_t|x_t, y_{c_t}, c_t)$ модели датчика явно указывает переменную соответствия c_t. Явное вычисление этой вероятности за постоянное время невозможно, поскольку включает приведение к пределу почти всех переменных на карте. Однако, тот же самый тип аппроксимации, который был важен для эффективного разрежения, можно применить и здесь.

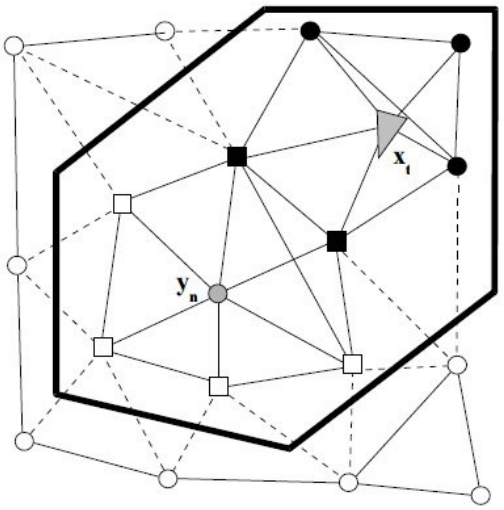


Рис. 12.12 Комбинированное марковское одеяло признака y_n и наблюдаемых признаков обычно достаточно для аппроксимации апостериорной вероятности местоположения признака, отбрасывая все остальные признаки.

В частности, обозначим как $m^+_{c_t}$ комбинированное марковское одеяло положения робота x_t и ориентира y_{c_t}. Марковское одеяло – это набор всех признаков на карте, соединённых с роботом или ориентиром y_{c_t}. На Рис. 12.12 показан такой набор. Стоит заметить, что $m^+_{c_t}$, по определению, включает

все активные ориентиры. Разрежённость $\bar{\Omega}_t$ гарантирует, что $m^+_{c_t}$ содержит только фиксированное количество признаков, независимо от размера карты N. Если марковские одеяла x_t и y_{c_t} не пересекаются, добавляются больше признаков, отражающих кратчайший путь в информационном графе между x_t и y_{c_t}.

Все оставшиеся признаки будут совместно относиться к $m^-_{c_t}$:

(12.42)
$$m^-_{c_t} = m - m^+_{c_t} - \{y_{c_t}\}$$

Множество $m^-_{c_t}$ содержит только признаки с малой степенью воздействия на целевые переменные x_t и y_{c_t}. SEIF аппроксимирует вероятность $p(x_t, y_{c_t}|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1})$ в выражении (12.41), игнорируя это непрямое влияние:

(12.43)
$$\begin{aligned}
&p(x_t, y_{c_t}|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) \\
&\quad = \iint p(x_t, y_{c_t}, m^+_{c_t}, m^-_{c_t}|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) dm^+_{c_t}\, dm^-_{c_t} \\
&\quad = \iint p(x_t, y_{c_t}|m^+_{c_t}, m^-_{c_t}, z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) \\
&\qquad\quad p(m^+_{c_t}|m^-_{c_t}, z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) p(m^-_{c_t}|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) dm^+_{c_t}\, dm^-_{c_t} \\
&\quad \approx \int p(x_t, y_{c_t}|m^+_{c_t}, m^-_{c_t} = \mu^-_{c_t}, z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) \\
&\qquad\quad p(m^+_{c_t}|m^-_{c_t} = \mu^-_{c_t}, z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}) dm^+_{c_t}
\end{aligned}$$

Эту вероятность можно вычислить за постоянное время, если набор переменных, задействованных в этом вычислении, не зависит от размера карты (обычно, зависит). Полностью аналогично различным приведённым выше выводам, заметим, что аппроксимация апостериорной вероятности получается простым извлечением субматрицы, соответствующей двум целевым переменным:

(12.44)
$$\Sigma_{t:c_t} = F^T_{x_t, y_{c_t}} (F^T_{x_t, y_{c_t}, m^+_{c_t}} \Omega_t F_{x_t, y_{c_t}, m^+_{c_t}})^{-1} F_{x_t, y_{c_t}}$$

(12.45)
$$\mu_{t:c_t} = \mu_t F_{x_t, y_{c_t}}$$

Это вычисление занимает постоянное время, поскольку включает матрицу размера, независимого от N. Из этой гауссовой функции легко можно получить искомую вероятность измерения в уравнении (12.41).

Так же как в алгоритме EKF SLAM, признаки помечаются новыми, когда правдоподобие $p(z_t|z_{1:t-1}, u_{1:t}, \hat{c}_{1:t-1}, c_t)$ остаётся ниже порога α. После этого просто установим $\hat{c}_t = N_{t-1}+1$ и $N_t = n_{t-1}+1$, поскольку, в противном случае, размер карты останется неизменным в силу $N_t = N_{t-1}$. Значение $\hat{c}_t$ выбирается таким образом, чтобы максимизировать вероятность ассоциации данных.

Последним препятствием является тот факт, что, иногда, комбинации марковских одеял оказывается недостаточно, поскольку в них нет пути между положением робота и ориентиром, который проверяется на соответствие. Обычно это происходит при замыкании большого цикла в среде. Здесь необходимо дополнить набор признаков $m^+_{c_t}$ набором ориентиров

вдоль, по крайней мере, одной траектории между m_{c_t} и положением робота x_t. В зависимости от размера цикла количество ориентиров, требуемых для результирующего множества, может зависеть от размера карты N. Оставим детальное рассмотрение такого дополнения в качестве упражнения.

12.8.2 Практические соображения

В общем, инкрементный метод жадной ассоциации данных неустойчив, как обсуждалось в главах, посвящённых EKF SLAM. Ошибочные измерения могут легко привести к возникновению неверных ассоциаций и вызвать существенные ошибки в оценке SLAM. Стандартным способом избежать этого (как для EKF, так и для SEIF) является создание временного списка ориентиров. Этот метод уже детально обсуждался в подразделе 10.3.3 в контексте EKF SLAM. Во временный список кандидатов, который сохраняется отдельно от SEIF, добавляется любой новый признак, который перед этим не наблюдался.

Рис. 12.13 Автомобиль, используемый в экспериментах, оборудован двухмерным лазерным дальномером и дифференциальной системой GPS. Собственное движение автомобиля измеряется датчиком линейного дифференциального преобразования для угла поворота колес, и энкодером измерения скорости на колесе. На заднем плане видна территория парка Виктория, используемая как среда для тестирования. Изображение принадлежит Хосе Гюванту и Эдуардо Неботу из Австралийского центра полевой робототехники (José Guivant and Eduardo Nebot, Australian Centre for Field Robotics).

На последующем этапе измерений вновь найденные кандидаты проверяются по всем кандидатам из списка ожидания, совпадения увеличивает вес соответствующих кандидатов, отсутствие наблюдения близлежащего признака уменьшало его вес. Когда вес кандидата превышает определённый порог, он пополняет сеть признаков SEIF.

Заметим, что ассоциация данных нарушает свойство постоянных затрат времени в SEIF. Это происходит при вычислении ассоциации данных, когда необходимо проверить множество признаков. Если бы удалось гарантировать, что все полезные признаки уже присоединены к SEIF короткой траекторией к множеству активных переменных, станет возможным выполнять ассоциацию данных за постоянное время. Таким образом, структура SEIF естественно обеспечивает поиск наиболее вероятного признака, заданного измерением. Однако, такое решение непригодно при закрытии цикла в первый раз, когда верная ассоциация может оказаться слишком далеко в графе соседства SEIF.

Кратко обратим внимание на реализацию SEIF алгоритма на физическом транспортном средстве. Используемые здесь данные являются общим ориентиром в области SLAM. Этот набор данных был собран с помощью оснащённого датчиками автомобилем, который проехал через парк в Сиднее, Австралия.

Автомобиль и окружающая среда показаны на Рис. 12.13 и 12.14, соответственно. Автомобиль оборудован лазерным датчиком расстояния SICK и системой измерения угла поворота руля и поступательной скорости. Лазер предназначался для обнаружения деревьев в окружающей среде, но также обнаружил сотни несуществующих признаков, такие, как углы движущихся по соседней автостраде машин. Данные исходной одометрии, используемые нами, очень некачественные и дают ошибку в несколько сотен метров при использовании интегрирования вдоль пути автомобиля длиной 3,5 км что показано на Рис. 12.14. Плохое качество данных одометрии вкупе с наличием множества ошибочных признаков делает этот набор данных очень подходящим для тестирования алгоритмов SLAM.

Рис. 12.14 Тестовая окружающая среда: Участок размером 350 на 350 метров в парке Виктория в Сиднее. Сверху наложен пройденный путь на основе показаний одометрии. Данные и снимок с воздуха принадлежат Хосе Гюванту и Эдуардо Неботу из Австралийского центра полевой робототехники (José Guivant and Eduardo Nebot, Australian Centre for Field Robotics). Результаты принадлежат Майклу Монтемерло из университета Стэнфорда (Michael Montemerlo, Stanford University).

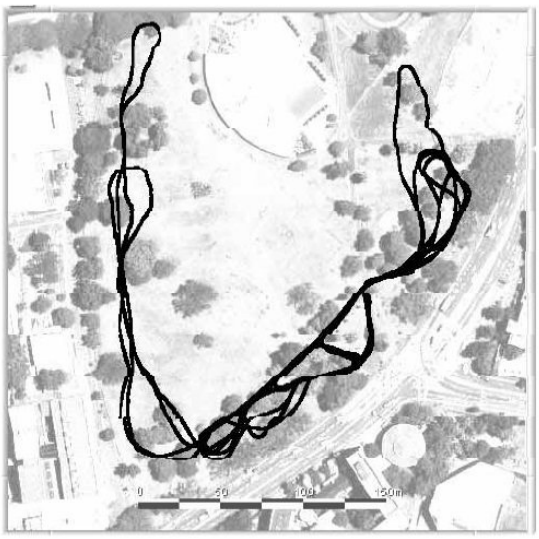


Рис. 12.15 Траектория пути, восстановленная с помощью SEIF с точностью ±1 м. Собственность Майкла Монтемерло из университета Стэнфорда (Michael Montemerlo, Stanford University)

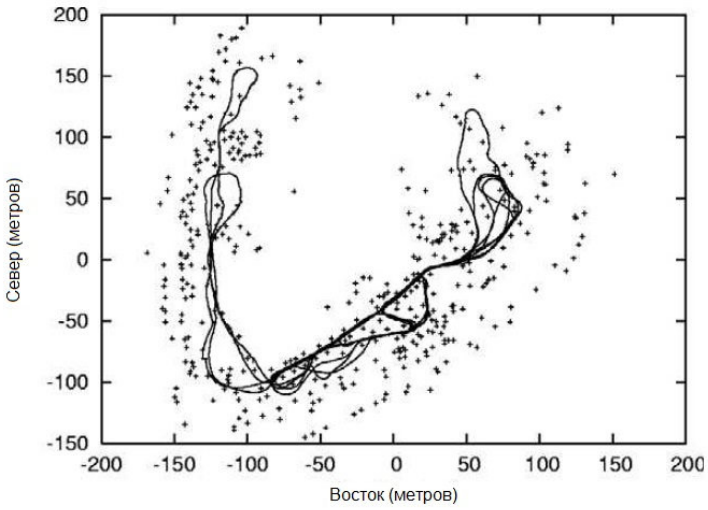


Рис. 12.16 Наложение оценок местоположений ориентиров и пути автомобиля. Изображения принадлежат Майклу Монтемерло из университета Стэнфорда (Michael Montemerlo, Stanford University)

Траектория, восстановленная с помощью SEIF, показана на Рис. 12.15. Эта траектория пути вычислительно неотличима от созданного EKF. Средняя ошибка местоположения, измеренная дифференциальным датчиком GPS, составляет менее 0,5 метра, что весьма мало по сравнении с общей длиной пути 3,5 км. Соответствующая карта ориентиров показана на Рис 12.16. Она тоже имеет точность, сравнимую с результатами наиболее современных алгоритмов EKF. В сравнении с алгоритмом EKF SEIF выполняется, примерно в два раза быстрее и потребляет около четверти памяти, требуемой для EKF. Эта экономия относительно мала, что является результатом

малого размера карты. Фактически, наибольшее количество времени было потрачено на предобработку данных датчиков и для карт большего размера относительная экономия будет выше.

12.9 Ассоциация данных методом ветвей и границ

SEIF позволяет определить полностью иной метод ассоциации данных, который даст оптимальные результаты (хотя, возможно, за экспоненциальное время). Метод построен на трёх ключевых принципах:

МЯГКИЕ ОГРАНИЧЕНИЯ АССОЦИАЦИИ ДАННЫХ

• Так же как GraphSLAM, SEIF позволяет добавлять мягкие ограничения ассоциации данных. Для двух заданных признаков m_i и m_j, мягкое ограничение ассоциации данных – это всего лишь информационная связь, которая принудительно уменьшает расстояние между m_i и m_j. Примеры таких мягких связей уже приводились в предыдущей главе. В разреженных обобщённых информационных фильтрах добавление такой связи представляет собой просто добавление значений в информационную матрицу.

• Мягкие ограничения ассоциации легко удалить. Также, как добавление нового ограничение представляет собой всего лишь локальное добавление в информационную матрицу, удаление представляет собой локальное вычитание. Такая «обратная» операция может быть применена к произвольным связям ассоциации данных, независимо от того, когда они были добавлены, или когда в последний раз наблюдался соответствующий признак. Это делает возможным пересмотр прошлых решений об ассоциации данных.

• Возможность свободно добавлять и удалять ассоциации данных позволяет выполнять поиск по дереву возможных ассоциаций данных эффективным и целостным способом, как это будет показано ниже.

Для разработки алгоритма ассоциации данных методом ветвей и границ, будет полезно определить дерево ассоциации данных, определяющее последовательность решений по ассоциации данных со временем. В каждый момент времени каждый наблюдаемый признак может быть ассоциирован с несколькими другими признаками или же считаться новым, прежде ненаблюдаемым признаком. Результирующее дерево решений об ассоциации данных, от начального момента в момент времени $t = 1$ до текущего момента, показано на Рис. 12.17a. Конечно, дерево экспоненциально разрастётся со временем, поэтому тщательный поиск в нем невозможен. Инкрементный жадный алгоритм, описанный в предыдущем разделе, напротив, следует по единственному пути вдоль дерева, определённому наиболее вероятными ассоциациями данных. Такой путь показан на Рис. 12.17a жирной серой кривой.

Очевидно, если инкрементный жадный метод будет успешно работать, результирующий путь оптимален. Однако, этот метод может и не работать, поскольку в случае неверного решения инкрементный метод неспособен восстановиться. Более того, неверные решения ассоциации данных вносят ошибки в карту, которые, затем, могут привести к ещё большим ошибкам ассоциации данных.

12.9.1 Рекурсивный поиск

Этот метод будет обсуждаться в оставшейся части главы и призван обобщить инкрементый жадный алгоритм до полномасштабного алгоритма поиска в оптимальном дереве. Конечно, поиск всех ветвей в дереве выполнять неразумно.

ПЕРИМЕТР

Однако, если сохранять логарифм правдоподобия для всех узлов по *периметру* разрастающегося дерева, оптимальность можно гарантировать. Эта идея показана на Рис. 12.17b: алгоритм SEIF методом ветвей и границ сохраняет не только один путь по дереву ассоциации данных, но весь периметр. Каждый раз при расширении узла (например, с помощью инкрементного максимума правдоподобия), также оцениваются все альтернативные решения и сохраняются все значения соответствующих правдоподобий. Это показано на Рис. 12.17b, где приводится логарифм правдоподобия для всего периметра дерева.

Нахождение максимума в выражении (12.41) предполагает, что логарифм правдоподобия избранного листа выше или равен другому листу на такой же глубине. Поскольку логарифм правдоподобия равномерно убывает с глубиной дерева, можно гарантировать, что ассоциация данных оптимальна, если логарифм правдоподобия избранного листа выше или равен логарифму правдоподобия любого другого узла на периметре. Другими словами, когда узел на периметре получает правдоподобие выше, чем у выбранного листа, это может означать возможность ещё больше увеличить правдоподобие данных, пересмотрев прошлые решения ассоциации данных. В нашем методе такие узлы периметра просто расширяются. Если расширение достигло листа и его значение выше, чем у текущего наилучшего листа, этот лист выбирается для новой ассоциации данных. Поиск прерывается, если все листы периметра имеют меньших или одинаковый параметр правдоподобия с текущим выбранным листом. Этот метод гарантирует сохранение лучшего множества значений переменных для ассоциации данных, но, время от времени, может требовать довольно долгого поиска.

12.9.2 Вычисление вероятности произвольной ассоциации данных

Для проверки необходимости связи между двумя признаками на карте нам понадобится метод вычисления вероятность их равенства. Этот метод проверки аналогичен методу проверки соответствия в GraphSLAM в Таблице 11.8 на странице 336. Однако, в SEIF эта проверка приблизительна, поскольку точное вычисление логарифма правдоподобия может потребовать дополнительных вычислений.

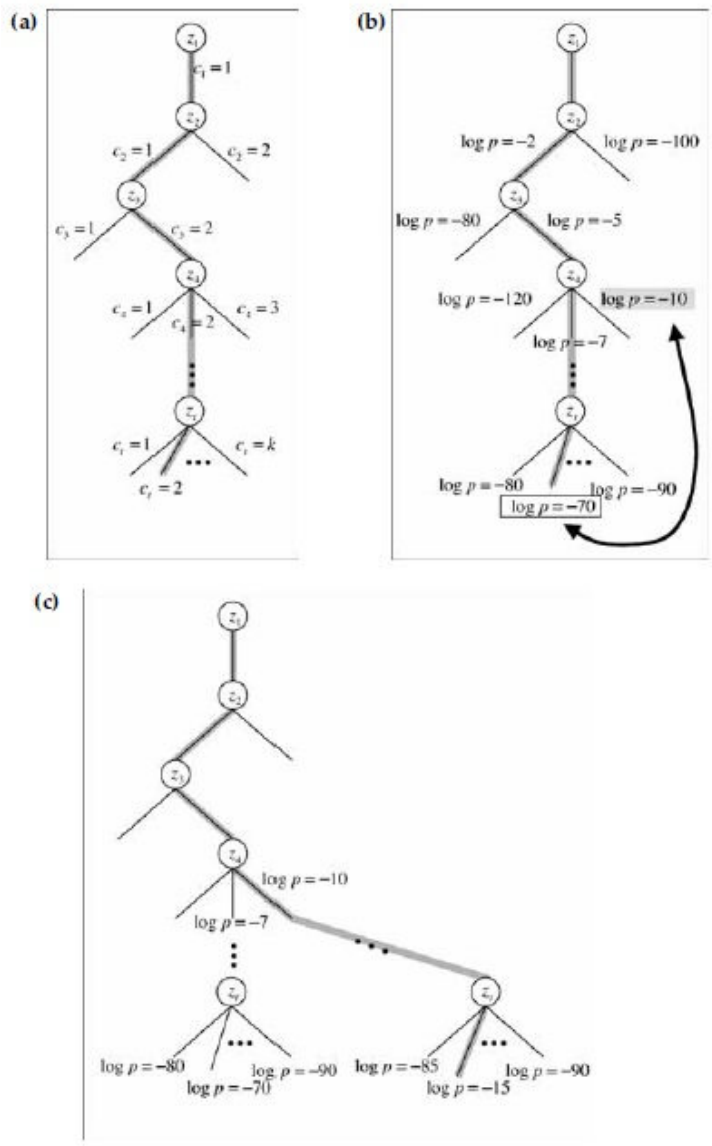


Рис. 12.17 Дерево ассоциации данных, коэффициент ветвления которого растёт с увеличением числа ориентиров на карте(a). Алгоритм SEIF на основе дерева поддерживает логарифм правдоподобия для всего периметра расширенных узлов, позволяя находить альтернативные пути(b). Улучшенная траектория пути (c).

1: **Algorithm SEIF_correspondence_test**(Ω, μ, m_j, m_k) :
2: *пусть* $B(j)$ *будет одеялом* m_j
3: *пусть* $B(k)$ *будет одеялом* m_k
4: $B = B(j) \cup B(k)$
5: *if* $B(j) \cap B(k) = \emptyset$
6: *добавить признаки по кратчайшему пути между* m_i *и* m_j *и* B
7: *endif*
8: $F_B = \begin{pmatrix} 0...0 & 1 & 0 & 0 & 0...0 & \cdots & & & & & \\ 0...0 & 0 & 1 & 0 & 0...0 & \cdots & & & & & \\ 0...0 & 0 & 0 & 1 & 0...0 & \cdots & & & & & \\ & \cdots & & & 0...0 & 1 & 0 & 0 & 0...0 & & \\ & \cdots & & & 0...0 & 0 & 1 & 0 & 0...0 & & \\ & \cdots & & & 0...0 & 0 & 0 & 1 & 0...0 & & \\ & & & & & & & & & \ddots & \\ & & & & \cdots & & & & & & 0...0 \\ & & & & \cdots & & & & & & 0...0 \end{pmatrix}$
9: $(\textit{размер}\,(3N+3)\ \textit{на}\, 3|B|)$
10: $\Sigma_B = (F_B \Omega F_B^T)^{-1}$
11: $\mu_B = \Sigma_B F_B \xi$
12: $F_\Delta = \begin{pmatrix} 0...0 & 1 & 0 & 0...0 & -1 & 0 & 0...0 \\ 0...0 & \underbrace{0}_{\text{признак } m_j} & \underbrace{1}_{\text{признак } m_j} & 0...0 & \underbrace{0}_{\text{признак } m_j} & \underbrace{-1}_{\text{признак } m_j} & 0...0 \end{pmatrix}$
13: $\Sigma_\Delta = (F_\Delta \Omega F_\Delta^T)^{-1}$
14: $\mu_\Delta = \Sigma_\Delta F_\Delta \xi$
15: *return* $\det(2\pi\Sigma_\Delta)^{-\frac{1}{2}} \exp\{-\frac{1}{2}\mu_\Delta^T \Sigma_\Delta^{-1} \mu_\Delta\}$

Таблица 12.6 Проверка соответствия для SEIF SLAM.

В Таблице 12.6 приведён алгоритм, который проверяет вероятность того, что два признака на карте соответствуют одному ориентиру. Такой проверки достаточно для реализации метода жадной ассоциации данных. Ключевое вычисление здесь состоит в восстановлении объединённой ковариации и среднего вектора на небольшом множестве признаков карты B. Для определения идентичности двух признаков на карте, в SEIF необходимо учитывать информационные связи между ними. Технически, чем больше связей принимается во внимание, тем более точным будет результат, но лишь ценой дополнительных вычислений. На практике обычно достаточно идентифицировать два *марковских одеяла* интересующих признаков. Марковское одеяло для признака - это сам признак и все прочие признаки, соединённые с ним ненулевым элементом информационной матрицы. В большинстве случаем марковские одеяла пересекаются. Если этого не происходит, алгоритм в Таблице 12.6 находит путь между ориентирами (который должен существовать, если оба признака наблюдались одним и тем же роботом).

МАРКОВСКОЕ ОДЕЯЛО

Затем алгоритм в Таблице 12.6 вырезает локальные информационную матрицу и информационный вектор, выполняя тот же самый математический «фокус», который позволял эффективный осуществлять разрежение: SEIF выбрасывает признаки за пределами марковских одеял. В результате, в алгоритме SEIF появляется эффективный метод вычисления искомой вероятности, пусть и приблизительный (из-за отбрасывания переменных),

но очень хорошо работающий на практике.

Результат интересен тем, что не только позволяет SEIF принимать решения по ассоциации данных, но и предоставляет способ вычисления логарифма правдоподобия такого решения. Логарифм результата такой процедуры соответствует логарифму правдоподобия отдельного элемента данных, а суммирование по траектории в дереве ассоциации данных даёт логарифм правдоподобия всех данных для конкретной ассоциации.

12.9.3 Ограничения эквивалентности

Как только два признака на карте были определены в качестве эквивалентных при поиске ассоциации данных, в SEIF добавляется мягкая связь в информационную матрицу. Допустим, первый признак m_i, а второй - m_j. Мягкая связь ограничивает их местоположение одним и тем же местом с помощью следующего экспоненциально-квадратичного ограничения

(12.46)
$$\exp\{-\frac{1}{2}(m_i - m_j)^T C(m_i - m_j)\}$$

Здесь C - диагональная матрица штрафа типа

(12.47)
$$C = \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix}$$

На практике диагональные элементы C большими положительными значениями, и чем больше эти значения, тем сильнее ограничение.

Легко увидеть, что ненормализованный гауссиан (12.46) может быть записан в информационной матрице в виде связи между m_i и m_j. Определим матрицу проекции

(12.48)
$$F_{m_i - m_j} = \begin{pmatrix} 0...0 & 1 & 0 & 0 & 0...0 & -1 & 0 & 0 & 0...0 \\ 0...0 & 0 & 1 & 0 & 0...0 & 0 & -1 & 0 & 0...0 \\ 0...0 & \underbrace{0 & 0 & 1}_{m_i} & 0...0 & \underbrace{0 & 0 & -1}_{m_j} & 0...0 \end{pmatrix}$$

Матрица проецирует состояние y_t в виде разницы $m_i - m_j$. Отсюда, выражение (12.46) становится

(12.49)
$$\begin{aligned} &\exp\{-\frac{1}{2}(F_{m_i - m_j} y_t)^T C\,(F_{m_i - m_j} y_t)\} \\ &\quad = \exp\{-\frac{1}{2} y_t^T [F_{m_i - m_j}^T C\, F_{m_i - m_j}] y_t\} \end{aligned}$$

Для реализации этого мягкого ограничения в SEIF необходимо добавить $F_{m_i - m_j}^T C\, F_{m_i - m_j}$ к информационной матрице, оставляя информационный вектор неизменным:

(12.50)
$$\Omega_t \longleftarrow \Omega_t + F_{m_i - m_j}^T C F_{m_i - m_j}$$

Конечно, добавочный член разрежен, поскольку содержит ненулевые элементы вне главной диагонали только между признаками m_i и m_j. После добавления мягкой связи ее можно удалить с помощью инверсии

(12.51)
$$\Omega_t \longleftarrow \Omega_t - F^T_{m_i - m_j} C F_{m_i - m_j}$$

Это удаление может быть выполнено вне зависимости от времени, которое прошло с момента добавления ограничения в фильтр. Однако, требуется внимательность, чтобы не позволить SEIF удалить несуществующее ограничение, в противном случае информационная матрица уже не будет неотрицательно определённой, и результирующая оценка может не соответствовать правильному вероятностному распределению.

12.10 Практические соображения

В любой конкурентной реализации этого метода обычно будет существовать только небольшое количество путей ассоциации данных, применимых в произвольный момент времени. При замыкании цикла в среде внутри помещения обычно имеется только три разумные гипотезы: закрытие, продолжить влево и продолжить вправо. Но вероятность быстро уменьшается, поэтому количество проходов поиска по дереву обычно достаточно мало.

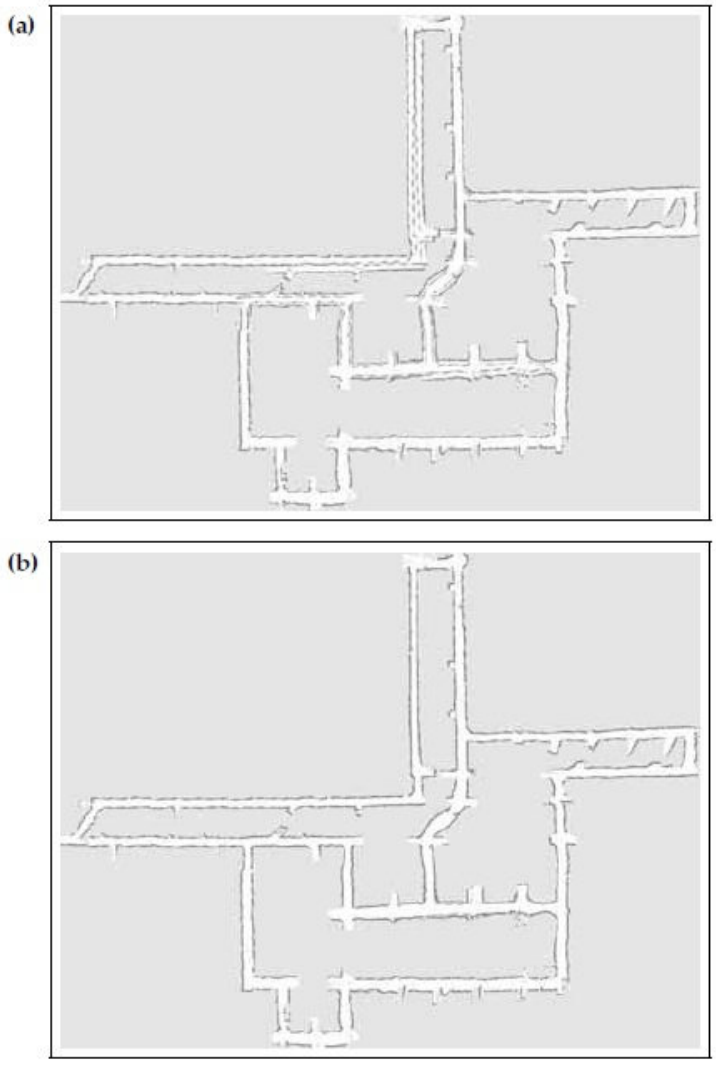


Рис. 12.18 Карта с инкрементным наложением сканирований по методу ML(a) и полной рекурсивной ассоциации данных по методу ветвей и границ (b). Изображения принадлежат Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

Одним из способов увеличения эффективности ассоциации данных является учёт отрицательной информации измерения. Датчики расстояния, используемые в предложенных реализациях, возвращают как положительную, так и отрицательную информацию о наличии объектов окружающего мира. Положительная информация – это обнаружения объектов, отрицательная описывает расстояние между датчиком и объектом. Тот факт, что робот не смог обнаружить объект ближе текущих показаний датчика, даёт информацию об отсутствии объекта в диапазоне измерений.

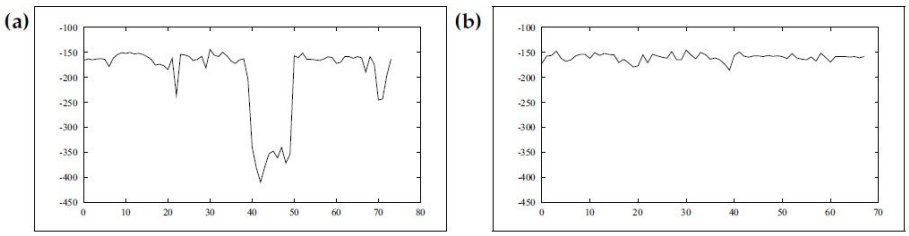


Рис. 12.19 Логарифм правдоподобия настоящего измерения как функция времени. Меньшее правдоподобие вызвано неверной ассоциацией (a). Логарифм правдоподобия при рекурсивном восстановлении с помощью поиска по дереву (b). Успешность определяется по отсутствию выраженных провалов.

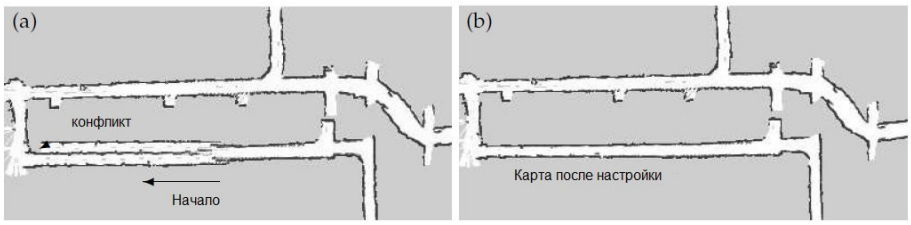


Рис. 12.20 Пример метода ассоциации данных на основе деревьев: при закрытии большого цикла, робот ошибочно принимает решение о наличии второго, параллельного пути (a). Эта модель теряет целостность, когда робот достигает правого поворота. В этой точке выполняется рекурсивный поиск лучшего метода ассоциации данных, что приводит к построению карты справа (b).

Метод, который оценивает эффект нового ограничения на общее правдоподобие, учитывает оба типа информации - и положительную, и отрицательную. Это выполняется вычислением попарного соответствия (или несоответствия) двух проходов сканирования при оценке положения. При использовании датчиков расстояния одним из способов получения комбинации положительной и отрицательной информации является наложение сканирования на локальную карту сетки занятости, построенную по результатам другого сканирования. Это очевидный способ определить примерную вероятность соответствия двух локальных карт, с учётом положительной, и отрицательной информации.

В оставшейся части этого раздела описаны практические результаты, полученные при использовании SEIF с ассоциацией данных методом деревьев. Слева на Рис. 12.18a показан результат инкрементной ассоциации данных по методу максимального правдоподобия, что эквивалентно регулярному инкрементному сравнению проходов сканирования. Очевидно, некоторые коридоры на карте показаны дважды, показывая недостатки метода максимального правдоподобия. Для сравнения справа показан результат. Хорошо видно, что карта более точная, по сравнению с созданной методом инкрементного максимального правдоподобия.

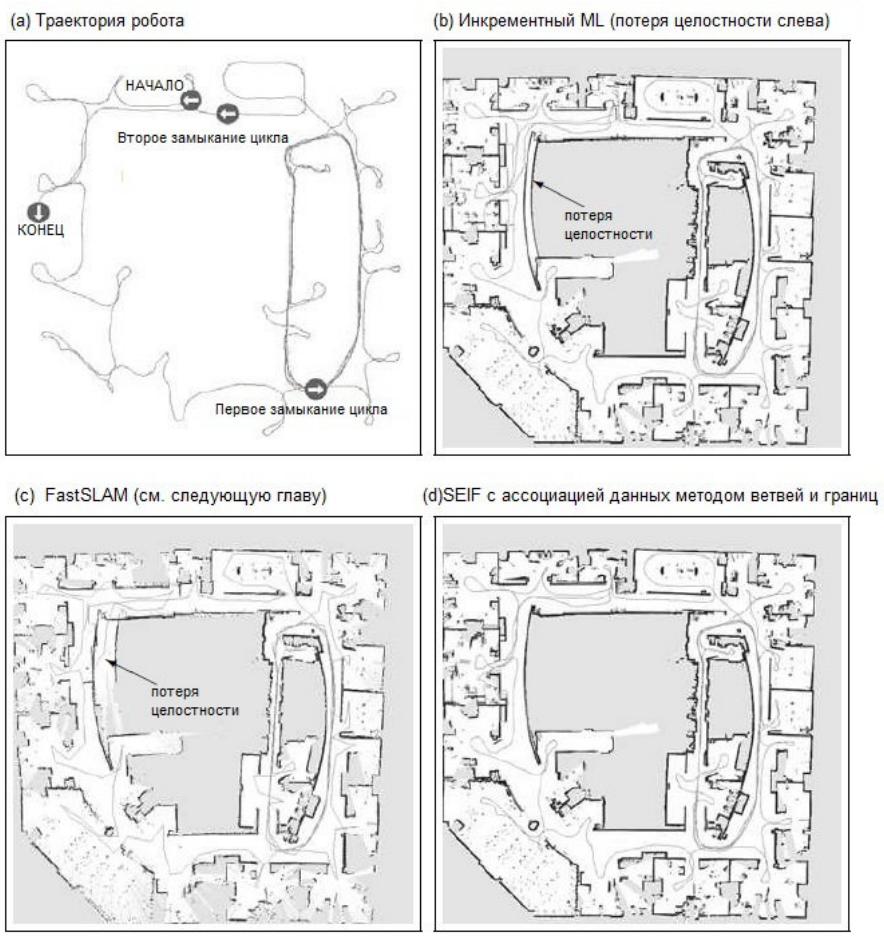


Рис. 12.21 Траектория робота (a). Инкрементный ML (сравнение сканирований)(b), Fast SLAM(c). SEIF с ленивой ассоциацией данных (d). Изображение принадлежит Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

На Рис. 12.19a показан логарифм правдоподобия самого последнего измерения (не всей траектории), который существенно падает с потерей целостности карты. В этой точке SEIF выполняет поиск альтернативных значений ассоциации данных. Он быстро обнаруживает "верное", создавая карту, изображённую на Рис. 12.18b. Рассматриваемая зона показана на Рис. 12.20, иллюстрирующем момент резкого спада логарифма правдоподобия. Сам логарифм правдоподобия измерения показан на Рис. 12.19b.

Наконец, на Рис. 12.21 сравниваются различные методы в контексте картографирования большого здания с несколькими циклами.

12.11 SLAM с несколькими роботами

Алгоритм SEIF также применим к задачам *SLAM с использованием нескольких роботов*. Задача использования нескольких роботов в SLAM включает независимое исследование и картографирование несколькими роботами с конечной целью интеграции нескольких карт в одну монолитную карту. Во многих отношениях задача SLAM с несколькими роботами напоминает

картографирование одним роботом, когда данные необходимо интегрировать в единое апостериорное распределение со временем. Однако, задача для нескольких роботов существенно сложнее в ряде аспектов:

• При отсутствии априорной информации об относительном местоположении двух роботов проблема соответствия становится *глобальной*. Главным образом, любые два признака на карте могут соответствовать, и робот способен определить верные соответствия только путём сравнения многих признаков.

• Все карты будут получены в локальных системах координат, которые могут отличаться по абсолютным местоположениям и ориентациям по направлению. Перед интегрированием двух карт их необходимо совместить путём смещения и поворота. В SEIF это потребует повторной линеаризации информационной матрицы и вектора.

• Степень наложения двух карт заранее неизвестна. Например, роботы могут действовать на разных этажах здания с одинаковой поэтажной планировкой. В такой ситуации возможность различить карты будет основана на мелких отличающихся признаках окружающей среды, например, по-разному расставленная мебель.

В этом разделе будут эскизно намечены некоторые идеи реализации алгоритма картографирования с помощью нескольких роботов. Будет представлен алгоритм интегрирования двух карт после установки соответствия. Также будут обсуждаться, без приведения доказательства, методы установки глобального соответствия для SLAM с использованием нескольких роботов.

12.11.1 Интегрирование карт

Критические моменты для слияния карт с известным соответствием показаны в Таблице 12.7. Этот алгоритм принимает на вход два локальных апостериорных Распределения, выраженные в информационном виде Ω^j, ξ^j и Ω^k, ξ^k, соответственно. Также требуются три других элемента:

1. Набор значений линейного смещения d
2. Относительный угол поворота α
3. Набор соответствия признаков $\mathcal{C}^{j,k}$

1: **Algorithm SEIF_map_fusion**$(\Omega^j, \xi^j, \Omega^k, \xi^k, d, \alpha, \mathcal{C})$:

2: $\Delta = (d_x\ d_y\ \alpha\ d_x\ d_y\ 0\ ...\ d_x\ d_y\ 0)^T$

3: $\mathcal{A} = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 & & & \dots & 0 \\ -\sin\alpha & \cos\alpha & 0 & & & & \vdots \\ 0 & 0 & 1 & & & & \\ & & & \ddots & & & \\ & & & & \cos\alpha & \sin\alpha & 0 \\ \vdots & & & & -\sin\alpha & \cos\alpha & 0 \\ 0 & \dots & & & 0 & 0 & 1 \end{pmatrix}$

4: $\Omega^{j\longrightarrow k} = \mathcal{A}\Omega^j\mathcal{A}^T$

5: $\xi^{j\longrightarrow k} = \mathcal{A}(\xi^j - \Omega^{j\longrightarrow k}\Delta)$

6: $\Omega = \begin{pmatrix} \Omega^k & 0 \\ 0 & \Omega^{j\longrightarrow k} \end{pmatrix}$

7: $\xi = \begin{pmatrix} \xi^k \\ \xi^{j\longrightarrow k} \end{pmatrix}$

8: *для любой пары* $(m_j, m_k) \in \mathcal{C}^{j,k}$ *выполнять*

9: $F = \begin{pmatrix} 0...0 & 1 & 0 & 0 & 0...0 & -1 & 0 & 0 & 0...0 \\ 0...0 & 0 & 1 & 0 & 0...0 & 0 & -1 & 0 & 0...0 \\ 0...0 & \underbrace{0}_{m_j} & \underbrace{0}_{m_j} & \underbrace{1}_{m_j} & 0...0 & \underbrace{0}_{m_k} & \underbrace{0}_{m_k} & \underbrace{-1}_{m_k} & 0...0 \end{pmatrix}$

10: $\Omega \longleftarrow \Omega + F^T \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix} F$

11: *endfor*

12: *return* Ω, ξ

Таблица 12.7 Цикл слияния карт для картографирования алгоритмом SEIF несколькими роботами.

Вектор смещения $d = (d_x\ d_y)^T$ и поворот α определяет относительную ориентацию карт двух роботов. В частности, j-е положение робота x^j и признаки карты j-го робота проектируются в систему координат k-го робота с помощью поворота на α с последующим параллельным переносом d. Здесь будем использовать обозначение "$j \longrightarrow k$" для координат элемента карты j-го робота, выраженного в системе координат k-го робота.

1. Для положения j-го робота x_t^j

(12.52)

$$\underbrace{\begin{pmatrix} x^{j\longrightarrow k} \\ y^{j\longrightarrow k} \\ \theta^{j\longrightarrow k} \end{pmatrix}}_{x_t^{j\longrightarrow k}} = \begin{pmatrix} d_x \\ d_y \\ \alpha \end{pmatrix} + \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} x^j \\ y^j \\ \theta^j \end{pmatrix}}_{x_t^j}$$

2. Для каждого признака карты j-го робота m_i^j

(12.53)

$$\underbrace{\begin{pmatrix} m_{i,x}^{j\longrightarrow k} \\ m_{i,y}^{j\longrightarrow k} \\ m_{i,s}^{j\longrightarrow k} \end{pmatrix}}_{m_i^{j\longrightarrow k}} = \begin{pmatrix} d_x \\ d_y \\ 0 \end{pmatrix} + \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} m_{i,x}^{j} \\ m_{i,y}^{j} \\ m_{i,s}^{j} \end{pmatrix}}_{m_t^j}$$

Эти две проекции выполняются в строках со 2 по 5 алгоритма **SEIF_map_fusion** в Таблице 12.7. На этом этапе выполняются локальный поворот и смещение информационной матрицы и информационного вектора, что сохраняет разрежённость SEIF. После этого выполняется слияние карт путём построения единой апостериорной карты в строках 6 и 7. На финальном шаге алгоритма слияния определяется список соответствия $C^{j,k}$. Это множество состоит из пар признаков (m_j, m_k), которые взаимно соответствуют карте робота j и роботу k. Слияние выполняется аналогично мягким ограничениям эквивалентности, упоминаемым в подразделе 12.9.3. Для любых двух соответствующих признаков просто выполняется добавление больших значений элементам информационной матрицы, соединяющим эти признаки.

Заметим, что альтернативным способом выполнить слияние карт будет свернуть соответствующие строки и столбцы результирующей информационной матрицы и вектора. Следующий пример показывает операцию сворачивания признаков 2 и 4 в фильтре, которая может произойти, когда состояния в списке соответствия признаков 2 и 4 идентичны:

(12.54)

$$\begin{pmatrix} \Omega_{11} & \Omega_{12} & \Omega_{13} & \Omega_{14} \\ \Omega_{21} & \Omega_{22} & \Omega_{23} & \Omega_{24} \\ \Omega_{31} & \Omega_{32} & \Omega_{33} & \Omega_{34} \\ \Omega_{41} & \Omega_{42} & \Omega_{43} & \Omega_{44} \end{pmatrix} \longrightarrow \begin{pmatrix} \Omega_{11} & \Omega_{12}+\Omega_{14} & \Omega_{13} \\ \Omega_{21}+\Omega_{41} & \Omega_{22}+\Omega_{42}+\Omega_{24}+\Omega_{44} & \Omega_{23}+\Omega_{43} \\ \Omega_{31} & \Omega_{32}+\Omega_{34} & \Omega_{33} \end{pmatrix}$$

(12.55)

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} \longrightarrow \begin{pmatrix} \xi_1 \\ \xi_2+\xi_4 \\ \xi_3 \end{pmatrix}$$

Сворачивание использует аддитивность информационного состояния.

12.11.2 Математический вывод интеграции карт

Для этого вывода будет достаточно заданных информационной матрицы и векторов сдвига, указанных в (12.52) и (12.53). Определим переменные δ_x, δ_m, и A следующим образом:

(12.56)

$$\delta_x = (d_x \; d_y \; \alpha)^T$$

(12.57)

$$\delta_m = (d_x \; d_y \; 0)^T$$

(12.58)

$$A = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Перепишем (12.52) и (12.53) в виде

(12.59)
$$x_t^{j\longrightarrow k} = \delta_x + Ax_t^j$$

(12.60)
$$m_t^{j\longrightarrow k} = \delta_m + Am_i^j$$

Для полного вектора состояний получаем

(12.61)
$$y_t^{j\longrightarrow k} = \Delta + \mathcal{A}y_t^j$$

с

(12.62)
$$\Delta = (\delta_r\ \delta_m\ \delta_m\ ...\ \delta_m)^T$$

(12.63)
$$\mathcal{A} = \begin{pmatrix} A_r & 0 & \cdots & 0 \\ 0 & A_m & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_m \end{pmatrix}$$

Требуемое преобразование координат в информационном пространстве выполняется схожим образом. Пусть апостериорная вероятность для j-го робота в момент времени t будет определена информационной матрицей Ω^j и информационным вектором ξ^j для сдвига и поворота используется следующее преобразование:

(12.64)

$$\begin{aligned} p(y^{j\longrightarrow k}|z_{1:t}^j, u_{1:t}^j) \\ &= \eta\ \exp\{-\frac{1}{2}y^{j\longrightarrow k,T}\Omega^{j\longrightarrow k}y^{j\longrightarrow k} + y^{j\longrightarrow k,T}\xi^{j\longrightarrow k}\} \\ &= \eta\ \exp\{-\frac{1}{2}(\Delta + \mathcal{A}y^j)^T\Omega^{j\longrightarrow k}(\Delta + \mathcal{A}y^j) + (\Delta + \mathcal{A}y^j)^T\xi^{j\longrightarrow k}\} \\ &= \eta\ \exp\{-\frac{1}{2}y^{jT}\mathcal{A}^T\Omega^{j\longrightarrow k}\mathcal{A}y^j + y^{jT}\Omega^{j\longrightarrow k}\Delta - \underbrace{\frac{1}{2}\Delta^T\Omega^{j\longrightarrow k}\Delta}_{\text{const.}} \\ &\qquad + \underbrace{\Delta^T\xi^{j\longrightarrow k}}_{\text{const.}} + y^{jT}\mathcal{A}^T\xi^{j\longrightarrow k}\} \\ &= \eta\ \exp\{-\frac{1}{2}y^{jT}\mathcal{A}^T\Omega^{j\longrightarrow k}\mathcal{A}y^j + y^{jT}\Omega^{j\longrightarrow k}\Delta + y^{jT}\mathcal{A}^T\xi^{j\longrightarrow k}\} \\ &= \eta\ \exp\left\{-\frac{1}{2}y^{jT}\underbrace{\mathcal{A}^T\Omega^{j\longrightarrow k}\mathcal{A}}_{\Omega^j}y^j + y^{jT}\underbrace{(\Omega^{j\longrightarrow k}\Delta + \mathcal{A}^T\xi^{j\longrightarrow k})}_{\xi^j}\right\} \end{aligned}$$

Таким образом, получается

(12.65)
$$\Omega^j = \mathcal{A}^T\Omega^{j\longrightarrow k}\mathcal{A}$$

(12.66)
$$\xi^j = (\Omega^{j\longrightarrow k}\Delta + \mathcal{A}^T\xi^{j\longrightarrow k})$$

Из $\mathcal{A}^{-1} = \mathcal{A}^T$ следует, что

(12.67)
$$\Omega^{j\longrightarrow k} = \mathcal{A}\Omega^j\mathcal{A}^T$$

$$\xi^{j\longrightarrow k} = \mathcal{A}(\xi^j - \Omega^{j\longrightarrow k}\Delta)$$

Это доказывает правильность участка алгоритма в строках со 2 до 7 в Таблице 12.7. Оставшиеся мягкие ограничения эквивалентности напрямую следуют из обсуждения в подразделе 12.9.3.

12.11.3 Установка соответствия

Оставшаяся часть задачи относится к установке соответствия между разными картами, и вычисление поворота α и параллельного переноса δ. Существует огромное множество подходящих методов, поэтому алгоритм будет приводиться эскизно. Очевидно, задача состоит в необходимости, потенциально, совмещать большое количество признаков на обеих локальных картах.

Канонический алгоритм для карт на основе признаков может выполнять кэшировние конфигурации достаточно близко расположенных ориентиров таким образом, чтобы локальные конфигурации давали хорошие признаки-кандидаты для соответствия. Например, можно идентифицировать множество m близко расположенных ориентиров (для небольшого числа m), и вычислить относительные расстояния и углы между ними. Такой вектор расстояний или углов послужит статистическим показателем, с помощью которого можно выполнить сравнение двух карт. Используя хэш-таблицы или kd-деревья, можно эффективно повысить их доступность, поэтому ответ на вопрос «соответствуют ли следующие m ориентиров для карты j-робота каким-либо m ориентирам на карте робота k?» можно получить очень легко, по крайней мере, оценочно. После определения начального соответствия можно легко вычислить d и α минимизируя квадрат расстояния между этими m признаками обеих карт.

Слияние выполняется следующим образом: сначала вызывается оператор слияния, используя d, α и C, вычисленных из m локальных признаков обеих карт. Затем идентифицируются дополнительные ориентиры, для которых проверка соответствия в Таблице 12.6 генерирует вероятность меньше пороговой. Если такие пары ориентиров не найдены, выполнение алгоритма прерывается.

Сравнение обоих компонентов объединённых карт, особенно близко расположенных ориентиров, для которых нет соответствия, даст критерий результирующего сравнения. Формально, слияние карт происходит сразу же после прерывания процесса поиска, если результирующее уменьшение общего правдоподобия в логарифмическом виде сдвинулось согласно количеству свёрнутых признаков с постоянным коэффициентом. Это эффективно реализует байесовскую функцию оценки MAP с экспоненциальным априорным распределением по количеству признаков окружающего пространства.

В общем случае, заметим, что поиск оптимального соответствия NP-сложен, но алгоритмы нахождения экстремума на практике работают чрезвычайно эффективно.

12.11.4 Пример

На Рис. 12.22 приведён пример из восьми локальных карт. Эти карты получены разбиением эталонного набора данных, обсуждаемого выше, на 8 непересекающихся последовательностей, и выполнением алгоритма SEIF на каждой по отдельности.

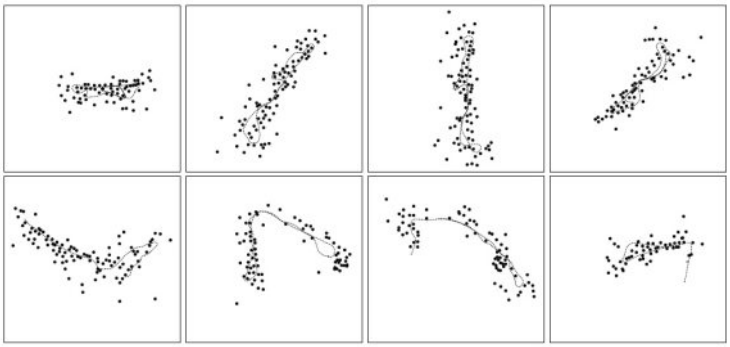


Рис. 12.22 Восемь локальных карт, полученных разбиением набора данных на восемь последовательностей.

Рис. 12.23 Результат работы SLAM с помощью нескольких роботов, описанного в этой главе. Изображение принадлежит Юфенг Лю (Yufeng Liu).

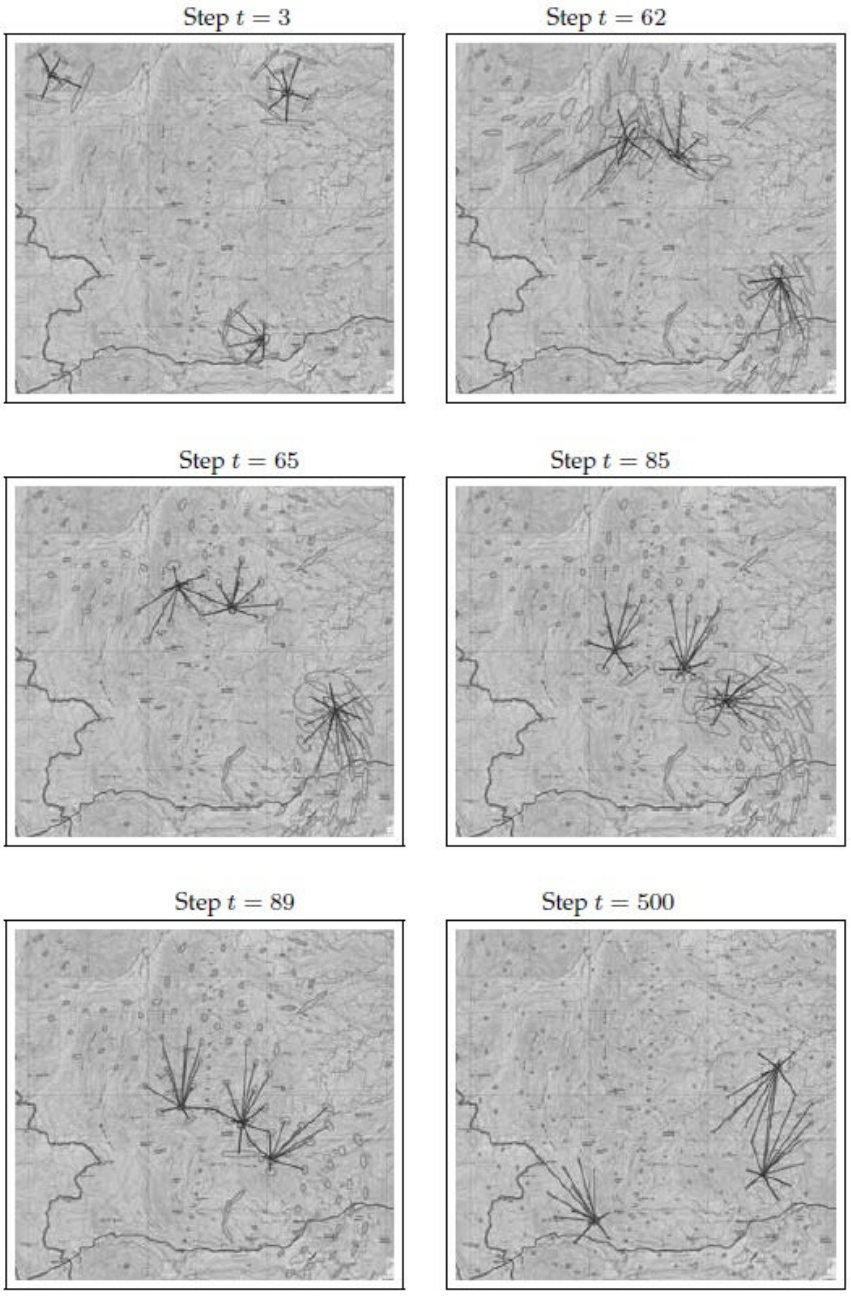


Рис. 12.24 Мгновенные снимки работы имитационного эксперимента SLAM в различные моменты времени. Во время выполнения шагов с 62 по 64, роботы 1 и 2 прошли через одну и ту же зону в первый раз. В результате, неопределённость их локальных карт уменьшилась. Позже, на шагах с 85 по 89, с робота 2 были обнаружены те же ориентиры, что и для транспорта 3, с похожим воздействием на общую неопределённость. После 500 шагов все ориентиры были локализованы точно.

Комбинируя локальные карты на основе $m = 4$ локальных признаков в таблице хэшей поиска соответствия, SEIF надёжно завершает работу, как показано на Рис. 12.23. Эта карта при вычислении через $\mu = \Omega^{-1}\xi$, являет-

ся не только суперпозицией отдельных локальных карт. Напротив, каждая локальная карта в процессе слияния слегка изменяется в результате комбинирования информации.

На Рис. 12.24 показана имитация работы трёх воздушных беспилотных аппаратов. На схеме показано, что в результате слияния карт неопределённость каждой отдельной карты уменьшилась.

12.12 Выводы

В этой главе описано эффективное решение проблемы онлайн SLAM – *разреженный обобщенный информационный фильтр* или SEIF. SEIF похож на GraphSLAM тем, что представляет апостериорную вероятность в информационном виде. Однако, он отличается отбрасыванием прошлых положений, и может выполняться как онлайн SLAM. Было показано, что:

• При отбрасывании прошлых положений признаки, обнаруженные из этих положений, приобретают прямые связи в информационной матрице.

• В информационной матрице, преимущественно, находится небольшое количество связей между признаками, которые соединяют физически близко расположенные признаки. Чем дальше друг от друга расположены два признака, тем слабее связь.

• Под разрежением матрицы понимается процесс смещения информации с помощью алгоритма SEIF в целях уменьшения количества связей, при этом информационная матрица всегда остаётся разреженной. Разрежённость означает, что каждый элемент матрицы соединяется ненулевым значением только с конечным множеством элементов, вне зависимости от общего размера карты N. Однако, разрежение является аппроксимацией, а не точной вычислительной операцией.

• Было замечено, что для разреженной информационной матрицы оба шага фильтрации (измерение и обновление движения) могут быть выполнены за постоянное время, независимо от размера карты. В обычных информационных фильтрах только этап обновления измерения требует постоянного времени. Этап обновления движения требует больше времени.

• Для выполнения многих шагов в SEIF все ещё требуется оценка состояния. В SEIF используется смягченный алгоритм восстановления этих оценок.

• Было описано два метода ассоциации данных. Первый идентичен уже описанному для EKF SLAM - это инкрементный максимум правдоподобия. Этот метод назначает измерение наиболее вероятному признаку в каждый момент времени, но неспособен пересмотреть решение об ассоциации.

• Улучшенный метод выполняет рекурсивный поиск по дереву всех ассоциаций данных, чтобы найти вектор ассоциации данных, максимизирующий правдоподобие всей совокупности данных. Это делается, используя онлайн версию алгоритма ветвей и границ с ленивым расширением дерева, когда логарифм правдоподобия данных запоминается по периметру частично расширенного дерева. Когда текущий наилучший лист достигает значения, меньшего среднему значению по периметру, периметр расширяется,

пока его значение не уменьшится или же пока не будет найдено лучшее глобальное решение проблемы ассоциации данных.

• Также приведено обсуждение использования SEIF в контексте картографирования с использованием нескольких роботов. Алгоритм используется в качестве внутреннего цикла метода для поворота и смещения карт, представленных в информационном виде, без необходимости вычисления самой подлежащей карты. Эта операция сохраняет разрежённость информационной матрицы.

• Эскизно был приведён алгоритм, который позволяет эффективно установить глобальное соответствие между двумя картами в задаче картографирования с помощью нескольких роботов. Он создаёт хэши локальных конфигураций признаков и использует технологии быстрого поиска для установки соответствия. Затем выполняется рекурсивное слияние карт, и, если карты хорошо накладываются, то и их слияние.

SEIF является первым эффективным онлайновым алгоритмом SLAM, описанным в книге. Он соединяет элегантность информационного выражения с идеей отбрасывания прошлых положений. Это "ленивый родственник" EKF: там, где в EKF информация каждого нового измерения активно распространяется через сеть признаков для вычисления точной общей ковариации, SEIF лишь собирает эту информацию, постепенно разрешая ее со временем. Ассоциация данных на основе дерева в SEIF также ленивая: она учитывает альтернативные пути только для наилучшего текущего пути и только при необходимости. Это строго противоположно методу, описанному в следующей главе, где к задаче ассоциации данных применяются многочастичные фильтры.

Для возможности эффективной работы онлайн в SEIF необходимо использовать ряд аппроксимаций, которые делают его более точным, чем GraphSLAM или EKF. В частности, у SEIF есть два ограничения: во-первых, линеаризация выполняется только однажды, так же, как в EKF. В Graph SLAM может выполняться повторная линеаризация, что, в общем, улучшает точность результата. Во-вторых, в SEIF используется аппроксимация для сохранения разрежённости информационной матрицы. Разрежённость является естественным свойством алгоритма GraphSLAM в силу природы информации, интегрирование которой выполняется при работе.

Хотя каждая из основных операций SEIF (с известным соответствием) может быть выполнена за "постоянное время", следует соблюдать осторожность. Если алгоритм SEIF применён к линейной системе (то есть нет необходимости выполнять разложение в ряд Тейлора, и ассоциация данных известна), обновление действительно займёт постоянное время. Однако, из-за необходимости линеаризации, понадобится оценить среднее μ_t, а также информационное состояние. Эта оценка не поддерживается в традиционном информационном фильтре, и ее восстановление требует определённого времени. Представленная реализация SEIF только аппроксимирует ее, и качество апостериорной оценки зависит от качества этой аппроксимации.

12.13 Библиографические примечания

Литература по информационно-теоретическим выражениям в SLAM уже обсуждалась в предыдущей главе, в той степени, которая относится к оф-

флайновому представлению. Информационные фильтры имеют сравнительно короткую историю в области исследования по SLAM. В 1997 году Цорба разработал информационный фильтр, сохраняющий относительную информацию между триплетами из трёх ориентиров. Возможно, он был первым, кто заметил, что такие информационные связи косвенно сохраняют глобальную корреляцию информации, и положил начало алгоритмам с квадратичными и линейными требованиями к памяти. Ньюман (Newman, 2000), Ньюман и Дюран-Уайт (Newman and Durrant-Whyte, 2001) разработали схожий информационный фильтр, но оставили открытым вопрос о том, как именно образуются информационные связи "ориентир-ориентир". Амбициозно назвав алгоритм «целостный, сходящийся и выполняемый за постоянное время SLAM» ("consistent, convergent, and constant-time SLAM"), Леонард и Ньюман эффективно выполнили выравнивание, успешно применив его в подводном аппарате, оснащённом сонаром с синтетической апертурой (Newman and Rikoski 2003).

ТОНКИЙ РАЗДЕЛИТЕЛЬНЫЙ ФИЛЬТР

Другой основополагающий алгоритм в этой области, это тонкий разделительный фильтр Паскина (Paskin, 2003), выражающий апостериорное распределение SLAM в виде разреженной сети, известной как тонкие разделительные деревья (Pearl 1988, Cowell et al. 1999). Та же идея была использована Фризи (Frese, 2004), который разработал древовидную факторизацию информационной матрицы для эффективного вывода.

ПЕРЕСЕЧЕНИЕ КОВАРИАЦИИ

Джулиер и Ульманн (Julier and Uhlmann) разработали масштабируемый метод под названием пересечение ковариаций ("covariance intersection"), выполняющий разреженную оценку апостериорного распределения таким образом, чтобы предотвратить чрезмерную степень "уверенности" алгоритма. Их алгоритм был успешно реализован в семействе роботов NASA MARS Rover (Uhlmann et al. 1999). Концепция информационного фильтра также относится к ранней работе Булата и Деви (Bulata and Devy, 1996), в методе которых модель ориентира сначала получалась локальных эталонных "ориентиро-центрических" координатах, и только затем присоединялась к целостной глобальной карте путём разрешения относительной информации между ориентирами. Наконец, некоторые "оффлайновые" алгоритмы SLAM, решающие полную задачу SLAM, которые были предложены Боссе (Bosse et al., 2004), Гутманом и Конолиги (Gutmann and Konolige, 2000), Фризи (Frese, 2004) и Монтемерло и Трун (Montemerlo and Thrun, 2004), оказались достаточно быстрыми для работы в режиме онлайн на ограниченных наборах данных.

Слияние карт нескольких роботов обсуждалось в работе Гутмана и Конолиги (Gutmann and Konolige, 2000). Неттлтон (Nettleton et al., 2003) был первым автором, который обобщил информационное выражение для проблемы SLAM с несколькими роботами. Они обнаружили, что аддитивность информации позволила выполнить асинхронную интеграцию локальных карт между роботами. Они также обнаружили, что добавление субкарт даёт эффективные алгоритмы связи, а интеграция карт будет выполняться за время, логарифмически зависящее от числа задействованных роботов. Однако, они не пояснили, как выравнивать эти карты, и эта проблема была позже исследована Труном и Лю (Thrun and Liu, 2003).

Алгоритм SEIF был разработан Труном (Thrun et al., 2002), и приводится в работе (Thrun et al., 2004a). По нашему мнению, это первый алгоритм, выводящий создание связей между парами признаков с точки зрения фильтрации. Жадный алгоритм ассоциации данных для SEIF был разработан Лю и Труном (Liu and Thrun, 2003), которая была впоследствии обобщена для задачи SLAM с несколькими роботами Труном и Лю (Thrun and Liu,

2003). Поиск ассоциации данных методом ветвей и границ принадлежит Хенелу (Hähnel et al., 2003a), и основывается на более ранних методах ветвей и границ, описанных Лоулером и Вудом (Lawler and Wood, 1966) и Наренда и Фукунага (Narendra and Fukunaga, 1977). Она согласуется с работой Куперса (Kuipers et al., 2004), который разработал похожий метод ассоциации данных, хотя и не в контексте информационных теоретических концепций. SEIF применялся для задач картографии заброшенных шахт (Thrun et al. 2004c), генерируя карты с 10^8 признаков.

Набор данных парка Виктории, на который приведены ссылки в этой главе, принадлежит Гюванту (Guivant et al., 2000).

12.14 Упражнения

1. Сравнить степень разрежённости в GraphSLAM и SEIF. Каковы преимущества и недостатки каждого метода? Описать условия, при которых каждый метод будет предпочтителен. Чем более конкретны аргументы, тем лучше.

ЦЕЛОСТНОСТЬ

2. Важной концепцией многих исследователей SLAM является целостность. В сообществе SLAM целостность определяется несколько иначе, чем в общей области статистики (где целостность является асимптотическим свойством).

Пусть x будет случайным вектором, а $\mathcal{N}(\mu, \Sigma)$ – гауссовой оценкой x. Гауссиан считается целостным, если соблюдаются следующие два свойства:

Условие 1: Несмещённость: Среднее μ является несмещенным

$$E[\mu] = x$$

Условие 2: Отсутствие «самоуверенности»: Ковариация Σ не вызывает чрезмерной доверительности алгоритма. Пусть Ξ будет истинной ковариацией оценочной функции μ:

$$\Xi = E[(\mu - E[\mu])(\mu - E[\mu])^T]$$

Тогда Σ чрезмерно доверительна, если существует вектор $\bar{x}$, для которого

$$\bar{x}^T \Sigma^{-1} \bar{x} > \bar{x}^T \Xi^{-1} \bar{x}$$

Чрезмерная доверительность образуется, если 95% эллипса доверительности оценённой ковариации Σ попадает внутрь или пересекается с истинным эллипсом доверительности оценочной функции.

Доказательство целостности обычно затруднено для алгоритмов SLAM. В данном случае необходимо доказать или опровергнуть, что разрежение сохраняет целостность (см. равенство (12.20)). В частности, доказать или опровергнуть следующее предположение: *для целостной объединённой вероятности $p(a, b, c)$ в гауссовой форме следующая аппроксимация также будет целостна:*

$$\bar{p}(a, b, c) = \frac{p(a, c)p(b, c)}{p(c)}$$

3. Требуется реализовать алгоритм SEIF для линейного гауссового SLAM. В линейном гауссовом SLAM, уравнение движения имеет простой аддитивный вид

$$x_t \sim \mathcal{N}(x_{t-1} + u_t, R)$$

а уравнение измерения - вид

$$z_t = \mathcal{N}(m_j - x_t, Q)$$

где R и Q - диагональные матрицы ковариации. Ассоциация данных известна в линейных гауссовых SLAM.

(a) Запустить алгоритм в простых средах имитации и проверить правильность реализации.

(b) Показать на графике ошибку SEIF в виде функции разрежённости информационной матрицы. Какие выводы можно сделать?

(c) Показать на графике вычислительное время для разработанной реализации SEIF в виде функции разрежённости информационной матрицы. Отметить особенности.

4. Правило разрежённости в SEIF отфильтровывает все пассивные признаки m^-, назначая $m^- = 0$. Зачем это делается? Каким будет уравнение обновления, если эти признаки не отфильтровать? Будет ли результат более или менее точным? Будет ли вычисление более или менее эффективным? Дать краткий ответ.

5. В описанном виде в SEIF линеаризация выполняется сразу после интеграции команды на движение в фильтре. Обсудить возможность алгоритма SEIF, позволяющего ретроспективное изменение линеаризации. Как будет выражено апостериорное распределение такого алгоритма? Как будет выражена информационная матрица?

13 Алгоритм FastSLAM

Пришло время обратить внимание на метод *многочастичного фильтра* в SLAM. Мы уже сталкивались с многочастичными фильтрами в нескольких главах книги и заметили, что они лежат в основе некоторых наиболее эффективных алгоритмов робототехники. Поэтому закономерно возникает вопрос применимости многочастичных фильтров для задачи SLAM. К сожалению, многочастичные фильтры подвержены «проклятью размерности»: там, где гауссовы фильтры требуют ресурсов, с зависимостью от линейной до квадратичной от числа измерений задачи оценки, сложность многочастичных фильтров возрастают экспоненциально! Прямолинейная реализация многочастичных фильтров для задачи SLAM обречена на провал, в силу большого числа переменных, используемых при описании карты.

Приведённый в этой главе алгоритм основан на важной характеристике задачи SLAM, которая ещё явно не обсуждалась в книге. Дело в том, что в

задаче SLAM с известным соответствием соблюдается условная независимость между любыми двумя непересекающимися множествами признаков на карте, если задано положение робота. Другими словами, если некий оракул даст информацию об истинном положении робота, станет возможным оценить местоположение всех признаков независимо друг от друга. Зависимость в этих оценках возникает *только* в силу неопределённости положения робота.

МНОГОЧАСТИЧНЫЙ ФИЛЬТР РАО-БЛЕКВЕЛЛА

Это наблюдение делает возможным применение варианта многочастичного фильтра, известного под названием *многочастичный фильтр Рао-Блеквелла в SLAM*. Многочастичные фильтры Рао-Блеквелла выражают апостериорную вероятность по нескольким переменным, а также их гауссианам (или другим параметрическим функциями) для выражения всех прочих переменных.

УСЛОВНАЯ ВЕРОЯТНОСТЬ

В FastSLAM многочастичные фильтры используется для оценки по пути робота. Как можно увидеть, для каждой из частиц ошибки отдельных карт *условно независимы*, поэтому задачу картографирования можно разбить на множество отдельных задач, по одной для каждого признака на карте. FastSLAM оценивает местоположение этих признаков на карте с помощью EKF, используя отдельный EKF низкой размерности для каждого отдельного признака. Это сильно отличается от алгоритмов SLAM, обсуждаемых в прошлых главах, в которых всегда использовался один гауссиан для совокупной оценки всех признаков.

Основной алгоритм может быть реализован за время, логарифмически зависящее от числа признаков. Поэтому, FastSLAM имеет определённые вычислительные преимущества перед обычными реализациями EKF и многими из их вариаций. Ключевое преимущество FastSLAM, однако, проистекает из возможности выполнить решение ассоциации данных на основе отдельных частиц. В результате, в фильтре сохраняются апостериорные вероятности по нескольким ассоциациям данных, а не только по самой вероятной. Это полностью противоположно всем алгоритмам SLAM, которые до сих пор обсуждались, и в которых отслеживалась только одна ассоциация данных в произвольный момент времени. Фактически, выполняя выборку по ассоциациям данных, FastSLAM аппроксимирует полную апостериорную вероятность, а не только ассоциацию данных, имеющую максимальное правдоподобие. Эмпирически подтверждено, что возможность сохранять несколько ассоциаций данных одновременно делает FastSLAM существенно повышает устойчивость к проблемам ассоциации данных, по сравнению с алгоритмами инкрементной ассоциации данных по максимальному правдоподобию.

Другое преимущество FastSLAM по сравнению с другими алгоритмами SLAM возникает из-за того, что многочастичные фильтры могут справляться с нелинейными моделями движения робота, там, где предыдущие методы аппроксимировали такие модели с помощью линейных функций. Это важно при высокой нелинейности кинематики, или же, когда неопределённость положения сравнительно высока.

Использование многочастичных фильтров создаёт необычную ситуацию, когда алгоритм FastSLAM решает сразу и *полную задачу SLAM*, и *онлайновую задачу SLAM*. Как будет показано далее, FastSLAM сформулирован для вычисления полной апостериорной вероятности пути, поскольку только наличие полного пути обеспечивает местоположениям признаков условную независимость. Однако, поскольку многочастичные фильтры выполняют одну оценку за один проход, FastSLAM в действительности является онлайновым алгоритмом и также решает онлайн задачу SLAM. Среди всех

обсуждаемых алгоритмов SLAM, FastSLAM является единственным алгоритмом, попадающим сразу в две категории.

В этой главе описано несколько версий алгоритма FastSLAM. FastSLAM 1.0 – это оригинальный алгоритм FastSLAM, который концептуально прост и лёгок в реализации. Но, в некоторых ситуациях, компонент многочастичного фильтра в FastSLAM 1.0 неэффективно генерирует выборку. В алгоритме FastSLAM 2.0 эта проблема решена улучшенным предполагаемым распределением, но ценою существенно более сложной реализации (как и математического вывода). В обоих алгоритмах FastSLAM используется модель датчика на основе признаков, обсуждаемая ранее. Применение FastSLAM для датчиков расстояния даст алгоритм, решающий задачу SLAM в контексте карт сетки занятости. Для всех алгоритмов в этой главе приведены методы оценки переменных ассоциации данных.

13.1 Основной алгоритм

Частица в базовом алгоритме FastSLAM имеют вид, показанный в Таблице 13.1. Каждая частица содержит оценку положения робота, обозначаемую $x_t^{[}k]$, и набор калмановских фильтров с математическим ожиданием $\mu_{j,t}^{[}k]$ и ковариацией $\Sigma_{j,t}^{[}k]$, по одному для каждого признака m_j на карте. Здесь $[k]$ – индекс частицы, а общее количество частиц обозначим M, как уже делалось раньше.

Шаг обновления базового FastSLAM приведён в Таблице 13.2. За множеством различных деталей обновления скрывается основной цикл, практически идентичный многочастичному фильтру, обсуждаемому в Главе 4. Первый шаг включает извлечение частицы, выражающей апостериорную вероятность в момент времени $t-1$, и выборку значений положения робота в момент времени t на основе вероятностной модели движения. На следующем шаге выполняется обновление EKF для наблюдаемых признаков, используя стандартное уравнение обновления EKF. Оно не является частью первоначального многочастичного фильтра, но необходимо в FastSLAM для изучения карты. Последние операции связаны с вычислением веса значимости, который затем используется для перевыборки частиц.

Разберём каждый шаг более детально и выведем их из базовых математических свойств задачи SLAM. Остановимся на том, что вывод изначально предполагает, что FastSLAM решает полную задачу SLAM, а не только онлайновую. Однако, в ходе дальнейших рассуждений станет ясно, что FastSLAM является решением для обеих задач. Каждую частицу можно рассматривать в виде элемента выборки в пространстве путей, что требуется для полной задачи SLAM, но обновление требует наличия только текущего положения, поэтому можно запускать FastSLAM и в качестве фильтра.

13.2 Факторизация апостериорной вероятности в SLAM

Ключевая математическая идея FastSLAM относится к тому факту, что полная апостериорная вероятность SLAM, $p(y_{1:t}|z_{1:t}, u_{1:t})$ в выражении (10.2) может быть переписана в виде множителей

(13.1)

$$p(y_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(m_n|x_{1:t}, z_{1:t}, c_{1:t})$$

Факторизация определяет, что вычисление апостериорной вероятности по путям и картам может быть разложена на $N + 1$ вероятностей.

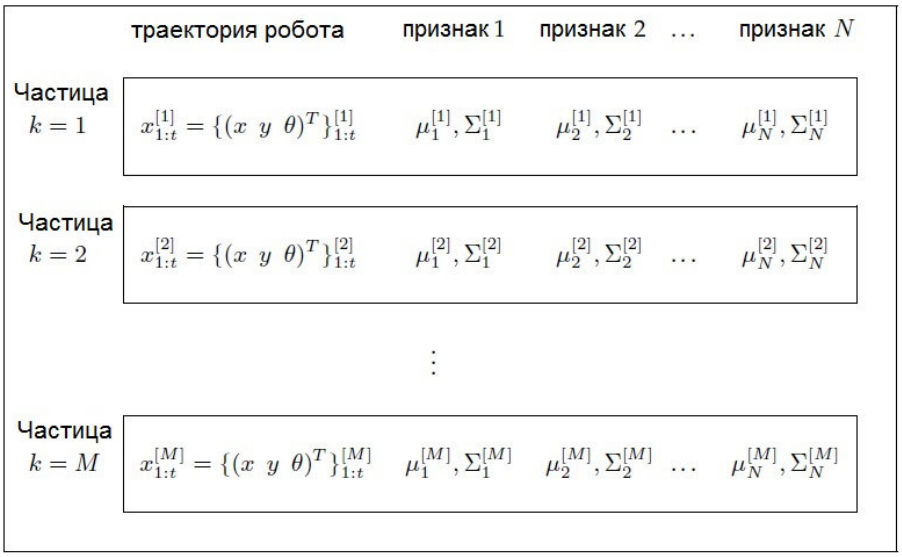


Рис. 13.1 Частицы в FastSLAM состоят из оценки пути и набора оценивающих функций местоположений отдельных признаков со связанными с ними ковариациями.

- Выполнить следующие шаги M раз:
  - **Извлечение.** Извлечь положение $x_{t-1}^{[k]}$ из набора частиц Y_{t-1}.
  - **Прогноз**. Сделать выборку нового положения $x_t^{[k]} \sim p(x_t|x_{t-1}^{[k]}, u_t)$.
  - **Обновление измерения.** Для каждого наблюдаемого признака z_t^i определить соответствие j измерения z_t^i, и учесть измерение z_t^i в соответствующем EKF, обновив среднее $\mu_{j,t}^{[k]}$ и ковариацию $\Sigma_{j,t}^{[k]}$
  - **Вес значимости**. Вычислить вес значимости $w^{[k]}$ для новой частицы.
- **Перевыборка.** Выполнить выборку с заменой M частиц, где каждая частица выбирается с вероятностью, пропорциональной $w^{[k]}$.

Рис. 13.2 Основные шаги алгоритма FastSLAM

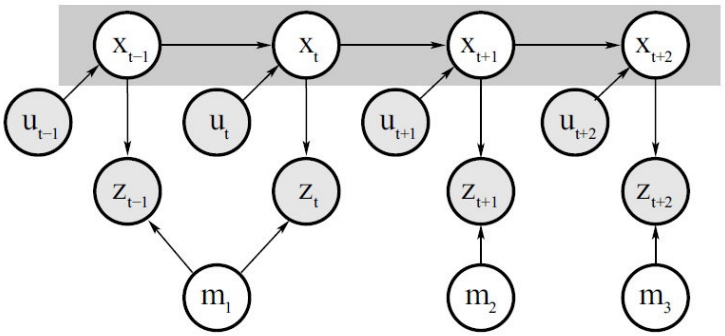


Рис. 13.3 Задача SLAM в виде графа байесовской сети. Робот перемещается из положения x_{t-1} в положение x_{t+2}, выполняя последовательность сигналов управления. В каждом положении x_t наблюдается близлежащий признак на карте $m = \{m_1, m_2, m_3\}$. На этой графической сети показано, что переменная положения «разделяет» отдельные признаки на карте друг от друга. Если положения известны, между двумя признаками на карте не останется путей с неизвестными переменными. Отсутствие пути обуславливает условную независимость апостериорной вероятности любых двух признаков (при заданном положении).

В FastSLAM используется многочастичный фильтр для вычисления апостериорного распределения по траекториям робота, обозначенной в виде $p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})$. Для каждого признака на карте в FastSLAM используются отдельные функции оценки по местонахождению $p(m_n|x_{1:t}, c_{1:t}, z_{1:t})$, для $n = 1, ..., N$. Поэтому, в сумме получится $N + 1$ апостериорных вероятностей FastSLAM. Оценивающие функции признаков зависят от пути робота, что означает наличие отдельной копии для каждой функции оценки признака, по одной для каждой частицы. Для M частиц количество фильтров будет равно $1 + MN$. Произведение этих вероятностей выражает искомое апостериорное распределение в виде множителей. Как будет показано ниже, выражение в виде множителей является точным, а не просто приближением, что является общей характеристикой задачи SLAM.

Для иллюстрации корректности этой факторизации, на Рис. 13.3 графически показан процесс получения данных в виде динамической байесовской сети. Как подразумевает этот граф, каждое измерение $z_1, ..., z_t$ является функцией соответствия признака, а также положения робота в момент времени, когда было выполнено измерение. Знание пути робота выделяет проблемы оценки отдельного признака и делает их независимыми друг от друга, в том смысле, что не существует прямого, в графическом выражении, пути от одного признака к другому, который бы не включал в себя переменных пути робота. Знание точного местонахождения одного признака, таким образом, ничего не скажет о местонахождении остальных признаков. Это указывает на *условную независимость* признаков при заданном пути, как указано в выражении (13.1).

Перед обсуждением выводов на основе этого свойства задачи SLAM приведём краткий математический вывод.

13.2.1 Математический вывод факторизованной апостериорной вероятности SLAM

Выведем уравнение (13.1) из основных закономерностей. Очевидно, имеется

(13.2)

$$p(y_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})\, p(m|x_{1:t}, z_{1:t}, c_{1:t})$$

Достаточно показать, что второй множитель с правой стороны можно выразить следующим образом:

(13.3)

$$p(m|x_{1:t}, c_{1:t}, z_{1:t}) = \prod_{n=1}^{N} p(m_n|x_{1:t}, c_{1:t}, z_{1:t})$$

Докажем это методом индукции. Вывод потребует выделения двух возможных случаев, в зависимости от того, наблюдается ли признак m_n в самом последнем измерении. В частности, если в последнем измерении $c_t \neq n$, то z_t не оказывает никакого эффекта на апостериорную вероятность, а также на положение робота x_t или соответствие c_t. Отсюда, получим:

(13.4)

$$p(m_n|x_{1:t}, c_{1:t}, z_{1:t}) = p(m_n|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})$$

Если $c_t = n$, а значит, и $m_n = m_{c_t}$, то есть признак наблюдался в самом последнем измерении z_t, возможно применить теорему Байеса с некоторыми стандартными упрощениями:

(13.5)

$$\begin{aligned} p(m_{c_t}|x_{1:t}, c_{1:t}, z_{1:t}) &= \frac{p(z_t|m_{c_t}, x_{1:t}, c_{1:t}, z_{1:t-1})p(m_{c_t}|x_{1:t}, c_{1:t}, z_{1:t-1})}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})} \\ &= \frac{p(z_t|x_t, m_{c_t}, c_t)p(m_{c_t}|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})} \end{aligned}$$

Это даёт следующее выражение для вероятности наблюдаемого признака m_{c_t}:

(13.6)

$$p(m_{c_t}|x_{1:t-1}, c_{1:t-1}, z_{1:t-1}) = \frac{p(m_{c_t}|x_{1:t}, c_{1:t}, z_{1:t})p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})}{p(z_t|x_t, m_{c_t}, c_t)}$$

Доказательство верности (13.3) выполняется через индукцию. Допустим, что апостериорная вероятность в момент времени $t-1$ уже факторизована:

(13.7)

$$p(m|x_{1:t-1}, c_{1:t-1}, z_{1:t-1}) = \prod_{n=1}^{N} p(m_n|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})$$

Это утверждение тривиально для $t = 1$, поскольку в начальный момент у робота нет информации ни об одном признаке, а, значит, все оценки независимы. В момент времени t, апостериорная вероятность имеет следующий

вид:

(13.8)

$$p(m_{c_t}|x_{1:t}, c_{1:t}, z_{1:t}) = \frac{p(z_t|m, x_{1:t}, c_{1:t}, z_{1:t-1})p(m|x_{1:t}, c_{1:t}, z_{1:t-1})}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})}$$
$$= \frac{p(z_t|x_t, m_{c_t}, c_t)p(m|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})}$$

Вставка этого выражения в гипотезу индукции (13.7) даст:

(13.9)

$$\begin{aligned}
&p(m|x_{1:t}, c_{1:t}, z_{1:t}) \\
&\quad = \frac{p(z_t|x_t, m_{c_t}, c_t)}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})} \prod_{n=1}^{N} p(m_n|x_{1:t-1}, c_{1:t-1}, z_{1:t-1}) \\
&\quad = \frac{p(z_t|x_t, m_{c_t}, c_t)}{p(z_t|x_{1:t}, c_{1:t}, z_{1:t-1})} \underbrace{p(m_{c_t}|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})}_{\text{форм. (13.6)}} \\
&\qquad\qquad \prod_{n \neq c_t} \underbrace{p(m_n|x_{1:t-1}, c_{1:t-1}, z_{1:t-1})}_{\text{форм. (13.4)}} \\
&\quad = p(m_{c_t}|x_{1:t}, c_{1:t}, z_{1:t}) \prod_{n \neq c_t} p(m_n|x_{1:t}, c_{1:t}, z_{1:t}) \\
&\quad = \prod_{n=1}^{N} p(m_n|x_{1:t}, c_{1:t}, z_{1:t})
\end{aligned}$$

Заметим, что были заменены выражения (13.4) и (13.6), что показывает правильность выражения (13.3). Правильность основного вида выражения (13.1) напрямую следует из результата и последующего общего преобразования:

(13.10)

$$\begin{aligned}
p(y_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) &= p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})p(m|x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \\
&= p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t})p(m|x_{1:t}, c_{1:t}, z_{1:t}) \\
&= p(x_{1:t}|z_{1:t}, u_{1:t}, c_{1:t}) \prod_{n=1}^{N} p(m_n|x_{1:t}, c_{1:t}, z_{1:t})
\end{aligned}$$

Заметим, что зависимость от всего пути $x_{1:t}$ действительно важна для результата. Зависимости от самого последнего положения x_t может быть недостаточно для переменной зависимости, поскольку зависимости могут возникать из предыдущих положений робота.

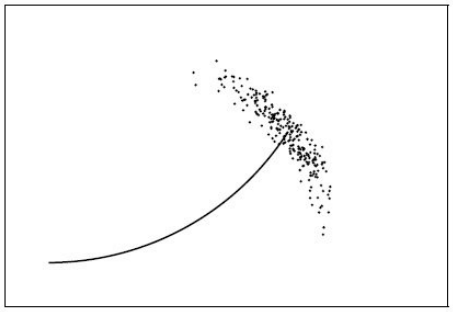


Рис. 13.4 Выборка значений из вероятностной модели движения.

13.3 FastSLAM с известной ассоциацией данных

Выражение апостериорной вероятности в виде множителей даёт существенные вычислительные преимущества над алгоритмами SLAM оценивающими неструктурированное апостериорное распределение. В FastSLAM используется факторизованное выражение в силу сохранения $MN + 1$ фильтров, M для каждого множителя в (13.1). В силу этого все $MN + 1$ имеют малую размерность.

Как было отмечено, в FastSLAM оценивается апостериорная вероятность по пути робота с помощью многочастичного фильтра. Местоположения признака карты оцениваются, используя EKF. В силу факторизации, FastSLAM может сохранять отдельный фильтр EKF для каждого признака, что делает обновление более эффективным по сравнению с EKF SLAM. Каждый отдельный EKF зависит от пути робота, поэтому каждая частица имеет собственный набор значений. В общей сложности имеется $N \cdot M$ EKF, по одному для каждого признака на карте и по одному для каждой частицы многочастичного фильтра.

Начнём с алгоритма FastSLAM для известной ассоциации данных. Частицы в FastSLAM будут определены

(13.11)
$$Y_t^{[k]} = \langle x_t^{[k]}, \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, ..., \mu_{N,t}^{[k]}, \Sigma_{N,t}^{[k]} \rangle$$

Как обычно, квадратные скобки $[k]$ указывают на индекс частицы, $x_t^{[k]}$ это оценка пути робота, а $\mu_{n,t}^{[k]}$ и $\Sigma_{n,t}^{[k]}$ - математическая ожидание и дисперсия гауссового выражения местоположения n-го признака, относительно k-й частицы. Вместе все эти показатели образуют k-ю частицу $Y_t^{[k]}$, которых в апостериорном распределении FastSLAM имеется M шт.

Фильтрация, или апостериорное распределение в момент времени t из таковых, взятых в момент времени $t-1$, включает генерацию нового набора частиц Y_t из Y_{t-1}, набора частиц из предыдущего шага. В новом наборе частиц учитывается новое управляющее воздействие u_t и измерение z_t с назначенным соответствием c_t. Это обновление выполняется в следующей последовательности:

1. **Расширение апостериорного распределения пути выборкой новых положений**. В FastSLAM 1.0 используется управляющее воздействие u_t для выборки нового положения робота x_t для каждой частицы Y_{t-1}. Обозначим k-ю частицу $Y_t^{[k]}$. FastSLAM 1.0 выполняет выборку положения x_t в соответствии с k-частицей, извлекая частицы в соответствии с

апостериорным распределением

(13.12)
$$x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$$

Здесь $x_{t-1}^{[k]}$ апостериорная оценка местоположения робота в момент времени $t-1$, принадлежащей k-й частице. Результирующая выборка $x_t^{[k]}$ добавляется к временному набору частиц, вместе с путём по предыдущим положениям, $x_{1:t-1}^{[k]}$. Этап выборки графически изображён на Рис. 13.4, где показан набор частиц положений, извлечённых из одного начального положения.

2. **Обновление наблюдаемой оценки признака**. Далее в FastSLAM 1.0 обновляется апостериорная вероятность по оценкам признака, выраженная средним $\mu_{n,t-1}^{[k]}$ и ковариацией $\Sigma_{n,t-1}^{[k]}$. Обновлённые значения добавляются к временному набору частиц вместе с новым положением робота.

Точные выражения обновления зависят от того, наблюдался ли признак m_n в момент времени t. Для $n \neq c_t$, когда признак n не наблюдался, на основании выражении (13.4) можно считать признак в апостериорном распределении неизменным. Это приводит к простому обновлению:

(13.13)
$$\langle \mu_{n,t}^{[k]}, \Sigma_{n,t}^{[k]} \rangle = \langle \mu_{n,t-1}^{[k]}, \Sigma_{n,t-1}^{[k]} \rangle$$

Для наблюдаемого признака $n = c_t$ обновление приводится в выражении (13.5) с нормирующим членом η:

(13.14)
$$\begin{aligned} & p(m_{c_t} | x_{1:t}, z_{1:t}, c_{1:t}) \\ & \quad = \eta\, p(z_t | x_t, m_{c_t}, c_t) p(m_{c_t} | x_{1:t-1}, z_{1:t-1}, c_{1:t-1}) \end{aligned}$$

Вероятность $p(m_{c_t} | x_{1:t-1}, c_{1:t-1}, z_{1:t-1})$ в момент времени $t-1$ выражена гауссианом с математическим ожиданием $\mu_{n,t-1}^{[k]}$ и ковариацией $\Sigma_{n,t-1}^{[k]}$. Для новой оценки в момент времени t вероятность также будет иметь вид гауссиана, FastSLAM реализует воспринимаемую модель $p(z_t | x_t, m_{c_t}, c_t)$ аналогично EKF SLAM. Как обычно, аппроксимируем функцию измерения h разложением в ряд Тейлора:

(13.15)
$$\begin{aligned} h(m_{c_t}, x_t^{[k]}) & \approx \underbrace{h(\mu_{c_t,t-1}^{[k]}, x_t^{[k]})}_{=: \hat{z}_t^{[k]}} + \underbrace{h'(x_t^{[k]}, \mu_{c_t,t-1}^{[k]})}_{=: H_t^{[k]}} (m_{c_t} - \mu_{c_t,t-1}^{[k]}) \\ & = \hat{z}_t^{[k]} + H_t^{[k]} (m_{c_t} - \mu_{c_t,t-1}^{[k]}) \end{aligned}$$

Здесь производная h' берётся по отношению к координатам признака m_{c_t}. Линейная аппроксимация выполняется тангенциально по h для $x_t^{[k]}$ и $\mu_{c_t,t-1}^{[k]}$. Согласно этой аппроксимации, апостериорное распределение местоположения признака c_t действительно имеет вид гауссовой функции. Новое математическое ожидание и ковариация получаются, используя стандартное обновление измерения EKF:

(13.16)
$$K_t^{[k]} = \Sigma_{c_t,t-1}^{[k]} H_t^{[k]T} (H_t^{[k]} \Sigma_{c_t,t-1}^{[k]} H_t^{[k]T} + Q_t)^{-1}$$

(13.17)
$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + K_t^{[k]} (z_t - \hat{z}_t^{[k]})$$

(13.18)
$$\Sigma_{c_t,t}^{[k]} = (I - K_t^{[k]} H_t^{[k]}) \Sigma_{c_t,t-1}^{[k]}$$

Шаги 1 и 2 повторяются M раз, в результате генерируя временный набор из M частиц.

3. **Перевыборка.** На финальном шаге FastSLAM выполняет перевыборку набора частиц. Мы уже сталкивались с перевыборкой во многих алгоритмах. FastSLAM извлекает их с заменой из временного набора из M частиц в соответствии с весом значимости, который будет объясняться ниже. На основании результирующего набора из M частиц формируется новый итоговый набор частиц Y_t. Необходимость перевыборки возникает в силу того, что частицы во временном наборе не распределены согласно искомой апостериорной вероятности: на шаге 1 генерируются положения x_t только в соответствии с самым последним управляющим воздействием u_t, не учитывая измерение z_t. Как читатель может уже хорошо знать, перевыборка является общепринятым методом определения таких ошибочных соответствий в многочастичных фильтрах.

Эта ситуация для одномерного случая показана на Рис. 13.5. Пунктирной линией показано предположительное распределение, сгенерированных частиц, а сплошной линией – искомое распределение. В FastSLAM предположительное распределение не зависит от z_t, но целевое - зависит. Взвешивание частиц показано внизу рисунка, перевыборка выполняется в соответствии с весами, а результирующий набор частиц приближает искомое распределение.

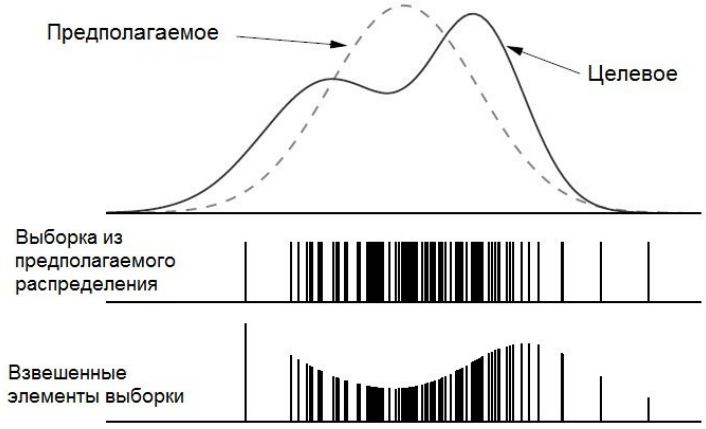


Рис. 13.5 Элементы почти невозможно извлекать напрямую из искомого распределения (показано сплошной линией). Вместо этого функция выборки извлекает образцы выборки из предполагаемого распределения (пунктирная линия), что значительно проще. В нижней части рисунка извлекаются образцы из предполагаемого распределения согласно их весам значимости.

Для определения фактора значимости будет полезно вычислить текущее предполагаемое распределение частиц пути во временном наборе. Допустив, что этот набор переменных пути из частиц Y_{t-1} распределён в соответствии с $p(x_{1:t-1}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$ (что является асимптотически верной аппроксимацией), частицы пути во временном наборе распределены согласно:

(13.19)

$$p(x_{1:t}^{[k]}|z_{1:t-1}, u_{1:t}, c_{1:t-1}) = p(x_t^{[k]}|x_{t-1}^{[k]}, u_t)p(x_{1:t-1}^{[k]}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$$

Множитель $p(x_t^{[k]}|x_{t-1}^{[k]}, u_t)$ показывает распределение выборки из выражения (13.12).

Целевое распределение принимает в расчёт распределение в момент времени z_t, а также соответствие c_t:

(13.20)

$$p(x_{1:t}^{[k]}|z_{1:t}, u_{1:t}, c_{1:t})$$

Процесс перевыборки принимает в расчёт разницу целевого и предполагаемого распределений. Как обычно, фактор значимости для перевыборки задан коэффициентами целевого и предполагаемого распределения:

(13.21)

$$\begin{aligned} w_t^{[k]} &= \frac{\text{целевое распределение}}{\text{предполагаемое распределение}} \\ &= \frac{p(x_{1:t}^{[k]}|z_{1:t}, u_{1:t}, c_{1:t})}{p(x_{1:t}^{[k]}|z_{1:t-1}, u_{1:t}, c_{1:t-1})} \\ &= \eta\, p(z_t|x_t^{[k]}, c_t) \end{aligned}$$

Последнее преобразование является прямым следствием следующего преобразования перечисления в (13.21):

(13.22)

$$\begin{aligned} &p(x_{1:t}^{[k]}|z_{1:t}, u_{1:t}, c_{1:t}) \\ &\quad = \eta\, p(z_t|x_{1:t}^{[k]}, z_{1:t-1}, u_{1:t}, c_{1:t})p(x_{1:t}^{[k]}|z_{1:t-1}, u_{1:t}, c_{1:t}) \\ &\quad = \eta\, p(z_t|x_t^{[k]}, c_t)p(x_{1:t}^{[k]}|z_{1:t-1}, u_{1:t}, c_{1:t-1}) \end{aligned}$$

Для вычисления вероятности $p(z_t|x_t^{[k]}, c_t)$ в (13.21) будет необходимо выполнить дальнейшие преобразования. В частности, эта вероятность эквивалентна следующему интегрированию, где мы снова удаляем переменные, иррелевантные для прогноза измерения датчика:

(13.23)

$$\begin{aligned} w_t^{[k]} &= \eta \int p(z_t|m_{c_t}, x_t^{[k]}, c_t)p(m_{c_t}|x_t^{[k]}, c_t)dm_{c_t} \\ &= \eta \int p(z_t|m_{c_t}, x_t^{[k]}, c_t) \underbrace{p(m_{c_t}|x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})}_{\sim\mathcal{N}(\mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} dm_{c_t} \end{aligned}$$

Здесь $\mathcal{N}(x; \mu, \Sigma)$ обозначает гауссово распределение по переменной x с математическим ожиданием μ и ковариацией Σ.

Интеграция в (13.23) включает оценку местоположения наблюдаемого признака в момент времени t и модель измерения. Для вычисления (13.23) в закрытом виде в FastSLAM выполняется та же самая линейная аппроксимация, используемая обновлении измерения на шаге 2. В частности, фактор значимости задан как

(13.24)

$$w_t^{[k]} \approx \eta \, |2\pi Q_t^{[k]}|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_t^{[k]}) Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]})\}$$

с ковариацией

(13.25)

$$Q_t^{[k]} = H_t^{[k]T} \Sigma_{n,t-1}^{[k]} H_t^{[k]} + Q_t$$

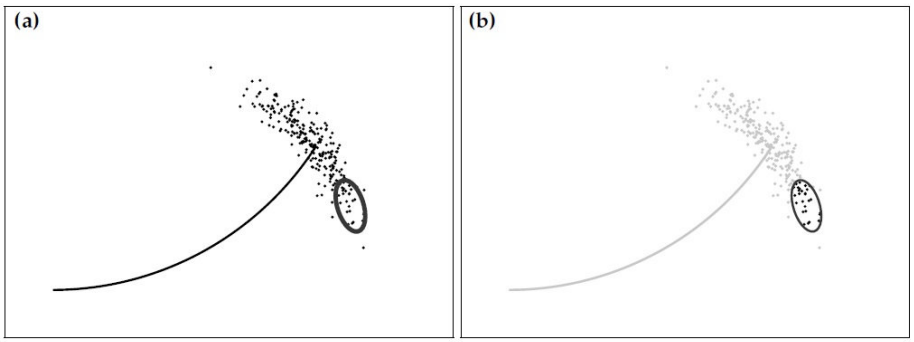


Рис. 13.6 Несовпадение между предполагаемым и апостериорным распределением: показаны элементы прямой выборки в FastSLAM 1.0, и апостериорное распределение измерения (эллипс)(a). На Рис. (b) показан набор выборки после шага перевыборки.

Это выражение является вероятностью текущего измерения z_t при гауссовом распределении. Оно является результатом свёртки распределений (13.23), используя линейную аппроксимацию по h. Результирующие веса значимости используются для извлечения M новых значений в временный набор значений выборки. С помощью этого процесса частицы сохраняются в пропорции их вероятности измерения.

Вместе эти три шага образуют правила обновления алгоритма FastSLAM 1.0 для задач SLAM с известной ассоциацией данных. Заметим, что время выполнения обновления не зависит от общей длины пути по времени t. Фактически, в процессе генерации частицы в момент времени t используется только последнее положение $x_{t-1}^{[k]}$. Следовательно, прошлые положения могут отбрасываться. Это имеет полезное следствие, что ни требования вычислительного времени, ни занимаемой памяти FastSLAM не зависят от общего количества тактов времени, затраченных на получение данных.

Выводы по алгоритму FastSLAM 1.0 с известной ассоциацией данных приводятся в Таблице 13.1. Для простоты, эта реализация допускает что только один признак измеряется в один момент времени. Этот алгоритм

упрощенно реализует разные шаги обновления. Сама реализация достаточно прямолинейна, ведь FastSLAM 1.0 является одним из простейших для реализации алгоритмов SLAM!

1: **Algorithm FastSLAM 1.0_known_correspondence**(z_t, c_t, u_t, Y_{t-1}) :
2: *for* $k = 1$ *to* M *do* //*цикл по всем частицам*
3: *получить* $\langle x_{t-1}^{[k]}, \langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \rangle, ..., \langle \mu_{N,t-1}^{[k]}, \Sigma_{N,t-1}^{[k]} \rangle \rangle$ *из* Y_{t-1}
4: $x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ //*элемента положения*
5: $j = c_t$ //*наблюдаемый ориентир*
6: *если признак* j *ранее не наблюдался*
7: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ //*инициализировать среднее*
8: $H = h'(x_t^{[k]}, \mu_{j,t}^{[k]})$ //*вычислить якобиан*
9: $\Sigma_{j,t}^{[k]} = H^{-1} Q_t (H^{-1})^T$ //*инициализировать ковариацию*
10: $w^{[k]} = p_0$ //*веса знаимости по умолчанию*
11: *иначе*
12: $\hat{z} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ //*прогноз измерения*
13: $H = h'(x_t^{[k]}, \mu_{j,t-1}^{[k]})$ //*вычислить якобиан*
14: $Q = H \Sigma_{j,t-1}^{[k]} H^T + Q_t$ //*ковариация измерения*
15: $K = \Sigma_{j,t-1}^{[k]} H^T Q^{-1}$ //*вычислить калмановское усиление*
16: $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z})$ //*обновить среднее*
17: $\Sigma_{j,t}^{[k]} = (I - K\,H)\Sigma_{j,t-1}^{[k]}$ //*обновить ковариацию*
18: $w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_n)^T$
$Q^{-1}(z_t - \hat{z}_n)\}$ //*фактор значимости*
19: *endif*
20: *для всех других признаков* $j' \neq j$ *do* //*ненаблюдаемых признаков*
21: $\mu_{j',t}^{[k]} = \mu_{j',t-1}^{[k]}$ //*оставить неизвменным*
22: $\Sigma_{j',t}^{[k]} = \Sigma_{j',t-1}^{[k]}$
23: *endfor*
24: *endfor*
25: $Y_t = \emptyset$ //*инициализировать новый набор частиц*
26: *повторить* M *раз* $\emptyset$ //*перевыборка* M *частиц*
27: *извлечь случайное* k *с вероятностью* $\propto w^{[k]}$ //*перевыборка*
28: *добавить* $\langle x_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \rangle, ..., \langle \mu_N^{[k]}, \Sigma_N^{[k]} \rangle \rangle$ *к* Y_t
29: *endfor*
30: *return* Y_t

Таблица 13.1 FastSLAM 1.0 с известным соответствием.

13.4 Улучшение предполагаемого распределения

FASTSLAM 2.0

FastSLAM 2.0, по большей части, эквивалентен FastSLAM 1.0, но с одним важным исключением: в его предполагаемом распределении принимается во внимание измерение z_t при выполнении выборки по положению x_t. Таким образом, можно обойти ключевое ограничение FastSLAM 1.0.

На первый взгляд, разница невелика: читатель может припомнить, что FastSLAM 1.0 выполняет выборку положений только на основании сигналов управления u_t, а затем использует измерение z_t для вычисления весов

значимости. Это проблематично, когда точность сигналов управления относительно низка по сравнению с точностью датчиков робота. Такая ситуация показана на Рис. 13.6. Предположение генерируется на основании большого спектра элементов выборки, показанных на Рис. 13.6a, но только малое подмножество этих элементов имеет высокое правдоподобие, показанное в виде эллипсоида. После перевыборки только частицы сохранятся с относительно высоким правдоподобием внутри эллипсоида. FastSLAM 2.0 обходит эту проблему выполнением выборки из положений на основе измерения z_t вдобавок к сигналу управления u_t. Таким образом, FastSLAM 2.0 более эффективен по сравнению с FastSLAM 1.0. С другой стороны, FastSLAM 2.0 более сложен в реализации по сравнению с FastSLAM 1.0, и его математический вывод также более труден.

13.4.1 Обобщение апостериорной вероятности пути выборкой нового положения

В FastSLAM 2.0 положение $x_t^{[k]}$ извлекается из апостериорного распределения

(13.26)
$$x_t^{[k]} \sim p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})$$

Это распределение отличается от предполагаемого распределения, представленного в (13.12), в котором (13.26) принимает измерение в расчёт z_t вместе с соответствием c_t. В частности, выражение в (13.26) зависит от $z_{1:t}$, а алгоритм выборки положения FastSLAM 1.0 зависит от $z_{1:t-1}$.

К сожалению, это требует и более сложной математики. В частности, механизм выполнения выборки из (13.26) требует дальнейшего анализа. Во-первых, перепишем (13.26) в терминах «известных» распределений, таких, как модели измерения и движения, и гауссова оценка признака в k-й частице.

(13.27)
$$\begin{aligned}
&p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t}) \\
&\quad \overset{\text{Bayes}}{=} \frac{p(z_t | x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t})}{p(z_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t})} \\
&\quad = \eta^{[k]} p(z_t | x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\quad \overset{\text{Markov}}{=} \eta^{[k]} p(z_t | x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}); p(x_t | x_{t-1}^{[k]}, u_t) \\
&\quad = \eta^{[k]} \int p(z_t | m_{c_t}, x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\qquad\qquad p(m_{c_t} | x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) dm_{c_t} p(x_t | x_{t-1}^{[k]}, u_t) \\
&\quad \overset{\text{Markov}}{=} \eta^{[k]} \int \underbrace{p(z_t | m_{c_t}, x_t, c_t)}_{\sim \mathcal{N}(z_t; h(m_{c_t}, x_t), Q_t)} \underbrace{p(m_{c_t} | x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})}_{\sim \mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} dm_{c_t} \\
&\qquad\qquad \underbrace{p(x_t | x_{t-1}^{[k]}, u_t)}_{\sim \mathcal{N}(x_t; g(x_{t-1}^{[k]}, u_t), R_t)}
\end{aligned}$$

Это выражение показывает, что наше распределение извлечения выборки действительно является свёрткой двух гауссианов, перемноженных на

третий. В общем случае SLAM распределение по выборке не имеет закрытого вида, из которого можно легко извлекать элементы выборки. Виной всему функция h: если бы она была линейна, вероятность была бы гауссовой, что станет понятным чуть ниже. Фактически, даже интеграл в (13.27) не имеет решения в закрытом виде, поэтому выборка из вероятности (13.27) затруднена.

Это наблюдение побуждает заменить h линейной аппроксимацией. Как часто встречается в книге, это приближение получается с помощью разложения в ряд Тейлора первого порядка, заданного следующей линейной функцией:

(13.28)

$$h(m_{c_t}, x_t) \approx \hat{z}_t^{[k]} + H_m(m_{c_t} - \mu_{c_t,t-1}^{[k]}) + H_x(x_t - \hat{x}_t^{[k]})$$

Здесь используем следующие сокращения:

(13.29)

$$\hat{z}_t^{[k]} = h(\mu_{c_t,t-1}^{[k]}, \hat{x}_t^{[k]})$$

(13.30)

$$\hat{x}_t^{[k]} = g(x_{t-1}^{[k]}, u_t)$$

Матрицы H_m и H_x являются якобианами h. Они также производные h по отношению к m_{c_t} и x_t, соответственно, вычисленные по ожидаемым значениям аргументов:

(13.31)

$$H_m = \nabla_{m_{c_t}} h(m_{c_t}, x_t)\big|_{x_t=\hat{x}_t^{[k]}; m_{c_t}=\mu_{c_t,t-1}^{[k]}}$$

(13.32)

$$H_x = \nabla_{x_t} h(m_{c_t}, x_t)\big|_{x_t=\hat{x}_t^{[k]}; m_{c_t}=\mu_{c_t,t-1}^{[k]}}$$

С этой аппроксимацией искомое распределение выборки (13.27) является гауссианом со следующими параметрами:

(13.33)

$$\Sigma_{x_t}^{[k]} = [H_x^T Q_t^{[k]-1} H_x + R_t^{-1}]^{-1}$$

(13.34)

$$\mu_{x_t}^{[k]} = \Sigma_{x_t}^{[k]} H_x^T Q_t^{[k]-1}(z_t - \hat{z}_t^{[k]}) + \hat{x}_t^{[k]}$$

где матрица $Q_t^{[k]}$ определена следующим образом:

(13.35)

$$Q_t^{[k]} = Q_t + H_m \Sigma_{c_t,t-1}^{[k]} H_m^T$$

Заметим, что для линейной аппроксимации теорема о свёртке имеет закрытый вид для интегрального члена в (13.27):

(13.36)

$$\mathcal{N}(z_t; \hat{z}_t^{[k]} + H_x x_t - H_x \hat{x}_t^{[k]}, Q_t^{[k]})$$

Распределение выборки (13.27) сейчас задано в виде произведения этого нормального распределения и самого правого члена в (13.27), нормального

$\mathcal{N}(x_t; \hat{x}_t^{[k]}, R_t)$. Записав в гауссовой форме, получим

(13.37)
$$p(x_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t}) = \eta \exp\{-P_t^{[k]}\}$$

с

(13.38)
$$\begin{aligned} P_t^{[k]} = \frac{1}{2}[&(z_t - \hat{z}_t^{[k]} - H_x x_t + H_x \hat{x}_t^{[k]})^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]} - H_x x_t + H_x \hat{x}_t^{[k]}) \\ &+ (x_t - \hat{x}_t^{[k]})^T R_t^{-1} (x_t - \hat{x}_t^{[k]})] \end{aligned}$$

Очевидно, это выражение квадратично по целевой переменной x_t, поэтому $p(x_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})$ является гауссианом. Математическое ожидание и ковариация этого гауссиана эквивалентны минимуму и кривизне функции $P_t^{[k]}$. Они идентичны вычислению первой и второй производной $P_t^{[k]}$ по x_t:

(13.39)
$$\begin{aligned} \frac{\partial P_t^{[k]}}{\partial x_t} &= -H_x^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]} - H_x x_t + H_x \hat{x}_t^{[k]}) + R_t^{-1} (x_t - \hat{x}_t^{[k]}) \\ &= (H_x^T Q_t^{[k]-1} H_x + R_t^{-1}) x_t - H_x^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]} + H_x \hat{x}_t^{[k]}) - R_t^{-1} \hat{x}_t^{[k]} \end{aligned}$$

(13.40)
$$\frac{\partial^2 P_t^{[k]}}{\partial x_t^2} = H_x^T Q_t^{[k]-1} H_x + R_t^{-1}$$

Ковариация $\Sigma_{x_t}^{[k]}$ распределения выборки получается обращением второй производной

(13.41)
$$\Sigma_{x_t}^{[k]} = [H_x^T Q_t^{[k]-1} H_x + R_t^{-1}]^{-1}$$

Математическое ожидание $\mu_{x_t}^{[k]}$ распределения выборки получается установкой первой производной в (13.39) в нуль. Это даст:

(13.42)
$$\begin{aligned} \mu_{x_t}^{[k]} &= \Sigma_{x_t}^{[k]} [H_x^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]} + H_x \hat{x}_t^{[k]}) + R_t^{-1} \hat{x}_t^{[k]}] \\ &= \Sigma_{x_t}^{[k]} H_x^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]}) + \Sigma_{x_t}^{[k]} [H_x^T Q_t^{[k]-1} H_x + R^{-1}] \hat{x}_t^{[k]} \\ &= \Sigma_{x_t}^{[k]} H_x^T Q_t^{[k]-1} (z_t - \hat{z}_t^{[k]}) + \hat{x}_t^{[k]} \end{aligned}$$

Этот гауссиан является аппроксимацией искомого распределения выборки (13.26) в FastSLAM 2.0. Очевидно, это предполагаемое распределение значительно сложнее, чем используемое для FastSLAM 1.0 в выражении (13.12).

13.4.2 Обновление оценки наблюдаемого признака

Так же, как и первая версия алгоритма FastSLAM, FastSLAM 2.0 выполняет обновление апостериорной вероятности по оценкам признака на основе измерения z_t и выборки по положению $x_t^{[k]}$. Оценки в момент времени $t-1$ снова выражены в виде математического ожидания $\mu_{j,t-1}^{[k]}$ и ковариации $\Sigma_{j,t-1}^{[k]}$. Обновлённые оценки обозначены $\mu_{j,t}^{[k]}$ и $\Sigma_{j,t}^{[k]}$. Последовательность обновления зависит от того, наблюдался или нет признак j в момент времени t. Для $j \neq c_t$ уже было определено выражение (13.4) и установлено, что апостериорное распределение остаётся неизменным. Это означает, что, вместо обновления оценки, ее можно просто скопировать.

Для наблюдаемого признака $j = c_t$ ситуация более сложная. В выражении (13.5) уже установлено апостериорное распределение по наблюдаемому признаку. Повторим его с индексом частицы k:

(13.43)

$$\begin{aligned} & p(m_{c_t} | x_t^{[k]}, c_{1:t}, z_{1:t}) \\ & \quad = \eta \underbrace{p(z_t | m_{c_t}, x_t^{[k]}, c_t)}_{\sim \mathcal{N}(z_t; h(m_{c_t}, x_t^{[k]}), Q_t)} \underbrace{p(m_{c_t} | x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})}_{\sim \mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} \end{aligned}$$

Так же как в выражении (13.27), нелинейность h не позволяет выразить апостериорную вероятность в виде гауссиана, что противоречит выражению для оценки признаков FastSLAM 2.0 в виде гауссовой функции. К счастью, решение лежит в той же самой линеаризации, которая уже обсуждалась выше:

(13.44)

$$h(m_{c_t}, x_t) \approx \hat{z}_t^{[k]} + H_m (m_{c_t} - \mu_{c_t,t-1}^{[k]})$$

Заметим, что x_t не является свободной переменной, поэтому третий член в выражении (13.28) можно отбросить. Эта аппроксимация даст вероятность вида (13.43). Гауссиан в целевой переменной m_{c_t}:

(13.45)

$$\begin{aligned} & p(m_{c_t} | x_t^{[k]}, c_{1:t}, z_{1:t}) \\ & \quad = \eta \exp \{ -\frac{1}{2} (z_t - \hat{z}_t^{[k]} - H_m (m_{c_t} - \mu_{c_t,t-1}^{[k]})) Q_t^{-1} \\ & \qquad (z_t - \hat{z}_t^{[k]} - H_m (m_{c_t} - \mu_{c_t,t-1}^{[k]})) \\ & \qquad - \frac{1}{2} (m_{c_t} - \mu_{c_t,t-1}^{[k]}) \Sigma_{c_t,t-1}^{[k]-1} (m_{c_t} - \mu_{c_t,t-1}^{[k]}) \} \end{aligned}$$

Новые математическое ожидание и ковариация получаются, используя стандартные уравнения обновления измерения EKF:

(13.46)

$$K_t^{[k]} = \Sigma_{c_t,t-1}^{[k]} H_m^T Q_t^{[k]-1}$$

(13.47)

$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + K_t^{[k]} (z_t - \hat{z}_t^{[k]})$$

(13.48)

$$\Sigma_{c_t,t}^{[k]} = (I - K_t^{[k]} H_m) \Sigma_{c_t,t-1}^{[k]}$$

Заметим, что это несколько сложнее, чем обновление в FastSLAM 1.0, но дополнительные усилия в реализации часто окупаются за счет улучшенной точности.

13.4.3 Вычисление факторов значимости

Пока что сгенерированные частицы не совпадают с искомой апостериорной вероятностью. В FastSLAM 2.0 это вызвано нормализующим членом $\eta^{[k]}$ в (13.27), который обычно различается для каждой частицы k. Эти различия ещё не учтены в процессе перевыборки. Также как в FastSLAM 1.0, фактор значимости выражен следующим коэффициентом:

(13.49)
$$w_t^{[k]} = \frac{\text{целевое распределение}}{\text{предполагаемое распределение}}$$

И снова, целевое распределение, которое бы хотелось выразить с помощью частиц, задано апостериорной вероятностью по пути $p(x_t^{[k]}|z_{1:t}, u_{1:t}, c_{1:t})$. Используя асимптотически корректные допущения, что пути $x_{1:t-1}^{[k]}$ были сгенерированы согласно целевому распределению, взятому на шаг ранее, $p(x_{1:t-1}^{[k]}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1})$, заметим, что предполагаемое распределение теперь задано произведением

(13.50)
$$p(x_{1:t-1}^{[k]}|z_{1:t-1}, u_{1:t-1}, c_{1:t-1})\, p(x_t^{[k]}|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})$$

Второй член произведения является распределением выборки по положению (13.27). Вес значимости получается следующим образом:

(13.51)
$$\begin{aligned}
w_t^{[k]} &= \frac{p(x_t^{[k]}|u_{1:t}, z_{1:t}, c_{1:t})}{p(x_t^{[k]}|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&= \frac{p(x_t^{[k]}|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})p(x_{1:t-1}^{[k]}|u_{1:t}, z_{1:t}, c_{1:t})}{p(x_t^{[k]}|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t})p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&= \frac{p(x_{1:t-1}^{[k]}|u_{1:t}, z_{1:t}, c_{1:t})}{p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&\overset{\text{Bayes}}{=} \eta\, \frac{p(z_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t})p(x_{1:t-1}^{[k]}|u_{1:t}, z_{1:t-1}, c_{1:t})}{p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&\overset{\text{Markov}}{=} \eta\, \frac{p(z_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t})p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})}{p(x_{1:t-1}^{[k]}|u_{1:t-1}, z_{1:t-1}, c_{1:t-1})} \\
&= \eta\, p(z_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t})
\end{aligned}$$

Читатель может заметить, что это выражение является инверсией нормировочной постоянной $\eta^{[k]}$ в (13.27). Дальнейшие преобразования ведут к следующему виду:

(13.52)

$$
\begin{aligned}
w_t^{[k]} &= \eta \int p(z_t|x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\qquad p(x_t|x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) dx_t \\
&\overset{\text{Markov}}{=} \eta \int p(z_t|x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) p(x_t|x_{t-1}^{[k]}, u_t) dx_t \\
&= \eta \int \int p(z_t|m_{c_t}, x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) \\
&\qquad p(m_{c_t}|x_t, x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}) dm_{c_t} p(x_t|x_{t-1}^{[k]}, u_t) dx_t \\
&\overset{\text{Markov}}{=} \eta \int \underbrace{p(x_t|x_{t-1}^{[k]}, u_t)}_{\sim \mathcal{N}(x_t; g(\hat{x}_{t-1}^{[k]}, u_t), R_t)} \int \underbrace{p(z_t|m_{c_t}, x_t, c_t)}_{\sim \mathcal{N}(z_t; h(m_{c_t}, x_t), Q_t)} \\
&\qquad \underbrace{p(m_{c_t}|x_{1:t-1}^{[k]}, u_{1:t-1}, z_{1:t-1}, c_{1:t-1})}_{\sim \mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} dm_{c_t} dx_t
\end{aligned}
$$

Мы обнаружили, что это выражение снова можно аппроксимировать гауссовой функцией по измерениям z_t путём линеаризации функции h. Как легко показать, математическое ожидание результирующего гауссиана $\hat{z}_t$, и его ковариация равны

(13.53)

$$L_t^{[t]} = H_x^T Q_t H_x + H_m \Sigma_{c_t,t-1}^{[k]} H_m^T + R_t$$

другими словами, не нормализованный множитель значимости k-й частицы задан следующим выражением:

(13.54)

$$w_t^{[k]} = |2\pi L_t^{[t]}|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_t) L_t^{[t]-1} (z_t - \hat{z}_t)\}$$

Так же, как FastSLAM 1.0, частицы, сгенерированные на этапах 1 и 2, вместе с фактором значимости, вычисленным на этапе 3, собираются во временный набор частиц.

Финальный шаг обновления FastSLAM 2.0 – перевыборка. Так же, как в FastSLAM 1.0, FastSLAM 2.0 извлекает (с заменой) M частиц из временного набора. Каждая частица извлекается с вероятностью, пропорциональной фактору значимости $w_t^{[k]}$. Результирующий набор частиц асимптотически выражает искомую факторизованную апостериорную вероятность t.

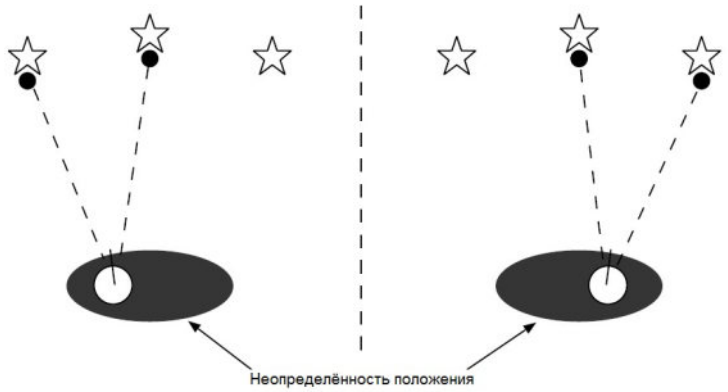


Рис. 13.7 Проблема ассоциации данных в SLAM. На рисунке показано, что лучшая ассоциация данных может быть различной даже внутри областей с высоким правдоподобием положения робота.

13.5 Неизвестная ассоциация данных

Этот раздел обобщает обе переменные алгоритма FastSLAM для случая, когда переменные соответствия $c_{1:t}$ неизвестны. Ключевым преимуществом использования многочастичных фильтров для SLAM являются отдельные для каждой частицы локальные решения ассоциации данных.

Напоминаем читателю, что задача ассоциации данных в момент времени t является задачей определения переменной c_t на основе доступных данных. Эта проблема показана на Рис. 13.7. Здесь робот наблюдает два признака окружающей среды. В зависимости от относительного положения робота и измерения этих признаков, эти измерения соответствуют различным признакам карты (показаны звёздочками на Рис. 13.7).

Пока что обсуждались различные методы ассоциации данных, используя ряд таких методов, как максимум правдоподобия. Эти методы объединяет то, что для всего фильтра имеется только одна ассоциация данных на измерение. FastSLAM, из-за использования нескольких частиц, способен определить соответствие для каждой частицы. Таким образом, фильтр не только выполняет выборку по траекториям робота, но также по возможным решениям ассоциации данных на этом пути.

Это ключевое свойство FastSLAM, отделяющее его от большого семейства алгоритмов SLAM. Пока малое подмножество частиц основано на верной ассоциации данных, ошибки ассоциации данных фатальными, как в случае методов EKF, не являются. Частицы, подверженные таким ошибкам, могут создавать противоречивые карты, которые увеличивают вероятность того, что они будут исключены при перевыборке из будущих карт.

Математическое определение ассоциации данных для каждой частицы очевидно, обобщая ассоциацию данных для каждого фильтра. Для каждой частицы сохраняется локальный набор переменных ассоциации данных, обозначаемых $\hat{c}_t^{[k]}$. При ассоциации данных методом максимального правдоподобия каждый $\hat{c}_t^{[k]}$ определяется максимизацией правдоподобия измерения z_t:

(13.55)
$$\hat{c}_t^{[k]} = \operatorname*{argmax}_{c_t} p(z_t | c_t, \hat{c}_{1:t-1}^{[k]}, x_{1:t}^{[k]}, z_{1:t-1} u_{1:t})$$

Альтернативой является алгоритм выборки ассоциации данных (data association sampler- DAS), выполняющий выборку переменной ассоциации данных в соответствии с правдоподобием

(13.56)
$$\hat{c}_t \sim \eta\, p(z_t|c_t, \hat{c}_{1:t-1}, x^{[k]}_{1:t}, z_{1:t-1}u_{1:t})$$

Оба метода, и метод максимального правдоподобия, и DAS, позволяют оценить количество признаков на карте. Методы SLAM на основе методов максимального правдоподобия, создают новые признаки на карте, если правдоподобие падает ниже порога p_0 для всех известных признаков на карте. DAS выполняет стохастическую ассоциацию наблюдаемого измерения с новым, прежде ненаблюдаемым признаком. Это выполняется с вероятностью, пропорциональной ηp_0, где η - нормализующий член, определённый в (13.56).

(13.57)
$$\hat{c}^{[k]}_t \sim \eta\, p(z_t|c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_{1:t}, z_{1:t-1}u_{1:t})$$

Для обоих методов правдоподобие вычисляется следующим образом:

(13.58)
$$\begin{aligned}
&p(z_t|c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_{1:t}, z_{1:t-1}, u_{1:t}) \\
&\quad = \int p(z_t|m_{c_t}, c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_{1:t}, z_{1:t-1}, u_{1:t}) \\
&\qquad\qquad p(m_{c_t}|c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_{1:t}, z_{1:t-1}, u_{1:t}) dm_{c_t} \\
&\quad = \int \underbrace{p(z_t|m_{c_t}, c_t, x^{[k]}_t)}_{\sim\mathcal{N}(z_t; h(m_{c_t}, x^{[k]}_t), Q_t)} \underbrace{p(m_{c_t}|\hat{c}^{[k]}_{1:t-1}, x^{[k]}_{1:t-1}, z_{1:t-1})}_{\sim\mathcal{N}(\mu^{[k]}_{c_t,t-1}, \Sigma^{[k]}_{c_t,t-1})} dm_{c_t}
\end{aligned}$$

Линеаризация функции h позволяет получить его в закрытом виде:

(13.59)
$$\begin{aligned}
&p(z_t|c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_t, z_{1:t-1}, u_{1:t}) \\
&\quad = |2\pi Q^{[k]}_t|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - h(\mu^{[k]}_{c_t,t-1}, x^{[k]}_t))^T Q^{[k]-1}_t (z_t - h(\mu^{[k]}_{c_t,t-1}, x^{[k]}_t))\}
\end{aligned}$$

Переменная $Q^{[k]}_t$ была определена в выражении (13.35) как функция переменной ассоциации данных c_t. Новые признаки добавляются к карте как было описано выше. В методе максимального правдоподобия новый признак добавляется, когда вероятность $p(z_t|c_t, \hat{c}^{[k]}_{1:t-1}, x^{[k]}_t, z_{1:t-1}, u_{1:t})$ падает ниже порогаp_0. DAS включает новую гипотезу, что наблюдение, соответствующее прежде ненаблюдаемое действие в набор гипотез, и выполняет выборку с вероятностью ηp_0.

13.6 Управление картой

Управление картой FastSLAM, в основном, аналогично, EKF SLAM с небольшим количеством частиц, поскольку в FastSLAM ассоциация данных выполняется на уровне отдельных частиц.

Так же как в альтернативных алгоритмах SLAM, любой вновь добавленный признак требует инициализацию нового фильтра Калмана. Во многих задачах SLAM функция измерения h *обратима*. Это происходит, например, когда робот выполняет измерение расстояния и направления на признаки на плоскости, и единичного измерения достаточно для появления новой (невырожденной) оценки по местонахождению признака. Инициализация EKF очевидна:

(13.60)
$$x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$$

(13.61)
$$\mu_{n,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$$

(13.62)
$$\Sigma_{n,t}^{[k]} = (H_{\hat{c}}^{[k]T} Q_t^{-1} H_{\hat{c}}^{[k]})^{-1} \qquad \text{где} \qquad H_{\hat{c}}^{[k]} = h'(\mu_{n,t}^{[k]}, x_t^{[k]})$$

(13.63)
$$w_t^{[k]} = p_0$$

Заметим, что для вновь обнаруженных признаков выборка по положению $x_t^{[k]}$ выполняется в соответствии с моделью движения $p(x_t | x_{t-1}^{[k]}, u_t)$. Это распределение эквивалентно выборке из распределения FastSLAM (13.26) в ситуациях, когда отсутствуют предыдущие оценки для наблюдаемого признака.

Методы инициализации для ситуаций, когда h необратимо, обсуждались в работах Динса и Хеберта (Deans and Hebert, 2002). В таких ситуациях требуется накопление нескольких измерений, чтобы получить хорошую оценку линеаризации h.

Чтобы обрабатывать ошибочно привнесённые на карту признаки, Fast SLAM имеет механизм уничтожения признаков, которые не поддержаны достаточным объёмом наблюдений. Так же как в EKF SLAM, в FastSLAM это выполняется отслеживанием логарифма шансов существования отдельных признаков на карте.

В частности, при наблюдении признака его логарифм шансов на существование увеличивается на фиксированное значение, которое вычисляется по стандартной формуле теоремы Байеса. Похожим образом, когда признак не наблюдается в ситуации, когда он должен был присутствовать, такая отрицательная информация вызывает уменьшение переменной существования признака на фиксированное значение. Признаки, значение переменной существования которых падает ниже порогового значения, удаляются из списка частиц. Так же возможно реализовать временный список признаков в FastSLAM, что технически тривиально, поскольку для каждого признака имеется собственная частица.

13.7 Алгоритмы FastSLAM

В Таблицах 13.2 и 13.3 приводятся итоговые алгоритмы FastSLAM с неизвестной ассоциацией данных. В обоих алгоритмах частицы заданы в виде

(13.64)
$$Y_t^{[k]} = \langle x_t^{[k]}, N_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \tau_1^{[k]} \rangle, ..., \langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, \tau_{N_t^{[k]}}^{[k]} \rangle \rangle$$

Вдобавок к положению $x_t^{[k]}$ и оценкам признаков $\mu_{n,t}^{[k]}$ и $\Sigma_{n,t}^{[k]}$ в каждой частице сохраняется некоторое количество признаков $N_t^{[k]}$ на локальной карте, а каждый признак несёт вероятностную оценку существования $\tau_n^{[k]}$. Для выполнения итерации фильтра требуется время, линейно зависящее от максимального количества признаков $max_k N_t^{[k]}$ на каждой карте, и от числа частиц M. Ниже будет обсуждаться более совершенная структура данных, которая позволяет создавать более эффективные реализации.

Заметим, что обе версии FastSLAM, описанные здесь, основаны на одном измерении в один момент времени. Как и прежде, это допущение сделано для удобства записи, как было выполнено для многих методов SLAM, описанных в предыдущих главах.

13.8 Эффективная реализация

На первый взгляд, может показаться, что каждое обновление в FastSLAM требует времени $O(MN)$, где M – число частиц, а N –число признаков на карте. Линейная сложность по M неизбежна, учитывая, что при каждом обновлении необходимо обработать M частиц. Линейная сложность по N является результатом выполнения перевыборки. Когда частица извлекается при перевыборке больше одного раза, наивная реализация может дублировать всю карту, связанную с частицей. Такой процесс дублирования линейно зависит от размера карты N. Более того, наивная реализация ассоциации данных может привести к оценке правдоподобности измерения для каждого из N признаков на карте, что снова повлечёт линейную зависимость вычислительной сложности от N. Заметим, что плохая реализация процесса выборки может легко увеличить сложность обновления ещё на $\log N$.

1: **Algorithm FastSLAM 1.0**(z_t, u_t, Y_{t-1}) :
2: *для* $k = 1$ *до* M *выполнять* //*цикл по всем частицам*
3: *получить* $\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \rangle, ...,$
$\langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]},t-1}^{[k]} \rangle \rangle$ *из* Y_{t-1}
4: $x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ //*элемент положения*
5: *для* $j = 1$ *до* $N_{t-1}^{[k]}$ *выполнять* //*правдоподобие измерения*
6: $\hat{z}_j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ //*прогноз измерения*
7: $H_j = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ //*вычислить якобиан*
8: $Q_j = H_j \Sigma_{j,t-1}^{[k]} H_j^T + Q_t$ //*ковариация измерения*
9: $w_j = |2\pi Q_j|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_j)^T$
$Q_j^{-1}(z_t - \hat{z}_j)\}$ //*правдоподобие соответствия*
10: *endfor*
11: $w_{1+N_{t-1}^{[k]}} = p_0$ //*фактор значимости, новый признак*
12: $w^{[k]} = \max w_j$ //*максимальное правдоподобие соответствия*
13: $\hat{c} = \operatorname{argmax} w_j$ //*индекс максимального правдоподобия признака*
14: $N_t^{[k]} = \max\{N_{t-1}^{[k]}, \hat{c}\}$ //*новое число признаков на карте*
15: *для* $j = 1$ *до* $N_t^{[k]}$ *выполнять* //*обновить фильтры Калмана*
16: *если* $j = \hat{c} = 1 + N_{t-1}^{[k]}$ то //*новый признак?*
17: $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ //*инициализировать среднее*
18: $H_j = h'(\mu_{j,t}^{[k]}, x_t^{[k]}); \Sigma_{j,t}^{[k]} = (H_j^{-1})^T Q_t H_j^{-1}$ //*инициализировать ковариацию*
19: $i_{j,t}^{[k]} = 1$ //*инициализировать счетчик*
20: *иначе если* $j = \hat{c} \leq N_{t-1}^{[k]}$ *то* //*уже наблюдаемый признак?*
21: $K = \Sigma_{j,t-1}^{[k]} H_j^T Q_{\hat{c}}^{-1}$ //*вычислить усиление Калмана*
22: $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}_{\hat{c}})$ //*обновить среднее*
23: $\Sigma_{j,t}^{[k]} = (I - K\, H_j)\Sigma_{j,t-1}^{[k]}$ //*обновить ковариацию*
24: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ //*инициализировать счетчик*
25: *else* //*всех прочих признаков*
26: $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ //*копировать прежнее среднее*
27: $\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]}$ //*копировать прежнюю ковариацию*
28: *if* $\mu_{j,t-1}^{[k]}$ *вне зоны восприятия*
расстояние $x_t^{[k]}$ *then* //*должен ли признак быть различим?*
29: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ //*инициализировать счетчик*
30: *else*
31: $i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ //*да, уменьшить счетчик*
32: *if* $i_{j,t-1}^{[k]} < 0$ *then*
33: *отбросить признак* j //*отбросить дублирующиеся признаки*
34: *endif*
35: *endif*
36: *endif*
37: *endfor*

Продолжение на следующей странице

Начало на предыдущей странице

38: *прибавить* $\langle x_t^{[k]}, N_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \rangle, ..., \langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \rangle \rangle$ *to* Y_{aux}
39: *endfor*
40: $Y_t = \emptyset$ *//создать новый набор частиц*
41: *повторить* M *раз* *//перевыборка* M *частиц*
42: *извлечь случайный* k *с вероятностью* $\propto w^{[k]}$ *//перевыборка*
43: *прибавить* $\langle x_t^{[k]}, N_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \rangle, ..., \langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \rangle \rangle$ *to* Y_t
44: *enddo*
45: *return* Y_t

Таблица 13.2 Алгоритм FastSLAM 1.0 с неизвестной ассоциацией данных. В этой версии не реализованы методы эффективного выражения в виде деревьев, обсуждаемые в главе.

1: **Algorithm FastSLAM 2.0**(z_t, u_t, Y_{t-1}) :
2: *для* $k = 1$ *до* M *выполнять* *//цикл по всем частицам*
3: *извлечь* $\langle x_{t-1}^{[k]}, N_{t-1}^{[k]}, \langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]}, i_1^{[k]} \rangle, ..., \langle \mu_{N_{t-1}^{[k]},t-1}^{[k]}, \Sigma_{N_{t-1}^{[k]},t-1}^{[k]}, i_{N_{t-1}^{[k]}}^{[k]} \rangle \rangle$ *из* Y_{t-1}
4: *для* $j = 1$ *до* $N_{t-1}^{[k]}$ *выполнять* *//вычислить распределение выборки*
5: $\hat{x}_{j,t} = g(x_{t-1}^{[k]}, u_t)$ *//прогноз положения*
6: $\bar{z}_j = h(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t})$ *//прогноз измерения*
7: $H_{x,j} = \nabla_{x_t} h(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t})$ *//якобиан по положению*
8: $H_{m,j} = \nabla_{m_j} h(\mu_{j,t-1}^{[k]}, \hat{x}_{j,t})$ *//якобиан по признаку карты*
9: $Q_j = Q_t + H_{m,j} \Sigma_{j,t-1}^{[k]} H_{m,j}^T$ *//информация измерения*
10: $\Sigma_{x,j} = [H_{x,j}^T Q_j^{-1} H_{x,j} + R_t^{-1}]^{-1}$ *//ковариация предполагаемого распред.-я*
11: $\mu_{x_t,j} = \Sigma_{x,j} H_{x,j}^T Q_j^{-1} (z_t - \bar{z}_j) + \hat{x}_{j,t}$ *//среднее предполагаемого распределения*
12: $x_{t,j}^{[k]} \sim \mathcal{N}(\mu_{x_t,j}, \Sigma_{x,j})$ *//элемент выборки по положению*
13: $\hat{z}_j = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ *//прогноз измерения*
14: $\pi_j = |2\pi Q_j|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_j)^T Q_j^{-1}(z_t - \hat{z}_j)\}$ *//правдоподобие соответствия*
15: *endfor*
16: $\pi_{1+N_{t-1}^{[k]}} = p_0$ *//правдоподобие нового признака*
17: $\hat{c} = \operatorname{argmax} \pi_j$ *//соответствие максимального правдоподобия*
18: $N_t^{[k]} = \max\{N_{t-1}^{[k]}, \hat{c}\}$ *//новое число признаков*
19: *для* $j = 1$ *до* $N_t^{[k]}$ *выполнять* *//обновить фильтры Калмана*
20: *если* $j = \hat{c} = 1 + N_{t-1}^{[k]}$ *то* *//новый признак?*

Продолжение на следующей странице

Начало на предыдущей странице

$x_t^{[k]} \sim p(x_t | x_{t-1}^{[k]}, u_t)$ //*элемент выборки по положению*
$\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ //*инициализировать среднее*
$H_{m,j} = \nabla_{m_j} h(\mu_{j,t}^{[k]}, x_t^{[k]})$ //*якобиан по признаку карты*
$\Sigma_{j,t}^{[k]} = (H_{m,j}^{-1})^T Q_t H_{m,j}^{-1}$ //*инициализировать ковариацию*
$i_{j,t}^{[k]} = 1$ //*инициализировать счетчик*
$w^{[k]} = p_0$ //*вес значимости*
иначе если $j = \hat{c} \leq N_{t-1}^{[k]}$ *то* //*признак уже наблюдался?*
$x_t^{[k]} = x_{t,j}^{[k]}$
$K = \Sigma_{j,t-1}^{[k]} H_{m,j}^T Q_j^{-1}$ //*вычислить калмановское усиление*
$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}_j)$ //*обновить среднее*
$\Sigma_{j,t}^{[k]} = (I - K H_{m,j}) \Sigma_{j,t-1}^{[k]}$ //*обновить ковариацию*
$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} + 1$ //*увеличить счетчик*
$L = H_{x,j} R_t H_{x,j}^T + H_{m,j} \Sigma_{j,t-1}^{[k]} H_{m,j}^T + Q_t$
$w^{[k]} = |2\pi L|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(z_t - \hat{z}_j)^T$
$L^{-1}(z_t - \hat{z}_j)\}$ //*вес значимости*
иначе //*все прочие признаки*
$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]}$ //*копировать прежнее среднее*
$\Sigma_{j,t}^{[k]} = \Sigma_{j,t-1}^{[k]}$ //*копировать прежнюю ковариацию*
если $\mu_{j,t-1}^{[k]}$ *вне зоны восприятия*
расстояние $x_t^{[k]}$ *then* //*наблюдался ли уже признак?*
$i_{j,t}^{[k]} = i_{j,t-1}^{[k]}$ //*нет, не менять*
else
$i_{j,t}^{[k]} = i_{j,t-1}^{[k]} - 1$ //*да, уменьшить счетчик*
if $i_{j,t-1}^{[k]} < 0$ *then*
отбросить признак j //*отбросить дублирующиеся признаки*
endif
endif
endif
endfor
прибавить $\langle x_t^{[k]}, N_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \rangle, ..., \langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \rangle \rangle$ *к* Y_{aux}
endfor
$Y_t = \emptyset$ //*создать новый набор частиц*
выполнить M *раз* //*перевыборка* M *частиц*
извлечь k *случайных с вероятностью* $\propto w^{[k]}$ //*перевыборка*
прибавить $\langle x_t^{[k]}, N_t^{[k]}, \langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, i_1^{[k]} \rangle, ..., \langle \mu_{N_t^{[k]},t}^{[k]}, \Sigma_{N_t^{[k]},t}^{[k]}, i_{N_t^{[k]}}^{[k]} \rangle \rangle$ *к* Y_t
enddo
return Y_t

Таблица 13.3 Алгоритм FastSLAM 2.0 для неизвестной ассоциации данных.

Эффективные реализации алгоритма FastSLAM требуют только $O(M \log N)$ на операцию обновления, то есть наблюдается логарифмическая зависимость от размера карты N. Сначала рассмотрим ситуацию с известной ассоциацией данных. Линейных затрат времени на копирование можно из-

бежать, используя структуру данных для представления частиц, которая позволит более выборочное обновление. Общая идея состоит в организации карты в виде *сбалансированного бинарного дерева*. На Рис. 13.8a показано такое дерево для одной частицы, в случае, когда число признаков $M = 8$. Заметим, что все гауссовы параметры $\mu_j^{[k]}$ и $\Sigma_j^{[k]}$ для всех j расположены в листах дерева. Считая, что дерево примерно сбалансировано, доступ к листу требует времени, логарифмически зависящего от N.

Пусть FastSLAM учитывает новое управляющее воздействие u_t и новое измерение z_t. Каждая новая частица в Y_t будет отличаться от соответствующей частицы в Y_{t-1} по двум параметрам: во-первых, у неё будет другая оценка местоположения, полученная с помощью (13.26), во-вторых, будет обновлён гауссиан наблюдаемого признака, как показано в выражениях (13.47) и (13.48). Однако, все прочие гауссовы оценки признаков будут эквивалентны сгенерированной частице. При копировании частицы, таким образом, изменится только один путь в дереве, выражающем все гауссианы, что даст логарифмическую зависимость времени обновления.

Иллюстрация такого «приёма» показана на Рис. 13.8b. Примем $c_t^i = 3$, поскольку обновляются только гауссовы параметры $\mu_3^{[k]}$ и $\Sigma_3^{[k]}$. Вместо генерации полностью нового дерева создаётся только один путь, ведущий к гауссиану $c_t^i = 3$, который представляет собой неполное дерево. Дерево дополняется копированием пропущенных указателей из дерева генерации частицы. Поэтому, ветви, отходящие от пути, будут иметь указатели на то же неизменное под-дерево, что и дерево генерации частицы. Очевидно, генерация такого дерева потребует времени, логарифмически зависящего от N. Более того, получение доступа к гауссиану также требует времени, логарифмически зависящего от N, поскольку количество шагов, требуемых для достижения листа дерева, эквивалентно длине пути, имеющего, по определению, логарифмическую сложность. В силу этого, и генерация частичного дерева, и навигация по нему могут быть выполнены за время $O(\log N)$. Поскольку на каждом шаге обновления будет создано M новых частиц, вся процедура обновления потребует времени $O(M \log N)$.

Организация частиц в виде деревьев открывает вопрос о необходимости освобождения памяти. Освобождение памяти также может быть выполнено менее чем за логарифмически зависимый промежуток времени. Нужно лишь назначить переменную каждому узлу, внутреннему или же листу, и выполнить подсчёт числа указателей на узел. Счётчик вновь созданного узла будет инициализирован единицей. По мере того, как другими частицами будут создаваться ссылки на узел, значение счётчика увеличится. Значение уменьшается при удалении указателя (например, указателей от частиц, которые были отсеяны при перевыборке). Как только значение счётчика достигнет нуля, значения всех дочерних счётчиков должна уменьшается, а память, занимаемая узлом - освобождается. Процесс рекурсивно применяется ко всем дочерним связям узла, значение счётчиков которых тоже может достичь нуля. Этот рекурсивный процесс, в среднем, занимает время $O(M \log N)$.

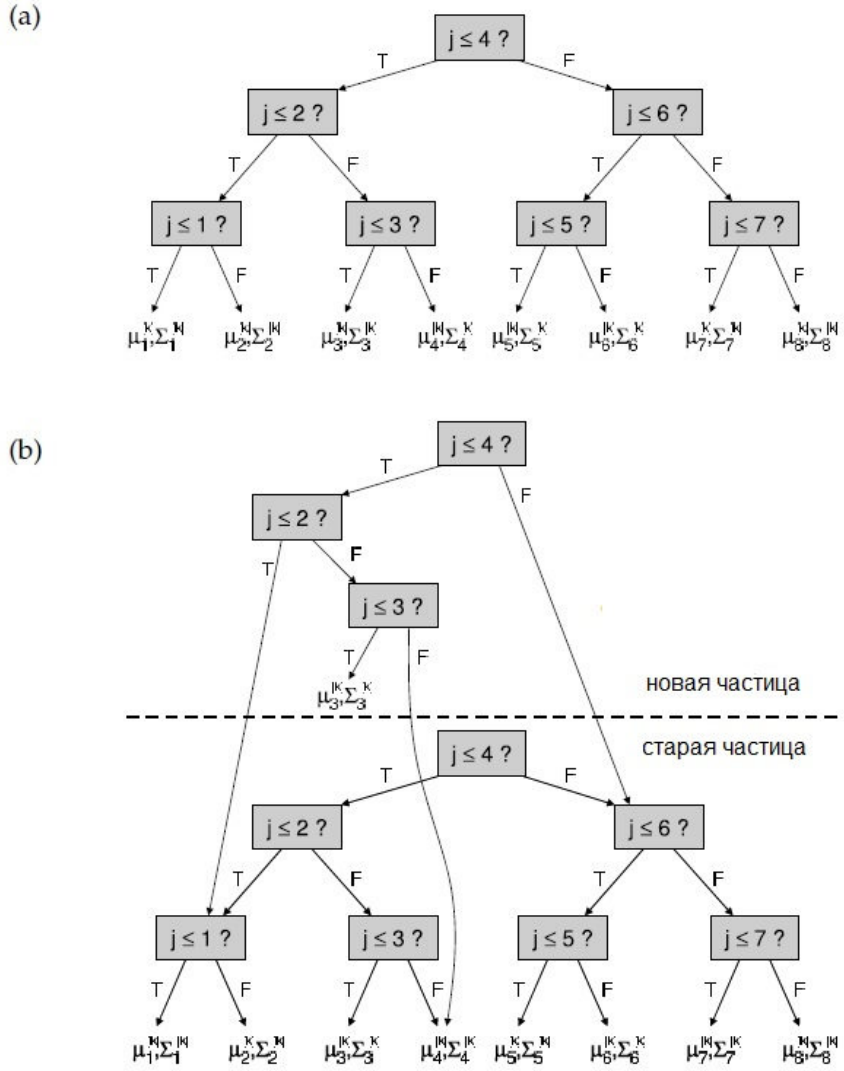


Рис. 13.8 Дерево, выражающее $N = 8$ оценок признаков в одной частице(a). Генерация новой частицы на основе старой, требует изменения лишь одного гауссиана (b). Новая частица получает лишь частичное дереве, состоящее из пути к измененному гауссиану. Все другие указатели копируются из генерирующего дерева, что может быть выполнено за время, логарифмически зависящее от N.

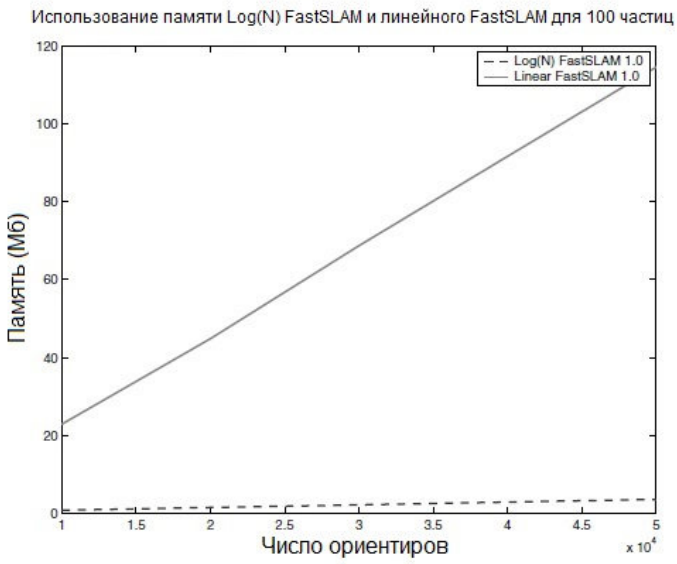


Рис. 13.9 Требования памяти для линейной и логарифмической $\log(N)$ версии FastSLAM 1.0.

Дерево позволяет существенно экономить память. На Рис. 13.9 показано влияние эффективной технологии отображения в виде дерева на требуемую для FastSLAM память. Этот граф является результатом реального использования FastSLAM 1.0 с $M = 100$ частицами для получения карты на основе признаков. Согласно графу, для карты с 50000 признаков экономия составляет почти два порядка. Относительная экономия времени обновления имеет близкое значение.

Получение логарифмической сложности по времени для FastSLAM с неизвестной ассоциацией данных более трудно. Потребуется метод ограничения поиска ассоциации данных локальной окрестностью признака, чтобы предотвратить циркуляцию вероятности ассоциации данных для всех N признаков карты. Кроме того, дерево должно оставаться приблизительно сбалансированным.

Конечно, существуют варианты деревьев ядерной оценки плотности (kernel density trees, kd-trees), способные удовлетворять этим требованиям, подразумевая, что дисперсия измерений датчика по сравнению с общим размером карты мала. Например, *bkd-дерево* предложенное Прокопюк (Procopiuc et al., 2003), сохраняет последовательность деревьев возрастающей сложности. Путём последовательного смещения элементов по этим деревьям можно гарантировать амортизационное логарифмическое время вызова для вставки новых признаков в карту. Другим алгоритмом является DP-SLAM, предложенный Елизаром и Парром (Eliazar and Parr, 2003), в котором для эффективного хранения и получения используются *деревья истории*, похожие на описанные.

13.9 FastSLAM для карт на основе признаков

13.9.1 Эмпирические наблюдения

Алгоритм FastSLAM применялся для большого количества выражений карт и данных датчиков. Наиболее общий вариант использования основан на картах признаков, считая, что робот оснащён датчиком для обнаружения расстояния и угла направления на ориентиры. Один из таких наборов данных, парк Виктории, уже обсуждался в Главе 12. На Рис. 13.10a показан путь робота на основе оценок управляющего воздействия. Управляющие воздействия плохо прогнозируют местоположение транспорта, после 30 минут движения оценочное местоположение автомобиля более чем на 100 метров отличалось от положения по GPS.

На оставшихся трёх врезках на Рис. 13.10 показан итог работы FastSLAM 1.0. На всех схемах путь по GPS показан пунктиром, а результат работы FastSLAM – сплошной линией. Среднеквадратичная ошибка результирующего пути составляет чуть больше 4 метров для перемещения длиной 4. Этот эксперимент проводился с числом частиц $M = 100$. Величина ошибки практически не отличается от ошибки других наиболее совершенных алгоритмов SLAM, обсуждаемых в предыдущих главах. Надёжность FastSLAM становится очевидным из Рис. 13.10d, на котором приводится результат эксперимента, в котором информация о движении просто игнорировалась. Вместо этого, модель движения на основе одометрии была заменена броуновской моделью движения. Средняя ошибка FastSLAM статистически неотличима от ошибки, полученной ранее.

При реализации FastSLAM в картах на основе признаков важно учитывать отрицательную информацию. Когда отрицательная информация используется для оценки вероятности существования каждого признака, как описано в подразделе 13.6, многие ошибочные признаки на карте можно удалить. На Рис. 13.11 показана карта парка Виктория с учётом отрицательной информации и без него. Здесь использование отрицательной информации привело к уменьшению количества признаков результирующей карты на 44% . Хотя истинное количество признаков недоступно, даже при беглом взгляде на карты видно, что множество ошибочных признаков было отброшено.

Имеет смысл сравнить FastSLAM с EKF SLAM, который остаётся популярным эталонным алгоритмом. Например, на Рис. 13.12 приводится сравнение точности FastSLAM 1.0 с EKF для различного числа частиц от 1 до 5000. Для сравнения ошибка EKF SLAM показана пунктирной горизонтальной линией на Рис. 13.12. Точность FastSLAM 1.0 достигает точности EKF по мере увеличения количества частиц, и становится статистически неотличима после, приблизительно, 10 частиц. Это интересно уже потому, что FastSLAM 1.0 с 10 и 100 частицами требует на порядок меньше параметров, чем EKF SLAM для достижения аналогичного уровня точности.

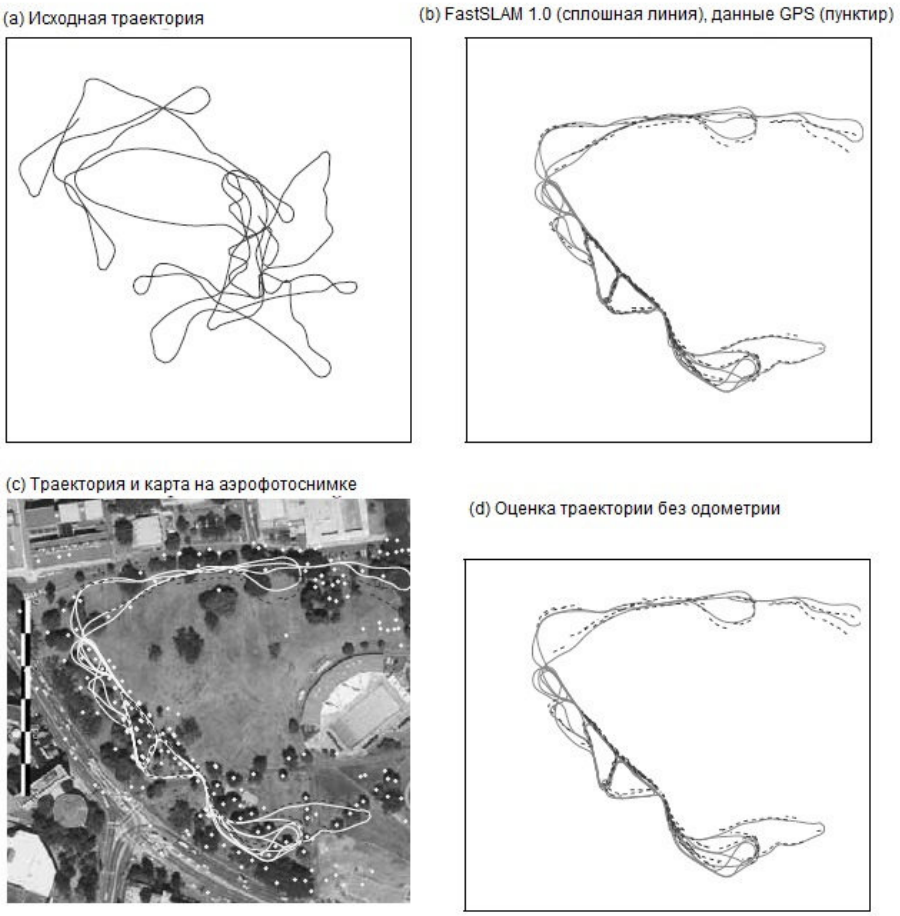


Рис. 13.10 Путь автомобиля на основе одометрии (a); Реальный путь (показан пунктиром) и путь согласно FastSLAM 1.0 (сплошная линия) (b); результаты обработки датасета наложенные на аэрофотосъемку, где путь согласно GPS показан синим (пунктиром), усреднённый путь FastSLAM 1.0 показан желтым (сплошная линия), а оценённые признаки показаны жёлтыми кругами (c). Набор данных созданный без информации об одометрии(d). Данные и аэрофотография принадлежат Хосе Гюванту и Эдуарду Неботу, австралийский центр полевой робототехники (José Guivant and Eduardo Nebot, Australian Centre for Field Robotics).

Рис. 13.11 FastSLAM 1.0 (a) без удаления признаков (b) с удалением признаков на основе отрицательной информации.

На практике, FastSLAM 2.0 даёт лучшие результаты по сравнению Fast SLAM 1.0, хотя улучшение существенно только при определённых обстоятельствах. Как правило, результаты обоих алгоритмов сравнимы при большом числе частиц M и когда шум измерений велик по сравнению с неопределённостью движения. Это показано на Рис. 13.13, где точность каждого варианта FastSLAM для $M = 100$ частиц изображена в виде функции шума измерения. Наиболее важное наблюдение состоит в сравнительно низкой эффективности FastSLAM 1.0 в экспериментах с малошумными датчиками. Одним из способов проверить, страдает ли реализация FastSLAM 1.0 от этой проблемы, является искусственное увеличение шума измерений в вероятностной модели $p(z|x)$. Если в результате такого увеличения ошибка уменьшается, время переключиться на FastSLAM 2.0.

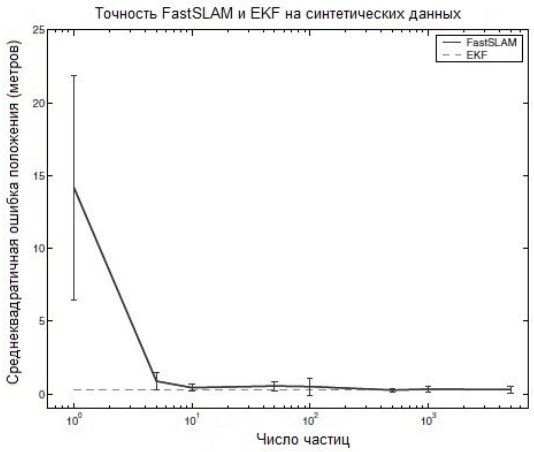


Рис. 13.12 Сравнительная точность FastSLAM 1.0 и EKF на синтетических данных.

13.9.2 Замыкание цикла

Совершенных алгоритмов не существует. Есть задачи, в которых FastSLAM проигрывает конкурентам на основе гауссовых моделей. Одна из таких задач включает *замыкание цикла*. В замыкании цикла робот движется через

неизвестную территорию и в некоторой точке сталкивается с признаками, которые не были видны долгое время. Сохранение корреляции в алгоритме SLAM оказывается важным, поэтому информация, полученная при закрытии цикла, может распространяться через всю карту. EKF и GraphSLAM напрямую сохраняют такие корреляции, а FastSLAM сохраняет их с помощью разнообразия в наборах частиц. Поэтому, возможность замыкания циклов зависит от числа частиц M. Большее разнообразие в наборе элементов выборки приводит к лучшим показателям при замыкании циклов, поскольку новые наблюдения влияют на информацию о положении робота дальше в прошлом.

К сожалению, при обрезке неправильных траекторий робота перевыборка однажды вызывает ситуацию, когда все частицы FastSLAM разделяют общую историю, начиная с некоторой точки в прошлом. Новые наблюдения неспособны повлиять на признаки, наблюдаемые до определённого момента. Общая точка истории может быть отодвинута назад во времени путём увеличения количества частиц M. Такой процесс отбрасывания данных корреляции со временем позволяет FastSLAM эффективно выполнять обновление датчика. Эта эффективность даётся ценой меньшей скорости сходимости. Отбрасывание информации корреляции означает, что для достижения заданного уровня точности требуется больше наблюдений. Очевидно, улучшенное предположительное распределение FastSLAM 2.0 гарантирует, меньшее количество уничтоженных частиц при перевыборке в сравнении с FastSLAM 1.0, но в решении проблемы это не поможет.

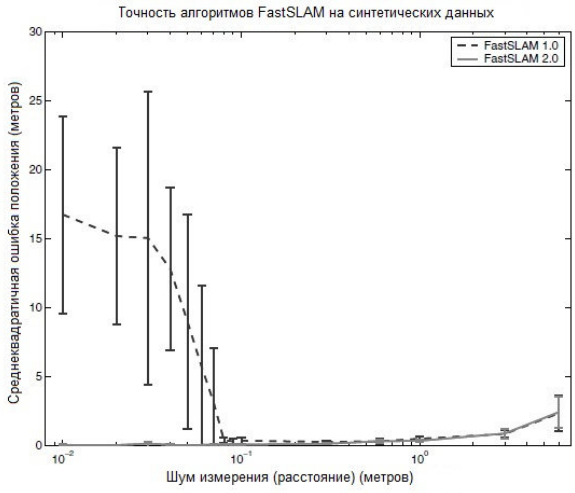


Рис. 13.13 FastSLAM 1.0 и 2.0 с разным уровнем шумов измерений: как и ожидалось, FastSLAM 2.0 практически равномерно превосходит FastSLAM 1.0. Разница особенно очевидна для малых наборов частиц, где улучшенное предположительное распределение значительно лучше фокусирует частицы.

На практике поддержка разнообразия важно, и стоит уделить внимание оптимизации алгоритма для поддержания максимального разнообразия. Примеры замыкания цикла показаны на Рис. 13.15. На схемах приводятся истории всех M частиц. На Рис. 13.15a частицы FastSLAM 1.0 имеют общую часть истории вокруг цикла. Новые наблюдения неспособны повлиять на положение признаков, наблюдаемых до порога. В случае FastSLAM 2.0 алгоритм в состоянии поддерживать разнообразие распространяя до на-

чала цикла. Это критично для надёжного замыкания цикла и быстрой сходимости.

На Рис. 13.16a показан результат эксперимента по сравнению эффективности по замыканию цикла в FastSLAM 1.0 и 2.0. по мере увеличения размера цикла ошибка обоих алгоритмов возрастает. Однако, FastSLAM 2.0 стабильно превосходит FastSLAM 1.0. Этот же результат можно перефразировать в терминах частиц. FastSLAM 2.0 требует меньше частиц для закрытия данного цикла FastSLAM 1.0.

Рис. 13.14 Карта парка Виктории, созданная FastSLAM 2.0 с $M = 1$ частицей.

На Рис. 13.16b показаны результаты эксперимента сравнения скорости сходимости FastSLAM 2.0 и EKF. FastSLAM 2.0 (с 1, 10 и 100 частиц) и EKF были запущены по 10 раз на большом имитационном цикле признаков, похожим на показанный на Рис. 13.16a и b. Для каждого запуска были использованы различные начальные параметры, вызывая генерацию различных наблюдений и управляющих воздействий в каждом цикле. Среднеквадратичная ошибка местоположения на карте на каждом такте времени была усреднена по 10 запускам для каждого алгоритма.

По мере перемещения робота по карте ошибка постепенно возрастает. Когда робот замыкает цикл на итерации 150, пересмотр старых признаков должен влиять на местоположение всех признаков вокруг цикла, вызывая уменьшение общей ошибки карты. Очевидно, так происходит при использовании EKF. FastSLAM 2.0 с единственной частицей неспособен повлиять на местоположение прошлых признаков, поэтому ошибка признака не уменьшается. По мере добавления частиц к FastSLAM 2.0 фильтр способен применять данные наблюдений к местоположению признаков для прошлых моментов времени, постепенно достигая скорости сходимости EKF. Очевидно, количество частиц, необходимых для достижения времени сходимости, близкого к EKF, будет возрастать с размером цикла. Недостаток долговременных корреляций в выражении FastSLAM, пожалуй, наиболее важный недостаток в сравнении с методами SLAM на основе гауссовых представлений.

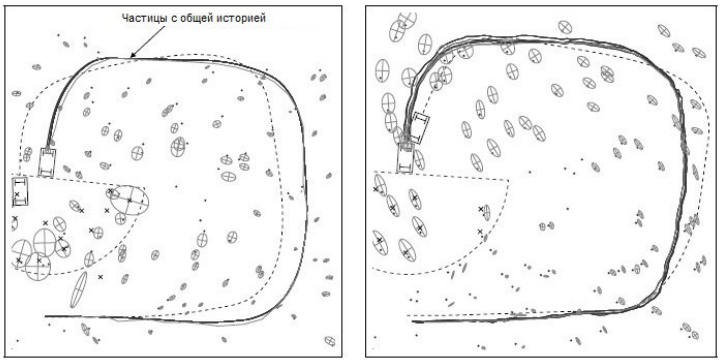


Рис. 13.15 FastSLAM 2.0 может закрывать циклы большего размера, чем FastSLAM 1.0 при одинаковом количестве признаков.

13.10 FastSLAM на основе сетки

13.10.1 Алгоритм

В Главе 9 были изучены *карты сеток занятости* как объёмное выражение окружающей среды робота. Преимуществом такого выражения является то, что для них не требуется предопределённых определений ориентиров. Вместо этого, можно моделировать произвольные типы сред. В оставшейся части главы будет выполнено обобщение FastSLAM для таких представлений сред.

Для адаптации алгоритма FastSLAM к картам сеток занятости, понадобится три функции, уже определённые в предыдущих разделах. Во-первых, необходимо выполнить выборку из апостериорной вероятности движения $p(x_t|x_{t-1}^{[k]}, u_t)$ как в выражении 13.12. Для этого необходим метод выборки. Во-вторых, понадобится метод оценки карты для каждой частицы. Выяснилось, что мы можем полагаться на картографирование на основе сеток занятости, описанных в Главе 9. Наконец, требуется вычислить веса значимости отдельных частиц. Для этого необходим метод вычисления правдоподобия $p(z_t|x_t^k, m^{[k]})$ наблюдения z_t зависящего от положения x_t^k, карты $m^{[k]}$, и самого последнего измерения z_t.

Как оказалось, обобщение FastSLAM для карт сеток занятости довольно очевидно. В Таблице 13.4 описан алгоритм FastSLAM с картами сеток занятости. Неудивительно, что в алгоритме присутствуют элементы локализации методом Монте-Карло (см. Таблицу 8.2) и картографирования на основе сеток занятости (см. Таблицу 9.1). Отдельные функции, используемые в алгоритме, являются вариантами аналогичных функций, применяемыми для локализации и картографирования.

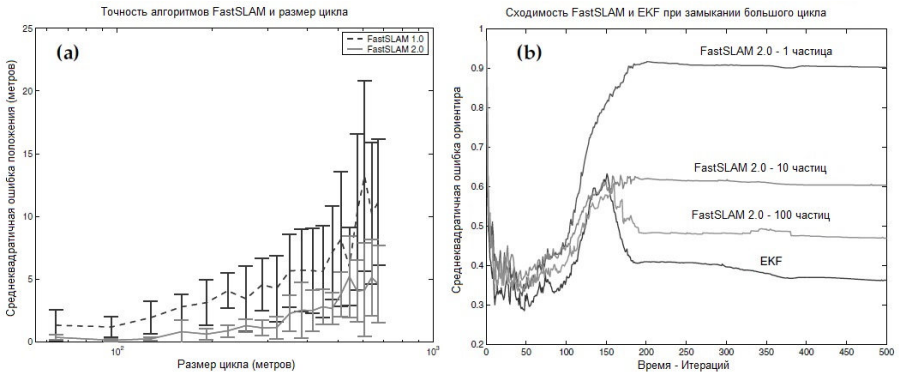


Рис. 13.16 Точность как функция размера цикла: FastSLAM 2.0 способен замыкать циклы большего размера, чем FastSLAM 1.0 при одинаковом количестве частиц (a). Сравнение скорости сходимости FastSLAM 2.0 и EKF (b).

В частности, функция **measurement_model_map**$(z_t, x_t^{[k]}, m_{t-1}^{[k]})$ вычисляет правдоподобие измерения z_t на основании положения $x_t^{[k]}$, выраженного k-й частицы и заданной картой $m_{t-1}^{[k]}$ предыдущего измерения и траекторией, выраженной этой частицей. Далее, функция, **updated_occupancy_grid**$(z_t, x_t^{[k]}, m_{t-1}^{[k]})$ вычисляет новую карту сетки занятости, на основании текущего положения $x_t^{[k]}$ k-ой частицы, связанной с ней карты $m_{t-1}^{[k]}$ и измерения z_t.

13.10.2 Эмпирические соображения

На Рис. 13.17 показана типичная ситуация использования алгоритма FastSLAM на основе сетки. Изображены три частицы со связанными с ними картами. Каждая частица выражает одну из потенциальных траекторий робота, что объясняет разницу карты сеток занятости. Центральная карта наилучшая с точки зрения глобальной целостности.

Типичная карта, полученная с помощью алгоритма FastSLAM, приведена на Рис. 13.19. Размер среды составляет 28 м×28 м. Длина траектории робота составляет 491 м, а средняя скорость – 0,19 м/сек. Разрешение карты составляет 10 см. Для обучения этой карты было использовано всего 500 частиц. В течение всего процесса робот перемещался по двум циклами. Карта, вычисленная только на основе данных одометрии, показана на Рис. 13.18, где чётко видна величина погрешности одометрии робота.

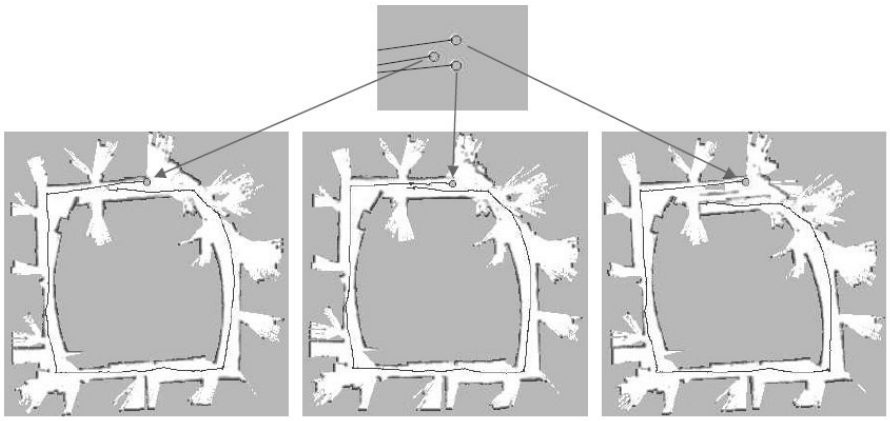


Рис. 13.17 Применение варианта алгоритма FastSLAM на основе сетки. Для каждой частицы имеется собственная карта, веса значимости части вычисляются на основе правдоподобия измерений, заданных собственными картами частицы.

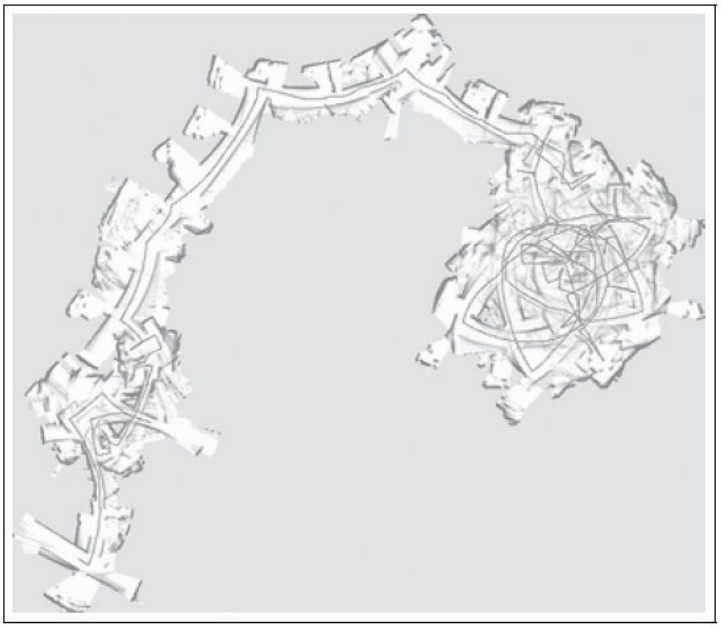


Рис. 13.18 Карта сетки занятости, сгенерированная из данных лазерного датчика расстояния и основанная только на одометрии. Все изображения принадлежат Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

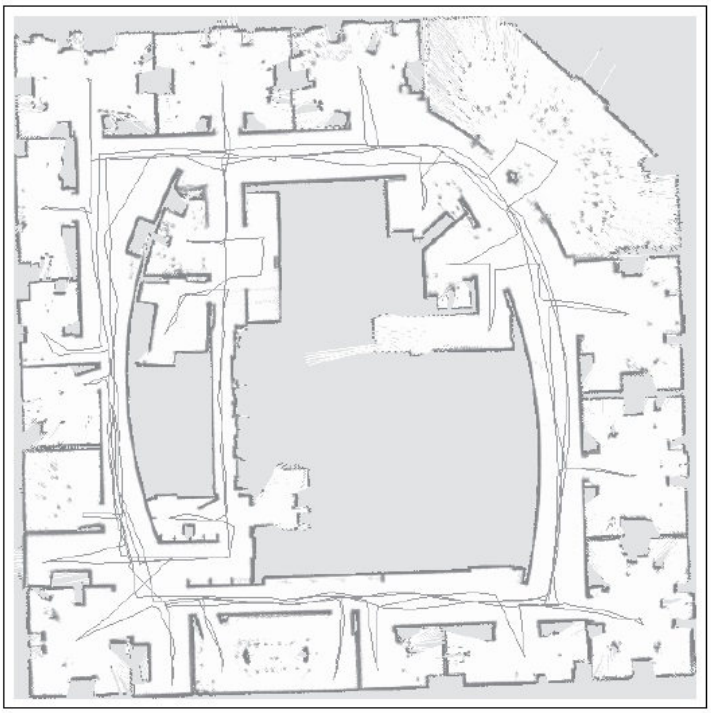


Рис. 13.19 Карта сетки занятости, соответствующая частице с наибольшим аккумулированным весом значимости, полученная алгоритмом, приведённым в Таблице 13.4 из данных, приведённых на Рис. 13.18. Количество частиц в этом эксперименте составляло 500. На рисунке также показан путь, выраженный частицей с максимальным аккумулированным весом значимости.

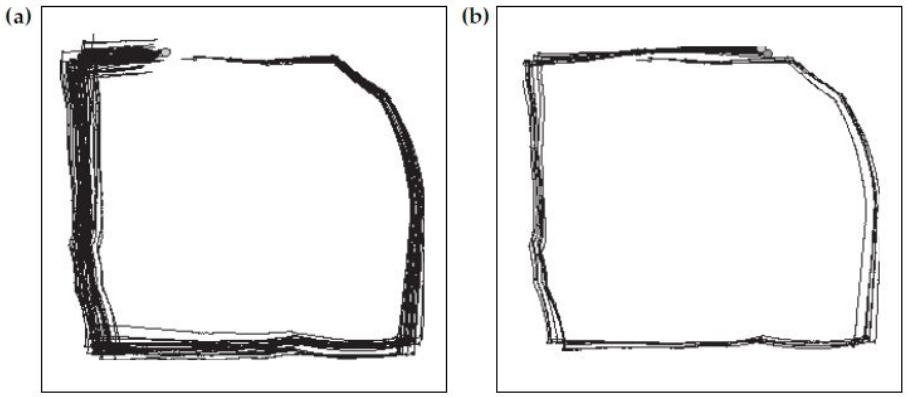


Рис. 13.20 Траектории по всей выборке перед (слева) и после (справа) замыкания внешнего цикла в среде, приведённой на Рис. 13.19. Изображения принадлежат Дирку Хенелу, университет Фрайбурга (Dirk Hähnel, University of Freiburg).

1: **Algorithm FastSLAM_occupancy_grids**$(\mathcal{X}_{t-1}, u_t, z_t)$:

2: $\quad \bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: $\quad$ *для* $k = 1$ *до* M *выполнять*
4: $\quad\quad x_t^{[k]} = \textbf{sample\_motion\_model}\,(u_t, x_{t-1}^{[k]})$
5: $\quad\quad w_t^{[k]} = \textbf{measurement\_model\_map}\,(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
6: $\quad\quad m_t^{[k]} = \textbf{updated\_occupancy\_grid}\,(z_t, x_t^{[k]}, m_{t-1}^{[k]})$
7: $\quad\quad \bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[k]}, m_t^{[k]}, w_t^{[k]} \rangle$
8: $\quad$ *endfor*
9: $\quad$ *для* $k = 1$ *до* M *выполнять*
10: $\quad\quad$ *извлечь* i *с вероятностью* $\propto w_t^{[i]}$
11: $\quad\quad$ *прибавить* $\langle x_t^{[i]}, m_t^{[i]} \rangle$ *к* $\mathcal{X}_t$
12: $\quad$ *endfor*
13: $\quad$ *return* $\mathcal{X}_t$

Таблица 13.4 Алгоритм FastSLAM для обучения карт сеток занятости.

Важность использования нескольких частиц становится очевидной на Рис. 13.20, иллюстрирующем траектории, полученные выборкой до и после замыкания цикла. Как видно на левом рисунке, положение робота достаточно неопределённо относительно начального, отсюда большой разброс к к моменту замыкания цикла. Однако, уже несколько шагов перевыборки после того, как робот оказался на уже знакомой территории, оказалось достаточно для значительного уменьшения неопределённости (правый рисунок).

13.11 Выводы

В этой главе был представлен метод многочастичного фильтра для решения задачи SLAM известный как алгоритм FastSLAM.

• Основная идея FastSLAM состоит в сохранении набора частиц. В каждой частице путь робота содержится в виде выборки элементов. В ней также содержится карта, но каждый признак на карте выражается собственным локальным гауссианом. Результирующее выражение требует памяти, линейно зависящей от размера карта и количества частиц.

• Основное преимущество представления карты в виде набора отдельных гауссиан, вместо одной общей функции, как для алгоритма EKF SLAM, возможно из-за факториальной структуры задачи SLAM. Было замечено, что признаки карты для данного пути условно независимы. Выполняя факторизацию по пути (по одному коэффициенту на частицу), можно просто считать каждый признак карты независимым, избежав затратный шаг сохранения корреляции между ними, который так вредит в методе EKF.

• Обновление в FastSLAM напрямую повторяет обновление обычного многочастичного фильтра: выполняется выборка по новому положению, затем обновляются обнаруженные признаки. Это обновление может выпол-

няться онлайн, а FastSLAM – служить решением онлайн задачи SLAM.

• Далее, было замечено, что FastSLAM решает обе задачи SLAM: онлайн SLAM и оффлайн SLAM. Метод вывода заставляет считать FastSLAM оффлайн алгоритмом, в котором частицы представляют выборку в пространстве пути, а не только положений текущего момента. Однако, ни для одного из шагов обновления не требуется знания положений, кроме текущего, поэтому оценки прежних положений можно отбросить. Это делает возможным запуск FastSLAM в виде фильтра и предотвращает линейный рост количества частиц с увеличением карты.

• Мы столкнулись с двумя вариантами FastSLAM, с номерами версий 1.0 и 2.0. FastSLAM 2.0 является расширенной версией FastSLAM. Он отличается от базовой версии одной ключевой идеей: FastSLAM 2.0 учитывает измерение при выборке нового положения. Математический вывод оказался несколько более сложным, но FastSLAM 2.0 превзошёл предшественника FastSLAM 1.0, поскольку ему требуется меньше частиц.

• Идея использования многочастичных фильтров делает возможным удаление переменных ассоциации данных на основе частиц. Каждая частица может основываться на разной ассоциации данных. Это даёт FastSLAM очевидный и мощный механизм решения проблем ассоциации данных в SLAM. Предыдущие алгоритмы, особенно EKF, GraphSLAM, и SEIF, могли использовать лишь единственное решение ассоциации данных для всего фильтра в произвольный момент времени, а значит, требовали большего внимания при выборе значения ассоциации данных.

• Для эффективного обновления частиц со временем обсуждались отображения карты в виде дерева. Эти отображения позволили уменьшить сложность обновления FastSLAM с линейной до логарифмической, позволив разделение идентичных участков карты между частицами. Идеи таких деревьев важны на практике, поскольку позволяют масштабировать FastSLAM для обработки 10^9 или даже более признаков на карте.

• Также обсуждались методы использования отрицательной информации. Один из них выполняет удаление с карты признаков, которые не поддерживаются существенными данными наблюдений. Здесь FastSLAM использует метод интеграции доказательств, знакомый с главы о картах сеток занятости. Другой касается взвешивания самих частиц. Если на карте частицы не удалось обнаружить искомую частицу путём измерения, значимость такой частицы может быть понижена умножением на соответствующий коэффициент.

• Было обсуждено множество практических свойств двух алгоритмов FastSLAM. Эксперименты показали работоспособность обоих алгоритмов на практике как для карт на основе признаков, так и для объёмных карт сеток занятости. С практической точки зрения, FastSLAM является одним из лучших вероятностных методов SLAM на текущий момент. Его возможности масштабирования сравнимы только с возможностями некоторых алгоритмов информационных фильтров, описанных в предыдущих двух главах.

• Алгоритм FastSLAM 2.0 был обобщён для различных представлений карты. В одном из представлений карта состояла из точек, обнаруженных

лазерным датчиком расстояния. В этом случае удалось отказаться от идеи моделировать неопределённость в признаках с помощью нормальных распределений и положиться на методы сравнения сканирований для реализации прямого процесса выполнения выборки в FastSLAM 2.0. Использование многочастичного фильтра дало надежный метод замыкания цикла.

Возможно, самым большим ограничением FastSLAM является неявная сохранение зависимости оценок местоположений признаков в виде разнообразия в наборе частиц. При определённых обстоятельствах это может негативно повлиять на скорость сходимости по сравнению с гауссовыми методами SLAM. При использовании FastSLAM следует предпринимать меры по уменьшению негативных эффектов истощения частиц в FastSLAM.

13.12 Библиографические примечания

Идея вычисления распределений по наборам переменных, комбинируя выборку с параметрической функцией плотности, принадлежит Рао (Rao (1945) и Блеквеллу (Blackwell, 1947). Сегодня эта идея превратилась в общепринятый инструмент в литературе по статистике (Gilks et al. 1996; Doucet et al. 2001). Первый алгоритм картографирования на основе многочастичных фильтров с возможностью замыкания циклов был найден Труном (Thrun et al., 2000b). Формальное введение в многочастичные фильтры Рао-Блеквелла в области SLAM было выполнено Мерфи (Murphy, 2000a), Мерфи и Расселом (Murphy and Russell, 2001), разработавшими эту идею в контексте карт сеток занятости.

Алгоритм FastSLAM был впервые разработан Монтемерло (Montemerlo et al., 2002a), который также предложил выражения в виде деревьев для эффективного сохранения нескольких карт. Обобщение этого алгоритма для карт высокого разрешения было выполнено Елизаром и Парром, чей алгоритм DP-SLAM генерировал карты из показаний лазерного датчика расстояния с беспрецедентной на тот момент точностью и детализацией. Центральная структура данных называлась "деревья наследования" и расширяла деревья FastSLAM для обновления карт на основе сеток занятости. Более эффективная версия известна как DP-SLAM 2.0 (Eliazar and Parr 2004). Алгоритм FastSLAM 2.0 был разработан Монтемерло (Montemerlo et al., 2003b) и основан на предыдущей работе ван дер Мерве (van der Merwe et al., 2001), впервые предложившем идею использования измерения как части предполагаемого распределения в теории многочастичных фильтров. Алгоритм FastSLAM на основе сеток в этой главе принадлежит Хенелу (Hähnel et al., 2003b), который интегрировал идею улучшенного предполагаемого измерения с фильтрами Рао-Блеквелла, применимыми для карт на основе сеток. Фильтр Рао-Блеквелла для отслеживания состояния дверей в динамической офисной среде описан в работе Авотса (Avots et al., 2002).

Одним из наиболее важных вкладов в FastSLAM является область ассоциации данных, которой уделялось много внимания в литературе по SLAM. В оригинальной работа по SLAM (Smith et al. 1990; Moutarlier and Chatila 1989a) ассоциация данных выполнялась методом максимального правдоподобия, как было детально выведено Диссанаяки (Dissanayake et al., 2001). Ключевым ограничением этих методов ассоциации данных была невозможность обеспечить взаимную исключительность: два разных признака в одном измерении датчика (или в течение короткого промежутке времени) не могут соответствовать одному и тому же физическому признаку окружающего мира. Воспользовавшись этом принципом, Ниера и Тардос разрабо-

тали методы проверки соответствия для наборов признаков, позволившие уменьшить количество ошибок ассоциации данных. Чтобы адаптировать алгоритм к огромному количеству потенциальных ассоциаций (экспоненциальному к количеству признаков, учитываемых в каждый момент времени), Ниера (Neira et al., 2003) предложил методы случайной выборки в пространстве ассоциации данных. Однако, все эти подходы сохраняли в апостериорном распределении SLAM только одну моду. Федер (Feder et al., 1999) применил идею жадной ассоциации данных к данным сонара, но для разрешения двойственности реализовал отложенное решение.

Идея сохранения многомодального апостериорного распределения в SLAM восходит к работам Дюрpран – Уайта (Durrant-Whyte et al., 2001), в алгоритме которого используется смесь нормальных функций для выражения апостериорной вероятности. Каждый компонент смеси соответствует разному следу в истории всех решений ассоциации данных. FastSLAM тоже следует этой идее, но вместо смеси компонент гауссиан используются частицы. Идею ленивой ассоциации данных можно отследить до других областей, например, популярного алгоритма RANSAC (Fischler and Bolles 1981) в машинном зрении. Алгоритм дерева, представленный в предыдущей главе, был разработан Хенелом (Hähnel et al., 2003a). Как уже упоминалось, он основан на работах Куперса (Kuipers et al., 2004). Совершенно иной подход в ассоциации данных описан в работе Шаткай и Келблинга (Shatkay and Kaelbling, 1997), Труна (Thrun et al., 1998b), которые использовали алгоритм *максимизации ожидания* для решения проблем соответствия (см. (Dempster et al. 1977)). Алгоритмы максимизации ожидания последовательно проходят такты ассоциации данных для всех признаков в фазе построения карты, таким, образом, одновременно выполняя поиск в пространстве численных параметров карты и дискретном пространстве соответствий. Аранеда (Araneda, 2003) успешно использовал методы MCMC для ассоциации данных в оффлайн SLAM.

Проблема ассоциации данных естественно возникает в контексте интеграции карт нескольких роботов. В большом количестве разработок были представлены алгоритмы для локализации одного робота относительно другого, допуская, что оба работают в одной среде, а их карты пересекаются (Gutmann and Konolige 2000; Thrun et al. 2000b). Рой и Дудек (Roy and Dudek, 2001) разработали метод, в котором для интеграции информации роботам необходимо сблизиться. В общем случае, однако, метод ассоциации данных должен учитывать возможность того, что карты не пересекаются. Стюарт (Stewart et al., 2003) разработал алгоритм многочастичного фильтра, явно моделирующего возможность непересекающихся карт. В их алгоритме для вычисления вероятности наличия в среде "общих" локальных карт использована байесовская оценочная функция. Идея наложения множеств признаков для ассоциации данных в картографировании с помощью нескольких роботов принадлежит Дедеоглу и Сухаттме (Dedeoglu and Sukhatme, 2000), а также Труну и Лю (Thrun and Liu, 2003). Новое неплоское выражение карт было предложено Говардом (Howard, 2004), и позволило избежать потери целостности в неполных картах. Его метод позволил получить весьма точные результаты картографирования несколькими роботами (Howard et al. 2004).

13.13 Упражнения

1. Назвать три ключевых, явных преимуществ каждого из алгоритмов SLAM: EKF, GraphSLAM и FastSLAM.

2. Описать обстоятельства, при которых FastSLAM 1.0 не будет сходиться, а FastSLAM 2.0 сойдётся к точной карте (с вероятностью 1).

3. На странице 408, было указано, что *зависимости от самого положения x_t вместо всего пути $x_{1:t}$ недостаточно, поскольку зависимости могут возникнуть в предыдущих положениях.* Доказать это утверждение, можно на примере.

4. FastSLAM генерирует множество различных карт, по одной для каждой частицы. Вопрос комбинирования таких карт в единую апостериорную карту остался открытым в этой главе. Предложить два метода, один для FastSLAM с известным соответствием, и один для FastSLAM с ассоциацией данных для каждой частицы.

5. Преимущество FastSLAM 2.0 над FastSLAM 1.0 заключается в природе предполагаемого распределения. Разработать аналогичное распределение для локализации методом Монте-Карло: Вывести похожее предполагаемое распределение для MCL в картах на основе признаков, и включить в результирующий алгоритм "MCL 2.0." Для этого упражнения может понадобиться использовать допущение об известном соответствии.

6. В подразделе 13.8 описана эффективная реализация в виде дерева, но псевдокод не приведён. В этом упражнении требуется описать соответствующие структуры данных и уравнения обновления для дерева, считая, что количество признаков заранее известно, а необходимости обрабатывать проблему соответствия нет.

7. В этом задании требуется эмпирически проверить, что FastSLAM действительно сохраняет корреляции между оценками признаков и оценкой положения робота. Требуется разработать простой алгоритм FastSLAM 1.0 для линейного гауссового SLAM. Можно вспомнить из предыдущих упражнений уравнения движения и измерения в линейном гауссовом SLAM с аддитивным гауссовым шумом:

$$x_t \sim \mathcal{N}(x_{t-1} + u_t, R)$$

$$z_t = \mathcal{N}(m_j - x_t, Q)$$

Запустить FastSLAM 1.0 в симуляторе. После t шагов, обучить гауссову функцию в объединённом пространстве местоположений признаков и положений робота. Вычислить матрицу корреляции на основании этого гауссиана и охарактеризовать степень корреляции как функцию по t. Что показали наблюдения?

8. Как упоминалось в тексте, FastSLAM является многочастичным фильтром Рао-Блеквелла. В этом упражнении требуется разработать фильтр Рао-Блеквелла для другой задачи – локализации робота, которого систематически заносит в сторону. Систематический занос является общим яв-

лением в одометрии – обратите внимание на Рис. 9.1 и 10.7 где показаны довольно явные проявления заноса. Допустим, дана карта среды. Можно ли разработать фильтр Рао-Блеквелла, который одновременно оценивает параметры заноса робота и глобальное местоположение робота в среде? В предлагаемом фильтре следует комбинировать многочастичные фильтры и фильтры Калмана.

Часть IV
Планирование и управление

14 Марковские процессы принятия решений

14.1 Цели

Это первая глава, посвящённая вероятностному *планированию* и *управлению* в книге. До сих пор предметом изложения было исключительно восприятие робота. Был описан широкий диапазон вероятностных алгоритмов для оценки интересующих параметров на основании данных датчиков. Однако, конечной целью работы любого программного обеспечения робота является выбор верных действий. В этой и следующих главах будут обсуждаться вероятностные алгоритмы по выбору действий.

Чтобы определить цели изучения вероятностных алгоритмов планирования, разберём следующие примеры.

1. Манипулятор робота хватает и собирает части, прибывающие случайным образом по ленте конвейера. Конфигурация части в момент появления неизвестна, но стратегия оптимального манипулирования требует знания конфигурации. Как робот должен манипулировать такими частями? Будет ли необходимо восприятие? Если да, все ли методы восприятия одинаково хороши? Могут ли существовать стратегии манипулирования, дающие полностью определённую конфигурацию без восприятия?

2. Подводный аппарат должен пройти от побережья Канады в Каспийское море. Должен ли он следовать кратчайшим маршрутом через Северный полюс, рискуя потерять ориентировку подо льдами? Или же необходимо следовать более длинным маршрутом по открытой воде, где возможна периодическая локализация с помощью GPS, глобальной космической системы позиционирования? До какой степени такие решения зависят от точности внутренних датчиков подводной лодки?

3. Группа роботов исследует неизвестную планету с целью получения единой карты. Должны ли роботы искать друг друга, чтобы определить относительные местоположения? Или же им следует друг друга избегать, что позволит изучить больше новой территории за более короткое время? Как изменится оптимальная стратегия исследования, если относительные стартовые местоположения роботов неизвестны?

Эти примеры иллюстрируют, что выбор действия во многих задачах робототехники тесно связан с понятием неопределённости. В некоторых задачах, таких, как исследование с помощью роботов, уменьшение неопреде-

лённости является прямой целью, определяющей выбор действий.

ЗАДАЧА СБОРА ИНФОРМАЦИИ

Такие задачи известны как *задачи сбора информации* и будут изучаться в Главе 17. В других случаях уменьшение неопределённости имеет целью достижение какой-либо иной цели, например, обеспечения надёжности прибытия в целевое местоположение. Эти задачи будут изучены в этой и следующих главах.

С точки зрения разработки алгоритма, удобно выделить два типа неопределённости: неопределённость, вызванная действием и неопределённость восприятия.

Во-первых, отделим *детерминированные* и стохастические эффекты действий. Многие теоретические результаты робототехники основаны на допущении, что эффекты действий управления детерминированы. Однако, на практике, действия порождают неопределённость, поскольку результат действий, на самом деле, не детерминирован. Неопределённость возникает в силу стохастической природы робота и его окружения, что требует восприятия параметров среды во время работы и возможности реагировать на неожиданные ситуации, даже если среда полностью наблюдаема. Недостаточно лишь запланировать единственную последовательность действий и слепо следовать ей.

Во-вторых, следует различать *полностью наблюдаемые* и *частично наблюдаемые системы*. В классической робототехнике часто считается, что датчики способны измерить полное состояние среды. Если бы это всегда было так, авторы бы не написали эту книгу! Фактически, ситуация строго противоположная. Практически во всех интересных задачах робототехники в реальном мире ограничения датчика являются ключевым фактором.

Очевидно, робот должен принимать во внимание текущую неопределённость, решая, что делать дальше. При выборе управляющего действия, как минимум, робот должен учитывать различные исходы (которые могут включать катастрофические отказы), взвешивая их согласно вероятностям того, что они действительно произойдут.

ОЖИДАЕМАЯ НЕОПРЕДЕЛЁННОСТЬ

Однако, управление роботом должно обрабатывать и будущую, *ожидаемую неопределённость*. Пример приводился выше, при обсуждении робота, который может выбрать более короткий путь в среде без возможности пользоваться GPS, или же более длинный, но с меньшим риском потеряться. Уменьшение ожидаемой неопределённости важно во многих приложениях робототехники.

В ходе изложения будем придерживаться свободной точки зрения без разделения планирование и управления, поскольку, изначально, и планирование и управление преследуют единую цель – выбрать действия.

Они отличаются в ограничениях времени, согласно которым следует выбирать действия, и роли восприятия при выполнении. Все алгоритмы, описанные в этой главе, схожи в том, что требует оффлайнового этапа оптимизации или планирования. Результатом этого этапа планирования является политика управления, предопределяющая действие управления для любой разумной ситуации. Другими словами, политика управления является эффективным контроллером, в том смысле, что ее можно использовать для определения действий робота с минимальным временем вычислений. Ни в коем случае выбор алгоритмов не предполагает, что это единственный способ обрабатывать неопределённость в робототехнике. Однако, он влияет на манеру использования алгоритмов, которые в настоящее время используются в области вероятностной робототехники.

В большей части алгоритмов, обсуждаемых в этой главе, подразумеваются конечные пространства состояний и действий. Непрерывные про-

странства аппроксимируются с помощью методов на основе сеток.

Следующие четыре главы организованы следующим образом.

• В этой главе детально обсуждается роль двух типов неопределённости и закладываются основные шаблоны разработки алгоритмов. В качестве первого решения для ограниченного класса задач будет представлен *итерационный алгоритм*, популярное решение для вероятностных систем. Обсуждение в этой главе затронет только первый тип неопределённости: неопределённость в движении робота. Рассуждения основаны на допущении о том, что состояние полностью наблюдаемо. Лежащий в основе математический аппарат известен под названием *марковские процессы принятия решений* (Markov decision processes - MDP).

• В Главе 15 метод итерационного алгоритма обобщается для обоих типов неопределённости, эффектов движения и восприятия. В этом виде итерационный алгоритм применяется для выражения состояния в виде гипотезы. Математический аппарат в основе метода носит название "марковские процессы принятия решений с частичной наблюдаемостью среды" (partially observable Markov decision processes - POMDP). Алгоритмы POMDP обрабатывают неопределённость, активно собирая информацию, для достижения произвольной цели. В Главе 15 также обсуждаются различные эффективные аппроксимации, позволяющие лучше вычислять политики управления.

• В Главе 16 вводятся некоторые итерационные алгоритмы для POMDP. Во всех решениях выполняется аппроксимация вероятностного процесса планирования для улучшения вычислительной эффективности. Один из таких алгоритмов позволит ускорить вероятностное планирование на основе допущения, что в какой-то момент в будущем состояние станет полностью наблюдаемым. Другой позволяет сжимать оценку состояния в представление с более низкой размерностью и выполняет планирование на основе этого представления. В третьем алгоритме для конденсации пространства задач используются многочастичные фильтры и методы машинного обучения. Все три этих алгоритма возможно существенно улучшить, уменьшив вычислительную сложность, хотя они уже достаточно применимы для практических задач робототехники.

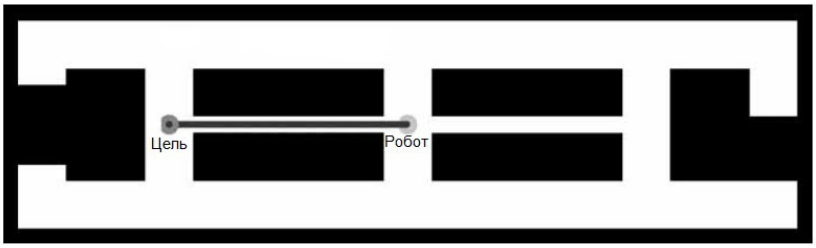


Рис. 14.1 Почти симметричная среда с узкими и широкими коридорами. Робот начинает работу в центре с неизвестной ориентацией по направлению. Его целью является передвижение в пункт назначения слева.

• Глава 17 посвящена специализированной проблеме исследования с помощью роботов. Здесь задачей робота является сбор информации об окружающей среде. Хотя методы исследования касаются проблемы неопреде-

лённости датчика, задача существенно проще по сравнению с POMDP, а, значит, может быть решена более эффективно. Вероятностные методы исследования популярны в робототехнике, поскольку роботов часто используют для получения информации о неизвестных местах.

14.2 Неопределённость в выборе действия

На Рис. 14.1 показана упрощённая среда для иллюстрации различных типов неопределённости, с которой может столкнуться робот. На рисунке показан мобильный робот в среде, состоящей из коридоров. Среда обладает высокой степенью симметричности, и единственным различимым признаком является дальний конец коридора, имеющий разную форму в зависимости от места на карте. Робот начинает в указанном месте и ищет способ достичь целевого местоположения. Заметим, что до цели существует несколько путей, более короткий, но узкий, и два более длинных, но имеющих большую ширину.

В классической парадигме планирования с помощью роботов неопределённости нет. Робот просто знает своё начальное положение и местоположение цели. Более того, действия, выполненные в физическом мире, имеют прогнозируемые последствия, которые могут быть предварительно запланированы. В такой ситуации нет необходимости в восприятии. Достаточно запланировать единственную последовательность действий, которая может выполняться в реальном времени. На Рис. 14.1 показан пример такого планирования. Очевидно, при отсутствии ошибок в движении робота, узкий путь предпочтительнее любого другого, более длинного пути. Поэтому «классический» планировщик всегда выберет первый, короткий путь.

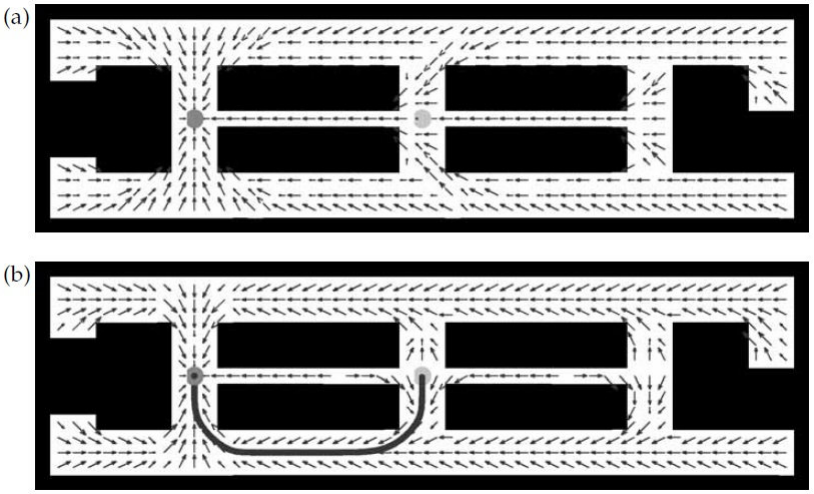


Рис. 14.2 Функция подкрепления и политика управления в MDP с детерминированными (a) и недетерминированными эффектами действия (b). В детерминированной модели робот совершенно свободно выполняет навигацию по узкому пути. Длинный путь оказывается предпочтительнее, если результат действия не гарантирован, для предотвращения столкновения со стеной. На Рис. (b) показан такой путь.

На практике такие планы не работают в силу нескольких причин. Робот,

слепо следующий по узкому проходу, подвержен опасности столкновения со стеной. Более того, слепое выполнение команд может привести к промаху мимо целевого местоположения из-за ошибок при выполнении плана. Поэтому, на практике такие алгоритмы планирования часто комбинируются с реактивным модулем управления на основе датчиков, который подстраивает траекторию робота так, чтобы избежать столкновений. Такой модуль может предотвратить столкновение робота со стеной в узком коридоре, но для этого ему может понадобиться замедлить движение робота, что сделает длинный путь более предпочтительным.

МАРКОВСКИЙ ПРОЦЕСС ПРИНЯТИЯ РЕШЕНИЙ

Парадигма с учётом неопределённости движений робота известна как *марковские процессы принятия решений* (Markov decision processes – MDP). В MDP предполагается, что состояние окружающей среды может полностью наблюдаться в любой момент времени. Другими словами, модель восприятия $p(z|x)$ детерминирована и взаимно однозначна. Однако, аппарат MDP позволяет учитывать и стохастические эффекты действия. Так, модель действия $p(x'u, x)$ может быть не детерминирована, а, следовательно, недостаточно просто запланировать одну последовательность действий. Вместо этого, планировщик должен генерировать действия для всего диапазона ситуаций, с которыми может столкнуться робот, в силу своих действий или непредсказуемой динамики окружающей среды. Одним из способов обработки результирующей неопределённости является генерация *политики выбора действия*, определённая для всех состояний, в которых робот может оказаться.

ПОЛИТИКА УПРАВЛЕНИЯ

Такая проекция состояний в действия известна под многими названиями, например, *политика управления*, *универсальные планы*, и *навигационные функции*. Пример политики показан на Рис. 14.2. Вместо единственной последовательности действий робот вычисляет проекции состояний в действия, показанные стрелками. После вычисления проекции робот способен обрабатывать отсутствие детерминированности восприятием состояния среды, и соответствующим образом реагировать. На Рис. 14.2a показана политика для робота с очень малой неопределённостью движения, когда проход по более узкому пути, действительно, допустим. На Рис. 14.2b показана та же ситуация, но для увеличившейся случайности в движениях робота. Здесь проход по узкому пути гораздо вероятнее приведёт к столкновению, и предпочтительным будет двигаться в обход. Этот пример иллюстрирует два факта: важность учёта неопределённости в процессе планирования движения, и возможность нахождения хорошей политики управления с помощью описанных алгоритмов.

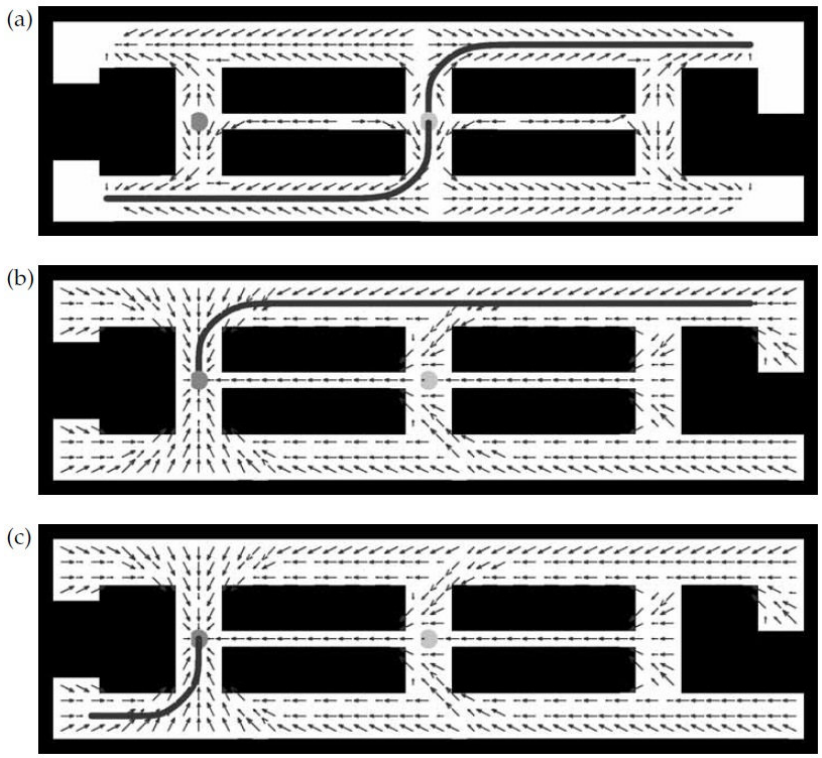


Рис. 14.3 Сбор информации в POMDP: Для достижения цели с более чем 50% вероятностью, планировщик в пространстве оценок сначала выполняет навигацию до местоположения, глобальную ориентацию которого можно определить. На Рис. (a) показана политика соответствия и возможный путь следования робота. На основании текущего местоположения робот может обнаружить, что находится в точке (b) или (c), из которой можно безопасно передвигаться к цели.

МАРКОВСКИЕ ПРОЦЕССЫ ПРИНЯТИЯ РЕШЕНИЙ С ЧАСТИЧНОЙ НАБЛЮДАЕМОСТЬЮ СРЕДЫ

Вернёмся к более общему, полностью вероятностному случаю, отбросив допущение о полной наблюдаемости состояния. Этот случай известен как *марковские процессы принятия решений с частичной наблюдаемостью среды* (partially observable Markov decision processes – POMDP). Во многих, если не во всех, робототехнических реализациях, измерения z являются зашумлёнными проекциями состояния x. В силу этого, состояние можно оценить лишь до некоторой степени. Проиллюстрируем это с помощью уже приведённого примера, но с другими допущениями. Допустим, робот осведомлен о своей начальной позиции, но не знает, ориентирован ли по направлению налево, или направо. Более того, датчика, способного определить прибытие к цели, у него нет.

Конечно, симметрия среды сильно затрудняет определение верной ориентации. Двигаясь напрямую к проекции цели впереди, робот имеет 50% шанс промахнуться мимо цели, придя в симметричное местоположение в правой части среды. Поэтому, оптимальный план – двигаться к любому из углов среды, где признаков уже будет достаточно для определения ориентации. Политика передвижения в эти местоположения показана на Рис. 14.3a. Основываясь на начальной ориентации по направлению, робот может передвинуться по любому из двух показанных путей. При достижении угла его датчики определяют ориентацию по направлению, получая текущее место-

положение относительно среды. Робот может обнаружить, что находится в одной из двух ситуаций, показанных на Рис 14.3b и c, откуда он может безопасно переместиться к цели.

На этом примере показан один из ключевых аспектов вероятностной робототехники. Роботу необходимо активно собирать информацию, и, чтобы это сделать, приходится идти в обход, что не является важным для робота, воспринимающего своё положение с абсолютной точностью. Это проблема очень важна в робототехнике. Почти для всех задач робототехники датчики робота характеризуются изначальными ограничениями того, что робот способен узнать, и где может быть получена информация. Похожие ситуации встречаются в задачах вида «обнаружить и вернуть», планетарных исследованиях, поиске и спасении в городе и так далее.

Возникает вопрос, каким же образом вывести алгоритм для выбора действий, способный работать с таким типом неопределённости. Как мы вскоре узнаем, вопрос далеко не тривиальный. На первый взгляд, кажется продуктивным проанализировать каждую ситуацию, возможную при текущем уровне осведомлённости. В нашем примере таких ситуаций две. В первом случае цель находится сверху слева относительно начального положения робота, а во втором – снизу справа. В обоих случаях, однако, оптимальной политикой является не перемещение агента к цели, а увод его к точке, где он будет способен разрешить двойственность восприятия своего положения. Поэтому, задача планирования в частично наблюдаемой среде не может быть решена с помощью учёта всех возможных сред и усреднения решения.

Вместо этого, базовой идеей будет генерация планов в *пространстве гипотез*, (синоним термина *информационное пространство*). Пространство гипотез состоит из пространства всех апостериорных гипотез о состоянии окружающего мира, которые могут быть у робота. Пространство гипотез в нашем простом примере соответствует трём врезкам на Рис. 14.3. На верхней врезке показан пример гипотезы в пространстве гипотез. На ней показана начальная политика, когда робот не осведомлен о своём положении в пространстве. Согласно этой политике, робот перемещается к одному из углов среды, где он в состоянии выполнить локализацию. После локализации он может безопасно следовать далее к цели, как показано на двух нижних врезках Рис. 14.3. Поскольку априорный шанс каждой ориентации в пространстве одинаковый, робот будет перемещаться, имея 50% шансы оказаться в одном из двух положений, показанных ниже.

В нашем упрощённом примере, количество разных оценок состояний конечно: робот точно знает своё местоположение или же не имеет о нем ни малейшего понятия. В практических реализациях это часто не так. В средах с конечным множеством состояний пространство гипотез обычно непрерывно, но имеет конечную размерность. Фактически, количество измерений пространства гипотез имеет тот же порядок, что и количество состояний в подлежащем пространстве состояний. Если пространство состояний непрерывно, пространство гипотез имеет бесконечно много измерений.

Этот пример иллюстрирует фундаментальное свойство, возникающее из невозможности идеального восприятия состояния среды роботом, важность которого в робототехнике так часто недооценивают. В неопределённых средах алгоритм планирования робота должен учитывать состояние осведомлённости в принятии решений управления. В общем случае, недостаточно учитывать только самое вероятное состояние. Устанавливая зависимость движения от гипотезы состояния, в отличие от наиболее вероятного состояния, робот может активно выполнять сбор информации. Фактически, оп-

тимальный план для гипотезы состояния предусматривает «оптимальный» сбор информации, поскольку ищет только новую информацию и лишь в той степени, которая будет полезна для возможного использования в действиях робота. Возможность получения оптимальных политик управления является ключевым преимуществом вероятностного подхода в робототехнике по сравнению с классическим детерминированным, "всеведающим" подходом. Однако, как скоро будет показано, за него приходится платить цену в виде возросшей сложности задачи планирования.

14.3 Итерационный алгоритм

Первый алгоритм для нахождения политик управления называется *итерационный алгоритм*. итерационный алгоритм рекурсивно выполняет вычисление полезности каждого действия относительно функции подкрепления. Обсуждение в этой главе будет ограничено первым типом неопределённости – стохастичностью робота и физического мира. Отложим до следующей главы рассмотрение неопределённости, возникающее из ограничений датчика. Пока что будем считать, что состояние среды полностью наблюдаемо в произвольный момент времени.

14.3.1 Цели и подкрепление

ЦЕЛЬ

Перед описанием конкретного алгоритма, определим задачу в более точных терминах. В общем случае, выбор действий робота обусловлен *целями*. Цели могут соответствовать определенным конфигурациям (например, часть была успешно подобрана и размещена манипулятором робота), или выражены условиями, которые необходимо сохранить в течение длительного времени (робот сохраняет баланс шеста).

СТОИМОСТЬ

В робототехнике иногда необходимо достичь некой целевой конфигурации, одновременно оптимизируя другие переменные, часто воспринимаемые как *стоимость*. Например, целью может быть перемещение конечного эффектора манипулятора в конкретное местоположение, одновременно минимизируя время, потребление энергии или количество столкновений с препятствиями.

На первый взгляд, может показаться затруднительным выражение этих условий двумя параметрами, один из которых максимизируется (например, бинарный флаг, показывающий, достиг ли робот целевого местоположения), а вторая – сводится к минимуму (например, общее энергопотребление робота).

ФУНКЦИЯ ПОДКРЕПЛЕНИЯ

Однако, обе можно выразить с помощью одной функции, которая называется *функция подкрепления*.

Доход, обозначаемый r, является функцией состояния и управления роботом. Например, простая функция подкрепления может иметь следующий вид:

(14.1)

$$r(x,u) = \begin{cases} +100 & \text{если } u \text{ ведет к целевой конфигурации или состоянию} \\ -1 & \text{во всех остальных случаях} \end{cases}$$

Эта функция «награждает» робота +100 при достижении целевой конфигурации, и «штрафует» робота -1 на каждый такт времени, когда он не

достиг искомой конфигурации. Такая функция подкрепления даст максимальный суммарный доход, если робот достигает целевой конфигурации за минимально возможное время.

Зачем использовать одну переменную награды для выражения достижения цели и стоимости? Главным образом, причин две. Во-первых, такая нотация позволяет избежать беспорядка в следующих формулах, поскольку должна объединять обработку стоимости и достижения цели в целях изложения в книге. Во-вторых, что более важно, в ней отдаётся должное фундаментальному компромиссу между достижением цели и стоимости прохода по пути. Поскольку роботы изначально неопределены, нет возможности определить была ли достигнута целевая конфигурация. Вместо этого, все, на что можно надеяться, это максимизация шансов достижения цели. Этот компромисс между достижением цели и стоимостью характеризуется такими вопросами, как *стоит ли увеличение вероятности достижения цели дополнительных усилий (например, в смысле времени или энергии)?* Считая достижение цели и стоимость одним числовым множителем позволяет разменивать и настраивать переменные, давая целостный способ выбора действий в условиях неопределённости.

Наша задача – вывести программу генерации действий таким образом, чтобы оптимизировать будущие ожидаемые затраты.

ПОЛИТИКА

Такую программу обычно называют *политикой управления*, или, проще, *политикой*. Обозначим ее следующим образом:

(14.2)
$$\pi : z_{1:t-1}, u_{1:t-1} \longrightarrow u_t$$

В случае полной наблюдаемости среды, допустим намного более простой случай:

(14.3)
$$\pi : x_r \longrightarrow u_t$$

Таким образом, π является функцией, которая проектирует прошлые данные в управляющие воздействия или же состояния в управляющие воздействия, когда состояние наблюдаемо.

Пока что, в нашем определении политики управления не было сделано никаких утверждений относительно вычислительных свойств. Это должен быть быстрый реактивный алгоритм, полагающийся на решения на основе самого последнего элемента, или же тщательно подобранный алгоритм планирования. На практике, однако, вычислительные соображения важны, поскольку любая задержка вычисления управляющего воздействия может негативно повлиять на работу робота. Определение политики π также не даёт никакого утверждение относительно ее детерминированности.

ГОРИЗОНТ ПЛАНИРОВАНИЯ

Интересной концепцией в контексте создания политик управления является *горизонт планирования*. Иногда достаточно выбрать управляющее действие таким образом, чтобы максимизировать следующее значение награды. В большинстве случаев действие не приведёт к немедленной награде. Например, робот, который перемещается к целевому местоположению, получит финальную награду при достижении цели только после самого последнего действия. Очевидно, что награда может быть отсрочена.

СУММАРНЫЙ ДОХОД

Тогда подходящей целью будет выбор таких действий, которые бы привели к максимальной сумме итогового дохода. Назовём эту сумму *суммарным доходом*. Поскольку мир не детерминирован, лучше всего оптимизиро-

вать *ожидаемый суммарный доход*, которую удобно записать в виде

(14.4)
$$R_T = E\left[\sum_{r=1}^{T} \gamma^T r_{t+\tau}\right]$$

Здесь ожидание $E[]$ берётся по моментальным значениям будущего дохода $r_{t+\tau}$ который робот может получить между моментами времени $t+T$.

КОЭФФИЦИЕНТ ПЕРЕОЦЕНКИ

Отдельные значения дохода $r_{t+\tau}$ умножаются на множитель γ^T, называемым *коэффициент переоценки*. Значение γ является параметром, различающимся в зависимости от задачи, и ограничивается интервалом $[0;1]$. Если $\gamma = 1$, получим $\gamma^T = 1$ для произвольных значений τ, а, значит, в выражении (14.4) можно опустить множитель. Меньшие значения γ экспоненциально уменьшают будущий доход, делая раннее получение награды экспоненциально более важным. Этот коэффициент переоценки, важность которого будет обсуждаться позже, напоминает концепцию денег, которые экспоненциально по времени теряют стоимость из-за инфляции.

ГОРИЗОНТ ПЛАНИРОВАНИЯ

Заметим, что R_T является суммой по T тактам времени. T называется *горизонтом планирования*, или, проще, *горизонтом*. Будем различать три важных случая:

ЖАДНЫЙ СЛУЧАЙ

1. $T = 1$. Ситуация, когда робот стремится минимизировать следующий штраф, называется *жадным случаем*. Хотя случай вырожденный и эффект действий, кроме как на следующем такте времени, не рассматривается, на практике он довольно важен, поскольку жадная оптимизация проста по сравнению с оптимизацией для нескольких тактов. Во многих задачах робототехники жадные алгоритмы являются лучшими на сегодняшний день решениями, которые могут быть вычислены за полиномиальное время. Очевидно, жадная оптимизация инвариантна по отношению к коэффициенту скидки γ, но требует, чтобы $\gamma > 0$.

СЛУЧАЙ КОНЕЧНОГО ГОРИЗОНТА

2. T больше 1, но конечно. Этот случай известен как *случай конечного горизонта*. Обычно, доход не падает со временем, поэтому $\gamma = 1$. Можно аргументировать, что случай конечного горизонта единственный, который имеет значение, поскольку во всех практических задачах время конечно. Однако, оптимальность конечного горизонта часто труднее достичь, чем оптимальность в случае бесконечного горизонта с переоценкой. Почему это так? Во-первых, заметим, что оптимальное управляющее действие является функцией временного горизонта. Например, около дальнего конца временного горизонта оптимальная политика может существенно отличаться от оптимального выбора в более ранний момент времени, даже при прочих одинаковых условиях (то же состояние и тот же прогноз). В результате, алгоритмы планирования с конечным горизонтом должны сохранять разные пути для разных горизонтов, что добавляет нежелательную сложность.

СЛУЧАЙ С БЕСКОНЕЧНЫМ ГОРИЗОНТОМ

3. T бесконечно. Этот случай известен как *случай с бесконечным горизонтом*. Он не подвержен проблемам, применимым для случая с конечным горизонтом, поскольку количество оставшихся шагов времени в любой момент одинаково (и бесконечно!). Однако, присутствие коэффициента переоценки γ крайне важно. Чтобы понять почему, давайте разберём случай, когда имеются две программы управления роботом, одна, зарабатывающая \$1 в час, и вторая, зарабатывающая \$100 в час. В случае конечного горизонта, вторая программа, очевидно, предпочтительнее первой. Неважно, каково значение горизонта, ожидаемая накопленная награда превышает награду первой программы в сто раз. Для случая бесконечного горизонта это не так. Без переоценки обе программы заработают бесконечно много денег,

что делает ожидаемый суммарный доход R_T несущественным при выборе программы.

При допущении, что каждая отдельная награда r ограничена по размеру (так, чтобы, $|r| < r_{\max}$ для некоторого $r_{\max}$), переоценка гарантирует, что R_∞ конечно, несмотря на факт того, что сумма состоит из бесконечного множества слагаемых. А именно, получим

(14.5)

$$R_\infty \leq r_{\max} + \gamma r_{\max} + \gamma^2 r_{\max} + \gamma^3 r_{\max} + ... = \frac{r_{\max}}{1-\gamma}$$

Это показывает, что R_∞ конечно до тех пор, пока γ меньше 1. Попутно заметим, что альтернатива переоценке включает максимизацию среднего дохода, вместо полного. Алгоритмы максимизации среднего дохода не будут рассматриваться в книге.

Иногда будем обращаться к суммарному доходу R_T, который зависит от состояния x_t. Это можно записать в следующем виде:

(14.6)

$$R_T(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^T r_{t+\tau} | x_t\right]$$

суммарный доход R_T является функцией политики выбора действий робота. Иногда выгодно сделать эту зависимость явной:

(14.7)

$$R_T^\pi(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^T r_{t+\tau} | u_{t+\tau} = \pi(z_{1:t+\tau-1}, u_{1:t+\tau-1})\right]$$

Такая запись позволяет сравнить две политики управления π и π', чтобы определить, какая лучше. Просто сравним R_T^π и $R_T^{\pi'}$ и выберем алгоритм с более высоким ожидаемым будущим доходом с учётом переоценки!

14.3.2 Нахождение оптимальных политик управления для случая полной наблюдаемости

Эту главу необходимо завершить алгоритмом итерации значений для вычисления политик управления в полностью наблюдаемых средах. На первый взгляд, такие алгоритмы отходят от общих допущений вероятностной робототехники, в частности о том, что состояние ненаблюдаемо. Однако, в некоторых реализациях вполне допустимо считать, что апостериорная вероятность $p(x_t|z_{1:t}, u_{1:t})$ хорошо выражается математическим ожиданием $E[p(x_t|z_{1:t}, u_{1:t})]$.

Случай полной наблюдаемости имеет ряд достоинств. Обсуждаемый алгоритм также подготовит почву для более общего случая частичной наблюдаемости.

Также заметим, что способы отображения стохастических сред с полностью наблюдаемым состоянием известны как марковские процессы принятия решений. Политики в MDP являются проекциями состояния на управляющие действия:

(14.8)

$$\pi : x \longrightarrow u$$

Факт того, что состояния достаточно для определения оптимального управления является прямым следствием марковского свойства, которое подробно обсуждалось в подразделе 2.4.4. Целью планирования в терминах MDP является обнаружение такой политики π, которая максимизирует будущий суммарный доход.

Начнём с определения оптимальной политики для горизонта планирования $T = 1$, поскольку нас интересует только политика, максимизирующая следующую награду. Эта политика будет обозначаться $\pi_1(x)$ и получаться путём максимизации ожидаемого дохода за один шаг по всем управляющим действиям:

(14.9)
$$\pi_1(x) = \underset{u}{\operatorname{argmax}}\, r(x, u)$$

Оптимальным действием будет то, которое то, которое обеспечит максимальный ожидаемый доход, получаемый немедленно. Политика, выбирающая такое действие, оптимальна по ожиданию.

ФУНКЦИЯ ДОХОДА

Для каждой политики есть связанная функция доходов, измеряющая ожидаемое значение (суммарный будущий доход со с переоценкой). Для π_1 функция доходов - это просто ожидаемая немедленная награда, уменьшенная коэффициентом переоценки γ:

(14.10)
$$V_1(x) = \gamma \max_u r(x, u)$$

Это значение для более дальних горизонтов планирования теперь определяется рекурсивно. Оптимальная политика для горизонта $T = 2$ выберет управляющее действие, которое максимизирует сумму дохода на одном шаге $V_1(x)$ и немедленной награды на одном шаге:

(14.11)
$$\pi_2(x) = \underset{u}{\operatorname{argmax}} \left[r(x, u) + \int V_1(x')\, p(x'|u, x) dx' \right]$$

Должно быть ясно, почему эта политика оптимальна. Значение этой политики зависит от состояния x, заданного следующим выражением с учётом переоценки:

(14.12)
$$V_2(x) = \gamma \max_u \left[r(x, u) + \int V_1(x')\, p(x'|u, x) dx' \right]$$

Оптимальная политика и ее функция дохода для $T = 2$ была рекурсивно построена из оптимальной функции дохода для $T = 1$. Это наблюдение подразумевает, что для любого конечного горизонта T оптимальная политика и связанная с ней функция дохода, может быть получена рекурсивно из оптимальной политики и функции дохода для $T - 1$.

(14.13)
$$\pi_T(x) = \underset{u}{\operatorname{argmax}} \left[r(x, u) + \int V_{T-1}(x')\, p(x'|u, x) dx' \right]$$

Результирующая политика $\pi_T(x)$ оптимальна для горизонта планирования T. Связанная функция дохода определена через следующую рекурсию:

(14.14)

$$V_T(x) = \gamma \max_u \left[r(x,u) + \int V_{T-1}(x')\, p(x'|u,x) dx' \right]$$

В случае бесконечного горизонта, оптимальная функция дохода стремится к достижению равновесия (за исключением редких детерминированных систем, где такого равновесия не существует):

(14.15)

$$V_\infty(x) = \gamma \max_u \left[r(x,u) + \int V_\infty(x')\, p(x'|u,x) dx' \right]$$

УРАВНЕНИЕ БЕЛЛМАНА

Эта инвариативность известна как *уравнение Беллмана*. Без приведения доказательства заметим, что для каждой функции дохода V, условие (14.15) необходимо и достаточно, чтобы получившаяся политика была *оптимальной*.

14.3.3 Вычисление функции дохода

Указанное соображение позволяет получить практический алгоритм вычисления оптимальной политики в стохастических системах с полностью наблюдаемым состоянием. Итерационный алгоритм выполняется путём последовательной аппроксимации функций оптимального дохода, как определено в (14.15).

Более детально, обозначим аппроксимацию функции дохода через $\hat{V}$. Изначально, $\hat{V}$ установлено в $r_{\min}$, минимально возможный немедленный доход:

(14.16)

$$\hat{V} \longleftarrow r_{\min}$$

Итерационный алгоритм затем последовательно обновляет аппроксимацию на основании следующего рекурсивного правила вычисления функции дохода для возрастающих горизонтов:

(14.17)

$$\hat{V}(x) \longleftarrow \gamma \max_u \left[r(x,u) + \int \hat{V}_(x')\, p(x'|u,x) dx' \right]$$

Поскольку при каждом обновлении информация распространяется в обратном временном порядке через функцию ценности, это обычно называется *обратный метод прогонки*.

ОБРАТНЫЙ МЕТОД ПРОГОНКИ

Итерационный алгоритм имеет сильное сходство с приведённым выше вычислением оптимальной политики для T-горизонта. Итерационный алгоритм сходится при $\gamma < 1$, а в некоторых особых случаях, даже при $\gamma = 1$. Порядок обновления состояний неважен до тех пор, пока каждое состояние обновляется бесконечно часто. На практике, сходимость обычно наблюдается после небольшого числа итераций.

В произвольный момент времени функция дохода $\hat{V}(x)$ определяет политику:

(14.18)

$$\pi(x) = \operatorname*{argmax}_u \left[r(x,u) + \int \hat{V}(x')\, p(x'|u,x) dx' \right]$$

После сходимости итерации ценности, жадная политика по отношению к конечной функции дохода является оптимальной.

Заметим, что все эти уравнения были сформулированы для общих пространств состояний. Для конечных пространств состояний, интеграл в каждом уравнении можно представить в виде конечной суммы по всем состояниям. Эту сумму часто можно эффективно вычислить, поскольку $p(x'|u,x)$ будет ненулевым для относительно небольшого количества состояний x и x'. В результате получается эффективное семейство алгоритмов вычисления функций дохода.

В Таблице 14.1 показаны три алгоритма: общий итерационный алгоритм **MDP_value_iteration** для произвольных пространств состояний и действий, его дискретный вариант для конечных пространств состояний **MDP_discrete_value_iteration**, и алгоритм получения оптимального управляющего действия из функции дохода, **policy_MDP**.

Algorithm MDP_value_iteration() :
для всех x *выполнять*
  $\hat{V}(x) = r_{\min}$
endfor
повторять до сходимости
  для всех x
    $\hat{V}(x) = \gamma \max_u \left[r(x,u) + \int \hat{V}(x')\, p(x'|u,x) dx' \right]$
  endfor
endrepeat
return $\hat{V}$

Algorithm MDP_discrete_value_iteration() :
для $i = 1$ *до* N *выполнять*
  $\hat{V}(x_t) = r_{\min}$
endfor
повторять до сходимости
  для $i = 1$ *до* N *выполнять*
    $\hat{V}(x_t) = \gamma \max_u \left[r(x_t,u) + \sum_{j=1}^{N} \hat{V}(x_j)\, p(x_j|u,x_i) \right]$
  endfor
endrepeat
return $\hat{V}$

1: **Algorithm policy_MDP**$(x, \hat{V})$:
2: *return* $\underset{u}{\text{argmax}} \left[r(x,u) + \sum_{j=1}^{N} \hat{V}(x_j)\, p(x_j|u, x_i) \right]$

Таблица 14.1 Итерационный алгоритм для MDP с конечными пространствами состояний и управляющих действий, показан в самом общем виде. Нижний алгоритм вычисляет наилучшее действие управления.

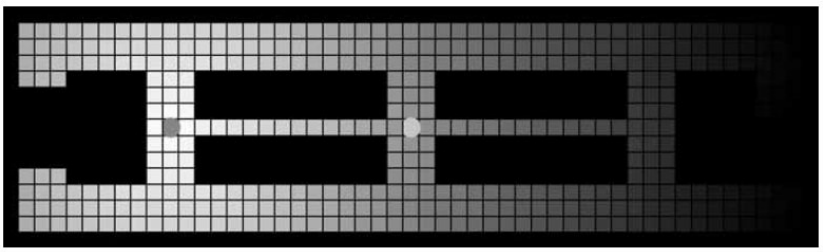


Рис. 14.4 Пример функции дохода с бесконечным горизонтом T_∞, подразумевая, что целевым состоянием является «поглощающее состояние». Эта функция дохода порождает политику, показанную на Рис. 14.2a.

Первый алгоритм **MDP_value_iteration** инициализирует функцию дохода в строке 3. В строках с 5 по 9 выполняются рекурсивные вычисления функции дохода. После сходимости итерационного алгоритма, результирующая функция дохода $\hat{V}$ порождает оптимальную политику. Если пространство состояний конечно, интеграл заменяется коечной суммой, как показано в **MDP_discrete_value_iteration**. Множитель γ является коэффициентом переоценки. Алгоритм **policy_MDP** обрабатывает функцию оптимального дохода по состоянию x, и возвращает управляющее воздействие u, которая максимизирует ожидаемый доход. На Рис. 14.4 показан пример функции дохода для обсуждаемого выше примера. Здесь заливка ячеек сети соответствует их значению, значение, отмеченное белым $V = 100$, чёрным $V = 0$. Функция поиска экстремума в функции дохода основана на выражении 14.18, и ведёт к политике, показанной на Рис. 14.2a.

14.4 Применение к управлению роботом

Простой итерационный алгоритм применим для задач с низкой размерностью управления и планирования движения робота. Для решения введём две аппроксимации. Во-первых, алгоритм в Таблице 14.5 определяет функцию дохода и требует максимизации и интеграции в непрерывном пространстве. На практике, принято аппроксимировать пространство состояний дискретным разложением, близким к гистограммированию, описанному в Главе 4.1. Похожим образом дискретизируется и пространство управляющих воздействий. Функция $\hat{V}$ легко реализуется в виде таблицы поиска, но такая декомпозиция работает только для пространств состояния и управления с низкой размерностью, в силу подверженности "проклятью размерности". В

ситуациях с большей размерностью для представления функции ценности вводятся обучающиеся алгоритмы.

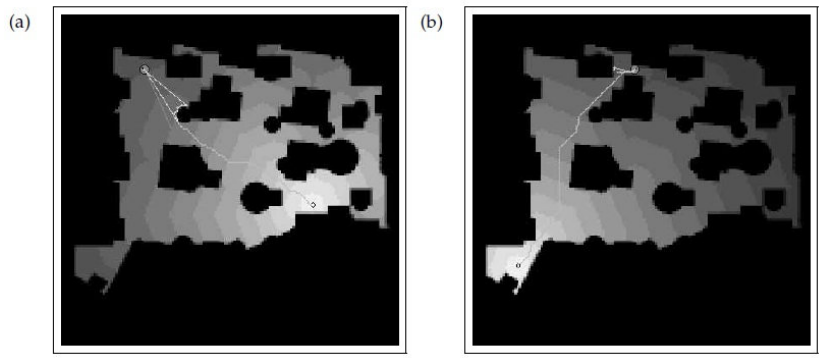


Рис. 14.5 Пример итерационного алгоритма в пространстве состояний при движении робота. Препятствия показаны чёрным. Функция дохода обозначена серой заливкой. Жадный выбор действия по отношению к функции дохода позволяет найти оптимальный сигнал управления, считая, что положение робота наблюдаемо. Также на схемах показаны примерные пути, полученные в результате следования жадной политике.

Во-вторых, необходимо знать состояние! Как уже обсуждалось выше, может оказаться выигрышной замена апостериорного распределения его модой

(14.19)
$$\hat{x}_t = E[p(x_t|z_{1:t}, u_{1:t})]$$

В контексте локализации робота такая аппроксимация работает хорошо, если мы сможем гарантировать, что робот всегда приблизительно локализован, а оставшаяся неопределённость апостериорного распределения локальна. Такой подход не работает, когда робот выполняет глобальную локализацию, или же был "похищен".

На Рис. 14.5 показан итерационный алгоритм в контексте задачи планирования пути робота. На схеме приведена двухмерная проекция пространства конфигурации круглого робота. Пространство конфигураций – это пространство всех координат (x, y, θ), в которых робот может физически оказаться. Для круглых роботов пространство конфигурации получается "выращиванием" препятствий на карте, основываясь на величине радиуса робота. Такие "возвышающиеся" препятствия показаны чёрным на Рис. 14.5.

Функция дохода показана градациями серого цвета, и чем светлее местоположение, тем выше её значение. Путь получен следованием оптимальной политике до соответствующего местоположения цели, как показано на Рис. 14.5. Из рисунка видно, что функция дохода определена по всему пространству состояний, и это позволяет роботу определять требуемое действие вне зависимости от своего положения. Это важно для недетерминированных сред, где действия стохастически влияют на состояние робота.

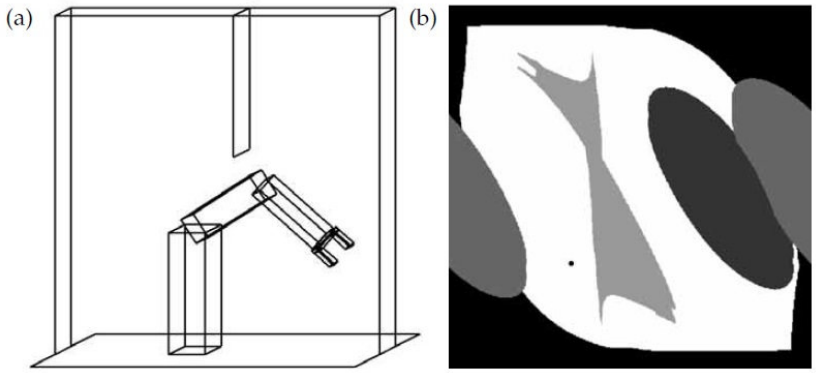


Рис. 14.6 (a) 2 DOF манипулятор робота в пространстве с препятствиями. (b) *Пространство конфигураций* этого манипулятора: горизонтальная ось соответствует плечевому сочленению, а вертикальная – конфигурации локтевого сочленения. Препятствия показаны серым. Маленькая точка на схеме соответствует конфигурации, показанной слева

В планировщике пути, с помощью которого был сгенерирован Рис. 14.5, приняты особые допущения для поддержания приемлемой вычислительной нагрузки. Для круглых роботов, способных повернуться на одном месте, часто используется вычисление функции ценности только в плоских евклидовых координатах, игнорируя затраты на вращение. Также игнорируются переменные состояния, такие, как скорость робота, несмотря на то, что величина скорости очевидным образом ограничивает места, куда робот может переместиться в заданный момент времени. Для превращения политики управления в реальные команды общепринятой практикой является комбинирование таких планировщиков пути с быстрыми, реактивными модулями предотвращения столкновений, способными генерировать требуемые значения скорости вращения двигателей при соблюдении динамических ограничений. Планировщик пути, учитывающий полное состояние робота, может действовать, минимум, в пяти измерениях, состоящих из полного положения (три измерения), поступательной и вращательной скоростей робота. В двух измерениях вычисление функции ценности для среды наподобие показанной выше, занимает лишь долю секунды на маломощном компьютере.

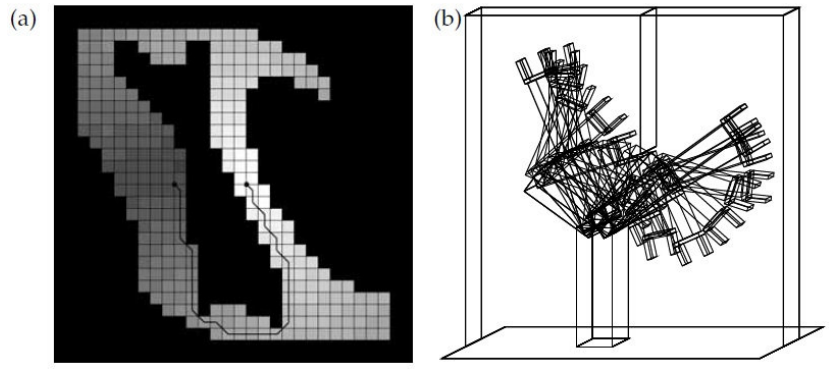


Рис. 14.7 Итерационный алгоритм, используемая для грубой дискретизации пространства конфигураций(a). Путь в локальных координатах рабочего пространства (b). Робот действительно избегает вертикального препятствия.

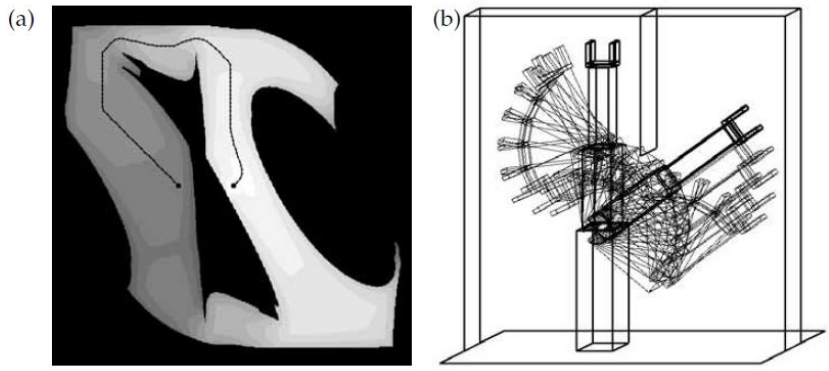


Рис. 14.8 Вероятностный итерационный алгоритм, показан на мелкой сетке(a). Соответствующий путь (b).

Второй пример приведён на Рис. 14.6a, на котором показана модель манипулятора робота с двумя степенями свободы вращения, в плечевом и локтевом сочленениях. Обычно возможно точно определить конфигурацию этих сочленений с помощью энкодеров на валах. Поэтому, аппроксимация в (14.19) является применимой. Однако, движение манипулятора робота часто подвержено зашумлению, и эти шумы управления необходимо учитывать при планировании движения. Это делает управление манипулятором основной задачей для вероятностных алгоритмов MDP.

ПРОСТРАНСТВО КОНФИГУРАЦИИ

Движение манипулятора робота обычно описывается в *пространстве конфигураций*. Пространство конфигураций для манипулятора показано на Рис. 14.6b. Здесь по горизонтальной оси показано вращательное положение плечевого сочленения, а по вертикали – ориентация в пространстве локтевого сочленения. Таким образом, каждая точка на диаграмме соответствует конкретной конфигурации. Маленькая точка на Рис. 14.6b соответствует конфигурации, показанной на Рис. 14.6a.

Принято разбивать пространство конфигураций на зоны, в которых робот может двигаться, и зоны, где может произойти столкновение. Это показано на Рис. 14.6b. Белая область на рисунке соответствует конфигурацион-

ному пространству, где столкновений не происходит, обычно называемому свободным пространством. Чёрная граница конфигурационного пространства является ограничением, накладываемым поверхностью стола и защитной клеткой. Вертикальное препятствие, проходящее в рабочее пространство робота сверху на Рис. 14.6a, соответствует светло-серому препятствию в центре Рис. 14.6b. Этот рисунок не так очевиден, и читатель может понадобиться некоторое время на визуализацию условий, при которых робот сталкивается с препятствием.

На Рис. 14.7a показан результат итерационного алгоритма на основе грубой дискретизации конфигурационного пространства. Здесь доход распространяются с помощью детерминированной модели движения, а также показан результирующий путь. Выполнение политики даёт движение, показанное на Рис. 14.7b. На Рис. 14.8 показан результат работы вероятностной модели движения, а также итоговое движение манипулятора. И снова, была применён итерационный алгоритм с допущением, что конфигурация манипулятора робота полностью наблюдаема – редкий случай, когда это действительно так!

14.5 Выводы

В этой главе представлен основной понятийный аппарат вероятностного управления.

• Было определено два основных типа неопределённости, с которым может столкнуться робот: неопределённость по отношению к управлению, и неопределённость в восприятии. Первое затрудняет прогнозирование будущих событий, а второе - текущей ситуации. Неопределённость от непредсказуемых событий среды также, по умолчанию, укладывается в эту классификацию.

• Цель управления была определена через *функцию дохода*, проектирующая сигналы управления и состояния в виде некоторого множества значений "желательности". Понятие "вознаграждения" позволяет выразить цели деятельности робота, а также "стоимость" такой деятельности. Общей целью управления является максимизация всех "вознаграждений", как немедленных, так и в будущие моменты времени. Чтобы избежать появления бесконечных сумм, вводится так называемый "коэффициент переоценки", экспоненциально уменьшающий будущий "доход".

• Обсуждался подход к решению вероятностных проблем управления путём определения политики управления. Политика управления определяет выбор управляющего действия в виде функции информации робота об окружающем мире. Политика оптимальна, если она максимизирует сумму всех накопленных вознаграждений в будущем. Политика вычисляется на этапе планирования, который предшествует действиям робота. После вычисления она определяет оптимальные действия для любой возможной ситуации, с которой может столкнуться робот.

• Был выведен конкретный алгоритм нахождения оптимальной политики управления для ограниченного случая полностью наблюдаемой среды, в которой состояние полностью наблюдаемо. Такие случаи называются марковскими процессами принятия решений. Алгоритм включает вычисление функции дохода, которая измеряет ожидаемый суммарный выигрыш.

Функция дохода определяет политику путём жадного выбора управляющего действия, которое максимизирует доход. Если функция дохода оптимальна, оптимальна и политика. Итерационный алгоритм последовательно выполняет улучшает функцию дохода с помощью рекурсивного обновления.

• Были обсуждены примеры итерационного алгоритма MDP в задачах вероятностной робототехники. Для этого была извлечена мода оценки состояния, и аппроксимировано значение функции дохода с помощью сетки низкой размерности. Результатом стал алгоритм планирования движения для стохастических сред, позволивший роботу выполнять навигацию даже если эффекты действий не детерминированы.

Материал данной главы закладывает основу для следующей, где мы коснёмся более общей задачи управления при наличии неопределённости измерений, также известной как задача марковских процессов принятия решений с частичной наблюдаемостью. На уровне догадок уже обсуждалось, почему эта проблема намного более трудна по сравнению с MDP. В любом случае, некоторые выводы и базовые алгоритмы применимы и для более общей задачи.

Завершим эту главу упоминанием о том, что существует множество альтернативных методов вероятностного планирования и управления в условиях наличия неопределённости. Наш выбор итерации значений в качестве базового метода основан на его популярности. Кроме того, методы итерации значений являются одними из наиболее хорошо изученных для более общего случая POMDP.

Итерационный алгоритм ни в коем случае не является самым эффективным алгоритмом для генерирования сигналов управления. Известные алгоритмы планирования включают A* алгоритм, использующий эвристику при вычислении функции дохода, или методы прямого поиска политики, которые идентифицируют оптимальную локальную политику с помощью градиентного спуска. Однако, итерационный алгоритм играет ключевую роль в следующей главе, посвящённой значительно более сложному случаю оптимального управления при наличии неопределённости датчиков.

14.6 Библиографические примечания

Идея динамического программирования восходит к работам Беллмана (Bellman, 1957) и Говарда (Howard, 1960). Беллман (Bellman, 1957) определил уравнение равновесия для итерационного алгоритма, позже названного его именем. Марковские процессы принятия решений (MDP) с неполной оценкой состояния впервые обсуждались Астромом (Astrom, 1965), а также Майном и Осаки (Mine and Osaki, 1970) в ранних работах по марковским процессам принятия решений. Начиная с этого момента динамическое программирование управления стало очень обширным направлением, как свидетельствует одна из недавних книг (Bertsekas and Tsitsiklis 1996). Последние улучшения базовой парадигмы включают методы итерационный алгоритм в реальном времени (Korf 1988), итерационных алгоритмов, управляемых взаимодействием со средой (Barto et al. 1991), без модели (Watkins 1989), с параметрическим выражением функции ценности (Roy and Tsitsiklis 1996; Gordon 1995), или использования деревьев (Moore 1991) (также см. (Mahadevan and Kaelbling 1996)). Иерархические представления итерационного алгоритма были разработаны Парром и Расселлом (Parr and Russell, 1998), а Дитерих (Dietterich, 2000) и Бутильер (Boutilier et al., 1998) улучшили его эффек-

тивность в MDP с помощью анализа достижимости. Существует богатый набор литературы по приложению итерационного алгоритма, например, работа Барнива (Barniv, 1990) по обнаружению движущейся цели. В свете этой обширной литературы, материал в этой главе даёт лишь базовое представление самых простых методов итерационного алгоритма, позволяя подготовиться к изложению в следующей главе.

В робототехнике проблема планирования движения робота обычно исследуется без применения вероятностного подхода. Как было отмечено, подразумевается, что и состояние робота, и окружающая среда полностью известны и имеют детерминированные эффекты. Затруднения возникают из того факта, что пространство состояний непрерывно и имеет высокую размерность. Общепризнанная работа в этой области принадлежит Латомбу (Latombe, 1991). За ним следует основополагающая работа по многим базовым методам планирования движения, графам видимости (Wesley and Lozano-Perez 1979), управлению потенциальными полями (Khatib 1986), и известный алгоритм силуэтов Канни (Canny, 1987). Рават (Rowat, 1979) представил идею графов Вороного в области управления роботом, а Гуйбас (Guibas et al., 1992) показал способ их эффективного вычисления. Чозет (Choset, 1996) развил эту парадигму в семейство эффективных алгоритмов исследования и картографии. Другой набор методов использует рандомизированный (но не вероятностный!) подход к поиску в пространстве возможных путей робота (Kavraki and Latombe 1994). Актуальная книга по этому вопросу была написана Чозетом (Choset et al., 2004).

АППРОКСИМАЦИЯ РАЗБИЕНИЯ НА ЯЧЕЙКИ

В области планирования движения робота в этой главе обсуждались методы *аппроксимации разбиения на ячейки*, которые, в детерминированном случае, не гарантируют полноту. Разбиения непрерывных пространств на конечные графы изучалось в робототехнике десятилетиями. Рейф (Reif, 1979) разработал множество методов разбиения непрерывных пространств на конечное множество ячеек, сохраняющих полноту в планировании движения. Идея пространств конфигураций, необходимых для проверки пересечений с другими методами в этой главе, была изначально предложена Лозано-Перезом (Lozano-Perez, 1983). Методы рекурсивного разбиения ячеек для планирования конфигурационного пространства можно найти в работах Брукса и Лозано-Переза (Brooks and Lozano-Perez, 1985), но для идеальных допущений об идеальных моделях среды и роботов. Действия при частичной осведомлённости были описаны Голдбергом (Goldberg, 1993), в кандидатской диссертации которого были разработаны алгоритмы для частичного ориентирования в отсутствии датчиков. Шарма (Sharma, 1992) разработал методы планирования пути робота со стохастическими препятствиями.

Функции политики, назначающие управляющее действие для каждого возможного состояния робота известны как *контроллеры*. Алгоритм вывода политики управления часто называют *оптимальным контроллером* (Bryson and Yu-Chi, 1975). Политики управления также называют *навигационной функцией* (Koditschek 1987, Rimon and Koditschek, 1992). В области искусственного интеллекта они известны как универсальные пути (Schoppers, 1987), и многие алгоритмы символического планирования касаются проблемы нахождения таких универсальных планов (Dean et al., 1995, Kushmerick et al., 1995, Hoey et al., 1999). Некоторые работы в робототехнике были посвящены соединению универсальных планов и последовательностей действий с открытой петлей, например, в кандидатской диссертации Нурбахша (Nourbakhsh, 1987).

Также заметим, что запись "управления" в книге употребляется в несколь-

ко узком смысле. В главе специально не описаны стандартные методы в широкой области управления, такие, как управление PID и другие популярные методы, которые легко найти в учебниках (Dorf and Bishop 2001). Очевидно, такие методы необходимы и применимы во многих робототехнических системах для реального мира. Сознательный отказ от описания таких методов основан не на ограничениях объёма, и на том, что эти методы не полагаются на явное выражение неопределённости.

14.7 Упражнения

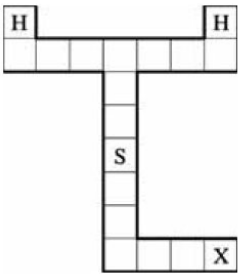
1. Алгоритм динамического программирования использует наиболее вероятное состояние для определения действия. Изобразить среду робота, в которой зависимость действий от наиболее вероятного состояния является неверным выбором? Можно ли указать конкретные причины, почему такой выбор может быть неверным?

2. Допустим, итерационный алгоритм для фиксированной функции дохода выполняется до окончания. Затем функция дохода меняется. Требуется изменить функцию дохода в дальнейших итерациях алгоритма, используя предыдущую функцию как начальную точку.

(a) Является ли это хорошим или плохим выбором? Зависит ли ответ от увеличения или уменьшения дохода?

(b) Можно ли, навскидку, сформулировать алгоритм, который будет более эффективным, чем просто продолжение итерационного алгоритма после изменения дохода? Если возможно, указать причины большей эффективности предложенного алгоритма. Если нет, указать причину, почему такой алгоритм существовать не может.

3. "Рай или Ад"? В этом упражнении требуется выполнить обобщение динамического программирования к среде с единственной переменной со скрытым состоянием. Среда представляет собой лабиринт с определенным стартом, обозначенным "S" и два возможных целевых состояния, отмеченных "H":



Агент не знает, какое из двух целевых состояний даст положительную награду. Одно состояние даст +100, а другое -100. Существует вероятность 0,5 что каждое из этих состояний истинно. Стоимость перемещения -1, агент может перемещаться только в четырёх направлениях - север, юг, запад, восток. Как только достигнуто состояние “H”, игра окончена.

(a) Реализовать итерационный алгоритм для этого сценария (игнорируя метку “X” на рисунке). Вычисляет ли предложенная реализация значение стартового состояния? Какова оптимальная политика?

(b) Изменить итерационный алгоритм для обработки вероятностной модели измерений: с вероятностью 0,9 агент двигается в указанном направлении, с вероятностью 0,1 – случайным образом перемещается по одному из трёх других направлений. Снова запустить итерационный алгоритм, вычислить значение в стартовом состоянии и оптимальную политику.

(c) Допустим, что в местоположении “X” стоит знак, информирующий агента о том, какая награда находится в каждом из состояний, обозначенных “H.” Как это повлияет на оптимальную политику?

(d) Как можно изменить итерационный алгоритм для нахождения оптимальной политики? Указать конкретные шаги. Указать изменения пространства, по которому изменяется значение дохода.

(e) Реализовать модификацию, вычислить значение в стартовом состоянии и оптимальную политику.

15 Марковские процессы принятия решений с частичной наблюдаемостью

15.1 Цель

В этой главе обсуждаются алгоритмы для решения задачи управления роботом с частичной наблюдаемостью. В этих алгоритмах учитывается как неопределенность измерения, так и неопределенность в эффектах управления. В них обобщается итерационный алгоритм, обсуждаемый в предыдущей главе, который был ограничен неопределённостью эффектов движения. Изучаемый набор уравнений называется *марковскими процессами принятия решений с частичной наблюдаемостью*, или *POMDP*. Это наименование закрепилось в литературе по исследованию операций. Термин *частичный* указывает на то, что состояние невозможно воспринять напрямую. Напротив, принимаемые роботом измерения являются неполными и, обычно, зашумлёнными проекциями этого состояния.

Как уже обсуждалось почти во всех главах книги, частичная наблюдаемость приводит к тому, что роботу необходимо оценивать апостериорное распределение по возможным состояниям среды. Алгоритмы для нахождения оптимальной политики управления существуют для сред с конечным количеством состояний, где пространство состояний, пространство действий, пространство наблюдений и горизонт планирования T конечны. К сожалению, именно эти методы вычислительно сложны. Все известные алгоритмы для более интересного непрерывного случая являются приблизительными.

Все алгоритмы, изучаемые в этой главе, основаны на методе итерационного алгоритма, который уже обсуждался выше. Ещё раз приведём выражение (14.14), которое является ключевым для такта обновления в MDP:

(15.1)
$$V_T(x) = \gamma \max_u [r(x,u) + \int V_{T-1}(x') p(x'|u,x) dx']$$

где $V_1(x) = \gamma \max_u r(x,u)$. В POMDP используется та же самая идея, но состояние x ненаблюдаемо. Роботу необходимо принять решение на основе пространства гипотез, которое является пространством апостериорных распределений по состояниям. В этой и следующей главах, кратко обозначим гипотезу символом b, вместо чуть более сложного bel, использованного в предыдущих главах.

POMDP вычисляет функцию дохода в пространстве гипотез следующим образом:

(15.2)
$$V_T(b) = \gamma \max_u [r(b,u) + \int V_{T-1}(b')p(b'|u,b)db']$$

где $V_1(b) = \gamma \max_u E_x[r(x,u)]$. Результирующая политика управления выглядит следующим образом:

(15.3)
$$\pi_T(b) = \gamma \operatorname*{argmax}_u [r(b,u) + \int V_{T-1}(b')p(b'|u,b)db']$$

Гипотеза является вероятностным распределением. Поэтому, каждое значение в POMDP является функцией всего вероятностного распределения, что весьма проблематично для вычислений. Если пространство состояний конечно, то пространство гипотез непрерывно, поскольку является пространством всех распределений по пространству состояний. Поэтому, существует непрерывное множество возможных значений, при том, что для MDP количество различных значений конечно. Ситуация ещё больше усложняется для непрерывных пространств состояний, где пространство гипотез представляет собой непрерывное множество бесконечной размерности.

Дополнительные трудности возникают из-за особенностей вычисления функции ценности. Выражения (15.2) и (15.3) интегрируются по всем гипотезам b'. Учитывая комплексную природу пространства состояний, далеко не всегда очевидно, что интегрирование можно провести в явном виде, или же есть возможность найти эффективную аппроксимацию. Неудивительно, что вычисление функции ценности V_T более сложно в пространстве гипотез, чем в пространстве состояний.

КУСОЧНО-ЛИНЕЙНАЯ ФУНКЦИЯ

К счастью существует точное решение для интересного особого случая конечных сред, в котором пространства состояний, действий, наблюдений и горизонт планирования конечны. Это решение выражает функции ценности в виде *кусочно-линейных функций* в пространстве гипотез. Как будет показано, линейность этого выражения напрямую следует из того факта, что ожидание является линейным оператором. Кусочное представление решения является следствием способности робота выбирать управляющее воздействие и в разных частях пространства гипотез будут выбраны разные управляющие воздействия. Все эти утверждения будут доказаны в ходе изложения в этой главе.

В главе обсуждается общий алгоритм POMDP для вычисления политик, определённых в пространстве всех распределений гипотез. Алгоритм вычислительно громоздкий, но применимый для конечных POMDP. Дополнительно будет обсуждаться очень легко вычислимый вариант. В следующей главе будут представлены более эффективные приближённые алгоритмы POMDP, пригодные для масштабирования в реальных задачах робототехники.

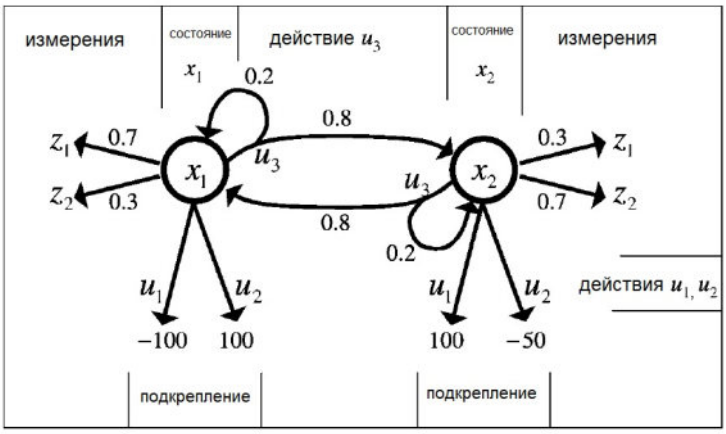


Рис. 15.1 Среда с двумя состояниями, используемая для иллюстрации итерационного алгоритма в пространстве гипотез.

15.2 Наглядный пример

15.2.1 Исходные данные

Проиллюстрируем итерационный алгоритм в пространствах гипотез с помощью числового примера. Пример является упрощением, но при его обсуждении можно определить все главные элементы итерационного алгоритма в пространствах гипотез.

Рис. 15.1 показана среда с двумя состояниями, обозначенными как x_1 и x_2. Робот способен выбирать между тремя управляющими действиями, u_1, u_2 и u_3. Действия u_1 и u_2 окончательны и при их выполнении мгновенно выдаётся следующая награда:

(15.4)
$$r(x_1, u_1) = -100 \qquad r(x_2, u_1) = +100$$

(15.5)
$$r(x_1, u_2) = +100 \qquad r(x_2, u_2) = -50$$

Дилемма состоит в том, что оба действия в каждом из состояний дают противоположные награды. В частности, в состоянии x_1 оптимальным действием является u_2, а действие u_1 - в состоянии x_2. Таким образом, знание состояния напрямую переносится на награду при выборе оптимального действия.

Чтобы получить такую информацию, у робота есть третье управляющее действие, u_3. Выполнение этого действия имеет небольшую стоимость -1:

(15.6)
$$r(x_1, u_3) = r(x_2, u_3) = -1$$

Будем считать, что это стоимость ожидания или восприятия. Действие u_3 недетерминировано воздействует на состояние среды в:

(15.7)
$$p(x'_1|x_1, u_3) = 0,2 \qquad p(x'_2|x_1, u_3) = 0,8$$

(15.8)
$$p(x'_1|x_2, u_3) = 0,8 \qquad p(x'_2|x_2, u_3) = 0,2$$

Другими словами, когда робот выполняет действие u_3, состояние переключается на противоположное с вероятностью 0,8, а робот платит единичную цену.

Тем не менее, выполнение действия u_3 имеет преимущества. Робот может выполнять действие восприятия перед выполнением каждого действия управления. С помощью восприятия робот приобретает знание о состоянии окружающей среды, и, в результате, сможет выбрать лучшее решение управления, которое, в перспективе, даст большую награду. Действие u_3 позволяет роботу выполнять восприятие без необходимости выполнения окончательного действия.

В нашем примере модель измерения управляется следующим вероятностным распределением:

(15.9)
$$p(z_1|x_1) = 0,7 \qquad p(z_2|x_1) = 0,3$$

(15.10)
$$p(z_1|x_2) = 0,3 \qquad p(z_2|x_2) = 0,7$$

Другими словами, если робот выполняет измерение z_1, возрастает его уверенность в нахождении в x_1, в то же время z_2 связано с x_2.

Пример с двумя состояниями был выбран потому, что он облегчает изображение функции в пространстве гипотез в виде графика. В частности, гипотеза состояния b характеризуется $p_1 = b(x_1)$ и $p_2 = b(x_2)$. Однако, мы знаем, что $p_2 = 1 - p_1$, и этого достаточно для изображения на графике p_1. Соответствующая политика управления π является функцией, отображающей единичный интервал $[0; 1]$ в пространство всех действий:

(15.11)
$$\pi : [0; 1] \longrightarrow u$$

15.2.2 Выбор управляющего действия

Чтобы определить нужный момент для выполнения управляющего действия, необходимо начать с условия наличия немедленной награды за каждое из трёх управляющих действий, u_1, u_2 и u_3. В предыдущей главе награда считалась функцией состояния и действий. Поскольку мы не знаем состояния, необходимость поменять выражение для награды, чтобы она соответствовала гипотезе состояния. Таким образом, для любой данный гипотезы $b = (p_1, p_2)$, ожидаемая для этой гипотезы награда задана следующим образом:

(15.12)
$$r(b, u) = E_x[r(x, u)] = p_1\, r(x_1, u) + p_2\, r(x_2, u)$$

НАГРАДА в POMDP

Функция $r(b, u)$ определяет *награду в POMDP.*

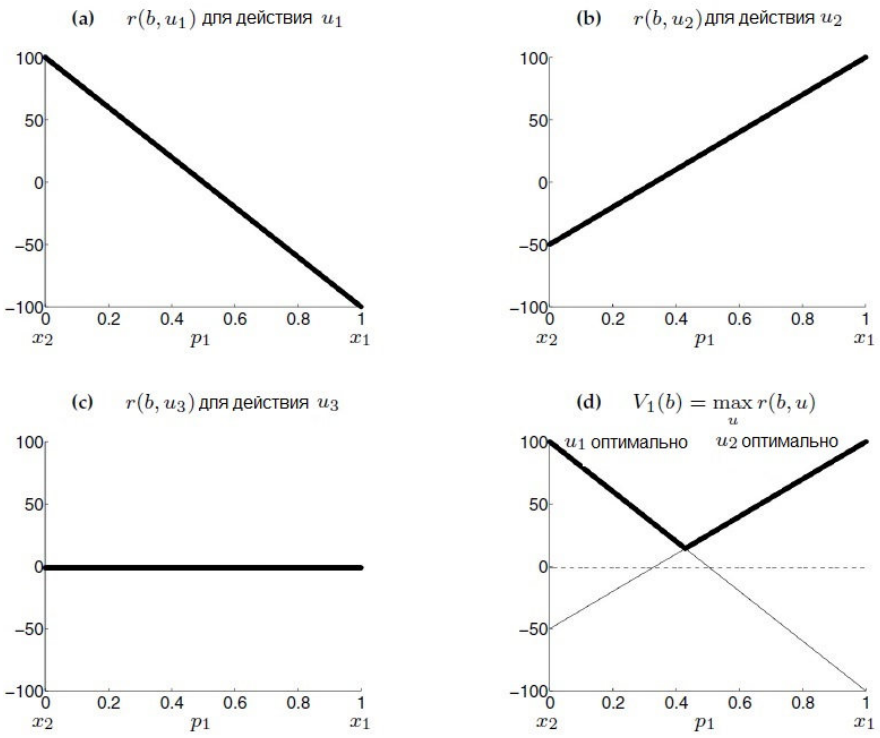


Рис. 15.2 На схемах (a), (b) и (c) показана ожидаемая награда r в виде функции параметра гипотезы состояния $p_1 = b(x_1)$ для каждого из трёх действий u_1, u_2 и u_3. Функция ценности для горизонта $T = 1$ соответствует максимуму этих трех линейных функций.(d)

На Рис. 15.2a показаны графики ожидаемой награды $r(b, u_1)$ при выборе управления u_1, заданного параметром p_1. С левой стороны схемы $p_1 = 0$, и робот абсолютно уверен, что среда находится в состоянии x_2. Выполнение действия u_1, таким образом, приводит к $r(x_2, u_1) = 100$, как задано в выражении (15.4). Справа $p_1 = 1$, поэтому состояние x_1. Соответственно, выбор элемента управления u_1 приведёт к результату $r(x_1, u_1) = -100$. Между тем, функция ожидания даст линейную комбинацию этих двух значений:

(15.13)

$$r(b, u_1) = -100p_1 + 100p_2 = -100p_1 + 100(1 - p_1)$$

Эта функция изображена на Рис. 15.2a.

На Рис.15.2b и (c) показаны соответствующие функции для действия u_2 и u_3, соответственно. Для u_2, получаем

(15.14)

$$r(b, u_2) = 100p_1 - 50(1 - p_1)$$

а для u_3 – постоянную функцию

(15.15)

$$r(b, u_3) = -1p_1 - 1(1 - p_1) = -1$$

Первым упражнением будет понимание функции итерационного алгоритма в пространствах гипотез для вычисление функции дохода V_1, которая

является оптимальной функцией ценности для процессов выбора с горизонтом $T = 1$. В одном цикле выбора робот имеет возможность остановиться на одном из трёх управляющих действий. Так какое следует предпочесть?

Ответ легко прочесть на показанных схемах. Для любой гипотезы состояния p_1 на схемах на Рис. 15.2a-c показаны графики ожидаемой награды для каждого выбора действий. Поскольку целью является максимизация награды, робот просто выбирает действие с наивысшим ожиданием награды. Это показано на Рис. 15.2d: на схеме представлены все три графика ожидаемой награды. В левой части оптимальным действием является u_1, поскольку преобладает его функция дохода. Переход происходит в момент, когда $r(b, u_1) = r(b, u_2)$, что разрешается до $p_1 = \frac{3}{7}$. Для значений p_1 больших, чем $\frac{3}{7}$, u_2 будет лучшим выбором. Поэтому, $(T = 1)$-оптимальная политика

(15.16)
$$\pi_1(b) = \begin{cases} u_1 & \text{если } p_1 \leq \frac{3}{7} \\ u_2 & \text{если } p_1 > \frac{3}{7} \end{cases}$$

Соответствующее значение показано жирной линией на графике на Рис. 15.2d. Это график кусочно-линейной выпуклой функции, которая является максимумом отдельных функций награды на Рис. 15.2a-c. Поэтому, можно записать ее в виде максимума трёх функций:

(15.17)
$$\begin{aligned} V_1(b) &= \max_u r(b, u) \\ &= \max \left\{ \begin{array}{rr} -100p_1 & +100(1-p_1) \\ 100p_1 & -50(1-p_1) \\ -1 & \end{array} \right\} \begin{array}{l} (*) \\ (*) \end{array} \end{aligned}$$

Очевидно, вклад вносят только линейные функции, обозначенные $(*)$ в (15.17). Оставшуюся линейную функцию можно смело отбросить:

(15.18)
$$V_1(b) = \max \left\{ \begin{array}{rr} -100p_1 & +100(1-p_1) \\ 100p_1 & -50(1-p_1) \end{array} \right\}$$

В рассматриваемом примере мы будем неоднократно использовать прием с обрезкой. Обрезка линейных ограничений показана пунктиром на Рис. 15.2d и многих рисунках в ходе изложения.

15.2.3 Восприятие

На следующем шаге добавим восприятие. Что, если робот способен воспринимать среду перед тем, как выполнить выбор действия? Как это повлияет на функцию оптимальной ценности? Очевидно, восприятие даст дополнительную информацию об окружающем мире, позволяя роботу выбрать верное действие. В частности, для наихудшей возможной гипотезы, $p_1 = \frac{3}{7}$, ожидаемая награда в наших примерах составила $\frac{100}{7} \approx 14,3$, соответствующему значение на перегибе графика на Рис. 15.2d. Очевидно, если бы можно было получить информацию об окружающем, гипотеза бы изменилась. Ценность этой гипотезы будет больше 14,3, но насколько?

Ответ несколько неожиданный. Допустим, результат восприятия z_1. На Рис. 15.3a показана гипотеза после восприятия z_1 как функция гипотезы

до измерения. Давайте проанализируем эту функцию. Если наша гипотеза до восприятия была $p_1 = 0$, гипотеза после измерения останется $p_1 = 0$, независимо от результатов измерения. Аналогично для $p_1 = 1$. Поэтому, на противоположных концах функция идентична. Но в середине графика имеется неопределённость оценки состояния среды, и процесс измерения z_1 смещает гипотезу. Величина смещения регулируется теоремой Байеса:

(15.19)

$$\begin{aligned} p_1' &= p(x_1|z) \\ &= \frac{p(z_1|x_1)p(x_1)}{p(z_1)} \\ &= \frac{0,7p_1}{p(z_1)} \end{aligned}$$

и

(15.20)

$$p_2' = \frac{0,3(1-p_1)}{p(z_1)}$$

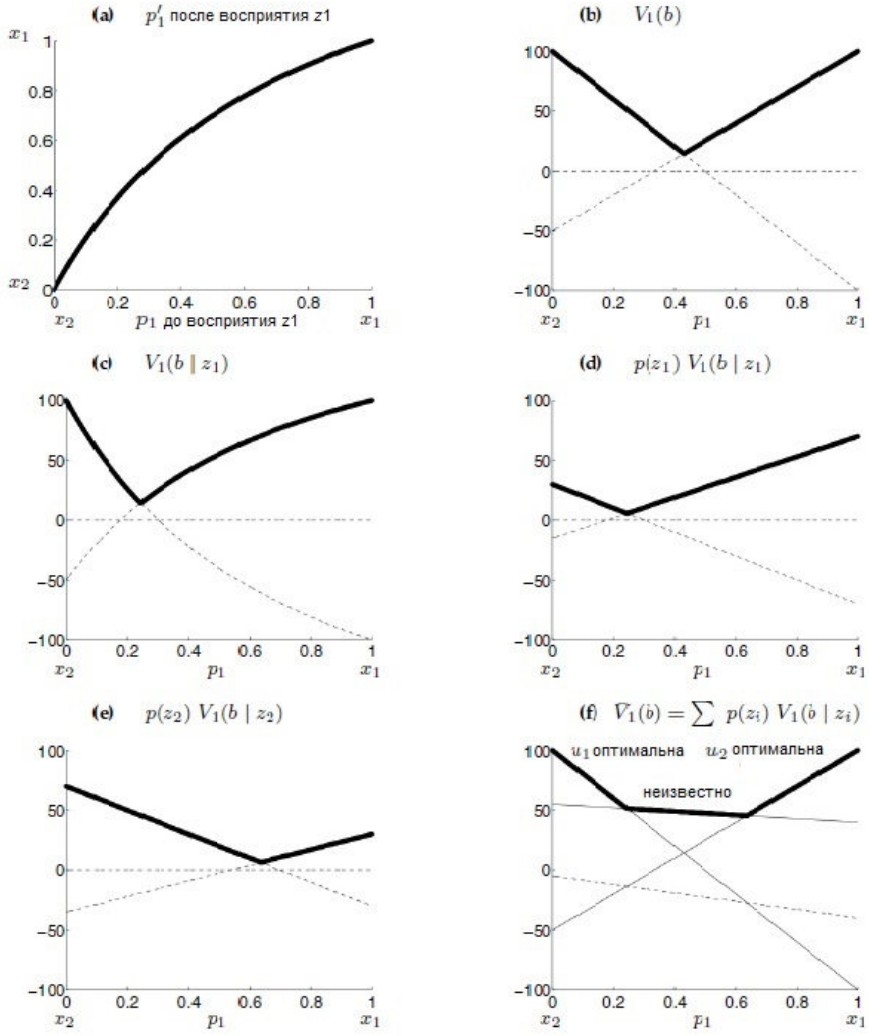
Нормирующий член $p(z_1)$ добавляет нелинейность на Рис. 15.3a. В нашем примере он разрешается до

(15.21)

$$p(z_1) = 0,7p_1 + 0,3(1-p_1) = 0,4p_1 + 0,3$$

поэтому $p_1' = \frac{0,7p_1}{0,4p_1+0,3}$. Однако, как будет показано ниже, этот нормирующий член свободно отбрасывается. Подробнее несколько ниже.

Давайте сначала изучим эффект этой передаточной функции на функцию дохода V_1. Допустим, известен результат измерения z_1, а затем необходимо сделать выбор действия. Каким будет этот выбор и как будет выглядеть соответствующая функция дохода? Ответ приведён в графическом виде на Рис. 15.3c. На рисунке показана кусочно-линейная функция дохода из Рис. 15.3b, спроектированная с помощью нелинейной функции измерения, обсуждаемой выше (и показанная на Рис. 15.3a). Читателю может понадобится какое-то время, чтобы сориентироваться: сначала берётся гипотеза p_1 и проектируется на соответствующую гипотезу p_1' согласно нелинейной функции, а её значение показано на Рис. 15.3b. Эта процедура для всех $p_1 \in [0;1]$ даст график, показанный на Рис. 15.3c.



Воздействие восприятия на функцию дохода: (a) Гипотеза после восприятия z_1 как функция гипотезы до восприятия z_1. Измерения z_1 увеличивает уверенность робота в состоянии x_1. Проекция функции дохода на график (b) с помощью этой нелинейной функции даст нелинейную функцию дохода, показанную на (c). (d) Деление функции дохода на вероятность наблюдения z_1 даст кусочно-линейную функцию. (e) Такая же кусочно-линейная функция для измерения z_2. (f) Ожидаемая функция дохода после восприятия.

Математически, график задан в виде

(15.22)

$$V_1(b|z_1) = \max \left\{ \begin{array}{cc} -100 \cdot \frac{0{,}7p_1}{p(z_1)} & +100 \cdot \frac{0{,}3(1-p_1)}{p(z_1)} \\ 100 \cdot \frac{0{,}7p_1}{p(z_1)} & -50 \cdot \frac{0{,}3(1-p_1)}{p(z_1)} \end{array} \right\}$$
$$= \frac{1}{p(z_1)} \max \left\{ \begin{array}{cc} -70p_1 & +30(1-p_1) \\ 70p_1 & -15(1-p_1) \end{array} \right\}$$

что является просто результатом замены p_1 на p_1' в функции дохода V_1,

приведённой в (15.18). Заметим, что на Рис. 15.3c гипотеза «наихудшего» значения смещена влево. Теперь худшая гипотеза - та, которая после восприятия z_1, заставляет думать с вероятностью $\frac{3}{7}$, что робот находится в состоянии x_1.

Однако, это соображение касается только одного из двух измерений, а значение до восприятия должно учитывать оба. Значение до восприятия, обозначенное $\bar{V}_1$, задано следующим ожиданием:

(15.23)
$$\bar{V}_1(b) = E_z[V_1(b|z)] = \sum_{i=1}^{2} p(z_i) V_1(b|z_i)$$

Сразу же можно заметить, что в этом ожидании каждая участвующая функция ценности $V_1(b|z_i)$ умножается на вероятность $p(z_i)$, которая была причиной нелинейности для функции ценности до измерения. Вставка (15.19) в это выражение даст

(15.24)
$$\begin{aligned}\bar{V}_1(b) &= \sum_{i=1}^{2} p(z_i) V_1(\frac{p(z_i|x_1)p_1}{p(z_i)}) \\ &= \sum_{i=1}^{2} p(z_i) \frac{1}{p(z_i)} V_1(p(z_i|x_1)p_1) \\ &= \sum_{i=1}^{2} V_1(p(z_i|x_1)p_1)\end{aligned}$$

Эта информация истинна, поскольку каждый элемент в V_1 линеен на $1/p(z_i)$, как показано на примере в (15.22). Теперь стало возможно вынести множитель $1/p(z_i)$ из функции максимизации. После решения выражения $p(z_i)$ просто сокращается!

В нашем примере два измерения, поэтому можно вычислить ожидание $p(z_i)V_1(b|z_i)$ для каждого из них. Читатель может вспомнить, что эти ожидания суммировались в выражении (15.23). Для z_1 мы уже вычислили $V_1(b|z_i)$ в (15.22), поэтому

(15.25)
$$p(z_1)V_1(b|z_1) = \max \left\{ \begin{array}{cc} -70p_1 & +30(1-p_1) \\ 70p_1 & -15(1-p_1) \end{array} \right\}$$

Эта функция показана на Рис. 15.3d и, действительно, является максимумом двух линейных функций. Аналогично, для z_2 получаем

(15.26)
$$p(z_2)V_1(b|z_2) = \max \left\{ \begin{array}{cc} -30p_1 & +70(1-p_1) \\ 30p_1 & -35(1-p_1) \end{array} \right\}$$

Эта функция приведена на Рис. 15.3e.

Искомая функция дохода до учёта восприятия получается сложением этих двух членов, согласно выражению (15.23):

(15.27)
$$\bar{V}_1 = \max \left\{ \begin{array}{cc} -70p_1 & +30(1-p_1) \\ 70p_1 & -15(1-p_1) \end{array} \right\} + \max \left\{ \begin{array}{cc} -30p_1 & +70(1-p_1) \\ 30p_1 & -35(1-p_1) \end{array} \right\}$$

Результирующая сумма показана на Рис. 15.3f. Она имеет примечательную форму: вместо одного перегиба на функции присутствует два, разделяющие функцию ценности на три разных линейных сегмента. Для левого сегмента оптимальным выбором является u_1, вне зависимости от информации, которую робот может получить с помощью восприятия. Аналогично для правого сегмента, где оптимальным выбором является u_2, вне зависимости от каких-либо обстоятельств. Однако, в центральном секторе восприятие робота имеет значение, определяющее оптимальное действие. В силу этого, в центральном сегменте определяется значение, которое существенно выше, чем соответствующее значение без учёта восприятия, что видно из Рис. 15.2d. Самое главное, что возможность восприятия поднимает значение функции дохода на более высокий уровень в целом регионе, где робот менее уверен в состоянии окружающего мира. Это существенное наблюдение показывает, что итерационный алгоритм в пространстве гипотез, действительно, зависит от восприятия, но только до тех пор, пока восприятие влияет на выбор действий управления в будущем.

Вернёмся к вычислению функции ценности, поскольку это может оказаться легче, чем ожидалось. Выражение (15.27) требует вычисления суммы двух максимумов линейных функций. Выражение в канонической форме, то есть в виде максимума линейных функций без суммы, потребует некоторых размышлений. В частности, новая функция дохода $\bar{V}_1$ будет ограничена снизу любой суммой, которая добавляет линейную функцию из первого выражения максимизации к линейной функции из второго выражения максимизации. Это даёт четыре возможные комбинации:

(15.28)

$$\begin{aligned}
\bar{V}_1(b) &= \max \left\{ \begin{array}{cccc}
-70p_1 & +30(1-p_1) & -30p_1 & +70(1-p_1) \\
-70p_1 & +30(1-p_1) & +30p_1 & -35(1-p_1) \\
70p_1 & -15(1-p_1) & -30p_1 & +70(1-p_1) \\
70p_1 & -15(1-p_1) & +30p_1 & -35(1-p_1)
\end{array} \right\} \\
&= \max \left\{ \begin{array}{cc}
-100p_1 & +100(1-p_1) \\
-40p_1 & -5(1-p_1) \\
40p_1 & +55(1-p_1) \\
100p_1 & -50(1-p_1)
\end{array} \right\} \begin{array}{l} (*) \\ \\ (*) \\ (*) \end{array} \\
&= \max \left\{ \begin{array}{cc}
-100p_1 & +100(1-p_1) \\
40p_1 & +55(1-p_1) \\
100p_1 & -50(1-p_1)
\end{array} \right\}
\end{aligned}$$

И снова, используем $(*)$ для обозначения ограничений, которые действительно вносят вклад в определение функции ценностей. Как показано на Рис. 15.3f, требуется только три из четырёх функций, а четвертую можно смело отсечь.

15.2.4 Прогнозирование

Финальный этап учитывает переход состояний. Когда робот выполняет действие, его состояние меняется. Для планирования на горизонте глубже $T = 1$, необходимо принимать во внимание и соответствующим образом проектировать функцию дохода. В нашем примере u_1 и u_2 оба действия окончательными. Теперь осталось только оценить эффект воздействия u_3.

К счастью, переходы состояний не всегда так сложны, как измерения в POMDP. На Рис. 15.4a показана проекция гипотезы после выполнения u_3.

Допустим, робот начинает в состоянии x_1 с абсолютной уверенностью, то есть $p_1 = 1$. Затем, согласно модели вероятности перехода в Выражении (15.7), получим $p'_1 = p(x'_1|x_1, u_3) = 0,2$. Аналогично, для $p_1 = 0$ получим $p'_1 = p(x'_1|x_2, u_3) = 0,8$. Между этими величинами ожидание линейно:

(15.29)

$$\begin{aligned} p'_1 &= E_x[p(x'_1|x, u_3)] \\ &= \sum_{i=1}^{2} p(x'_1|x_i, u_3)p_i \\ &= 0,2p_1 + 0,8(1 - p_1) = 0,8 - 0,6p_1 \end{aligned}$$

Эта функция графически представлена на Рис. 15.4a. Если теперь отобразить функцию ценности на Рис. 15.4b (что эквивалентно функции, показанной на Рис. 15.3f), мы получим функцию ценности на Рис. 15.4c. Эта функция ценности более плоская, чем та, что была до проекции, выражая потерю информации из-за смены состояний. Она также отражена зеркально, поскольку ожидание состояния изменяется при выполнении u_3.

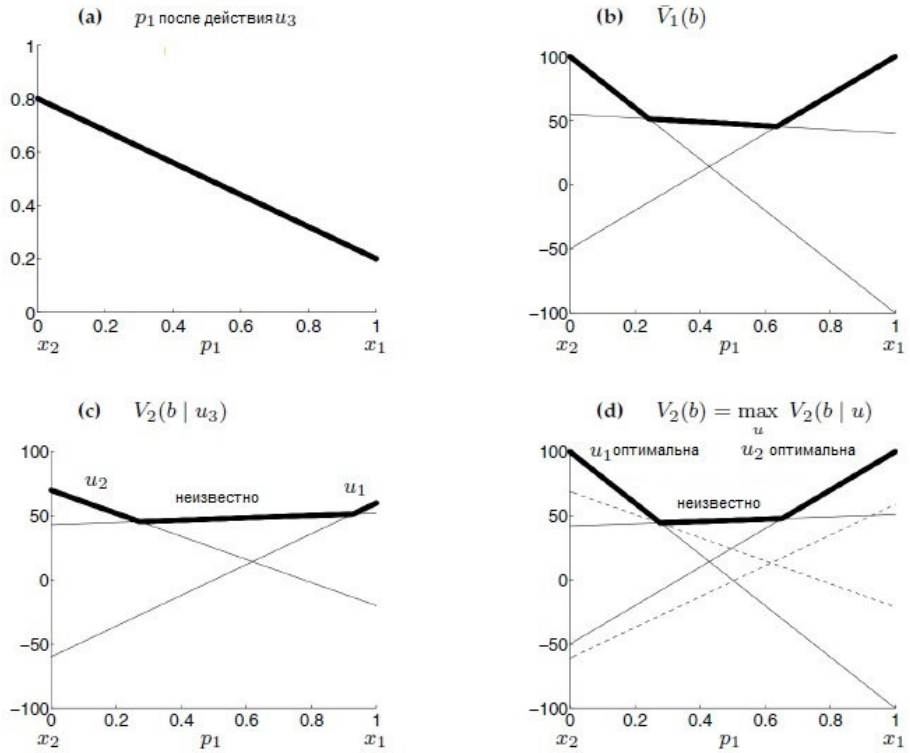


Рис. 15.4 (a) Параметр гипотезы состояния p'_1 после выполнения действия u_3, как функция параметра p_1 до выполнения действия. Распространение гипотезы показано на схеме (b) с помощью инверсии результатов проекции гипотезы, показанной на (c). (d) Функция дохода V_2 получена максимизацией функции гипотезы после ее прохождения, и награды двух оставшихся действий, u_1 и u_2.

Математически, эта функция ценности вычисляется проектированием (15.28) через (15.29).

(15.30)

$$\begin{aligned}\bar{V}_1(b|u_3) &= \max\left\{\begin{array}{rr} -100(0,8-0,6p_1) & +100(1-(0,8-0,6p_1)) \\ 40(0,8-0,6p_1) & +55(1-(0,8-0,6p_1)) \\ 100(0,8-0,6p_1) & -50(1-(0,8-0,6p_1)) \end{array}\right\} \\ &= \max\left\{\begin{array}{rr} -100(0,8-0,6p_1) & +100(0,2+0,6p_1) \\ 40(0,8-0,6p_1) & +55(0,2+0,6p_1) \\ 100(0,8-0,6p_1) & -50(0,2+0,6p_1) \end{array}\right\} \\ &= \max\left\{\begin{array}{rr} 60p_1 & -60(1-p_1) \\ 52p_1 & +43(1-p_1) \\ -20p_1 & +70(1-p_1) \end{array}\right\}\end{aligned}$$

Эти преобразования легко проверить вручную. Такая функция вместе с оптимальными управляющими действиями показаны на Рис. 15.4c.

Теперь конструирование функции дохода V_2 с горизонтом планирования $T = 2$ почти завершено. И снова перед роботом стоит выбор, выполнить ли действие u_3, или же одно из окончательных действий u_1 или u_2. Как и прежде, этот выбор выполняется дополнением в набор условий двух новых вариантов в виде двух линейных функций $r(b, u_1)$ и $r(b, u_2)$. Мы также должны вычесть стоимость выполнения действия u_3 из функции ценности.

Это даст схему, изображённую на Рис. 15.4d, которая имеет вид

(15.31)

$$\bar{V}_2(b) = \max\left\{\begin{array}{rr} -100p_1 & +100(1-p_1) \\ 100p_1 & -50(1-p_1) \\ 59p_1 & -61(1-p_1) \\ 51p_1 & +42(1-p_1) \\ -21p_1 & +69(1-p_1) \end{array}\right\} \begin{array}{l} (*) \\ (*) \\ \\ (*) \\ \\ \end{array}$$

Заметим, что было просто добавлено два варианта (строки один и два), и выполнено вычитание равномерной стоимости u_3 из всех других линейных ограничений (строки с третьей по пятую). И снова, необходимы только три из этих ограничений, обозначенных $(*)$. Результирующее значение может быть переписано в виде

(15.32)

$$\bar{V}_2(b) = \max\left\{\begin{array}{rr} -100p_1 & +100(1-p_1) \\ 100p_1 & -50(1-p_1) \\ 51p_1 & +42(1-p_1) \end{array}\right\}$$

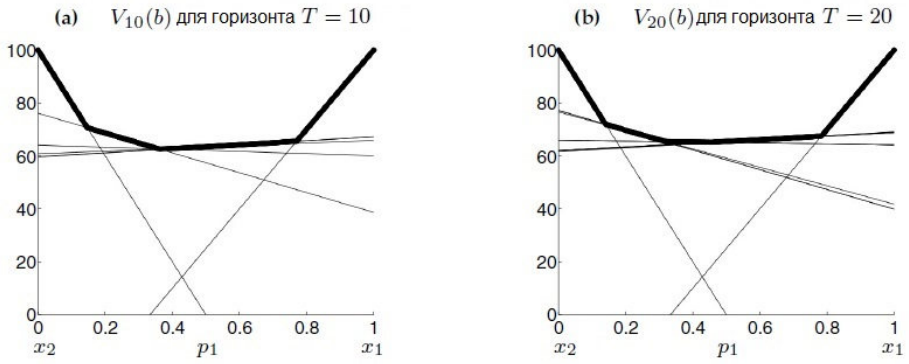


Рис. 15.5 Функция ценности V для горизонтов $T = 10$ и $T = 20$. Заметим, что масштаб по вертикальной оси на этих графиках отличает от предыдущих графиков функции ценности.

15.2.5 Глубокие горизонты и отсекание

ОБРАТНЫЙ ШАГ В ПРОСТРАНСТВЕ ГИПОТЕЗ

Мы выполнили полный обратный шаг в пространстве гипотез. Этот алгоритм легко выполняется рекурсивно. На Рис. 15.5 показана функция дохода для горизонтов $T = 10$ и $T = 20$, соответственно. Обе функции дохода выглядят похоже. С соответствующим отсеканием, в V_{20} имеется только 13 компонент:

(15.33)

$$\bar{V}_{20}(b) = \max \left\{ \begin{array}{rr} -100p_1 & +100(1-p_1) \\ 100p_1 & -50(1-p_1) \\ 64,1512p_1 & +65,9454(1-p_1) \\ 64,1513p_1 & +65,9454(1-p_1) \\ 64,1531p_1 & +65,9442(1-p_1) \\ 68,7968p_1 & +62,0658(1-p_1) \\ 68,7968p_1 & +62,0658(1-p_1) \\ 69,0914p_1 & +61,5714(1-p_1) \\ 68,8167p_1 & +62,0439(1-p_1) \\ 69,0369p_1 & +61,6779(1-p_1) \\ 41,7249p_1 & +76,5944(1-p_1) \\ 39,8427p_1 & +77,1759(1-p_1) \\ 39,8334p_1 & +77,1786(1-p_1) \end{array} \right\}$$

Сверху можно увидеть две уже знакомые функции. Все остальные соответствуют специфическим последовательностям измерений и выбора действий.

Этот простой пример показывает важность отсечения. Без него каждое новое обновление добавляет ещё два линейных ограничения (выбор действий), а затем возводит в квадрат количество ограничений (измерение). Поэтому, значение функции без отсечения для $T = 20$ определяется по 10^{547864} линейным ограничениям, а для $T = 30$ - по $10^{561012337}$. Обрезанная функция дохода, для сравнения, содержит всего 13 таких ограничений.

Этот невероятный взрывной рост является ключевой причиной непрактичности использования простых алгоритмов POMDP. На Рис. 15.6 приводится пошаговое сравнение этапов, которые образуют функцию дохода V_2. В левой части показаны функции после отсечения, в правой – все линейные

функции. Хотя в этом вычислении всего один шаг измерения, количество неиспользуемых функций уже достаточно велико. Мы вернёмся к этой теме позже, при выводе эффективных приближенных алгоритмов POMDP.

Последним наблюдением в нашем анализе является то, что оптимальная функция дохода для любого конечного горизонта непрерывная, кусочно-линейная и выпуклая. Каждый линейный участок соответствует разному выбору действий в некоторой точке в будущем. Выпуклость функции дохода указывает на довольно очевидное наблюдение, о том, что знание всегда лучше неведения. Если даны две гипотезы состояния b и b', смешанное значение гипотезы состояний больше или равно значению смешанной оценки состояния для некоторого параметра смешивания β, где $0 \leq \beta \leq 1$:

(15.34)
$$\beta V(b) + (1 - \beta)V(b') \geq V(\beta b + (1 - \beta)b')$$

Эта формализация применима только для случая конечных горизонтов. Для бесконечного горизонта функция ценности может быть прерывистой и нелинейной.

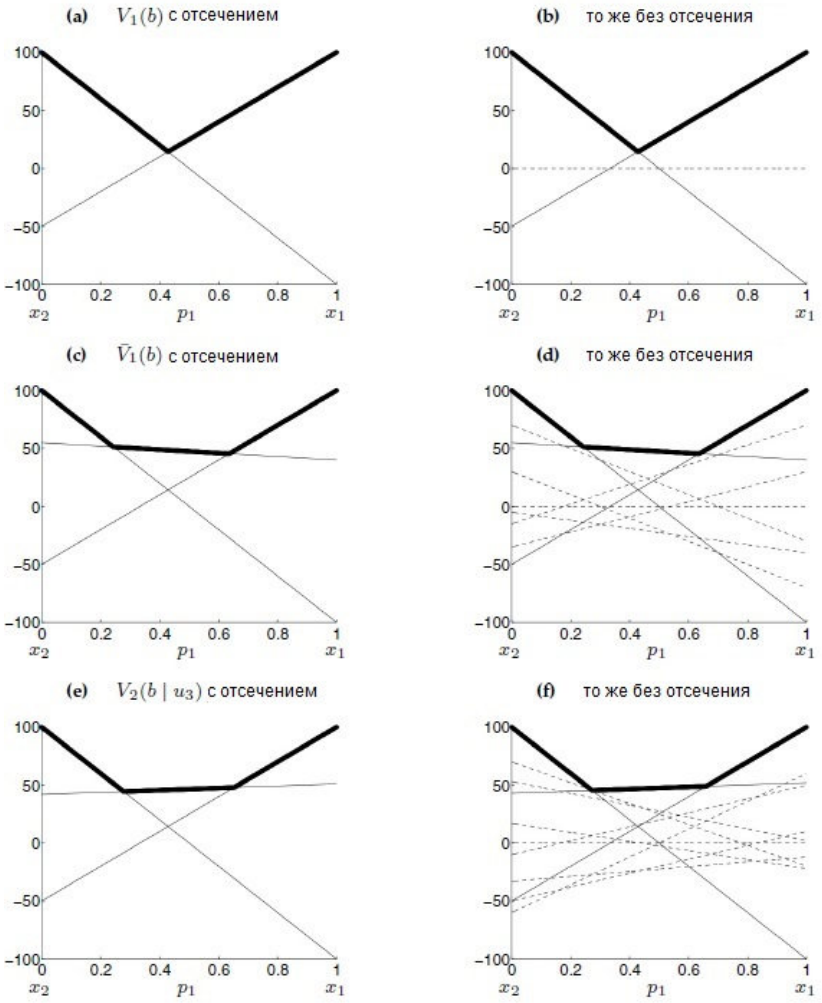


Рис. 15.6 Сравнения точного алгоритма отсечения (левый столбец) с алгоритмом POMDP без обрезки (правый столбец) для первых нескольких шагов алгоритма планирования POMDP. Очевидно, количество линейных ограничений без отсечения существенно увеличивается. При $T = 20$ значение функции без обрезки определено 10^{547864} линейными функциями, а с обрезкой - всего тринадцатью.

15.3 Алгоритм POMDP для конечной среды

В предыдущем разделе было показано на примере, как вычисляются функции ценности для конечных сред. Здесь мы кратко обсудим общий алгоритм для вычисления функции ценности, перед тем, как вывести его из общих положений.

Алгоритм POMDP приведён в Таблице 15.1. Он принимает на вход единственный параметр T, горизонт планирования для POMDP, и возвращает набор векторов параметров, каждый в виде

(15.35)
$$(v_1, ..., v_N)$$

Каждый из этих параметров определяет линейную функцию в простран-

стве гипотез вида

(15.36)
$$\sum_i v_i p_i$$

Фактическое значение определяется *максимумом всех этих линейных функций*:

МАКСИМУМ ЛИНЕЙНЫХ ФУНКЦИЙ

(15.37)
$$\max_{(p_1,\ldots,p_N)} \sum_i v_i p_i$$

Алгоритм POMDP рекурсивно вычисляет значение функций. Начальный набор для псевдо-горизонта $T = 0$ устанавливается в строке 2 Таблицы 15.1. Затем алгоритм POMDP рекурсивно вычисляет новый набор во вложенном цикле в строках 3-24. Ключевые вычисления выполняются в строке 9: Здесь вычисляются коэффициенты линейных функций $v^k_{u,z,j}$, необходимые для следующего набора линейных ограничений. Каждая линейная функция возникает из выполнения управляющего действия u, за которым следует измерение восприятия z, и выполнение действия управления u'. Линейное ограничение, соответствующее u', было вычислено в предыдущей итерации для меньшего горизонта планирования (взятого в строке 5). Поэтому, по достижении строки 14, была сгенерирована одна линейная функция для каждой комбинации действия управления, измерения и линейного ограничения предыдущей функции ценности.

Линейные ограничения новой функции дохода вычисляются взятием ожиданий по измерениям, как это сделано в строках 14-21. Для каждого действия управления алгоритм в строке 15 генерирует K^M таких линейных ограничений. Такое большое количество возникает из-за того, что каждое ожидание берётся по M возможным измерениям, каждое из которых можно «комбинировать» с любым из K ограничений, содержащихся в предыдущей функции значимости. В строке 17 вычисляется ожидание для каждой из таких комбинаций. Результирующее ограничение добавляется к новому набору ограничений в строке 19.

1: **Algorithm POMDP**(T) :
2: $\Upsilon = (0; 0, ..., 0)$
3: *для* $\tau = 1$ *до* T *выполнять*
4: $\Upsilon' = \emptyset$
5: *для всех* $(u'; v_1^k, ...v_N^k)$ *в* Υ *выполнить*
6: *для всех управляющих действий и выполнить*
7: *для всех измерений* z *выполнить*
8: *для* $j = 1$ *до* N *выполнять*
9: $v_{u,z,j}^k = \sum_{i=1}^N v_i^k p(z|x_i) p(x_i|u, x_j)$
10: *endfor*
11: *endfor*
12: *endfor*
13: *endfor*
14: *для всех управляющих действий и выполнить*
15: *для всех* $k(1), ..., k(M) = (1, ..., 1)$ *до* $(|\Upsilon|, ..., |\Upsilon|)$ *выполнить*
16: *для* $i = 1$ *до* N *выполнить*
17: $v_i' = \gamma[r(x_i, u) + \sum_z v_{u,z,i}^{k(z)}]$
18: *endfor*
19: *прибавить* $(u; v_1', ..., v_N')$ *к* Υ'
20: *endfor*
21: *endfor*
22: *опционально: обрезать* Υ'
23: $\Upsilon = \Upsilon'$
24: *endfor*
25: *return* Υ

Таблица 15.1 Алгоритм POMDP для дискретных сред. Этот алгоритм выражает оптимальную функцию значимости в виде набора вычисляемых рекурсивно линейных ограничений.

1: **Algorithm policy POMDP**$(\Upsilon, b = (p_1, ..., p_N))$:
2: $\hat{u} = \underset{(u; v_1^k, ..., v_N^k)}{\text{argmax}} \sum_{i=1}^N v_i^k p_i$
3: *return* $\hat{u}$

Таблица 15.2 Алгоритм определения оптимального действия для политики, выраженной набором линейных функций Υ.

Алгоритм для нахождения оптимального управляющего действия показан в Таблице 15.2. На вход алгоритма поступает гипотеза состояния, заданная в виде $b = (p_1, ..., p_N)$, а также набор линейных функций Υ. Оптимальное действие определяется поиском по всем линейным функциям, и нахождением той, которая имеет максимальное значение для b. Это значение возвращается в строке 3 алгоритма **policy_POMDP** в Таблице 15.2.

15.4 Математический вывод POMDP

15.4.1 Итерационный алгоритм в пространстве гипотез

Общее обновление функции дохода выполняется в (15.2), приводится здесь для удобства.

(15.38)
$$V_T(b) = \gamma \max_u \left[r(b,u) + \int V_{T-1}(b') p(b'|u,b) db' \right]$$

Начнём с преобразования этого равенства в более практичный вид, чтобы избежать интегрирования по пространству всех возможных гипотез.

Ключевым фактором этого обновления является условная вероятность $p(b'|u,b)$. Она определяет конкретное распределение из множества распределений на основании данных гипотез b и действий управления u. Это необходимо потому, что конкретная гипотеза b' также основана на следующем измерении, а само измерение генерируется стохастически. Необходимость иметь дело с распределениями из распределений добавляет излишнюю сложность.

Если зафиксировать измерение, апостериорная оценка b' будет уникальна, а вероятность $p(b'|u,b)$ вырождается до точечного распределения. Почему так? Ответ даст байесовский фильтр. Из гипотезы b перед выполнением действия u и последующего наблюдения z байесовский фильтр вычисляет единственную апостериорную оценку b', которая является единственной и верной гипотезой. Поэтому, можно заключить, что знание z устраняет необходимость интегрирования по всем гипотезам в (15.38).

Эту идею можно использовать, переписав выражение

(15.39)
$$p(b'|u,b) = \int p(b'|u,b,z) p(z|u,b) dz$$

где $p(b'|u,b,z)$ - точечное распределение из единственной гипотезы, вычисленной байесовским фильтром. Вставка этого интеграла в уравнение (15.38) даёт

(15.40)
$$V_T(b) = \gamma \max_u \left[r(b,u) + \int \left[\int V_{T-1}(b') p(b'|u,b,z) db' \right] p(z|u,b) dz \right]$$

Внутренний интеграл

(15.41)
$$\int V_{T-1}(b') p(b'|u,b,z) db'$$

содержит только один ненулевой член. Этот член b' является распределением, вычисленным из b, u, и z с помощью байесовского фильтра. Назовём это распределение $B(b,u,z)$:

(15.42)

$$\begin{aligned} B(b,u,z)(x') &= p(x'|z,u,b) \\ &= \frac{p(z|x',u,b)p(x'|u,b)}{p(z|u,b)} \\ &= \frac{1}{p(z|u,b)} p(z|x') \int p(x'|u,b,s)p(x|u,b)dx \\ &= \frac{1}{p(z|u,b)} p(z|x') \int p(x'|u,x)b(x)dx \end{aligned}$$

Читатель должен помнить знакомый уже вывод байесовского фильтра, который подробно обсуждался в Главе 2, на этот раз с нормирующим членом в явном виде.

Теперь можно переписать (15.40) в следующем виде. Заметим, что это выражение больше не интегрируется по b'.

(15.43)

$$V_T(b) = \gamma \max_u \left[r(b,u) + \int V_{T-1}(B(b,u,z))p(z|u,b)dz \right]$$

Такая форма более удобна по сравнению с первоначальной в (15.38), поскольку требует интегрирования по всем возможным измерениям z, вместо всех возможных распределений гипотез b'. Это преобразование было косвенно использовано в примере выше, где новая функция ценности получалась путём смешивания конечного множества кусочно-линейных функций.

Дальше будет удобнее отделить максимизацию по действиям от интегрирования. Заметим, что (15.43) можно переписать в виде двух равенств:

(15.44)

$$V_T(b,u) = \gamma \left[r(b,u) + \int V_{T-1}(B(b,u,z))p(z|u,b)dz \right]$$

(15.45)

$$V_T(b) = \max_u V_T(b,u)$$

Здесь $V_T(b,u)$ - горизонт T для функции дохода по гипотезе b, считая, что следующее действие u.

15.4.2 Выражение функции дохода

Как и в примере, выразим функцию дохода в виде максимума набора линейных функций. Мы уже обсуждали, что любая линейная функция по одиночной гипотезе может быть выражена набором коэффициентов $v_1, ..., v_N$:

(15.46)

$$V(b) = \sum_{i=1}^{N} v_i p_i$$

где, как обычно, $p_1, ..., p_N$ - параметры распределения гипотез b. Как и в примере, кусочно-линейная выпуклая функция $V_T(b)$ может быть выражена в виде максимума конечного набора линейных функций

(15.47)

$$V(b) = \max_k \sum_{i=1}^{N} v_i^k p_i$$

где $v_1^k, ..., v_N^k$ означает параметры k-й линейной функции. Читатель должен быстро убедиться, что максимум конечного набора линейных функций, действительно представляет собой выпуклую, непрерывную и кусочно-линейную функцию.

15.4.3 Вычисление функции дохода

Выведем рекурсивное выражение для вычисления функции дохода $V_T(b)$. Исходя из соображений индукции, что $V_{T-1}(b)$, функция ценности для горизонта $T-1$, выраженная кусочно-линейной функцией, как указывалось выше. Как часть вывода, продемонстрируем, что при допущении, что $V_{T-1}(b)$ является кусочно-линейной и выпуклой, $V_T(b)$ также кусочно-линейная и выпуклая. Индукция по горизонту планирования T доказывает, что все функции дохода с конечным горизонтом, действительно, кусочно-линейные и выпуклые.

Начнём с выражений (15.44) и (15.45). Если пространство измерений конечно, можно заменить интегрирование по z конечной суммой.

(15.48)

$$V_T(b,u) = \gamma \left[r(b,u) + \sum_z V_{T-1}(B(b,u,z)) p(z|u,b) \right]$$

(15.49)

$$V_T(b) = \max_u V_T(b,u)$$

Гипотеза $B(b,u,z)$ получается, используя следующее выражение, выведенное из выражения (15.42) заменой интеграла конечной суммой.

(15.50)

$$B(b,u,z)(x') = \frac{1}{p(z|u,b)} p(z|x') \sum_x p(x'|u,x) b(x)$$

Если гипотеза b выражена параметрами $\{p_1, ..., p_N\}$, а гипотеза $B(b,u,z)$ - параметрами $\{p'_1, ..., p'_N\}$, следовательно j-й параметр гипотезы b' вычисляется следующим образом:

(15.51)

$$p'_j = \frac{1}{p(z|u,b)} p(z|x_j) \sum_{i=1}^{N} p(x_j|u,x_i) p_i$$

Для вычисления обновления функции ценности (15.48), найдём более практичное выражение для члена $V_{T-1}(B(b,u,z))$, используя описанные выше конечные суммы. Наш вывод начинается с определения V_{T-1} и замены p'_j в соответствии с выражением (15.51):

(15.52)

$$\begin{aligned} V_{T-1}(B(b,u,z)) &= \max_k \sum_{j=1}^{N} v_j^k p'_j \\ &= \max_k \sum_{j=1}^{N} v_j^k \frac{1}{p(z|u,b)} p(z|x_j) \sum_{i=1}^{N} p(x_j|u,x_i) p_i \\ &= \frac{1}{p(z|u,b)} \max_k \sum_{j=1}^{N} v_j^k p(z|x_j) \sum_{i=1}^{N} p(x_j|u,x_i) p_i \\ &= \frac{1}{p(z|u,b)} \max_k \overbrace{\sum_{i=1}^{N} p_i \underbrace{\sum_{j=1}^{N} v_j^k p(z|x_j) p(x_j|u,x_i)}_{(*)}}^{(**)} \end{aligned}$$

Член гипотезы, обозначенный $(*)$? независим. Поэтому, функция, обозначенная $(**)$? является линейной в параметрах пространства гипотез $p_1, ..., p_N$. Член $1/p(z|u,b)$ нелинейный и трудно вычисляется, поскольку содержит полную гипотезу b как условную переменную. Однако, красота POMDP состоит в том, что это выражение можно отбросить. В частности, замена этого выражения обратно в (15.48) даст следующее равенство обновления:

(15.53)

$$V_T(b,u) = \gamma \left[r(b,u) + \sum_z \max_k \sum_{i=1}^{N} p_i \sum_{j=1}^{N} v_j^k p(z|x_j) p(x_j|u,x_i) \right]$$

Поэтому, несмотря на нелинейность, возникающую при обновлении измерения, $V_T(b,u)$ тоже кусочно-линейная функция.

Наконец, заметим, что $r(b,u)$ задан ожиданием

(15.54)

$$r(b,u) = E_x[r(x,u)] = \sum_{i=1}^{N} p_i r(x_i,u)$$

Здесь подразумевается, что гипотеза b выражена параметрами $\{p_1, ..., p_N\}$.

Искомая функция ценности V_T теперь получается максимизацией $V_T(b,u)$ по всем действиям u, как показано в (15.49):

(15.55)

$$V_T(b) = \max_u V_T(b,u)$$
$$= \gamma \max_u \left(\left[\sum_{i=1}^{N} p_i r(x_i,u) \right] + \sum_z \max_k \sum_{i=1}^{N} p_i \underbrace{\sum_{j=1}^{N} v_j^k p(z|x_j) p(x_j|u,x_i)}_{=:v_{u,z,i}^k} \right)$$
$$= \gamma \max_u \left(\left[\sum_{i=1}^{N} p_i r(x_i,u) \right] + \underbrace{\sum_z \max_k \sum_{i=1}^{N} p_i v_{u,z,i}^k}_{(*)} \right)$$

где

(15.56)

$$v_{u,z,i}^k = \sum_{j=1}^{N} v_j^k p(z|x_j) p(x_j|u,x_i)$$

как было показано ранее. Но выражение пока ещё не задано в виде максимума линейных функций. В частности, понадобится поменять выражение "сумма-максимум-сумма", обозначенное $(*)$ в (15.55) на выражение вида "максимум-сумма-сумма", которое является максимумом по множеству линейных функций.

Используется то же самое преобразование, которое уже было показано в примере в подразделе 15.2.3. А именно, допустим, требуется вычислить максимум

(15.57)

$$\max\{a_1(x), ..., a_n(x)\} + \max\{b_1(x), ..., b_n(x)\}$$

для некоторых функций $a_1(x), ..., a_n(x)$ и $b_1(x), ..., b_n(x)$ по переменной x. Максимум получается в виде

(15.58)

$$\max_i \max_j [a_i(x) + b_j(x)]$$

Это следует из того факта, что $a_i + b_j$, в действительности, нижняя граница. Далее, для каждого x должны существовать такие i и j, что $a_i(x) + b_j(x)$ определяет максимум. Включая все такие потенциальные пары в (15.58), получим плотную нижнюю границу, то есть решение.

Теперь легко выполнить обобщение на произвольные суммы по выражениям максимума:

(15.59)

$$\sum_{j=1}^{m} \max_{i=1}^{N} a_{i,j}(x) = \max_{i(1)=1}^{N} \max_{i(2)=1}^{N} ... \max_{i(m)=1}^{N} \sum_{j=1}^{m} a_{i(j),j}$$

и применить этот приём к вычислению функции дохода POMDP, выразив $(*)$ в (15.55). Пусть M будет общим количеством измерений.

(15.60)

$$\sum_z \max_k \sum_{i=1}^{N} p_i v_{u,z,i}^{k} = \max_{k(1)} \max_{k(2)} ... \max_{k(M)} \sum_z \sum_{i=1}^{N} p_i v_{u,z,i}^{k(z)}$$
$$= \max_{k(1)} \max_{k(2)} ... \max_{k(M)} \sum_{i=1}^{N} p_i \sum_z v_{u,z,i}^{k(z)}$$

Здесь каждая $k()$ является отдельной переменной, принимая значения k с левой стороны знака равенства. Этих переменных будет столько же, сколько измерений. В результате, искомая функция ценности теперь получается следующим образом:

(15.61)

$$V_T(b) = \gamma \max_u \left[\sum_{i=1}^{N} p_i \, r(x_i, u) \right] + \max_{k(1)} \max_{k(2)} ... \max_{k(M)} \sum_{i=1}^{N} p_i \sum_z v_{u,z,i}^{k(z)}$$
$$= \gamma \max_u \max_{k(1)} \max_{k(2)} ... \max_{k(M)} \sum_{i=1}^{N} p_i \left[r(x_i, u) + \sum_z v_{u,z,i}^{k(z)} \right]$$

Другими словами, каждая комбинация

$$\left(\left[r(x_1, u) + \sum_z v_{u,z,1}^{k(z)} \right] \left[r(x_2, u) + \sum_z v_{u,z,2}^{k(z)} \right] \; ... \; \left[r(x_N, u) + \sum_z v_{u,z,N}^{k(z)} \right] \right)$$

образует новое линейное ограничение в функции ценности V_T.

Для каждого уникального объединённого множества переменных $k(1), k(2), ..., k(M)$ будет по одному такому ограничению. Очевидно, максимум этих линейных функций снова кусочно-линейный и выпуклый, что доказывает, что этого выражения достаточно для выражения верной функции ценности по базовому непрерывному пространству гипотез. Далее, количество линейных участков будет дважды экспоненциально относительно пространства измерений, по крайней мере, для нашей наивной реализации, которая сохраняет все такие ограничения.

15.5 Практические соображения

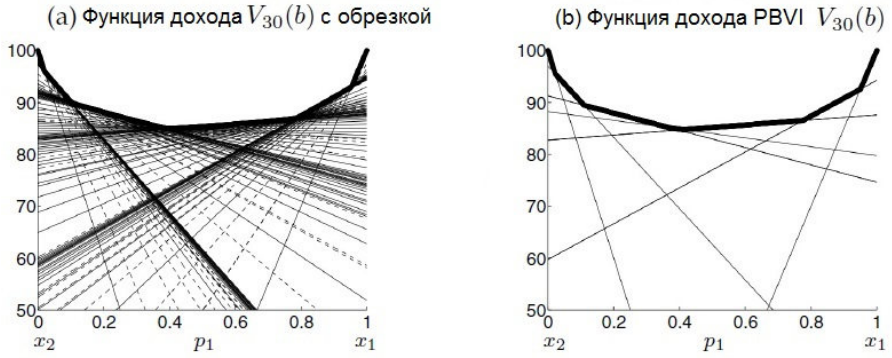


Рис. 15.7 Преимущества итерационного алгоритма на основе точек над общим итерационным алгоритмом: На схеме (a) показано точное значение функции для горизонта $T = 30$ для другого примера, состоящего из 120 ограничений после отсечения. С правой стороны результат работы алгоритма PBVI сохраняющего только 11 линейных функций. Результаты обоих функции, в части применимости к управлению, практически неразличимы.

Пока что обсуждаемый итерационный алгоритм далёк от практического. Для любого разумного количества явных состояний, измерений и действий управления сложность функции дохода чрезмерная, даже при относительно небольших горизонтах планирования.

Существует ряд возможностей реализовать более эффективные алгоритмы. Одна такая возможность уже обсуждалась в примере: количество линейных ограничений быстро растет до чрезмерных значений. К счастью, большое количество линейных ограничений можно смело игнорировать, поскольку они не участвуют в определении максимума.

Другим связанным недостатком итерационного алгоритма является вычисление функций дохода *для всех* гипотез состояния, не только для релевантных. Когда робот начинает с хорошо определённой гипотезой состояния, набор доступных гипотез состояния часто достаточно мал. Например, если робот пытается пройти через две двери, для которых неизвестно, открыты они или закрыты, состояние первой двери станет очевидным при достижении второй. Поэтому, гипотеза состояния, в которой известно только состояние второй двери, когда первой – неизвестно, физически невозможно. Во многих областях применения недостижимы огромные участки пространства гипотез.

Даже для достижимых гипотез некоторые могут быть достижимы лишь с малой вероятностью. Другие могут быть просто нежелательны, и робот будет избегать их. Итерационный алгоритм таких различий не делает. Фактически, время и ресурсы, вложенные в вычисления дохода, не зависят от шансов релевантности гипотезы состояния.

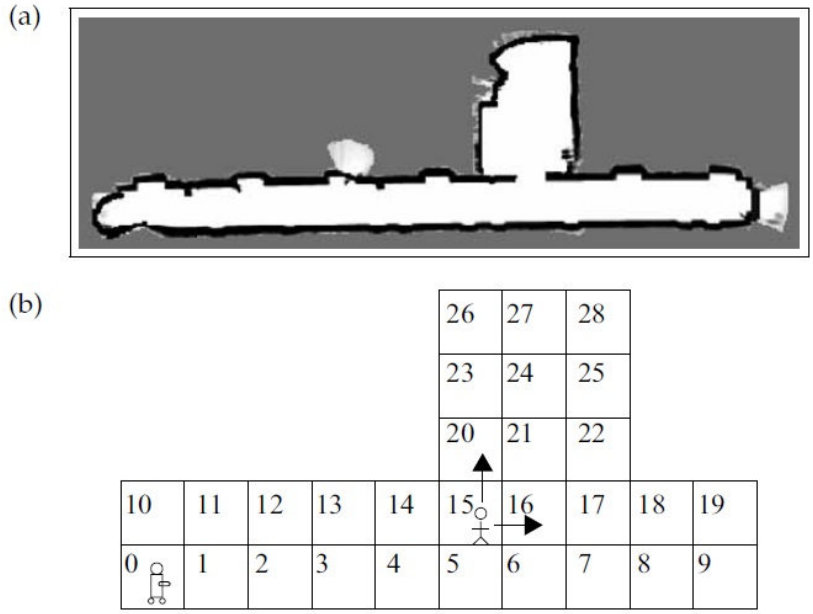


Рис. 15.8 Среда внутри помещения, в которой выполняется поиск политики обнаружения движущегося нарушителя. Карта сетки занятости(a), и дискретное пространство состояния POMDP (b). Робот отслеживает своё положение достаточно хорошо, чтобы игнорировать неопределенность положения. Оставшаяся неопределенность относится к местоположению человека. Собственность Джолли Пиню, университет МакГилла (Joelle Pineau, McGill University).

ИТЕРАЦИОННЫЙ АЛГОРИТМ НА ОСНОВЕ ТОЧЕК

Существует огромное множество алгоритмов, которые более разборчивы относительно областей пространства гипотез, для которых вычисляется функция дохода. Один из них называется итерационный алгоритм на основе точек *(point-based value iteration – PBVI)* и основан на идее сохранения набора эталонных гипотез состояний, и ограничения функции дохода лишь теми ограничениями, которые максимизируют её значение для, по крайней мере, одной из этих гипотез состояний. Представим, что дано множество гипотез состояния $B = b_1, b_2, ...,$ называемых *точками гипотез*. Затем, *приведённая функция дохода* V по отношению к B является набором ограничений $v \in V$ для которого можно найти, по крайней мере, один $b_i \in B$ такой, чтобы $v(b_i) = V(b_i)$. Другими словами, линейные сегменты, которые не совпадают с какой-либо дискретной точкой гипотезы в B, отбрасываются. Оригинальный алгоритм PBVI эффективно вычисляет функцию дохода даже не генерируя ограничения, которые не поддерживаются какой-либо из точек. Однако, та же самая идея может быть реализована отсечением всех линейных сегментов после их генерации в стандартном обратном шаге POMDP.

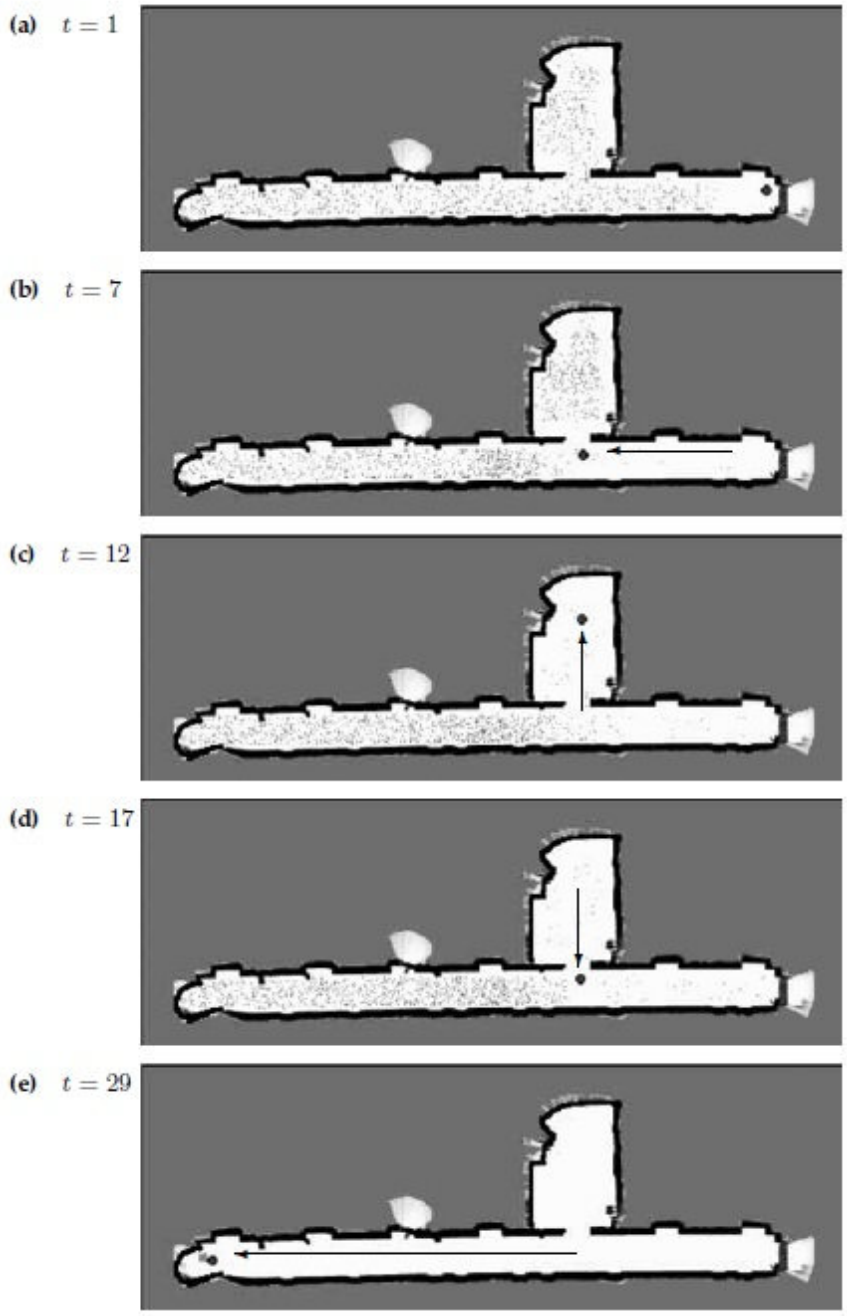


Рис. 15.9 Успешная политика поиска. Здесь отслеживание нарушителя реализовано с помощью многочастичного фильтра, который затем проектируется на выражение в виде гистограммы, подходящее для POMDP. Робот сначала осматривает комнату сверху, а затем проходит ниже по коридору. Собственность Джоелли Пиню, университет МакГилла (Joelle Pineau, McGill University).

Идея сохранения множества точек гипотез может существенно улучшить эффективность итерационного алгоритма. На Рис. 15.7a показана функция дохода для задачи, отличной от примера в подразделе 15.2 только в одном

аспекте: функция перехода состояний детерминирована (нужно просто заменить 0,8 на 1,0 в (15.7) и (15.8)). Функция дохода на Рис. 15.7a оптимальна по отношению к горизонту $T = 30$. Тщательное отсечение уменьшает её до 120 ограничений, вместо $10^{561012337}$ которые могла бы дать реализация без отсечений, конечно, при достаточном терпении. С простым набором точек $B = \{p_1 = 0, 0, p_1 = 0, 1, p_1 = 0, 2, ..., p_1 = 1\}$, получится функция дохода, показанная справа на Рис. 15.7b. Эта функция ценности приблизительна, и состоит только из 11 линейных функций. Что более важно, ее вычисление выполняется более чем в 1000 раз быстрее.

Использование точечных гипотез имеет второе важное следствие. Решатель задач может отобрать точечные гипотезы, признанные релевантными для процесса планирования. Для определения точечных гипотез существует самая разнообразная эвристика. Главным способом является идентификация достижимых гипотез (например, с помощью имитационного моделирования робота в POMDP), и нахождение гипотез, которые расположены достаточно далеко друг от друга. С помощью таких методов можно создавать алгоритмы POMDP, которые будут на много порядков быстрее. Фактически, возможно инкрементно конструировать множество B и постепенно строить множество функций дохода $V_1, v_2, ..., V_T$, добавляя новые ограничения к каждой функции при добавлении новой точечной гипотезы. Таким образом, алгоритм планирования становится *независимым от времени*, и его результаты со временем улучшаются.

Перспективной точкой зрения в робототехнике является мнение, что количество разумных гипотез состояния превышает количество состояний только на постоянный коэффициент. Как следствие, методы с активным отбором подходящих областей в пространстве гипотез для обновления при планировании, имеют фундаментально совершенно другие свойства масштабирования, нежели плоские «неразборчивые» методы итерационных алгоритмов.

Типичный результат применения PBVI в робототехнике показан на Рис. 15.8 и 15.9. На Рис. 15.8a показана карта сетки занятости среды внутри помещения, которая состоит из длинного коридора и комнаты. Робот начинает с правой стороны схемы с задачей обнаружить нарушителя, который перемещается случайным образом. Чтобы приспособить эту задачу для алгоритма планирования с помощью PBVI требуется пространство состояний низкой размерности. Используемое пространство состояний показано на Рис. 15.8b. В нем карта сетки разбита на 22 дискретных области. Такого представления разбиения достаточно для решения задачи, при этом вычисление функции дохода PBVI становится возможным. Задача обнаружения нарушителя изначально носит вероятностный характер. В любой политике управления требуется учитывать неопределённость среды и попытаться ее уменьшить. Кроме того, изначально присутствует динамика и одно лишь передвижение по ещё неизвестным местам уже не работает. На Рис. 15.9 показан типичный результат планирования с помощью POMDP. Робот уже определил последовательность управления, в которой сначала обследуется сравнительно небольшая комната, а затем выполняется перемещение в коридор. Такая политика управления основана на предположении, что у нарушителя недостаточно времени, чтобы пройти через коридор, пока робот обследует комнату. Для такой политики вероятность успеха достаточно велика.

Это хрестоматийный пример использования итерационного алгоритма POMDP в реальной задаче управления роботом. Даже при использовании агрессивного отсечения, как в PBVI, результирующие функции дохода все

ещё ограничены несколькими десятками состояний. Однако, если удаётся найти выражение среды с низкой размерностью, методы POMDP показывают прекрасные результаты, обрабатывая изначальное значение неопределённости в робототехнике.

15.6 Выводы

В этом разделе был представлен базовый итерационный алгоритм для управления роботом в условиях неопределённости.

• POMDP характеризуется несколькими типами неопределённости: эффектов управления, восприятия, и неопределённость, вносимая динамикой среды. Однако в алгоритмах POMDP подразумевается, что дана вероятностная модель действий и восприятия.

• Функция дохода в POMDP определена в пространстве всех гипотез робота относительно окружающей среды. Для сред с N состояниями, гипотеза определена в $(N-1)$-мерном подпространстве гипотез, характеризующимся вероятностью, назначенного для каждого из N состояний.

• Для конечных горизонтов функция дохода кусочно-линейная для параметров пространства гипотез, а также непрерывная и выпуклая. Поэтому, ее можно представить в виде максимума конечного набора линейных функций. Более того, эти линейные ограничения легко вычислить.

• Алгоритм планирования POMDP вычисляет последовательность функций дохода для расширения горизонтов планирования. Каждое такое вычисление рекурсивно: при заданной функции дохода для горизонта $T-1$, алгоритм выполняет вычисление оптимальной функции дохода для горизонта T.

• В каждой рекурсивной итерации комбинируется несколько элементов – выбор действия реализуется с помощью максимизации по множеству линейных ограничений, где каждое действие содержит собственное множество. Предполагаемое измерение учитывается путём комбинирования наборов линейных ограничений, по одному для каждого измерения. Прогнозирование выполняется линейными манипуляциями с набором линейных ограничений. Доход обобщается в пространстве гипотез вычислением его ожидания, что, опять же, линейно в пространстве параметров гипотез. Результатом является процедура обратного прохода, которая управляет линейными ограничениями.

• Мы обнаружили, что при каждом базовом обновлении возникает недопустимо много новых линейных ограничений. В каждом отдельном обратном проходе этап измерения увеличивает количество линейных ограничений на коэффициент, величина которого экспоненциально зависит от числа возможных измерений. Большинство из этих ограничений, обычно, не оказывают никакого влияния, и их отбрасывание не меняет функцию дохода.

• Итерационный алгоритм на основе точек (PBVI) – это приблизительный алгоритм, сохраняющий только те ограничения, которые требуются для поддержания конечного набора репрезентативных гипотез состояния. В таком случае количество ограничений остаётся постоянным, вместо то-

го, чтобы расти, в худшем случае, экспоненциально. Эмпирически, PBVI обеспечивает хороший результат для случаев, когда отобраны репрезентативные точки в пространстве гипотез с достаточным промежутком между ними.

Во многом, описанный в этой главе материал представляет лишь теоретический интерес. Итерационный алгоритм определяет общий механизм обновления, который поддерживается в большом количестве алгоритмов принятия решений. Однако, сам по себе он вычислительно нереализуем, поэтому эффективные реализации носят приближенный характер, как в случае обсуждаемого метода PBVI.

15.7 Библиографические примечания

ЭКСПЕРИМЕНТАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Тема выбора решений в условиях неопределённости интенсивно изучалась в статистике, где она известна под названием *экспериментальное проектирование*. Ключевыми учебниками в этой области являются книги Винера (Winer et al.,1971), Кирка и Кирка (Kirk and Kirk, 1995), и более новая работа Кона (Cohn, 1994).

Итерационный алгоритм, описанный в этой работе, восходит к Сондику (Sondik, 1971) и работе Смолвуда и Сондика (Smallwood and Sondik, 1973), которые одними из первых стали изучать проблему POMDP. Другую раннюю работу можно найти у Монохана (Monahan, 1982), а раннюю аппроксимацию на основе сетки - у Лавджоя (Lovejoy, 1991). Нахождение политик для POMDP долгое время было невозможным из-за чрезмерной требуемой вычислительной сложности. Проблема была представлена в области искусственного интеллекта Кэлблингом (Kaelbling et al, 1998). Алгоритмы отсечения в работах Кассандры (Cassandra et al., 1997) и Литтмана (Littman et al., 1995) позволили существенно улучшить предыдущие решения. В сочетании с заметным увеличением скорости и объёма памяти компьютеров, их работа позволила превратить POMDP в инструмент для решения небольших задач ИИ. Хоскрехт (Hauskrecht, 1997) представил границы сложности решения проблемы POMDP.

Наиболее значимый виток развития произошёл с появлением приближенных методов, некоторые из которых будут описаны в следующей главе. Улучшенная аппроксимация гипотез состояния POMDP по сетке была выведена Хоскрехтом (Hauskrecht, 2000), сетки с переменным разрешением были представлены Брафманом (Brafman, 1997). Анализ достижимости стал играть роль при вычислениях политик. Пун (Poon, 2001) и Жан и Жан (Zhang and Zhang, 2001) разработали методы POMDP на основе точек, с ограниченным набором гипотез состояния. В отличие от работы Хоскрехта (Hauskrecht, 2000), эти методы основаны на кусочно-линейной функции при выражении функции дохода. Эта работа получила развитие в определении точки на основе итерационного алгоритма Пиню (Pineau et al., 2003b), который разработал новые, не зависящие от времени методы нахождения релевантных пространств гипотез для решения POMDP. Их работа позже была расширена представлениями в виде деревьев (Pineau et al. 2003a).

Жеффнер и Боне (Geffner and Bonet, 1998) решили многие сложные проблемы, используя динамическое программирование на дискретном пространстве гипотез. Эта работа была обобщена Лихачевым (Likhachev et al., 2004), применившим алгоритм A* (Nilsson, 1982) к ограниченному кругу POMDP. Фергюсон (Ferguson et al., 2004) обобщил этот метод до D* плани-

рования в динамических средах (Stentz 1995).

В другом семействе методов для вычисления политик используются частицы, соединённые с ближайшим соседом в пространстве наборов частиц для определения аппроксимации функции ценности (Thrun 2000a). Частицы также были применены для алгоритмов мониторинга POMDP Паупартом (Poupart et al., 2001). Паупарт и Бутильер (Poupart and Boutilier, 2000) вывели алгоритм аппроксимации функции дохода, чувствительный к самому значению дохода, что дало выдающиеся результаты. Метод Дердена и Бутильера (Dearden and Boutilier, 1994) стал более эффективным повторяя такты планирования и выполнения в частичных политиках. Более подробно об этом можно прочесть в работах Смита и Симмонса (Smith and Simmons, 2004), посвящённым исследованиям перемежающихся эвристических методов планирования и выполнения на основе поиска. Использование отраслевых знаний было исследовано Пиню (Pineau et al., 2003c), а Вашингтон (Washington, 1997) представил инкрементные методы с границами. Дополнительная работа по решению приближенных алгоритмов POMDP обсуждается у Абердина (Aberdeen, 2002), Мерфи (Murphy, 2000b). Одним из немногих робототехнических систем, управляемых POMDP с итерационным алгоритмом, является CMU Nursebot, диалоговый менеджер и высокоуровневый контроллер которого основаны на POMDP (Pineau et al. 2003d; Roy et al. 2000).

Альтернативным подходом к нахождению политик управления POMDP является поиск прямо в пространстве политик, без вычисления функции дохода. Эта идея принадлежит Вилльямсу (Williams, 1992), который разработал идею градиентного поиска политики в контексте MDP. Современные методы градиентного поиска политики описаны в работах Бакстера (Baxter et al., 2001) и Нг и Джордана (Ng and Jordan, 2000). Багнелл и Шнайдер (Bagnell and Schneider, 2001) и Нг (Ng et al., 2003) успешно применили этот метод для управления автономным вертолётом в режиме висения. Фактически, Нг (Ng et al., 2003) отметил, что разработка контроллера на основе методов POMDP и обученной модели заняла всего 11 дней. В более новой работе Нг (Ng et al., 2004) применил эти методы для определения контроллера, способного поддерживать установившийся режим полёта вертолёта, вопрос, который был прежде нерешенным. Рой и Трун (Roy and Thrun, 2002) использовали методы поиска политики для навигации мобильного робота, и описали комбинирование поиска политики и методов итерации ценности.

Сравнительно небольшой прогресс был достигнут в обучающихся моделях POMDP. Ранние попытки обучить модель POMDP на основе взаимодействия со средой, в основном, провалились (Lin and Mitchell 1992; Chrisman 1992) из-за чрезмерной сложности задачи. Несколько более новых работ по обучению иерархических моделей показались многообещающими (Theocharous et al. 2001). В последних работах отошли от обучения HMM-подобных моделей в пользу альтернативных представлений. Методы выражения и обучения структуры частично наблюдаемых стохастических сред можно найти в работах Ягера (Jaeger, 2000), Литтмана (Littman et al., 2001), Джеймса и Сингха (James and Singh, 2004), Розенкранца (Rosencrantz et al., 2004). Хотя ни в одной из этих работ проблема POMDP до конца не была решена, они оказались интеллектуально связаны, и предоставили новые идеи к проблеме управления роботами, которая, в основном, остаётся открытой.

15.8 Упражнения

ЗАДАЧА ТИГРА

1. Эта задача известна как "задача тигра"и описана Кассандра, Литтманом и Кэлблингом (Cassandra et al., 1994). Человек находится перед двумя дверьми. За одной из них находится тигр, за другой - награда +10. Человек может либо прислушаться, либо открыть одну из дверей. Если открыть дверь комнаты с тигром, человек будет съеден, что в терминах задачи будет иметь стоимость -20. Стоимость "прислушивания1. В процессе прислушивания человек услышит рычащий звук, означающий присутствие тигра, но верно локализовать его сможет только с вероятностью 0,85. С вероятностью 0,15 ему будет казаться, что звук исходит из-за двери, за которой, в действительности, находится награда.

Вопросы:

(a) Описать формальную модель POMDP, в которой определить пространства состояний, действий и измерений, функцию стоимости и связанные вероятностные функции.

(b) Какова совокупная награда/стоимость для последовательности действий с открытым циклом: “Слушать, Слушать, Открыть дверь 1”? Объяснить вычисления.

(c) Какова совокупная награда/стоимость для последовательности действий с открытым циклом: “Слушать, затем открыть дверь, за которой не слышно шума”? Объяснить вычисления.

(d) Вручную выполнить одиночный обратный проход POMDP. Изобразить графики результирующих линейных функций на схеме, так же, как в разделе 15.2. Показать графики всех промежуточных шагов и нанести на оси единицы измерения.

(e) Вручную выполнить второй обратный проход, описав все с помощью графиков и вычислений.

(f) Реализовать задачу, вычислив решение для горизонтов планирования $T = 1, 2, ..., 8$. Убедиться в том, что была полностью выполнена обрезка пространства всех линейных функций. Для каких последовательностей измерений человек все ещё выберет "Слушать даже после 8 последовательных действий "Слушать"?

2. Показать правильность Выражения (15.26).

3. Какова будет, в наихудшем случае, вычислительная сложность для одного значения функции POMDP при обратном проходе? Предоставить ответ, используя нотацию $O()$, аргументами могут быть число линейных функций перед обратным проходом, количество состояний, действий и измерений в дискретном POMDP.

4. В литературе по POMDP часто используется коэффициент пересчёта, аналогичный фактору скидки из предыдущего раздела. Показать, что, даже при использовании коэффициента пересчёта, результирующие функции значимости все ещё кусочно-линейными.

5. Рассмотрим задачи POMDP с конечными пространствами состояний, действий, измерений, но с горизонтом $T \uparrow \infty$.

(a) Будет ли функция ценности все ещё кусочно-линейной?

(b) Будет ли она все ещё непрерывной?

(c) Останется ли она выпуклой?

Для всех трёх вопросов обосновать, почему это так, и привести пример, в котором это не так.

6. На странице 34 был представлен пример робота, который воспринимает и открывает дверь. В этом упражнении требуется реализовать алгоритм POMDP для оптимальной политики управления. Большую часть информации можно найти в примере на странице 34. Чтобы превратить ее в задачу управления, допустим, что у робота есть третье действие: **идти**. Когда он идёт, то получает награду +10 если дверь открыта, и -100 – если закрыта. Действие **идти** прерывает раунд. Действие **ничего_не_делать** стоит -1, а **толкнуть** -5. Изобразить функции ценности для разных временных горизонтов до $T = 10$, и объяснить оптимальную политику.

16 Приближенные методы POMDP

16.1 Цели

В предыдущих главах мы изучили два основных подхода для выбора действий в условиях неопределённости: MDP и POMDP. Оба подхода предназначены для случая с недетерминированными результатами действий, но различаются способностью обрабатывать ограничения датчиков. Только алгоритмы POMDP способны функционировать в условиях неопределённости восприятия, а в алгоритмах MDP подразумевается, что состояние полностью наблюдаемо. Однако, вычислительные затраты для точного планирования методами POMDP сводят на нет применимость практически для любых задач.

В этой главе описаны масштабируемые алгоритмы POMDP. Как мы увидим в этой главе, MDP и POMDP представляют собой крайние значения диапазона возможных алгоритмов вероятностного планирования и управления. В этой главе анализируются несколько приближенных методов POMDP, которые попадают в промежуток между MDP и POMDP. В описываемых алгоритмах, так же, как в POMDP используется итерационный алгоритм в пространстве гипотез. Однако. Функция дохода аппроксимируется несколько отличными способами, поэтому, они стали значительно быстрее, чем базовое решение POMDP.

Описываемые в этой главе методы были выбраны потому, что они характеризуют различные стили приближения. В частности, будут обсуждаться три следующих алгоритма:

• *QMDP* – это гибрид MDP и POMDP. Этот алгоритм обобщает MDP-оптимальную функцию дохода, определённую по состояниям, в POMDP-подобную функцию дохода по гипотезам. QMDP будет точным при (обычно,

неверном) допущении о том, что после каждого действия состояние становится полностью наблюдаемым. Итерационный алгоритм в QMDP имеет ту же вычислительную сложность, что и в MDP.

1: **Algorithm QMDP**$(b = (p_1, ..., p_N))$:
2: $\hat{V} =$ **MDP_discrete_value_iteration**()//см. стр. 458
3: *для всех действий управления и выполнить*
4: $Q(x_i, u) = r(x_i, u) + \sum_{j=1}^{N} \hat{V}(x_j) p(x_j | u, x_i)$
5: *endfor*
3: *return* $\underset{u}{\text{argmax}} \sum_{i=1}^{N} p_i Q(x_i, u)$

Таблица 16.1 Алгоритм QMDP вычисляет ожидаемый результат каждого действия управления u, а затем выбирает действие u с наивысшим доходом. Используемая функция дохода MDP-оптимальна, что нивелирует неопределённость состояния в POMDP.

• *Дополненный MDP (augmented MDP – AMDP).* Этот алгоритм проектирует пространство гипотез в необходимые статистики низкой размерности и выполняет итерационный алгоритм в этом пространстве низкой размерности. Самая базовая реализация включает представление, объединяющее самое вероятное состояние и степень неопределённости, выраженную энтропией. Таким образом, планирование лишь чуть менее эффективно по сравнению с MDP, но уже это является значительным улучшением!

• *Монте-Карло POMDP (Monte Carlo POMDP - MC-POMDP).* Эта версия алгоритма POMDP с многочастичным фильтром, где гипотезы аппроксимируются, используя частицы. Динамически конструируя набор точечных гипотез (точно, как в алгоритме PBVI, описанном ближе к концу предыдущей главы) MC-POMDP может сохранять относительно небольшой набор гипотез. Алгоритмы MC-POMDP применимы для состояний с непрерывными значениями состояний, действий и измерений, но подвержены тем же ограничениям, с которыми сталкиваются все многочастичные фильтры в книге, а также ещё нескольким дополнительным.

Эти алгоритмы позволяют проиллюстрировать основные методы для аппроксимации функций ценности, встречающиеся в обширной литературе по вероятностному планированию и управлению.

16.2 QMDP

QMDP – это попытка объединить лучшее в MDP и POMDP. Функции дохода легче вычислить для MDP, чем для POMDP, но MDP полагаются на допущение о полной наблюдаемости состояния. QMDP вычислительно практически так же эффективен, как MDP, но возвращает политику, определённую в пространстве гипотез.

Используемый математический «приём» достаточно прямолинеен. Алгоритм MDP, обсуждаемый в Главе 14, предоставляет функцию дохода на основе состояния, которая оптимальна при допущении полной наблюдаемости состояния. Результирующая функция дохода $\hat{V}$ определена по простран-

ствам состояний. QMDP обобщает это значение для пространства гипотез с помощью математического ожидания:

(16.1)
$$\hat{V}(b) = E_x[\hat{V}(x)] = \sum_{i=1}^{N} p_i \hat{V}(x_i)$$

Здесь используем уже знакомую запись $p_i = b(x_i)$. Таким образом, функция дохода линейна с параметрами

(16.2)
$$u_i = \hat{V}(x_i)$$

Эта линейная функция имеет в точности ту же самую форму, что и применяемая в итерационном алгоритме POMDP. Поэтому функция дохода в пространстве гипотез задана следующим линейным равенством:

(16.3)
$$\hat{V}(b) = \sum_{i=1}^{N} p_i u_i$$

Функция ценности MDP даёт одно линейное ограничение в пространстве гипотез, что позволяет использовать алгоритм **policy_POMDP** в Таблице 15.2 с одним линейным ограничением.

Самая базовая версия этой идеи ведёт к алгоритму QMDP, показанному в Таблице 16.1. Здесь используется чуть другая запись, чем приведённая в Таблице 15.2: Вместо кэширования одной линейной функции для каждого действия u и использования **policy_POMDP** для определения действия, наша формулировка QMDP напрямую вычисляет оптимальную функцию дохода с помощью функции Q. Значение $Q(x_i, u)$, вычисленное в строке 4 в Таблице 16.1 - это значение MDP действия управления u в состоянии x_i. Обобщение до гипотезы состояний выполняется в строке 6, где берётся ожидание по состоянию гипотезы. Строка 6 также выполняет максимизацию по всем действиям, и возвращает действие управления с самым большим значением ожидания.

Мысль о том, что MDP-оптимальная функция ценности может быть обобщена для пространства гипотез позволяет произвольно комбинировать обратные проходы MDP и POMDP. В частности, MDP-оптимальная функция дохода $\hat{V}$ может быть использована на входе алгоритма POMDP в Таблице 15.1 (страница 483). Для T дальнейших обратных проходов POMDP, результирующая политика может выполнять активный сбор информации, до тех пор, пока информация оказывается полезной для следующих T шагов. Даже для очень малых значений T, обычно получаются надёжные вероятностные алгоритмы, которые по вычислительной производительности сильно превосходят полное решение POMDP.

16.3 Дополненные марковские процессы принятия решений

16.3.1 Дополненное пространство состояний

Дополненный MDP или AMDP, является альтернативой алгоритму QMDP. Он также аппроксимирует функцию дохода POMDP. Однако, вместо игнорирования неопределённости состояния для малого горизонта времени T,

AMDP сжимает пространство состояния в более компактное представление, а затем выполняет полный вероятностный обратный проход как POMDP.

Фундаментальным допущением AMDP является то, что пространство гипотез может суммироваться в «достаточную» статистику с низкой размерностью f путём проектирования распределения гипотез в пространство более низкой размерности. Значения ценности и статистики вычисляются из статистики $f(b)$ вместо оригинальной гипотезы b. Чем более компактна статистика, тем более эффективен алгоритм итерации ценности.

Во многих ситуациях хорошим выбором статистики является кортеж

(16.4)
$$\bar{b} = \begin{pmatrix} \operatorname{argmax}_x b(x) \\ H_b(x) \end{pmatrix}$$

Здесь $\operatorname{argmax}_x b(x)$ является наиболее вероятным состоянием при распределении гипотез b, а

(16.5)
$$H_b(x) = -\int b(x) \log b(x) dx$$

ДОПОЛНЕННОЕ ПРОСТРАНСТВО СОСТОЯНИЙ

это *энтропия* гипотезы. Это пространство будем называть *дополненным пространством состояний*, поскольку оно дополняет пространство состояний единичным значением, энтропией распределения гипотезы. Вычисление функции ценности по дополненному пространству состояний, вместо пространства гипотез даёт огромное изменение сложное. Дополненное состояние позволяет избежать высокой размерности пространства гипотез, что позволяет сильно сэкономить при вычислении функции ценности (в худшем случае с дважды экспоненциальной до почти полиномиальной).

Выражение дополненного состояния математически разрешимо, если $f(b)$ является достаточной статистикой b по отношению к оценке ценности:

(16.6)
$$V(b) = V(f(b))$$

для всех гипотез b с которыми может столкнуться робот. На практике это допущение редко соблюдается. Однако, результирующая функция дохода может быть все ещё достаточно хороша для осознанного выбора действия управления.

Как альтернативный вариант, можно использовать другие статистики, например, моменты распределения гипотез (математическое ожидание, дисперсия, ...), множеству собственных значений и векторам ковариации, и так далее.

16.3.2 Алгоритм AMDP

Алгоритм AMDP выполняет итерацию ценности в дополненном пространстве состояний. Чтобы это сделать, необходимо преодолеть два препятствия. Во-первых, точное значение обновления для выражения дополненного состояния нелинейно. Это происходит потому, что энтропия является нелинейной функцией параметров гипотезы. Поэтому становится необходимым аппроксимировать обратный проход значимости. AMDP дискретизирует дополненное состояние, выражая функцию дохода $\hat{V}$ через поиск в таблице. Мы уже сталкивались с такой аппроксимацией при обсуждении

MDP. В AMDP эта таблица на одно измерение больше, чем таблица пространства состояний, используемая в MDP.

Второе препятствие возникает из-за перехода вероятностей и функции награды в дополненном пространстве состояний. Обычно заданы вероятности, например, модель движения $p(x'|u,x)$, модель измерения $p(z|x)$ и функция дохода $r(x,u)$. Но для итерационного алгоритма в дополненном пространстве состояний необходимо определить аналогичные функции для дополненного пространства состояний.

AMDP использует своего рода «хитрость» при конструировании необходимых функций. Она состоит в изучении переходных вероятностей и наград из имитационного моделирования. Обучающийся алгоритм основан на частотной статистике, которая подсчитывает, как часто дополненная статистика $\bar{b}$ переходит в другую гипотезу $\bar{b}'$ для управляющего воздействия u, и какую среднюю награду вызывает такой переход.

В Таблице 16.2 приведён базовый алгоритм **AMDP_value_iteration**, разбитый на две фазы. В первой фазе (строки 2–19), конструируется таблица переходных вероятностей $\hat{\mathcal{P}}$ для перехода из дополненного состояния $\bar{b}$ и управляющего действия u в возможное последующее дополненное состояние $\bar{b}'$. Также конструируется функция награды $\hat{\mathcal{R}}$ которая измеряет ожидаемую немедленную награду r, когда выбирается действие u в дополненном состоянии $\bar{b}$.

1: **Algorithm AMDP_value_iteration**() :
2: *для всех* $\bar{b}$ *выполнить* //обучение модели
3: *для всех* u *выполнить*
4: *для всех* $\bar{b}$ *выполнить* //инициализация модели
5: $\hat{\mathcal{P}}(\bar{b}, u, \bar{b}') = 0$
6: *endfor*
7: $\hat{\mathcal{R}}(\bar{b}, u) = 0$
8: *повторить* n *раз* //обучение модели
9: *генерировать* b *с* $f(b) = \bar{b}$
10: *выборка* $x \sim b(x)$ //выборка гипотез
11: *выборка* $x' \sim p(x'|u, x)$ //модель движения
12: *выборка* $z \sim p(z|x')$ //модель измерения
13: *вычислить* $b' = B(b, u, z)$ //байесовский фильтр
14: *вычислить* $\bar{b}' = f(b')$ //статистика гипотезы состояния
15: $\hat{\mathcal{P}}(\bar{b}, u, \bar{b}') = \hat{\mathcal{P}}(\bar{b}, u, \bar{b}') + \frac{1}{n}$ //обучение вероятностям перехода
16: $\hat{\mathcal{R}}(\bar{b}, u) = \hat{\mathcal{R}}(\bar{b}, u) + \frac{r(u,s)}{n}$ //обучение модели награды
17: *endrepeat*
18: *endfor*
19: *endfor*
20: *для всех* $\bar{b}$ //инициализация функции дохода
21: $\hat{V}(\bar{b}) = r_{\min}$
22: *endfor*
23: *повторять до сходимости* //итерационный алгоритм
24: *для всех* $\bar{b}$ *выполнять*
25: $\hat{V}(\bar{b}) = \gamma \max_u [\hat{\mathcal{R}}(u, \bar{b}) + \sum_{\bar{b}'} \hat{V}(\bar{b}')\hat{\mathcal{P}}(\bar{b}, u, \bar{b}')]$
26: *endfor*
27: *return* $\hat{V}, \hat{\mathcal{P}}, \hat{\mathcal{R}}$ //вернуть функцию ценности и модель

1: **Algorithm policy_AMDP**$(\hat{V}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, b)$:
2: $\bar{b} = f(b)$
3: *return* $\operatorname{argmax}_u [\hat{\mathcal{R}}(u, \bar{b}) + \sum_{\bar{b}'} \hat{V}(\bar{b}')\hat{\mathcal{P}}(\bar{b}, u, \bar{b}')]$

Таблица 16.2 Сверху: итерационный алгоритм для дополненного MDP. Снизу: алгоритм выбора управляющего действия.

Эти функции оцениваются с помощью процедуры выборки, в которой генерируется n значений для каждой комбинации $\bar{b}$ и u (строка 8). Для каждого из этих выборок по модели Монте-Карло, алгоритм сначала генерирует гипотезу b для каждого $f(b) = \bar{b}$. Этот шаг довольно необычен (фактически, не определён): В оригинальной модели AMDP, создатели алгоритма просто решили установить b в виде симметричного гауссиана, с параметрами, выбранными для соответствия $\hat{b}$. Затем, в алгоритме AMDP выполняется выборка по положению x, последующему положению x', и измерению z, все очевидными способами. Затем используется байесовский фильтр для генерации апостериорной гипотезы $B(b, u, z)$, для которой вычисляются до-

полненные статистики (строка 14). Таблицы $\hat{\mathcal{P}}$ и $\hat{\mathcal{R}}$ обновляются в строках 15 и 16, используя простой взвешенный подсчёт частоты (в случае награды) с настоящей наградой в данном элементе выборки по методу Монте-Карло.

Когда обучение закончено, алгоритм AMDP продолжается итерационным алгоритмом, реализованном в строках 20-26. Как обычно, функция дохода инициализируется большим отрицательным значением. Итерация уравнения обратного прохода в строке 25 даёт функцию дохода, определённую в дополненном пространстве состояний.

При использовании AMDP отслеживание состояния обычно имеет место в оригинальном пространстве гипотез. Например, при использовании AMDP для движения робота, можно использовать MCL для отслеживания гипотезы по положению робота. Алгоритм **policy_AMDP** в Таблице 16.2 каким образом извлекается действие политики из функции ценности AMDP. Извлекается выражение дополненного состояния из полной гипотезы в строке 2, а затем просто выбирается действие управления, которое максимизирует ожидаемую ценность (строка 3).

16.3.3 Математический вывод AMDP

Вывод AMDP относительно прямолинеен, допуская, что f является достаточной статистикой гипотезы состояния b. Подразумевается, что среда имеет марковское отношение к состоянию $f(b)$. Начнём с приближенной модификации обратного прохода POMDP в Выражении (15.2). Пусть f - функция, которая извлекает статистику $\bar{b}$ из b, поэтому $\bar{b} = f(b)$ для произвольных гипотез b. Допуская, что f является достаточной статистикой, выражение для итерационного алгоритма POMDP (15.2) может быть определено в пространстве состояний AMDP

(16.7)
$$V_T(\bar{b}) = \gamma \max_u \left[r(\bar{b}, u) + \int V_{T-1}(\bar{b}') p(\bar{b}'|u, \bar{b}) d\bar{b}' \right]$$

где $\bar{b}$ относится к статистике низкой размерности b определённой в (16.4). Здесь $V_{T-1}(\bar{b}')$ и $V_T(\bar{b})$ реализованы через поиск по таблицам.

Это выражение содержит вероятность $p(\bar{b}'|u, \bar{b})$, которая требует дальнейшего объяснения. А именно, имеется

(16.8)
$$\begin{aligned}
p(\bar{b}'|u, \bar{b}) &= \int p(\bar{b}'|u, b) p(b|\bar{b}) db \\
&= \iint p(\bar{b}'|z, u, b) p(z|b) p(b|\bar{b}) dz\, db \\
&= \iint p(\bar{b}' = f(B(b, u, z))) p(z|b) p(b|\bar{b}) dz\, db \\
&= \iint p(\bar{b}' = f(B(b, u, z))) \int p(z|x') \int p(x'|u, x) b(x) dx\, dx'\, dz\, p(b|\bar{b}) dz\, db
\end{aligned}$$

Эта модификация основана на факте того, что апостериорная гипотеза b' становится уникально определённой, если известны предыдущая гипотеза b, управление u, и измерение z. Аналогичная «хитрость» уже применялась в нашем выводе POMDP. Она позволяет заменить апостериорные распределения по гипотезам применением байесовского фильтра $B(b, u, z)$. В дополненном пространстве состояний можно заменить $p(\bar{b}'|z, u, b)$ точечным распределением с центром в $f(B(b, u, z))$.

Обучающийся алгоритм в Таблице 16.2 аппроксимирует это уравнение с помощью выборки методом Монте-Карло, выполняя замену каждого интеграла выборкой. Читателю может понадобиться некоторое время для понимания соответствия: каждый из вложенных интегралов в (16.8) напрямую проецируется в одну из выборок в Таблице 16.2.

В тех же самых строках можно вывести выражение ожидаемой награды $r(\bar{b}, u)$:

(16.9)

$$r(\bar{b}, u) = \int r(b, u) p(b|\bar{b}) db = \iint r(x, u) b(x) dx \, p(b|\bar{b}) db$$

И снова, алгоритм **AMDP_value_iteration** аппроксимирует интеграл с помощью выборки. Результирующая обученная функция награды выполняет просмотр $\mathcal{R}$. Обратный проход итерационного алгоритма в строках 20–26 алгоритма **AMDP_value_iteration**, в основном, идентичен выводу MDP.

Как отмечалось выше, эта аппроксимация методом Монте-Карло применима только для случая, когда $\bar{b}$ является достаточной статистикой для b, а система – марковской по отношению к $\bar{b}$. На практике это обычно не так, и правильная выборка в дополненных состояниях должна быть обусловлена прошлыми действиями и измерениями (математический кошмар!). В алгоритме AMDP это игнорируется, и выполняется генерация b, где $f(b) = \bar{b}$. В нашем примере выше используется симметричный гауссиан, с параметрами, соответствующими $\bar{b}$. Альтернативным вариантом, который выводится из оригинального алгоритма AMDP, но может быть математически более обоснован, будет использована имитация всех траекторий гипотез состояний на основе модели движений и измерений, используя последующие пары смоделированных гипотез состояния для обучения $\mathcal{P}$ и $\mathcal{R}$. Ниже, при обсуждении MC-POMDP, мы ещё столкнёмся с таким методом. MC-POMDP обходит эту проблему, используя моделирование для генерации применимых пар гипотез состояния.

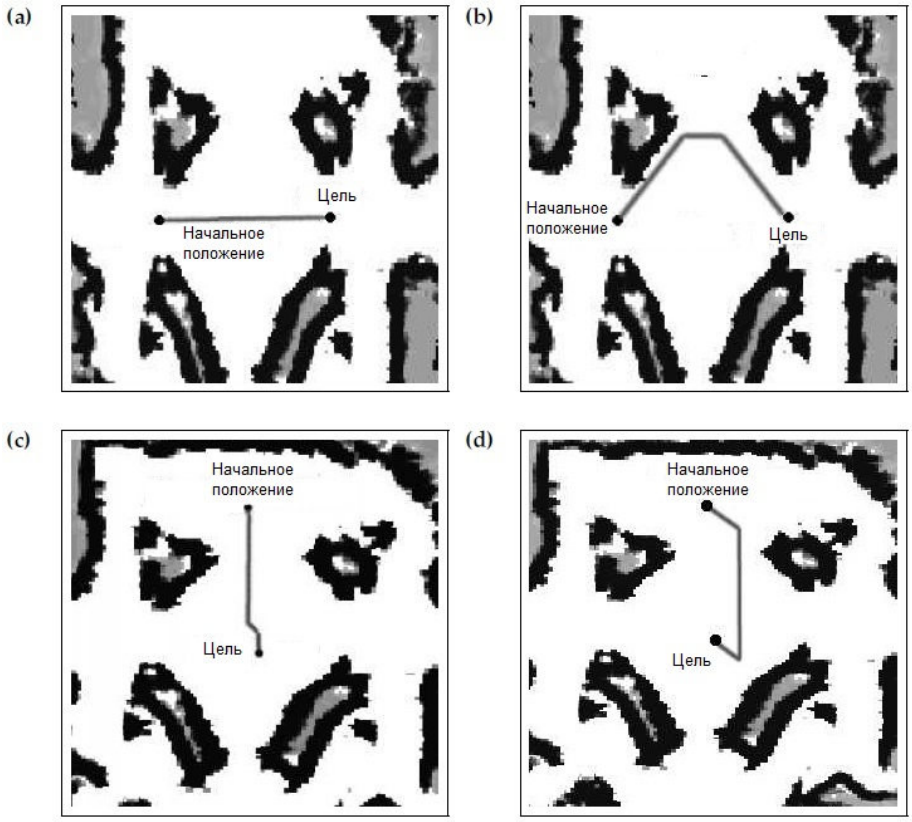


Рис. 16.1 Примеры пути робота в большой, открытой среде для двух разных конфигураций (верхний и нижний ряд). На схемах (a) и (c) показаны пути, сгенерированные обычным динамическим программирующим планировщиком пути, игнорирующем неопределённость восприятия робота. Схемы (b) и (d) получены с использованием дополненного MDP планировщика, который учитывает неопределённость и избегает области, где робот имеет большую вероятность заблудиться. Собственность Николаса Роя, МИТ (Nicholas Roy, MIT).

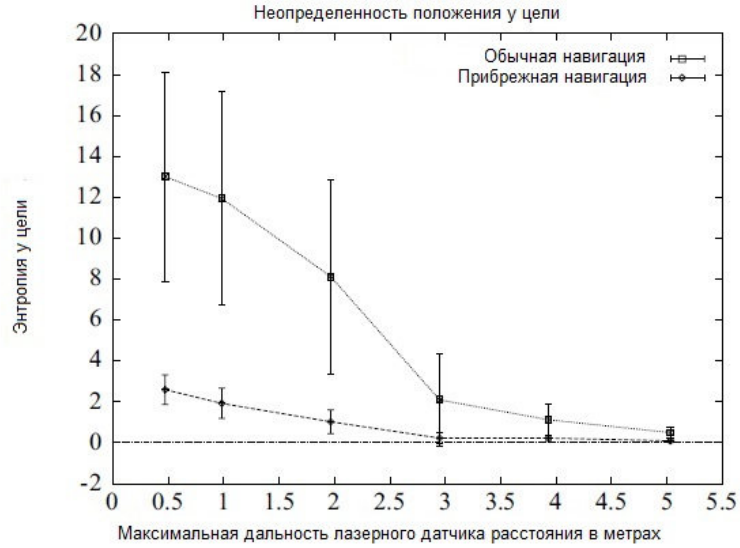


Рис. 16.2 Сравнение эффективности планирования MDP и дополненного планирования MDP. Здесь показана неопределённость (энтропия) целевого местоположения как функции дальности согласно показаниям датчика. Собственность Николаса Роя, МИТ (Nicholas Roy, MIT).

16.3.4 Применение в навигации мобильного робота

Алгоритм AMDP весьма практичен. В контексте навигации мобильного робота AMDP позволяет роботу учитывать общий уровень «замешательства» при выборе действий. Это относится не только к текущей мгновенной неопределённости, но и к будущей ожидаемой неопределённости, с которой робот может столкнуться при выборе действий.

Рассматриваемый пример включает навигацию робота в известной среде и уже приводился во вводной части книги, на Рис. 1.2 на странице 14. Очевидно, уровень «замешательства» зависит от того, где именно выполняется навигация. Робот проходит через большую зону, лишённую признаков, и постепенно утрачивает информацию о том, где именно он находится. Это отражено условной вероятностью $p(\bar{b}'|u, \bar{b})$, которая в таких зонах с высоким правдоподобием увеличивает энтропию гипотезы. В зонах с признаками, пригодными для локализации, то есть вблизи стен с различимыми признаками, неопределённость, скорее всего, уменьшится. AMDP обрабатывает такие ситуации и генерирует политики, минимизирующие время прибытия, с одновременной максимизацией определённости в момент прибытия к целевому местоположению. Поскольку неопределённость является оценкой ошибки по отношению к настоящему местоположению, она является хорошей мерой шансов того, что робот действительно прибудет к требуемой локации.

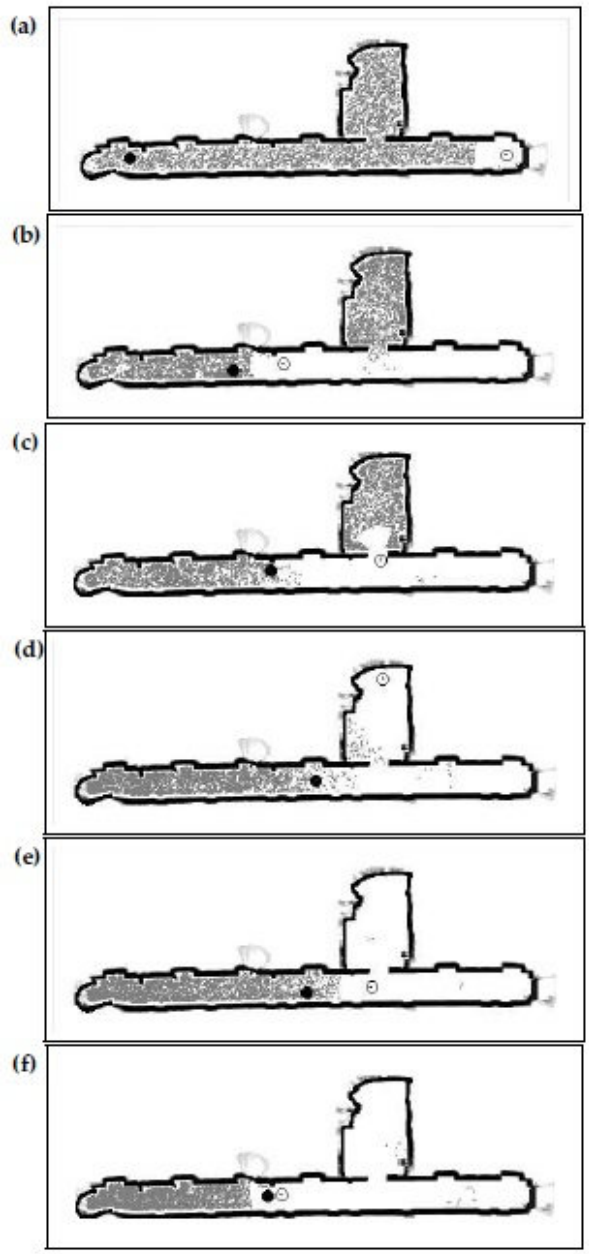


Рис. 16.3 Политика, вычисленная с помощью расширенной версии AMDP с обученным представлением состояния. Заданием является обнаружение нарушителя. Серые частицы извлекаются из распределения возможных местоположений человека, вначале равномерно распределенных по (a). Чёрная точка - истинное (ненаблюдаемое) положение человека. Открытый круг – наблюдаемое положение робота. Эта политика успешно работает с высокими значениями правдоподобия. Собственность Николаса Роя, МИТ и Джеффри Гордона, СМУ (Nicholas Roy, MIT and Geoffrey Gordon, CMU).

На Рис. 16.1 показаны примеры траекторий для двух групп (сочетаний начальных и конечных местоположений). Схемы слева соответствуют пла-

нировщику MDP, который не учитывает неопределённость робота. Дополненный планировщик MDP генерирует траектории наподобие показанных на фото. На Рис. 16.1a и b, от робота требуется продвинуться через большое открытое пространство, примерно 40 метров шириной. Алгоритм MDP, не учитывая возросший риск заблудиться на открытом пространстве, генерирует политику кратчайшего пути от точки старта до целевой локации. Планировщик AMDP, напротив, генерирует политику передвижения вблизи препятствий, где у робота есть большая вероятность получить информативные измерения датчиков ценой возросшего времени передвижения. Аналогично, на Рис. 16.1c и d показана ситуация, где целевая локация близка к центру открытой области без признаков. Здесь планировщик AMDP распознает, что проход около знакомых объектов уменьшает неопределённость положения, успешно увеличивая шансы достижения целевой локации.

На Рис. 16.2 показано сравнение эффективности между стратегией навигации AMDP и подходом MDP. В частности, показана энтропия гипотезы робота в целевой локации, как функции характеристик датчика. На этом графике варьируется максимальное расстояние восприятия для изучения воздействия ухудшения датчиков. Как показано на графе, у AMDP существенно более высокие шансы успеха и разница наиболее велика для плохих датчиков. Для датчиков с большой дальностью действия разница постепенно исчезает. Это неудивительно, поскольку при наличии хороших датчиков количество информации, которая может быть получена, меньше зависит от конкретного положения робота.

ПРИБРЕЖНАЯ НАВИГАЦИЯ

Способность учитывать неопределённость и избегать её привела к созданию так называемой *прибрежной навигации* в алгоритмах использования AMDP для навигации робота. Название указывает на схожесть с кораблями, которые, до изобретения спутниковой навигации, часто оставались вблизи береговой линии, чтобы не потерять возможность отслеживать своё местоположение.

Завершим обсуждение AMDP замечанием, что выбор статистики f, в некоторой степени, произволен. Признаки могут добавляться по необходимости, но это вызывает ожидаемый рост вычислительной сложности. Современные исследования позволили создать алгоритмы, которые обучаются статистикам f, используя нелинейные методы уменьшения размерности. На Рис. 16.3 показан результат такого обучающегося алгоритма, применяемого к задаче поиска движущегося нарушителя в здании. Здесь обучающийся алгоритм идентифицирует шестимерное представление состояния, отражающее гипотезу робота для любой применимой стратегии преследования. Серые частицы представляют собой гипотезы робота относительно местонахождения нарушителя. Как показано в этом примере, AMDP с обучением выражению состояния успешно генерирует довольно сложную стратегию: сначала робот осматривает часть коридора, оставаясь достаточно близко от комнаты, чтобы нарушитель не мог покинуть ее. Затем робот осматривает комнату за достаточно короткое время, чтобы нарушитель не сумел пройти по коридору незамеченным. Наконец, после осмотра комнаты, робот продолжает преследование по коридору.

16.4 Monte Carlo POMDP

16.4.1 Использование наборов частиц

Последний алгоритм, обсуждаемый в этой главе, это решение многочастичного фильтра для POMDP, называемое MC-POMDP, сокращение от "Monte Carlo POMDP". MC-POMDP находит функцию дохода, определённую на-

борами частиц. Пусть $\mathcal{X}$ будет набором частиц, выражающим гипотезу b. Тогда функция дохода может выражаться в следующем виде

(16.10)
$$V : \mathcal{X} \longrightarrow \Re$$

Это выражение имеет ряд преимуществ, но создаёт и затруднения. Ключевым преимуществом является возможность выражения функций ценности по произвольным пространствам состояний. Фактически, среди всех алгоритмов, которые обсуждались до сих пор, MC-POMDP – единственный, в котором не требуется конечное пространство состояний. Далее, в MC-POMDP для отслеживания гипотез используются многочастичные фильтры. Мы уже видели несколько примеров применения многочастичных фильтров. MC-POMDP расширяет использование многочастичных фильтров для задач планирования и управления.

Основная трудность использования наборов частиц в POMDP состоит в выражении функции дохода. Пространство всех наборов частиц любого данного размера M имеет размерность M. Более того, вероятность того, что какой-либо из наборов частиц будет наблюдаться дважды, равна нулю в силу стохастической природы генерации частиц. В результате, необходимо выражение для V, которое можно обновлять с помощью некоторого набора частиц, но затем представить значение для другого набора частиц, который алгоритм MC-POMDP ещё никогда не встречал. Другими словами, требуется *обучающийся алгоритм*. MC-POMDP использует алгоритм *ближайшего соседа*, используя локально взвешенную интерполяцию при интерполяции между разными гипотезами.

16.4.2 Алгоритм MC-POMDP

В Таблице 16.3 приводится базовый алгоритм MC-POMDP с использованием нескольких вложенных циклов. Самый внутренний цикл, в строках с 6 по 16 в Таблице 16.3, обновляет функцию дохода V для конкретной гипотезы $\mathcal{X}$. Это делается моделированием набора возможных последующих гипотез для каждого применимого действия управления u. Это моделирование выполняется в строках с 9 по 12. После этого собирается значение локального дохода для каждого из применимых действий (строка 13). Обновление функции дохода происходит в строке 16, где V просто устанавливается в значение, максимальное для всех Q_u.

```
1: Algorithm MC-POMDP(b_0, V) :
2:   повторять до сходимости
3:     выборка x ~ b(x)                          // инициализация
4:     инициализировать X with M выборки из b(x)
5:     повторять до конца эпизода
6:       для всего управления и выполнить//функция обновления дохода
7:         Q(u) = 0
8:         повторить n раз
9:           выбрать случайный x ∈ X
10:          выборка x' ~ p(x'|u, x)
11:          выборка z ~ p(z|x')
12:          X' = Particle_filter (X, u, z)
13:          Q(u) = Q(u) + (1/n) γ[r(x, u) + V(X')]
14:        endrepeat
15:      endfor
16:      V(X) = max_u Q(u)            //функция обновления дохода
17:      u* = argmax_u Q(u)           //выбор жадного действия
18:      выборка x' ~ p(x'|u, x)      //моделирование перехода состояния
19:      выборка z ~ p(z|x')
20:      X' = Particle_filter (X, u, z)  //вычислить новую гипотезу
21:      установить x = x'; X = X'       //обновление состояния и гипотезы
22:    endrepeat
23:  endrepeat
24:  return V
```

Таблица 16.3 Алгоритм MC-POMDP.

После этого выполняется локальный обратный проход в котором MC-POMDP моделирует физическую систему для генерации нового набора частиц $\mathcal{X}$. Это моделирование выполняется в строках с 17 по 21. В нашем примере при обновлении всегда выбирается жадное действие (строка 17), однако, на практике может оказаться выигрышнее выполнение случайного действия. С помощью перехода к новой гипотезе $\mathcal{X}$, в итерационном алгоритме MC-POMDP выполняется обновление на другую гипотезу состояния. Выполняя итерации для целых эпизодов (внешние циклы в строках со 2 по 5), функция дохода постепенно обновляется везде.

Ключевой открытый вопрос касается выражения функции V. MC-POMDP использует локальный обучающийся алгоритм, напоминающий алгоритм ближайших соседей. В нём конструируется набор эталонных гипотез $\mathcal{X}_i$ со связанными значениями V_i. При появлении запроса с прежде невидимым набором частиц $\mathcal{X}_{\text{query}}$, MC-POMDP идентифицирует K «ближайший» набор частиц в памяти. Определение подходящей концепции близости для наборов частиц требует дополнительных допущений. В оригинальной реализации, MC-POMDP сворачивает каждую частицу гауссовой функцией с малой, фиксированной ковариацией, а затем вычисляет KL-дивергенцию между результирующими смесями гауссиан. В двух словах, этот шаг даст возможность определить K ближайших эталонных наборов частиц $\mathcal{X}_1, ..., \mathcal{X}_K$, с соответствующими мерами расстояния, обозначаемыми $d_1, ..., d_K$ (заметим, что KL- дивергенция, технически, говоря, не расстояние, поскольку она ас-

симетрична). Ценность запроса набора частиц $\mathcal{X}_{\text{query}}$, получается согласно следующей формуле

(16.11)
$$V(\mathcal{X}_{\text{query}}) = \eta \sum_{k=1}^{K} \frac{1}{d_k} V_k$$

где $\eta = [\Sigma_k \frac{1}{d_k}]^{-1}$. Здесь $\mathcal{X}_k$ это k-ая эталонная гипотеза в наборе K ближайших соседей, и d_k –соответствующая дистанция до запрашиваемого набора.

МЕТОД ИНТЕРПОЛЯЦИИ ШЕПАРДА

Эта формула интерполяции, известная как *метод интерполяции Шепарда*, объясняет, как вычислить $V(\mathcal{X}')$ в строке 13 Таблицы 16.3.

Обновление в строке 16 включает неявную дифференциацию. Если эталонный набор уже содержит K наборов частиц, расстояние меньше определяемого пользователем порога, соответствующие V -значения обновляются просто пропорционально их вкладу в интерполяцию:

(16.12)
$$V_k \longleftarrow V_k + \alpha\, \eta\, \frac{1}{d_k} (\max_u Q(u) - V_k)$$

где α – скорость обучения. Выражение $max_u Q(u)$ является «целевым» значением для функции V, а $\eta \frac{1}{d_k}$ вклад k-й эталонной частицы набора в методе интерполяции Шепарда.

Если имеется меньше, чем K наборов частиц, расстояние которых падает ниже порогового значения, запрашиваемый набор частиц просто добавляется к эталонному, с ассоциированным значением $V = max_u Q(u)$. Таким образом, набор эталонных наборов частиц растёт со временем. Значение K и определяемый пользователем порог расстояния определяет гладкость функции ценности MC-POMDP. На практике, выбор соответствующих значений потребует некоторого осмысления, поскольку очень легко превысить объем памяти обычного персонального компьютера эталонным набором, если выбрать порог слишком плотно.

16.4.3 Математический вывод MC-POMDP

Алгоритм MC-POMDP основан на нескольких аппроксимациях: использование наборов частиц образует первое приближение. Вторая – это использование локального алгоритма обучения для выражения V, которое исключительно приближенное. Третья аппроксимация - использование обратного прохода по методу Монте-Карло функции ценности. Каждая из этих аппроксимаций ставит под угрозу сходимость базового алгоритма.

Математическая обоснованность использования многочастичных фильтров была уже описана в Главе 4. Шаг обновления по методу Монте-Карло следует из общего уравнения обновления POMDP (15.43) на странице 485, и повторно приводится ниже:

(16.13)
$$V_T(b) = \gamma \max_u \left[r(b,u) + \int V_{T-1}(B(b,u,z)) p(z|u,b) dz \right]$$

Аппроксимация по методу Монте-Карло выводится полностью аналогично выводу AMDP. Начнём с вероятности измерения $p(z|u,b)$, которая

разрешается следующим образом:

(16.14)
$$p(z|u,b) = \iint p(z|x')p(x'|u,x)b(x)dx\,dx'$$

Похожим образом, получаем для $r(b,u)$:

(16.15)
$$r(b,u) = \int r(x,u)b(x)dx$$

Это позволяет переписать (16.13) следующим образом:

(16.16)
$$\begin{aligned} V_T(b) &= \gamma \max_u [\int r(x,u)b(x)dx \\ &\quad + \int V_{T-1}(B(b,u,z))[\iint p(z|x')p(x'|u,x)b(x)dx\,dx']dz] \\ &= \gamma \max_u \iiint [r(x,u) + V_{T-1}(B(b,u,z))p(z|x')p(x'|u,x)]b(x)dx\,dx'\,dz \end{aligned}$$

Аппроксимация по методу Монте-Карло для этого интеграла сейчас представляет алгоритм выборки с несколькими переменными, что требует выполнения выборки $x \sim b(x), x' \sim p(x'|u,x)$ и $z \sim p(z|x')$. Как только мы получим x, x', и z, станет возможно вычислить $B(b,u,z)$ с помощью байесовского фильтра. Затем вычислим $V_{T-1}(B(b,u,z))$, используя локальный обучающийся алгоритм, а $r(x,u)$ – простым поиском. Заметим, что все эти шаги реализованы в строках с 7 по 14 Таблицы 16.3, а финальная максимизация выполняется в строке 16.

Локальный обучающийся алгоритм, который играет центральную роль в MC- POMDP, может легко нарушить любую сходимость алгоритма Монте-Карло. Не будем пытаться охарактеризовать условия, при которых локальное обучение может дать точную аппроксимацию, а, вместо этого, просто отметим необходимость соблюдения осторожности при установке различных параметров.

16.4.4 Практические соображения

Из трёх приближений POMDP, приведённых в этой главе, MC-POMDP разработано хуже всех и, потенциально, наименее эффективно. Его аппроксимация основана на обучающимся алгоритме для выражения функции дохода. Реализация алгоритма MC-POMDP может быть весьма затруднена. Требуется хорошее понимание гладкости функции ценности, а также, количества частиц, которое будет использоваться.

Оригинальная реализация алгоритма MC-POMDP даёт результат, показанный на Рис. 16.4. Робот, показанный на Рис. 16.4a, находится около объекта, который можно взять, и который можно обнаружить с помощью камеры. Однако, изначально объект расположен вне поля восприятия робота. Успешная политика, таким образом, будет включать три этапа. Этап поиска, во время которого робот поворачивается, пока не обнаружит объект, этап движения, во время которого робот центрует своё положение относительно объекта так, чтобы его можно было захватить, и финальное действие захвата. Комбинация активного восприятия и поведения, нацеленного на

достижение цели превращают это в относительно сложную вероятностную проблему управления.

На Рис. 16.4b показаны несколько примерных эпизодов, в которых робот успешно поворачивался, двигался и хватал объект. Показанные траектории проектируют движение в 2D. Численные результаты приведены на Рис. 16.4c, где показан коэффициент успешности как функция обновления итерационного алгоритма MC-POMDP. 4000 итераций обратного прохода параметра дохода потребовали примерно 2 часа вычислительного времени на маломощном персональном компьютере, а средняя эффективность алгоритма находится на уровне 80%. Оставшиеся 20% неудачных попыток относятся, в основном, к конфигурациям, в которых робот оказался неспособен определить своё положение для захвата объекта. Отчасти, это стало следствием множества аппроксимаций в MC-POMDP.

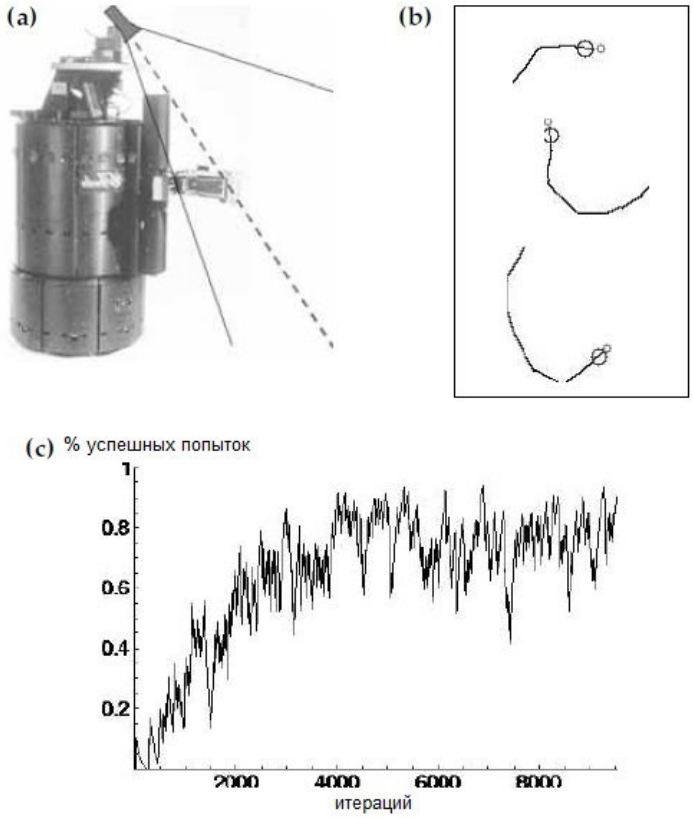


Рис. 16.4 Задание робота «найти и взять»: Мобильный робот с захватом и камерой, держащий целевой предмет(a). Представление на плоскости трёх успешных выполнений политики, в которых робот вращался, пока не увидит объект, а затем захватывал его(b). Коэффициент успешности, как функция количества шагов планирования, оценённая моделированием.(c)

16.5 Выводы

В этом разделе было представлено три приближенных вероятностных алгоритма планирования и управления, с различными показателями практической применимости. Все три алгоритма основаны на аппроксимации функции ценности POMDP, но различаются способом этой аппроксимации.

• Аппарат QMDP учитывает неопределённость только для одного выбора действия. Он основан на допущении, что после следующего действия управления, состояние среды внезапно становится наблюдаемым. Полная наблюдаемость делает возможным применение MDP-оптимальной функции дохода. QMDP обобщает функцию ценности MDP в пространства гипотез с помощью оператора математического ожидания. В результате, планирование в QMDP отличается от такового в MDP, но функция дохода, в общем, переоценивает настоящее значение гипотезы состояния.

• Обобщения алгоритма QMDP объединяют MDP-оптимальную функцию дохода с последовательностью обратных проходов POMDP. При комбинировании с T *шагами обратных проходов* POMDP, результирующая политика подразумевает действия сбора информации с горизонтом T, а затем полагается на допущение QMDP о полной наблюдаемости состояния. Чем больше горизонт T, тем ближе результирующая политика к полному решению POMDP.

• Алгоритм AMDP использует другую аппроксимацию. Он проецирует гипотезу в отображение более низкой размерности, по которой затем выполняется точный итерационный алгоритм. «Классическое» выражение состоит из наиболее вероятного состояния гипотезы, и энтропией гипотезы. В этом выражении, AMDP работают так же, как MDP с одним добавленным измерением в представлении состояния, измеряющим глобальное состояние неопределённости робота.

• Для реализации AMDP становится необходимым *обучить* переход состояний и функцию награды в пространстве гипотез низкой размерности. AMDP достигает этого в течение первой фазы, в которой статистики кэшируются в таблицы с возможностью поиска, выражая переход состояний и функцию дохода. Поэтому, AMDP работает по обученной модели, и обладает точностью, определяемой её точностью.

• Применение AMDP в навигации в известных средах называется *прибрежной навигацией*. В этом методе навигации учитывается неопределённость, и движение подбирается так, чтобы найти компромисс между общей длиной пути и неопределённости, накопленной вдоль траектории. Результирующие траектории существенно отличаются от решений без использования вероятности. Робот, выполняющий «прибрежную навигацию» избегает областей, в которых шансы заблудиться велики. Временная потеря ориентации допустима, если позже робот способен выполнить повторную локализацию с высокой вероятностью.

• Алгоритм MC-POMDP это версия POMDP с использованием многочастичного фильтра. В нем вычисляется функция дохода, определённая по наборам частиц. Для реализации такой функции дохода MC-POMDP необходимо использовать локальный обучающийся метод, использующий локальное взвешенное правило обучения в комбинации с проверкой на близость на основе KL-дивергенции. В MC-POMDP затем используется выборка методом Монте-Карло для реализации приблизительного обратного прохода итерационного алгоритма. Результирующий алгоритм представляет собой полноценный алгоритм POMDP, вычислительная точность и сложность которого являются функциями параметров обучающегося алгоритма.

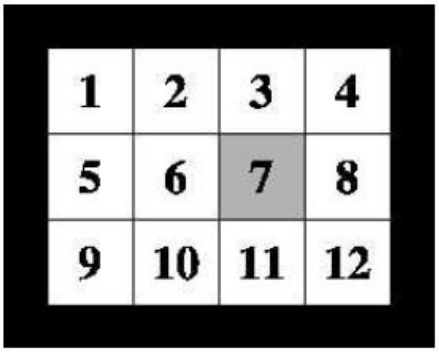
Ключевой урок, который можно вынести из этой главы, состоит в том, что существует несколько аппроксимаций, вычислительная сложность которых очень близка к MDP, но которые все ещё учитывают неопределённость состояния. Неважно, насколько груба аппроксимация, алгоритмы, которые учитывают неопределённость состояния, обычно существенно более надёжны по сравнению с алгоритмами, которые полностью ее игнорируют. Даже один новый элемент в векторе состояний, который измеряет глобальную неопределённость в одномерном виде, может создать огромную разницу в производительности робота.

16.6 Библиографические примечания

Литература по решению приближенной задачи POMDP подробно обсуждалась в предыдущей главе (15.7). Алгоритм QMDP, описанный в этой главе, принадлежит Литтману (Littman et al., 1995). Алгоритм AMDP для выражения фиксированного дополненного состояния был разработан Роем (Roy et al., 1999). Позже, Рой (Roy et al., 2004) обобщил его до выражения обученного состояния. Трун (Thrun, 2000a) вывел алгоритм Монте-Карло POMDP.

16.7 Упражнения

1. В этом упражнении требуется разработать AMDP, решающий простую задачу навигации. Рассмотрим следующую среду с 12 дискретными состояниями.



Вначале, робот помещён в некоторое случайное местоположение, равномерно выбранных из 12 состояний. Целью является достижение состояния 7. В любой момент времени робот перемещается на север, восток, запад или юг. Его единственный датчик - это бампер: при столкновении с препятствием, бампер срабатывает, и робот не меняет состояние. Робот не может воспринимать состояние, в котором находится, и не может определить, с какой стороны сработал бампер. В этой задаче нет шума и неопределённости начального положения (распределение будем считать равномерными).

(a) Какое минимальное количество состояний необходимо иметь AMDP? Описать все.

(b) Сколько из этих состояний достижимы из начального состояния AMDP? Описать все.

(c) Сейчас допустим, что робот начинает в состоянии 2 (которого он не знает, поэтому его внутренняя гипотеза другая). Нарисовать схему перехода между всеми состояниями AMDP, которые можно достичь за четыре действия.

(d) Для этого типа задачи (датчики и движение робота без шума, конечные пространства состояний, действий, измерений), можно ли придумать более компактное выражение по сравнению с AMDP, которого все ещё будет достаточным для нахождения оптимального решения?

(e) Для этого типа задачи (датчики и движение робота без шума, конечные пространства состояний, действий, измерений), можно ли сконструировать пространство состояний, для которого AMDP окажется неспособен найти оптимальное решение?

2. В предыдущей главе, мы узнали о *задаче тигра* (Упражнение 1 на странице 497). Какие изменения этой задачи позволят QMDP найти оптимальное решение? Подсказка: Возможных ответов несколько.

3. В этом упражнении требуется определить *размер* пространства гипотез состояний. Используем следующую таблицу:

Номер задачи	Количество состояний	датчики	Переход состояний	Начальное состояние
#1	3	совершенный	без шума	известно
#2	3	совершенный	зашумленный	известно
#3	3	без шума	без шума	неизвестно (равномерное)
#4	3	зашумленный	без шума	известно
#5	3	зашумленный	без шума	неизвестно (равномерное)
#6	3	нет	без шума	неизвестно (равномерное)
#7	3	нет	зашумленный	известно
#8	1-мерное непрерывное	совершенный	зашумленный	известно
#9	1-мерное непрерывное	зашумленный	зашумленный	известно
#10	2-мерное непрерывное	зашумленный	зашумленный	неизвестно (равномерное)

Совершенный датчик всегда даёт полную информацию о состоянии. Датчик без шума может предоставлять частичную информацию о состоянии, но делает это без элемента случайности. Зашумлённый датчик может не только давать частичную информацию, но и подвержен помехам. Переход состояний без шума детерминирован, а стохастический переход состояний оказывается зашумлённым. Наконец, будем различать только два типа начальных условий, в одном из которых начальное состояние известно с абсолютной уверенностью, а в другом – совершенно неизвестно и априорное распределение однородно.

Вопрос: Каков размер достижимого состояния гипотез для всех 10 задач?
Подсказка: Он может быть конечным или бесконечным, в бесконечном случае указать размерность пространства гипотез.

4. Требуется поразмышлять о критических режимах планировщика AMDP. В частности, AMDP *изучает* переход состояний и функции дохода. Поразмышлять о том, что может нарушить работу такой обученной модели, использующей итерационный алгоритм. Указать, по меньшей мере, три вида проблем, и детально их обсудить.

17 Исследование

17.1 Введение

Эта глава посвящена задаче исследования с помощью роботов. Исследование – это задача управления роботом таким образом, чтобы максимизировать его осведомленность об окружающем его мире. Во многих отраслях робототехники роботы изначально предназначены для доставки информации нам, пользователям. Некоторые среды могут быть просто недоступны для людей, в других посылать людей может быть невыгодно, и роботы представляют собой самый наилучший способ сбора информации. Задача исследования является одной из основополагающих в робототехнике. Роботы исследуют заброшенные шахты, места радиоактивных выбросов, и даже Марс.

Задачи исследования могут быть самыми разнообразными. Например, целью робота может быть составление карты статической среды. Если представить среду с помощью карты сетки занятости, задача исследования является задачей максимизации общего количества информации по каждой ячейке.

Более динамичная версия задачи включает движущихся актеров: например, робот пытается найти в известной среде движущегося человека, как часть *задачи ухода от преследования*. Целью может быть максимизация информации о местонахождении человека, а обнаружение человека может потребовать исследования среды. Однако, поскольку человек может двигаться, политикой исследования может быть предусмотрено повторное исследование областей несколько раз. Третья задача возникает, когда робот пытается определить своё местоположение в процессе локализации.

ЗАДАЧА УХОДА ОТ ПРЕСЛЕДОВАНИЯ

АКТИВНАЯ ЛОКАЛИЗАЦИЯ

Эта задача обычно называется *активной локализацией* , а ее целью является максимизация информации о собственном положении робота. При манипулировании предметами с помощью робота задача исследования возникает, когда перед манипулятором с датчиком возникает незнакомый предмет. Как показывает это короткое обсуждение, задачи исследования в робототехнике возникают практически повсеместно.

На первый взгляд, можно подумать, что задача исследования уже полностью охвачена подходом POMDP, обсуждаемом в предыдущих главах. Как было показано, POMDP изначально ориентирован на сбор информации. Для превращения POMDP в алгоритм, единственной целью которого является максимизация информации, все, что нужно – это соответствующая функция дохода. Подходящим выбором является усиление информации, измеряющее уменьшение энтропии гипотезы робота в виде функции

его действий. С такой функцией награды POMDP решает задачу исследования.

Однако, часто исследование с помощью алгоритма POMDP не такая уж хорошая идея. Это происходит потому, что во многих задачах количество неизвестных переменных состояния огромно, как и количество возможных измерений. Возьмём, для примера задачу исследования неизвестной планеты. Количество переменных, требуемых для описания поверхности планеты будет просто гигантским. Таким же будет и количество измерений, которые может совершить робот. Кака мы уже обнаружили, в общем алгоритме POMDP сложность планирования возрастает дважды экспоненциально относительно количества измерений (в худшем случае), и вычисление функции дохода просто невозможно. Фактически при столь огромном количестве *возможных* значений для неизвестных переменных состояния в исследовании, любой алгоритм, выполняющий интегрирование по всем возможным значениям, в конце концов, станет непригодным для задач исследования высокой размерности просто по причинам вычислительной сложности.

В этой главе описано семейство практических алгоритмов, способных решать проблемы исследования высокой размерности. Все обсуждаемые методы являются *жадными*. Другими словами, их горизонт ограничен следующим действием исследования. Однако, действие исследования может включать последовательность действий управления.

ДЕЙСТВИЕ ИССЛЕДОВАНИЯ

Например, будут обсуждаться алгоритмы, выбирающие местоположение для исследования на всей карте, при этом передвижение к точке считается одним *действием управления*. Обсуждаемые алгоритмы также аппроксимируют усиление информации из восприятия, в целях уменьшения вычислительных затрат.

Изложение организовано следующим образом

• Начнём с общего определения усиления информации в исследовании для дискретного и непрерывного случаев. Определим общий жадный алгоритм исследования, выбирающий действия так, чтобы максимизировать усиление информации.

• Затем проанализируем первый особый случай исследования с помощью робота: *активную локализацию*. При активной локализации выбор действий выполняется во время глобальной локализации робота. Применение такого жадного алгоритма исследования при соответствующем определении пространства действий, даст практический способ решения задачи.

• Также рассмотрим задачу исследования в построении сеток занятости. Мы выведем популярный метод, называемый *исследованием на основе границ (frontier-based exploration)*, в котором робот двигается в направлении ближайшей границы.

• Далее будет описано обобщение алгоритма исследования для систем из нескольких роботов и показано, каким образом можно управлять группой мобильных роботов для эффективного исследования неизвестной среды.

• Наконец, рассмотрим приложение методов исследования к задаче SLAM. На примере FastSLAM будет показано, каким образом можно управлять роботом, чтобы минимизировать неопределённость в SLAM.

Описанные ниже методы исследования широко освещены в литературе и

используются во множестве практических реализаций. Они также применимы для большого количества различных выражений и задач робототехники.

17.2 Основные алгоритмы исследования

17.2.1 Прирост информации

Информация – ключ к исследованию. Мы уже сталкивались с разнообразным использованием информации в вероятностной робототехнике.

ОЖИДАЕМАЯ ИФОРМАЦИЯ

В контексте исследования определим энтропию $H_p(x)$ вероятностного распределения p как ожидаемую информацию $E[-\log p]$

(17.1)

$$H_p(x) = -\int p(x) \log p(x) dx \quad \text{или} \quad -\sum_x p(x) \log p(x)$$

Энтропия уже кратко обсуждалась в разделе 2.2. Значение $H_p(x)$ максимально, если p представляет собой равномерное распределение. Она достигает минимума, когда p – точечное распределение. Однако, в некоторых непрерывных случаях, например, гауссовских, можно никогда не достичь полной уверенности.

УСЛОВНАЯ ЭНТРОПИЯ

Условная энтропия определяется как энтропия условного распределения. В исследовании мы стремимся минимизировать ожидаемую энтропию гипотезы после выполнения действия, потому, что она естественным образом зависит от измерения z и управляющего воздействия u, которое определяет переход состояния гипотезы.

Придерживаясь приведённой выше записи, обозначим $B(b, z, u)$ гипотезу после выполнения управляющего действия u и наблюдения z при имеющейся гипотезе b. Условная энтропия состояния x' после выполнения управляющего действия u и измерения z задана как

(17.2)

$$H_b(x'|z, u) = -\int B(b, z, u)(x') \log B(b, z, u)(x') dx'$$

Здесь $B(b, z, u)$ вычисляются, используя байесовский фильтр. В робототехнике выбор имеется только для управляющего действия u, а выбрать измерение z невозможно. Следовательно, во внимание принимается только условная энтропия управления u, а измерения отбрасываются:

(17.3)

$$\begin{aligned} H_b(x'|u) &\approx E_z[H_b(x'|z, u)] \\ &= \iint H_b(x'|z, u) p(z|x') p(x'|u, x) b(x) dz\, dx'\, dx \end{aligned}$$

Заметим, что это лишь приближение, поскольку в финальном выражении порядок суммирования и логарифмирования обратный. *Прирост информации* связанный с действием u гипотезы b задан разницей

ПРИРОСТ ИНФОРМАЦИИ

(17.4)

$$I_b(u) = H_p(x) - E_z[H_b(x'|z, u)]$$

17.2.2 Жадные методы

Планируемый прирост информации позволяет перефразировать задачу исследования как теоретическую проблему принятия решений похожую на рассматриваемые в предыдущих главах. В частности, пусть $r(x, u)$ будет стоимостью применения управляющего действия u в состоянии x. Здесь будем считать, что $r(x, u) < 0$ для сохранения целостности описания. Тогда оптимальное жадное исследование для гипотезы b максимизирует разницу между приростом информации и стоимостью, взвешенной коэффициентом α.

(17.5)

$$\pi(b) = \underset{u}{\operatorname{argmax}}\, \alpha \underbrace{(H_p(x) - E_z[H_b(x'|z, u)])}_{\text{expected information gain}} + \underbrace{\int r(x, u)b(x)dx}_{\text{expected costs}}$$

Коэффициент α соотносит информацию и стоимость выполнения действия u. Он определяет ценность информации для робота, а также цену, которую алгоритм готов платить в смысле стоимости получения информации.

Выражение (17.5) разрешается как

(17.6)

$$\begin{aligned}\pi(b) &= \underset{u}{\operatorname{argmax}} - \alpha\, E_z[H_b(x'|z, u)] + \int r(x, u)\, b(x)\, dx \\ &= \underset{u}{\operatorname{argmax}} \int [r(x, u) - \alpha \iint H_b(x'|z, u)p(z|x')p(x'|u, x)dz\, dx']\, b(x)\, dx\end{aligned}$$

В двух словах, для понимания полезности управляющего действия u требуется вычислить ожидаемую энтропию после выполнения u и наблюдения. Ожидаемая энтропия получается интегрированием по всем возможным измерениям z, которые может получить робот, умноженным на их вероятность. Она транслируется в показатель полезности с помощью константы α. Затем вычитается ожидаемая стоимость выполнения действия r.

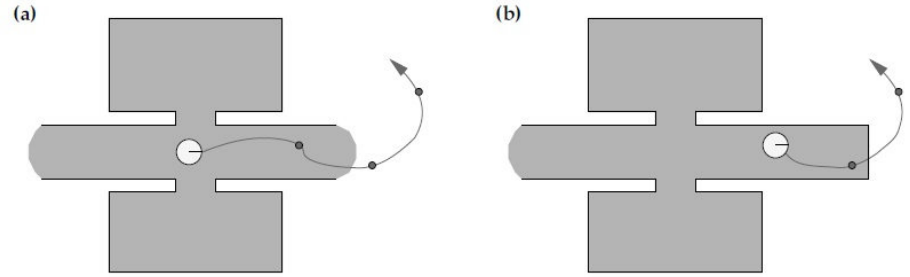


Рис. 17.1 Непредсказуемость задачи исследования: Робот на схеме (а) может выполнить последовательность трёх управляющих действий, но их исполнимость зависит от ситуации, которую ему только предстоит обнаружить на пути. Любая политика исследования должна иметь возможность реагировать различными способами.

В большинстве методов исследования используется подобная жадная политика, которая, действительно, оптимальна для горизонта 1. Причина такой приверженности жадным методам состоит в чудовищном количестве разветвлений в процессе исследования, что делает планирование на несколько шагов вперёд невозможным. Коэффициент разветвления так велик в силу самой природы проблемы исследования, ведь его целью является получение новой информации, но как только информация была получена, робот оказывается с новой гипотезой состояния, и вынужден перестраивать политику. Поэтому, измерения изначально непредсказуемы.

На Рис. 17.1 показана такая ситуация. Здесь робот выполнил картографирования двух комнат и части коридора. В этой точке оптимальное исследование может включать исследование коридора, а соответствующая последовательность действий может совпадать с показанной на Рис. 17.1a. Однако, исполнима эта последовательность или нет – остаётся совершенно непредсказуемым. Например, робот может обнаружить, что оказался в тупике, как показано на Рис. 17.1b, и выбранная последовательность действий неприменима.

17.2.3 Исследование методом Монте-Карло

Алгоритм **Monte_Carlo_exploration** – это простой вероятностный алгоритм исследования. В Таблице 17.1 приводится аппроксимация жадного алгоритма исследования в выражении (17.6) методом Монте-Карло. Этот алгоритм просто заменяет интегралы в жадном методе выборкой. В строке 4 выполняется выборка состояния x из текущей гипотезы b, за которой следует выборка следующего состояния x' и соответствующее наблюдение z'. Затем новая апостериорная гипотеза вычисляется в строке 8, а компромисс энтропия-стоимость кэшируется в строке 9. В строке 12 возвращается действие с наибольшей величиной прироста показателя "информация-стоимость" по методу Монте-Карло.

1: **Algorithm Monte_Carlo_exploration**(b) :
2: *установить* $\rho_u = 0$ *для всех действий* u
3: *для* $i = 1$ *до* N *выполнять*
4: *выборка* $x \sim b(x)$
5: *для всех действий управления* u *выполнить*
6: *выборка* $x' \sim p(x'|u, x)$
7: *выборка* $z \sim p(z|x')$
8: $b' = \textbf{Bayes\_filter}\,(b, z, u)$
9: $\rho_u = \rho_u + r(x, u) - \alpha H_{b'}(x')$
10: *endfor*
11: *endfor*
12: *return* $\underset{u}{\text{argmax}}\ \rho_u$

Таблица 17.1 Реализация жадного алгоритма исследования методом Монте-Карло. Действия выбираются максимизацией компромисса между приростом информации и стоимостью.

В общем случае жадный алгоритм Монте-Карло все ещё может требовать значительного времени на выполнение, что может привести к невоз-

можности использования. Основная сложность возникает при выполнении выборки измерений z. При исследовании неизвестной среды в процессе картографирования количество возможных измерений может быть огромным. Например, для робота, оборудованного 24 ультразвуковыми датчиками, каждый из которых передаёт один байт данных расстояния, количество потенциальных сканирований сонара, полученных в конкретном местоположении, составляет 256^{24}. Очевидно, не все эти измерения применимы, но количество применимых измерений, по меньшей мере, так же велико, как число применимых локальных карт. А для любой реалистичной проблемы картографирования количество возможных карт просто беспредельно! Ниже будут обсуждаться методы исследования, которые обходят это интегрирование с помощью анализа ожидаемого прироста информации в закрытом виде.

17.2.4 Многоступенчатые методы

В ситуациях, когда пространства состояний и измерений ограничены, может оказаться возможным обобщить принцип сбора информации для произвольного коечного горизонта $T > 1$. Допустим, необходимо оптимизировать баланс между приростом информации и стоимостью для горизонта T. Это можно выполнить, определив следующую функцию дохода при исследовании:

(17.7)
$$r_{\text{exp}}(b_t, u_t) = \begin{cases} \int r(x_t, u_t) b(x_t) dx_t & \text{при } t < T \\ \alpha\, H_{b_t}(x_t) & \text{при } t = T \end{cases}$$

Для этой функции планировщик POMDP находит политику управления, которая минимизирует энтропию финальной гипотезы b_T за вычетом стоимости достижения этой гипотезы в соответствующем масштабе. В данном случае применимы все обсуждаемые методы POMDP.

Читатель может заметить сходство с дополненным алгоритма MDP, обсуждаемом в предыдущей главе. Разница состоит в том, что здесь определяется только функция дохода, но не выражение гипотез состояния. Поскольку большинство задач исследования становятся вычислительно невыполнимыми для общей модели POMDP, этот подход далее обсуждаться не будет.

17.3 Активная локализация

Простейший случай исследования происходит при оценке состояния переменной с низкой размерностью. Такой задачей является *активная локализация*: здесь выполняется поиск информации о положении робота x_t, но дана карта среды. Активная локализация особенно интересна при глобальной локализации, поскольку в ней выбор управления может иметь огромное воздействие на прирост информации. Во многих средах бесцельное перемещение по окрестностям затрудняет глобальную локализацию, а передвижение к верному местоположению позволяет выполнить ее очень быстро.

Такая среда показана на Рис. 17.2a. Здесь робот помещается в симметричный коридор, что делает невозможным определение положения сколь долго он перемещается по нему. Единственным способом оказывается действия ухода из коридора через одну из открытых дверей. Таким образом, любое решение задачи активной локализации должно обязательно направить робота из коридора в одну из комнат.

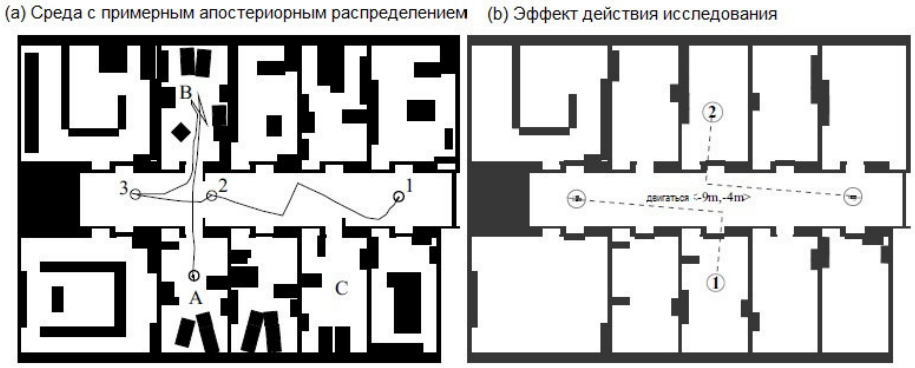


Рис. 17.2 Активная локализация в симметричной среде: Здесь показана среда с симметричным коридором, но ассиметричной обстановкой комнат, обозначенных A, B и C. На рисунке также показан путь исследования(a). Пример действия исследования «пройти назад 9 метров, налево 4 метра». Если апостериорное положение робота имеет две различимые моды, как показано на картинке, выполнение управляющего действия в глобальных координатах среды может привести в два разных места.(b)

Активная локализация может быть решена жадным методом с помощью только что представленного алгоритма. Ключевой идеей является выбор выражения действия. Очевидно, что, если действия определять как низкоуровневые, как в большей части книги, любой разумный план разрешения неопределенности положения путём исследования будет состоять из длинной цепочки управляющих действий. Для решения задачи активной локализации с помощью жадного исследования требуется определение действий, с помощью которых робот способен выполнять сбор информации.

Одним из возможных решений является определение *действия исследования* через целевые местоположения, выраженные в локальной системе отсчёта робота. Например, действием может считаться *"двигаться в точку* $\Delta x = -9$ *м и* $\Delta y = 4$ *м относительно текущего местоположения в локальной системе отсчёта"*, до тех пор, пока имеется низкоуровневый модуль навигации, способный проецировать это действие в низкоуровневые сигналы управления. На Рис. 17.2b показан потенциальный эффект такого действия в глобальной системе координат. В этом примере апостериорное распределение имеет две моды, поэтому действие может перенести робота в два разных местоположения.

Определение действий относительного передвижения позволяет решить задачу активной локализации с помощью алгоритма, практически аналогичного жадному алгоритму исследования в Таблице 17.1. Опишем его на примере. На Рис. 17.3a показан путь активной локализации и несколько помеченных местоположений. Начнём с середины: на Рис. 17.3b показана гипотеза после передвижения из местоположения, отмеченного "1" в местоположение с меткой "2". В этой гипотезе имеется шесть мод, каждая обозначена кружком на Рис. 17.3b. Для этой гипотезы ожидаемая занятость по координатам робота показана на Рис. 17.3c. Этот рисунок получен сложением известной карты сетки занятости для каждого из положений робота, взвешенных соответствующими вероятностями. Поскольку робот не имеет чёткого понимания своего положения, знать занятости местоположений он не может, отсюда и «нечёткость» карты ожидаемой стоимости. Однако, с большой вероятностью робот находится в зоне, напоминающей коридор, и

она проходима.

Хотя на Рис. 17.3c показана стоимость *нахождения* в целевом местоположении, необходима стоимость *передвижения* в такое целевое местоположение. Алгоритм для вычисления такой стоимости передвижения уже приводился, как и вычисление оптимального пути: *итерация согласно критерию* (см. Главу 14). На Рис. 17.3d показан результат итерационного алгоритма, применённый к карте на Рис. 17.3b в виде функции стоимости. Здесь значение распространяется от робота (противоположно цели, как было в записи в Главе 14). Это делает возможным вычисление стоимости передвижения в *любую* потенциальную целевую точку карты.

Как показано на рисунке, существует большая зона около робота, где перемещение безопасно. Фактически, эта область соответствует коридору, вне зависимости от того, где действительно находится робот. Перемещение за границы этой области и в одну из комнат имеет более высокую ожидаемую стоимость, поскольку применимость такого движения основана на знании точного местоположения робота, которое неизвестно.

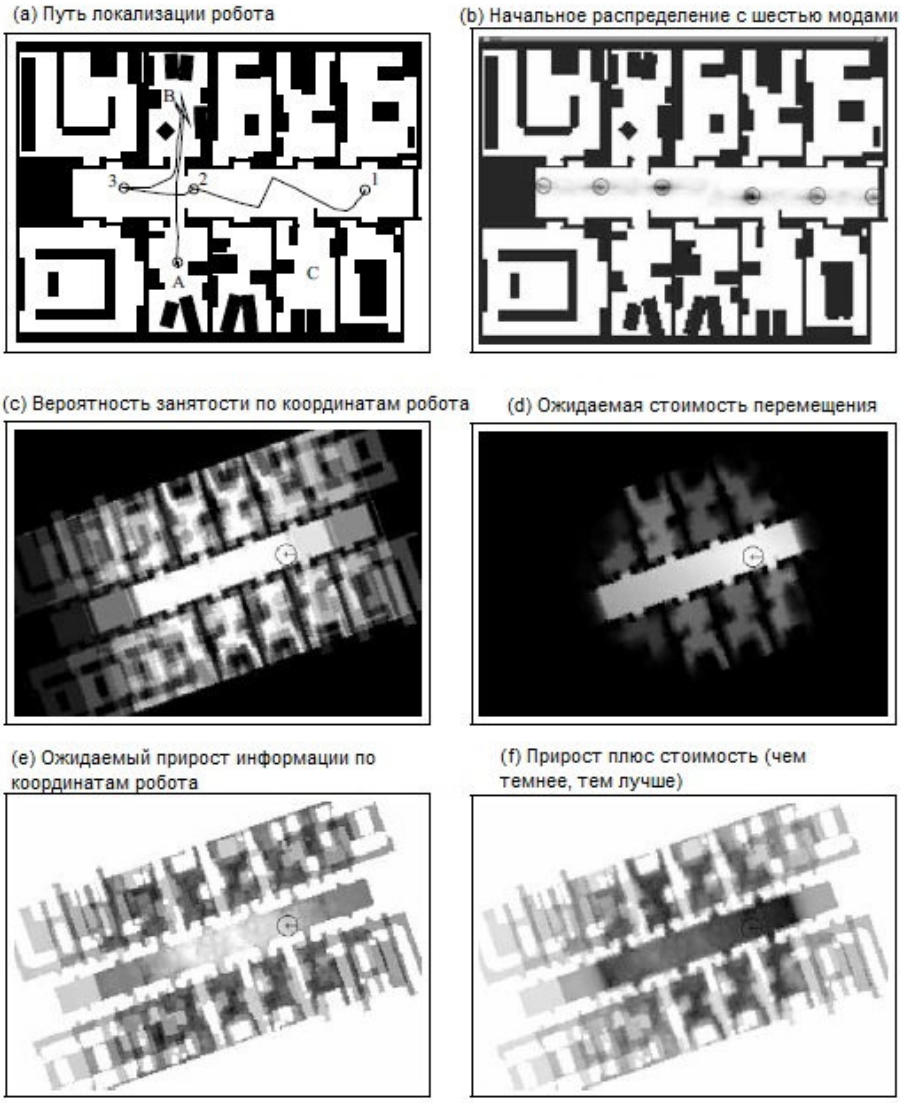


Рис. 17.3 Иллюстрация активной локализации. На рисунке показаны несколько вспомогательных функций для вычисления оптимального действия в распределении положений с несколькими гипотезами.

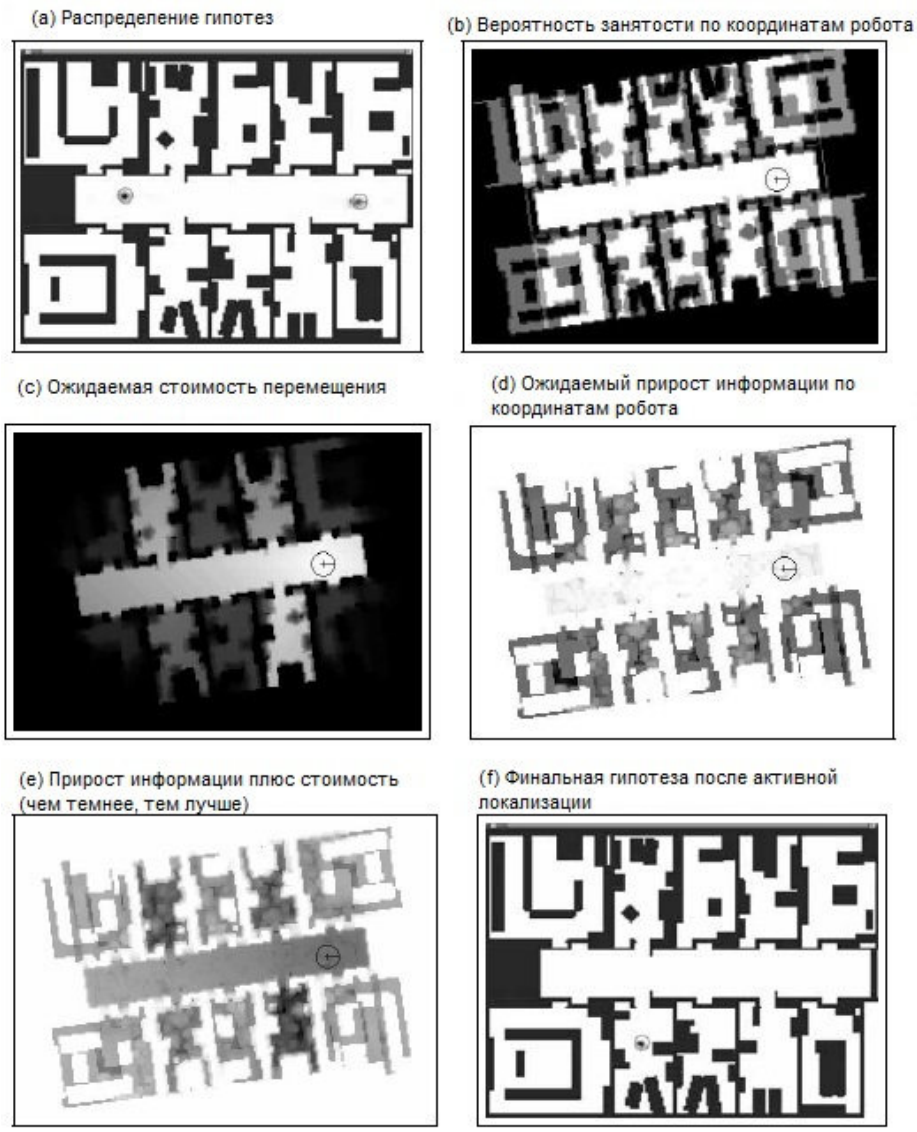


Рис. 17.4 Иллюстрация активной локализации в более поздний момент времени для гипотезы с двумя выраженными модами.

Для жадного исследования теперь требуется определить ожидаемый прирост информации. Его можно аппроксимировать, поместив робота в какое-либо положение, смоделировав возможные измерения расстояния, учтя результат, и измерив количество информации после байесовского обновления. Повторение этого этапа оценки для каждого возможного местоположения даст карту ожидаемого прироста информации. На Рис. 17.3e показан результат: чем темнее местоположение на карте, тем больше информации оно предоставляет, относительно оценки положения робота. Очевидно, любая из комнат будет наиболее информативна, по сравнению с торцами коридора. Поэтому, исходя их чистой перспективы прироста информации, роботу необходимо искать возможность переместиться в одну из комнат.

Добавление карты стоимости к ожидаемому приросту информации даст график на Рис. 17.3f: чем темнее целевое местоположение, тем лучше. Хотя

показатели внешних комнат для комбинированной функции тоже высоки, их полезность уменьшена относительно высокой стоимостью передвижения в них. В этой точке, местоположения в концах коридора имеют наибольшую полезность.

Теперь робот перемещается в местоположение с наивысшим комбинированным значением, что приводит его во внешнюю часть коридора, куда все ещё безопасно перемещаться. На Рис. 17.3a, это соответствует перемещению из местоположения, отмеченного “2”, к точке, отмеченной “3”.

Выполняется следующая итерация жадного алгоритма исследования с помощью активной локализации. Гипотеза в местоположении “3” показана на Рис. 17.4a. Очевидно, предыдущее действие исследования позволило сократить количество мод апостериорного распределения с шести до двух. На Рис. 17.4b показана новая карта занятости в локальных координатах робота. На Рис. 17.4c показана соответствующая функция дохода. Ожидаемый прирост информации теперь равномерно высок только в комнатах, как показано на Рис. 17.4d. На Рис. 17.4e приведена комбинированная карта прироста стоимости. В этой точке стоимость передвижения в любую из симметричных открытых комнат самая низкая, поскольку робот переместился в точку, в которой неопределённость положения, по большей части, разрешена. Финальная гипотеза после ещё одного шага и ещё одного уточнения показана на Рис. 17.4f.

Этот жадный алгоритм активной локализации имеет и недостатки. Первый недостаток возникает в силу самого подхода на основе жадности, поскольку он неспособен объединять несколько действий исследования, которые вместе максимизируют прирост информации. Другой недостаток – результат определения действия. Алгоритм не может учитывать измерения, которые будут получены на пути к целевому местоположению. Вместо этого, каждое такое действие воспринимается как открытый цикл, в котором робот неспособен реагировать на измерения. Конечно, встретив закрытую дверь, настоящий робот может отменить целевую точку даже до момента ее достижения. Однако при планировании этого не делается. Этим объясняется почти оптимальный выбор в примере выше, где робот исследует комнату, отмеченную “B” до того, как исследует комнату с меткой “A.” В результате, локализация занимает больше времени, чем это, теоретически, необходимо. В любом случае, алгоритм хорошо работает на практике.

17.4 Исследование для обучающихся карт сетки занятости

17.4.1 Вычисление прироста информации

Жадное исследование может также использоваться в картографировании с помощью роботов. Задачи построения карт включают намного больше неизвестных переменных, чем локализация робота, поскольку требуется метод вычисления ожидаемого прироста информации, масштабируемый для задач оценки высокой размерности. Как мы увидим, «хитрость» для карт сеток занятости ровно такая же, что привела к определению эффективного алгоритма обновления для карт сетки занятости. Нужно считать, что прирост информации между разными ячейками сетки *независим*.

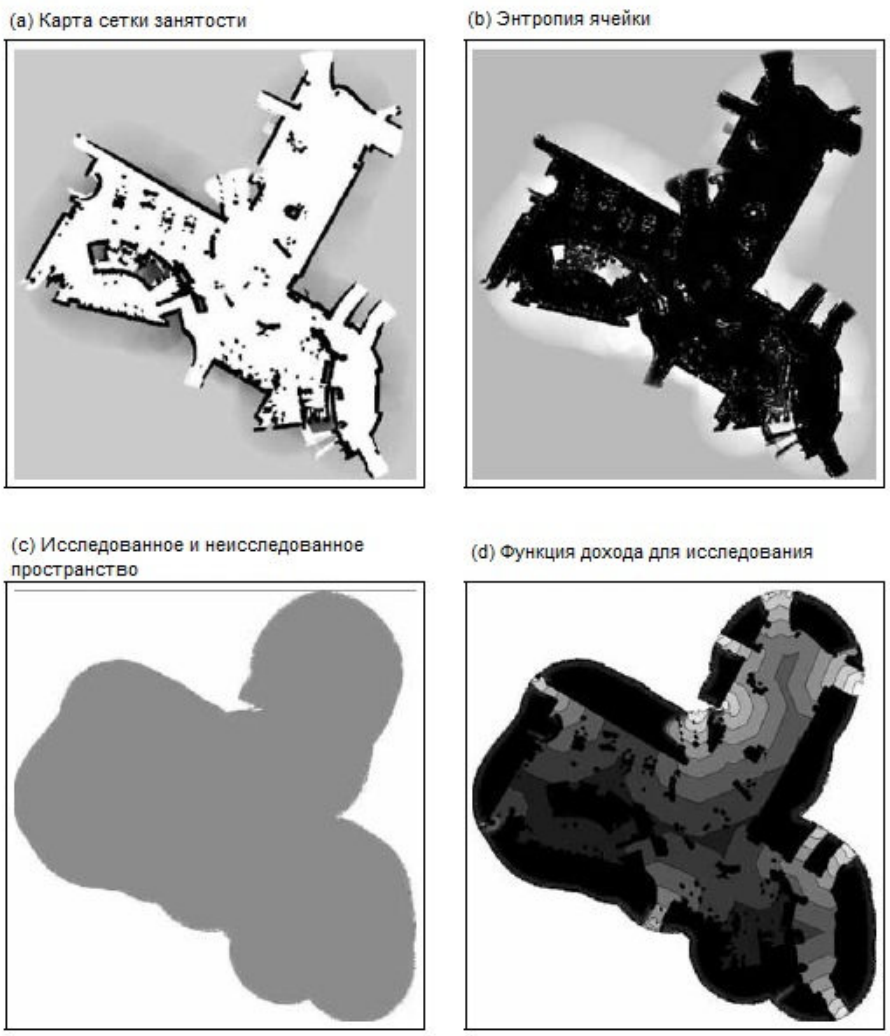


Рис. 17.5 Пример важных шагов в исследовании для построения карт. Показана частичная карта сетки(a), показана энтропия карты (b), показано пространство, для которой информация нулевая (c), и показана функция ценности для оптимального исследования(d).

Рис. 17.6 Карта, энтропия и ожидаемый прирост информации. На рисунке показано соответствующее масштабирование, энтропия и ожидаемый прирост информации почти неразличимы.

Возьмём карту сетки занятости, наподобие показанной на Рис. 17.5a. Некоторые части этой карты хорошо исследованы, например, большая свободная зона в центре карты и многие стены и препятствия, расположение которых хорошо известно. Другие части остаются не исследованными, например, большая серая зона за пределами карты. Жадное исследование направляет робота к ближайшей неисследованной зоне, где прирост информации максимален. Это поднимает вопрос о способе вычисления прироста информации.

Обсудим три возможных метода. Все три подхода имеют общую черту, что вычисляется прирост информации *для ячейки сети*, а не функцию действия робота. Это даёт карту прироста информации, которая представляет собой двухмерную карту, определённую по той же сетке, что и оригинальная сетка карты. Разница между этими методами состоит в качестве аппроксимации.

• **Энтропия.** Вычисление энтропии для каждой ячейки очевидно. Обозначим i-ю ячейку как m_i, а вероятность занятости $p_i = p(m_i)$. Тогда энтропия бинарной переменной занятости задана следующей суммой:

(17.8)
$$H_p(\mathbf{m}_i) = -p_i \log p_i - (1 - p_i) \log(1 - p_i)$$

На Рис. 17.5b показана энтропия для каждой из ячеек на карте, показанной на Рис. 17.5a. Чем светлее местоположение, тем выше энтропия. Большая часть центральных областей карты показывается с низкой энтропией, за исключением нескольких зон около или внутри препятствий. Это соответствует интуитивному пониманию, так как большая часть карты уже хорошо изучена. Энтропия внешних областей высокая, что указывает на высокую пригодность для исследования. Поэтому, карта энтропии назначает высокие значения ценности для мест, которые остались неисследованными.

• **Ожидаемый прирост информации**. Технически, энтропия способна измерить только текущую информацию. Она не обозначает информацию, которую робот может получить с помощью датчиков, в непосредственной видимости от ячейки. Вычисление ожидаемого прироста информации чуть более сложно, и требует дополнительных допущений относительно природы информации, приходящей с датчиков робота.

Допустим, датчик даёт измерение верной занятости с вероятностью p_{true}, и ошибается с вероятностью $1 - p_{true}$. Тогда можно ожидать измерения «за-

нято» со следующей вероятностью:

(17.9)
$$p^+ = p_{true}p_i + (1 - p_{true})(1 - p_i)$$

стандартное обновление сетки занятости даст новую вероятность, которая напрямую следует из алгоритма картографирования сетки занятости, обсуждаемого в Главе 9:

(17.10)
$$p'_i = \frac{p_{true}p_i}{p_{true}p_i + (1 - p_{true})(1 - p_i)}$$

Теперь энтропия апостериорного распределения:

(17.11)
$$\begin{aligned} H^+_{p'}(\mathbf{m}_i) = &-\frac{p_{true}p_i}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \log \frac{p_{true}p_i}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \\ &- \frac{(1 - p_{true})(1 - p_i)}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \log \frac{(1 - p_{true})(1 - p_i)}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \end{aligned}$$

По аналогии, датчик покажет «свободно» с вероятностью

(17.12)
$$p^- = p_{true}(1 - p_i) + (1 - p_{true})p_i$$

в этом случае апостериорное распределение станет

(17.13)
$$p'_i = \frac{(1 - p_{true})p_i}{p_{true}(1 - p_i) + (1 - p_{true})p_i}$$

Это апостериорное распределение имеет энтропию

(17.14)
$$\begin{aligned} H^-_{p'}(\mathbf{m}_i) = &-\frac{(1 - p_{true})p_i}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \log \frac{(1 - p_{true})p_i}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \\ &- \frac{p_{true}(1 - p_i)}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \log \frac{p_{true}(1 - p_i)}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \end{aligned}$$

сводя все предыдущие выражения вместе, получим ожидаемую энтропию после обновления:

(17.15)
$$\begin{aligned} E[H_{p'}(\mathbf{m}_i)] &= p^+ H^+_{p'}(\mathbf{m}_i) + p^- H^-_{p'}(\mathbf{m}_i) \\ &= -p_{true}p_i \log \frac{p_{true}p_i}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \\ &\quad - (1 - p_{true})(1 - p_i) \log \frac{(1 - p_{true})(1 - p_i)}{p_{true}p_i + (1 - p_{true})(1 - p_i)} \\ &\quad - (1 - p_{true})p_i \log \frac{(1 - p_{true})p_i}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \\ &\quad - p_{true}(1 - p_i) \log \frac{(1 - p_{true})p_i}{p_{true}(1 - p_i) + (1 - p_{true})p_i} \end{aligned}$$

Согласно определению в выражении (17.4), ожидаемый прирост информации после восприятия ячейки сетки m_i - это просто разница $H_p(\mathbf{m}_i) - E[H_{p'}(\mathbf{m}_i)]$.

Так насколько лучше ожидаемый прирост информации, по сравнению с энтропией, из которой он был выведен? Ответ: ненамного. На Рис. 17.6 энтропия показана на схеме (b), рядом с ожидаемым приростом информации на схеме (c), все для сегмента карты, показанной на схеме (a). Визуально, энтропия и ожидаемый прирост информации почти неразличимы, хотя их значения различны. Это оправдывает общую практику использования энтропии как функции направления исследования вместо ожидаемого прироста информации.

• **Бинарный прирост**. Третий метод самый простой из всех и самый популярный. Очень грубая аппроксимация ожидаемого прироста информации отмечает ячейки, которые обновлялись, по меньшей мере, единожды, как «исследованные», а все остальные – «неисследованные». Таким образом, прирост становится бинарной функцией.

На Рис. 17.5c показана такая бинарная карта: только внешняя белая зона даст новую информацию. Внутренняя часть карты считается полностью исследованной. Хотя эта карта прироста информации, очевидно, самая грубая из всех обсуждаемых, на практике она хорошо работает, поскольку постоянно толкает робота вглубь неисследованной территории.

ИССЛЕДОВАНИЕ НА ОСНОВЕ ГРАНИЦ

Бинарная карта лежит в основе популярного метода исследования, который называется *исследованием на основе границ*, когда робот постоянно двигается к ближайшей неисследованной границы исследованного пространства.

17.4.2 Распространение прироста информации

Оставшийся вопрос относится к разработке жадного метода исследования, использующего эти карты. Как и в примере активной локализации, это требует определения соответствующего действия исследования.

Простое, но эффективное определение *действия исследования* включает передвижение в местоположение $x - y$ по пути с минимальной стоимостью, а затем восприятие по всем ячейкам сети на небольшом расстоянии по всем направлениям от робота. Таким образом, каждое местоположение на карте определяет потенциальное действие исследования.

Вычисление наилучшего жадного действия исследования теперь легко выполняется с помощью итерационного алгоритма. На Рис. 17.5d показана результирующая функция дохода для карты, показанной на Рис. 17.5a, а бинарная карта прироста информации для этой функции – на Рис. 17.5c. Итерационный алгоритм подразумевает такую бинарную сетку прироста, поскольку прирост может быть только в не обследованных местоположениях.

Центральное обновление значений выполняется с помощью следующей рекурсии:

(17.16)

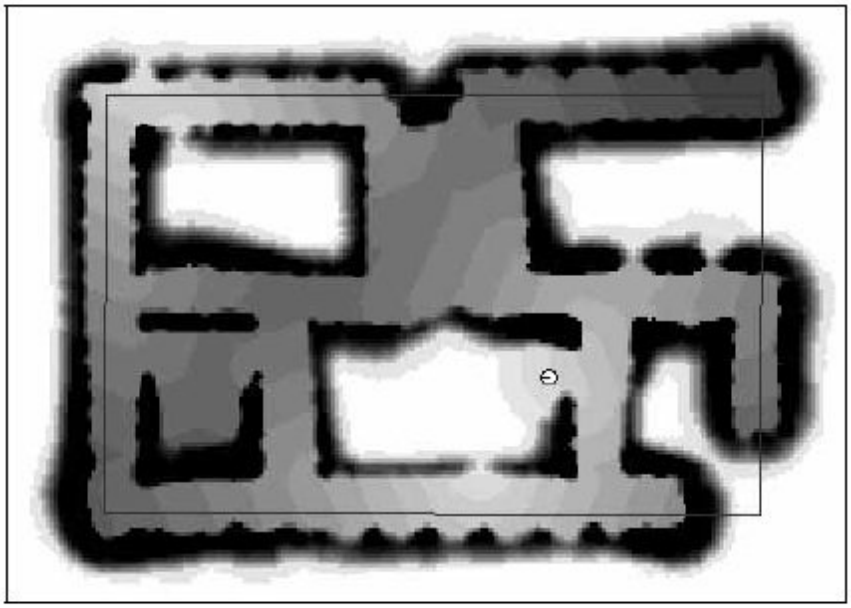
$$V_T(\mathbf{m}_i) = \begin{cases} \max_j \, r(\mathbf{m}_i, \mathbf{m}_j) + V_{T-1}(\mathbf{m}_j) & \text{при } I(\mathbf{m}_i) = 0 \\ I(\mathbf{m}_i) & \text{при } I(\mathbf{m}_i) > 0 \end{cases}$$

Здесь V функция дохода, j индексирует все значения ячеек с поправкой $\mathbf{m}_i$, r, измеряя стоимость перемещения до точки (обычно функция карты сетки занятости), а $I(\mathbf{m}_i)$ информацию, которую можно получить в ячейке $\mathbf{m}_i$. Условие прекращения $I(\mathbf{m}_i) > 0$ справедливо только для неисследованных ячеек на бинарной карте прироста информации.

На Рис. 17.5d показана такая функция ценности после сходимости. Здесь ценность наиболее высока в открытых частях карты, и ниже внутри карты. Метод исследования теперь просто определяет оптимальный путь методом поиска экстремума на карте. Этот путь приведёт робота к ближайшей неисследованной границе.

Очевидно, этот метод исследования является лишь грубой аппроксимацией. В нем полностью игнорируется информация, полученная робот при перемещении к целевому местоположению, и ошибочно считается, что робот при перемещении не воспринимает окружающее. Однако, он хорошо работает на практике. На Рис. 17.7 показана функция дохода и карта робота, выполняющего исследование. Это историческая карта: она была сделана на Соревновании мобильных роботов 1994 *(1994 AAAI Mobile Robot Competition)*, которое включало получение карты окружающей среды с высокими скоростями. Робот был оборудован массивом из 24 ультразвуковых датчиков, подходящих для относительно низкой точности карты. Наиболее интересный аспект этой карты – траектория пути, пройденного роботом. Как видно на Рис. 17.7b, в начале исследование очень эффективно, и робот движется по неисследованным коридорам. Однако, позже робот начинает выбирать между различными целевыми локациями. Такое выборочное поведение характерно для жадных алгоритмов исследования, и самые современные реализации имеют дополнительные механизмы, которые позволяют избежать такого поведения.

(a) Функция дохода исследования



(b) Путь исследования

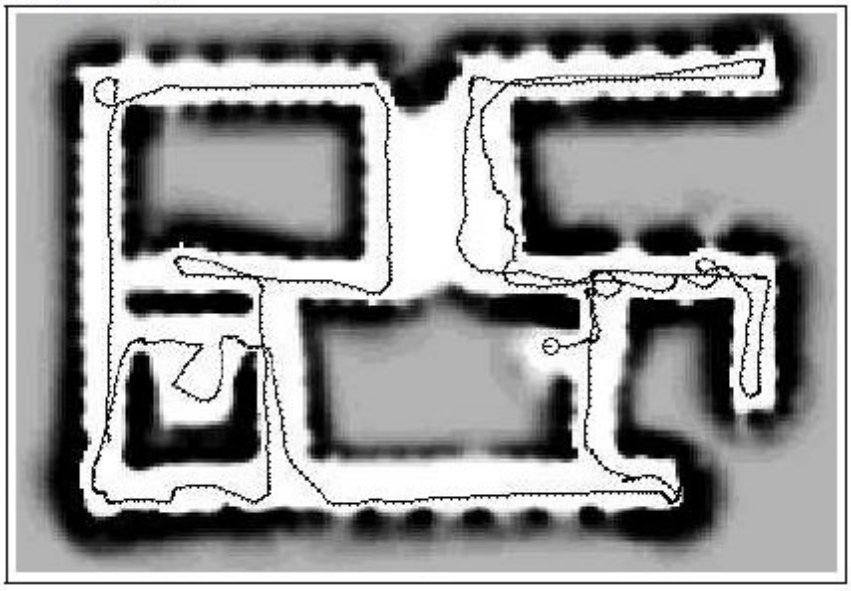


Рис. 17.7 Иллюстрация автономного исследования. Значения дохода V вычисляются с помощью итерации критерия. Белые области полностью не исследованы. Двигаясь по серому градиенту, робот двигается к следующему не открытому месту по пути с минимальной стоимостью. Большой чёрный прямоугольник обозначает глобальную ориентацию стен θ_{wall} (a). Текущий путь при выполнении исследования показана на схеме, вместе с результирующей метрической картой(b).

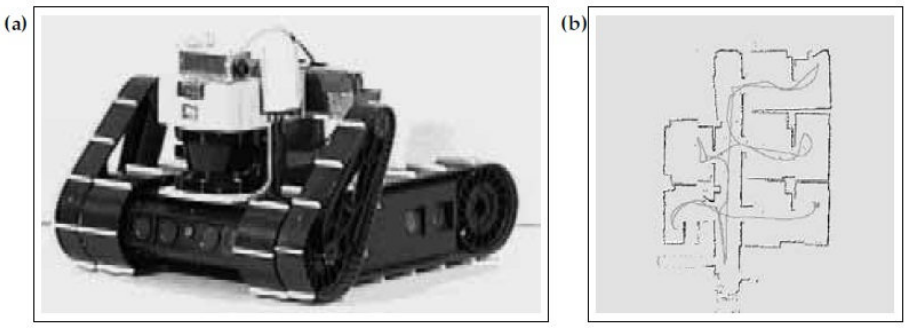


Рис. 17.8 Городской робот для исследования внутри и вне помещений. Одометрия робота достаточно плохая(a). Путь исследования автономного исследовательского робота на основе методов, описанных в тексте.(b)

Второй пример показан на Рис. 17.8. Путь робота, показанный справа, иллюстрирует эффективность жадного алгоритма исследования.

17.4.3 Обобщение для систем нескольких роботов

Правило исследования на основе прироста информации часто обобщается до системы из нескольких роботов, в которой роботы стремятся получить карту, используя исследование группой. В общем, ускорение от использования K роботов линейное. Оно может быть и *более, чем линейным*: K роботов могут ускорять исследование больше, чем в K раз по сравнению с одним роботом. Столь большое ускорение по сравнению с одним роботом вызвано тем, что один робот вынужден переходить через одни и те же места несколько раз - первый раз, чтобы пройти к месту исследования, и второй раз, чтобы вернуться для исследования другой части среды. При соответствующем количестве роботов, возвращение может оказаться ненужным, и степень ускорения будет близкой к $2K$. Коэффициент $2K$ является ограничением сверху для роботов, которые свободно передвигаются во всех направлениях.

Ключевым условием исследования с помощью нескольких роботов является координация действий. При статическом исследовании это легко достигается с помощью жадных методов распределения заданий. Рассмотрим ситуацию, в которой K роботов помещены на частично исследованную карту. В данной ситуации существует несколько мест для исследования на границе и необходим алгоритм, назначающий роботам места, где исследование будет наиболее эффективно.

Алгоритм **multi_robot_exploration** в Таблице 17.2 является одним из простейших алгоритмов такого рода. Для набора из K роботов вычисляется набор K целей исследования в виде координат мест, куда роботы должны переместиться для исследования.

1: **Algorithm multi_robot_exploration**$(m, x_1, ..., x_K)$:
2: *для каждого робота* $k = 1$ *до* K *выполнить*
3: *Пусть* $\mathbf{m}_k$ *будет ячейкой сетки для* x_k
4: $V_k(\mathbf{m}_i) = \begin{cases} \infty & \text{при} \mathbf{m}_i \neq \mathbf{m}_k \\ 0 & \text{при} \mathbf{m}_i = \mathbf{m}_k \end{cases}$
5: *выполнять до* V_k *сходимости*
6: *для всех* i *выполнить*
7: $V_k(\mathbf{m}_i) \longleftarrow \min_j \{V_k(\mathbf{m}_i), r(\mathbf{m}_i, \mathbf{m}_j) + V_k(\mathbf{m}_j)\}$
8: *endfor*
9: *endwhile*
10: *endfor*
11: *вычислить бинарную карту прироста информации* $\bar{m}$ *из карты* m
12: *для каждого робота* $k = 1$ *до* K *выполнять*
13: *установить* $\text{цель}_k = \underset{i \text{ таким образом, что } \bar{\mathbf{m}}_i = 1}{\text{argmin}} V_k(\mathbf{m}_i)$
14: *для всех ячеек* $\bar{\mathbf{m}}_j$ *in* $\varepsilon -$ *соседних с* целью_k
15: *установить* $\bar{\mathbf{m}}_j = 0$
16: *endfor*
17: *endfor*
18: *return* $\{\text{цель}_1, \text{цель}_2, ..., \text{цель}_K\}$

Таблица 17.2 Алгоритм исследования с помощью нескольких роботов.

Сначала алгоритм вычисляет значение функций дохода V_k, по одной для каждого робота (строки со 2 по 10). Однако, функции дохода теперь определяются не так, как было описано раньше: минимальное значение назначается согласно положению робота. Чем дальше о него находится ячейка, тем выше значение дохода. На Рис. 17.9 и 17.10 показаны несколько примеров таких функций дохода. В каждом случае местоположение самого робота имеет минимальную ценность, и значение постепенно возрастает во всем достижимом пространстве.

Легко показать, что эти функции дохода для каждой ячейки сетки измеряют стоимость передвижения к ней. Для каждого отдельного робота оптимальная ячейка для исследования вычисляется в строке 13. Согласно бинарной карте прироста, вычисленной в строке 11, это ячейка с минимальной стоимостью V_k, которая ещё не исследована. Она и используется как целевая точка. Но, для того, чтобы, «не позволить» другим роботам выбрать аналогичное или близкое местоположение, алгоритм назначает нулевое значение прироста вблизи цели. Это происходит в строках с 14 по 16.

Механизм координации в **multi_robot_exploration** можно описать следующим образом: каждый робот подбирает наилучшую из возможных точек исследования, а затем запрещает другим роботам выбрать ту же или близлежащую точку. На Рис. 17.9 и 17.10 показан эффект такой координации. Оба робота на Рис. 17.9, хотя и расположены в разных местах, определяют одну и ту же ячейку для исследования. Поэтому, при исследовании без координации они могут начать передвигаться в одну и ту же зону. Ситуация отличается на Рис. 17.10. Здесь первый робот делает выбор, запрещая выбранное местоположение для второго робота. В свою очередь, второй робот выбирает новое наиболее перспективное местоположение. Результирующее

объединённое действие исследования позволяет избежать конфликта и, в результате, более эффективно выполнить исследование.

Очевидно, механизм координации довольно сильно упрощён и и подвержен застреванию в локальном минимуме. Что произойдёт, если на Рис. 17.10 позволить второму роботу выбирать первым? Это может заставить первого робота выбрать удалённую точку, а, значит пути обоих роботов могут пересечься посередине. Очевидно, пересечение путей является хорошим показателем неоптимальности решения, хотя и отсутствие пересечений не гарантирует, что решение оптимально.

В улучшенных методах координации такие конфликты обрабатываются, что позволяет обменивать цели между роботам. Популярное семейство методов позволяет такой обмен, если это уменьшает общую стоимость исследования. Такие методы часто описываются в терминах *механизмов аукциона* , а результирующие алгоритмы характеризуют как *алгоритмы на основе рынка*.

МЕХАНИЗМ АУКЦИОНА

На Рис. 17.11 показано применение алгоритма **multi_robot_exploration** в реальном эксперименте, где три робота исследовали неизвестную среду. На самом левом рисунке показаны начальные положения роботов. На других изображениях показаны различные ситуации при координированном исследовании. Карта, сгенерированная этими роботами во время дополнительного прохода, показана на Рис. 17.12. Как можно увидеть, роботы хорошо распределились для исследования среды.

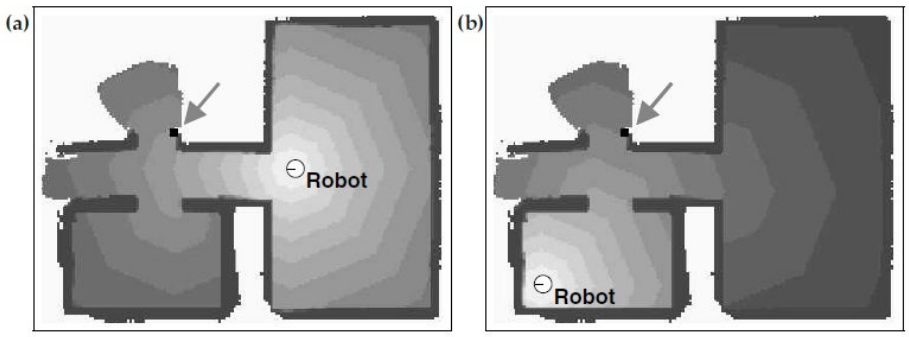


Рис. 17. Два робота, исследующих окружающую среду. Без координации оба робота могут принять решение о передвижении в одну и ту же точку. На каждом изображении показаны робот, карта и функция ценности. Чёрным квадратом обозначена точка с минимальной стоимостью.

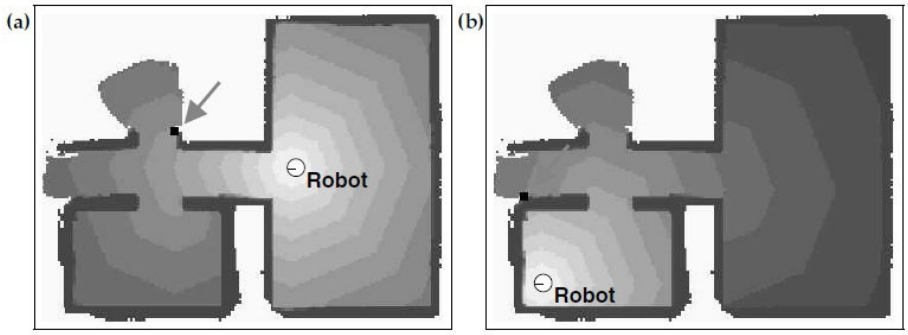
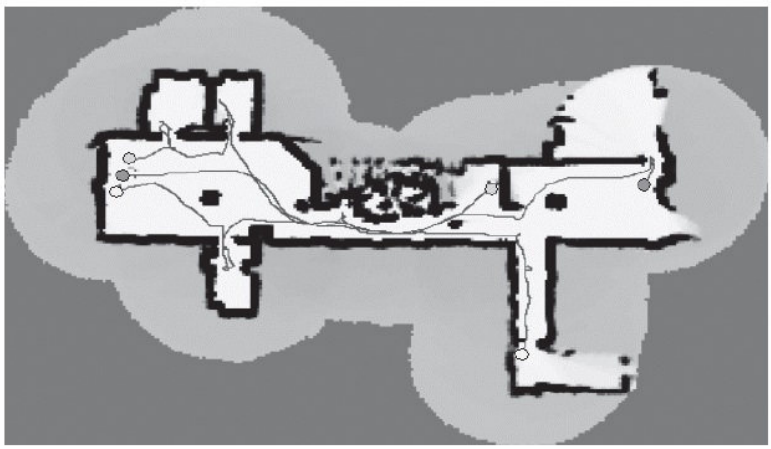


Рис. 17.10 Целевые позиции, полученные методом координации. В этом случае целевая точка для второго робота расположена слева в коридоре.

На Рис. 17.13 показана эффективность этого алгоритма по сравнению с группой роботов, выполняющих алгоритм **Monte_Carlo_exploration** безо всякой координации. По горизонтальной оси показано количество роботов в группе, по вертикальной- количество тактов времени, необходимых для завершения задачи исследования. В этих экспериментах допускалось, что роботы всегда разделяют между собой локальные карты. Кроме того, все роботы стартовали близко друг от друга. Результат очень нагляден, и, очевидно, некоординированные группы роботов гораздо менее эффективны.

Рис. 17.11 Координированное исследование группой мобильных роботов. Роботы распределяются в среде.



Карта 17.12 Карта объёмной среды 62×43 м, изученная тремя роботами за 8 минут.

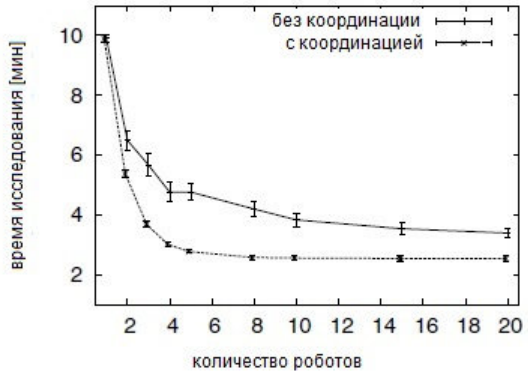


Рис. 17.13 Время исследования, показанное в имитационном эксперименте, в котором различные группы роботов исследовали среду, показанную выше.

Пока что обсуждаемые стратегии координации предполагают, что роботы обмениваются картами и знают относительные местоположения. Принимая во внимание неопределённость относительных положений роботов, исследование с помощью нескольких роботов может быть обобщено для случая, когда роботы стартуют из разных, неизвестных заранее положений.

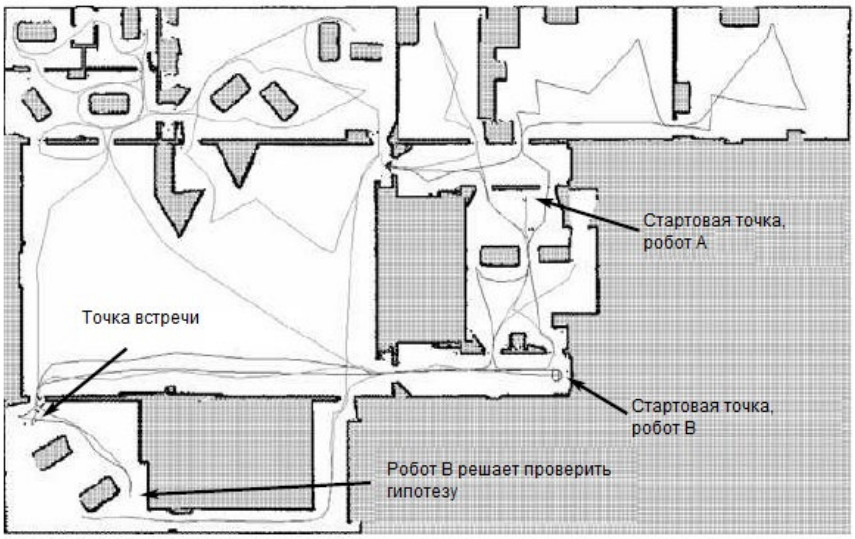


Рис. 17.14 Координированное исследование при неизвестных начальных положениях. Роботы устанавливают общую систему отсчёта, оценивая и проверяя относительное положение друг друга с помощью метода рандеву. При встрече они обмениваются картами и координатами исследований. Собственность Джонатана Ко и Бенсона Лимкеткай (Jonathan Ko and Benson Limketkai).

На Рис. 17.14 показан эксперимент исследования, использующий такой обобщенный метод координации. Два робота, A и B, не имеют никакой информации о своём местоположении и выполняют исследование независимо

друг от друга. По мере исследования, каждый робот оценивает положение другого робота относительно своей карты, используя модифицированную версию MCL локализации. Выбирая, куда двигаться дальше, оба робота A и B оценивают, что будет «выгоднее» - двинуться в неисследованную зону, или проверить гипотезу о местонахождении другого робота. В какой-то момент времени, робот B принимает решение проверить гипотезу об определении местонахождении робота A. Он посылает роботу A команду остановиться и двигается в предполагаемую точку его местонахождения (помеченную как точка встречи на Рис. 17.14). По прибытии в эту точку, оба робота проверяют наличие друг друга с помощью лазерных датчиков расстояния (роботы помечены светоотражающей лентой). После обнаружения друг друга они объединяют карты и обобщают систему отсчета. С этого момента они исследуют среду с помощью алгоритма **multi_robot_exploration**. Такой метод исследования из неизвестных начальных местоположений можно применять для сценариев с более чем двумя роботами.

17.5 Исследование для SLAM

В последнем алгоритме этой книги идея жадного исследования применяется к полному алгоритму SLAM. В предыдущих разделах мы всегда подразумевали, что карта или положение робота известны. В SLAM, однако, неизвестно ни то, ни другое. Соответственно, требуется учитывать неопределённость карты и положения робота при выборе способа исследования, поскольку можно получить или потерять информацию об обоих величинах. Очевидно, без знания положения робота интеграция информации датчиков в карту может привести к серьёзным ошибкам. С другой стороны, робот, который стремится только уменьшить неопределённость положения, просто не будет двигаться и никогда не получит информации о среде за пределами действия датчиков.

17.5.1 Разложение энтропии в SLAM

Ключевой идеей оптимального исследования в SLAM является то, что энтропия апостериорного распределения SLAM может быть *разложена* на две части. Одна часть относится к энтропии апостериорного распределения положения, а вторая – ожидаемой энтропии карты. Таким образом, робот, выполняющий исследование в задаче SLAM, разменивает неопределённость положения робота на неопределённость карты. Действия управления могут уменьшить только одну из двух. При замыкании цикла робот уменьшает неопределённость положения. При перемещении в открытую неисследованную область он, в основном, уменьшает неопределённость карты. Учитывая значения обоих видов неопределённости, робот принимает решение выполнить действие, которое будет в максимальной степени ее уменьшать, поэтому робот иногда будет двигаться на открытое место, а иногда – выполнять локализацию, возвращаясь в уже исследованные области.

Разложение энтропии, фактически, универсально для полной задачи SLAM. Будем считать апостериорное распределение SLAM в виде.

(17.17)
$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

Его можно разложить следующим образом:

(17.18)

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t})p(m|x_{1:t}, z_{1:t}, u_{1:t})$$

Это тривиально и уже приводилось в выражении (13.2) на странице 406. Менее очевидно следствие

(17.19)

$$\begin{aligned} &H[p(x_{1:t}, m|z_{1:t}, u_{1:t})] \\ &\quad = H[p(x_{1:t}|z_{1:t}, u_{1:t})] + E[H[p(m|x_{1:t}, z_{1:t}, u_{1:t})]] \end{aligned}$$

где ожидание берётся по апостериорной вероятности $p(x_{1:t}|z_{1:t}, u_{1:t})$.

Записав $p(x, m)$ как сокращённую форму полной апостериорной вероятности $p(x_{1:t}, m|z_{1:t}, u_{1:t})$, можно вывести разложение в аддитивном виде:

(17.20)

$$\begin{aligned} H(x, m) &= E_{x,m}[-\log p(x, m)] \\ &= E_{x,m}[-\log(p(x)p(m|x))] \\ &= E_{x,m}[-\log p(x) - \log p(m|x)] \\ &= E_{x,m}[-\log p(x)] + E_{x,m}[-\log p(m|x)] \\ &= E_x[-\log p(x)] + \int_{x,m} -p(x, m) \log p(m|x) dx\, dm \\ &= E_x[-\log p(x)] + \int_{x,m} -p(m|x)p(x) \log p(m|x) dx\, dm \\ &= E_x[-\log p(x)] + \int_x p(x) \int_m -p(m|x) \log p(m|x) dx\, dm \\ &= H(x) + \int_x p(x) H(m|x) dx \\ &= H(x) + E_x[H(m|x)] \end{aligned}$$

Это преобразование напрямую даст разложение в (17.19). Оно доказывает, что энтропия SLAM – это сумма энтропии пути и ожидаемой энтропии карты.

17.5.2 Исследование в FastSLAM

Теперь переложим разложение энтропии на алгоритм исследования SLAM. Метод основан на алгоритме FastSLAM, описанном в Главе 13 книги, а именно, FastSLAM на основе сеток в подразделе 13.10. Напомним, что FastSLAM отображает апостериорную вероятность SLAM в виде набора частиц. В каждой частице содержится путь робота. В случае реализации на основе сетки, каждая частица также содержит карту сетки занятости. Это позволяет применить меру энтропии для карт сеток занятости, как было описано в предыдущем разделе.

Алгоритм определения последовательности действий исследования приведён в Таблице 17.3. Он оставляет открытыми рад важных вопросов реализации, поскольку служит только схематической иллюстрацией, но характеризует все основные шаги реальной реализации идеи исследования SLAM.

Алгоритм исследования FastSLAM, в основном, выполняет оценку и проверку. В нем предлагается последовательность действий для исследования,

которая, затем, оценивается путём измерения остаточной энтропии. Основываясь на фундаментальных принципах, описанных выше, энтропия вычисляется в виде суммы двух членов. Первое слагаемое соответствует положению робота в конце предлагаемой последовательности исследования, а второе – описывает ожидаемую энтропию карты. Затем алгоритм исследования выполняет выбор действий управления, минимизирующих энтропию.

1: **Algorithm FastSLAM_exploration**(Y_t) :
2: *инициализировать* $\hat{h} = \infty$
3: *повторять*
4: *предложить последовательность действий для исследования* $u_{t+1:T}$
5: *выбрать случайную частицу* $y_t \in Y_t$
6: *для* $\tau = t+1$ *до* T
7: *извлечь* $x_\tau \sim p(x_\tau | u_\tau, x_{\tau-1})$
8: *извлечь* $z_\tau \sim p(z_\tau | x_\tau)$
9: *вычислить* $Y_\tau = \textbf{FastSLAM}\,(z_\tau, u_\tau, Y_{\tau-1})$
10: *endfor*
11: *обучить гауссову функцию* μ_x, Σ_x *для всех частиц положений* $\{x_T^{[k]}\} \in Y_T$
12: $h = \frac{1}{2}\log\det(\Sigma_x)$
13: *для частиц* $k = 1$ *до* M *выполнять*
14: *пусть* m *- карта* $m_T^{[k]}$ *из* k*-й частицы в* Y_T
15: *обновить* $h = h + \frac{1}{M}H[m]$
16: *endfor*
17: *если* $h < \hat{h}$ *то*
18: *установить* $\hat{h} = h$
19: *установить* $\hat{u}_{t+1:T} = u_{t+1:T}$
20: *endif*
21: *до сходимости*
22: *return* $\hat{u}_{t+1:T}$

Таблица 17.3 Алгоритм исследования для версии FastSLAM на основе сеток. На вход принимается набор частиц Y_t. Каждая частица $y_t^{[k]}$ содержит выборку пути робота $x_{1:t}^{[k]}$ и связанную карту сетки занятости $m^{[k]}$. На выходе создаётся путь исследования, выраженный в виде команд относительного движения.

Более детально, алгоритм **FastSLAM_exploration** принимает на вход набор частиц, а возвращает предполагаемую последовательность действий управления для исследования. В строке 4 Таблицы 17.3 предложена потенциальная последовательность управления. Проверка последовательности управления выполняется в строках с 5 по 16. Она состоит из трёх частей. Во-первых, робот смоделирован на основе случайной частицы в наборе частиц. В этом моделировании использована стохастическая модель робота и среды. Результатом является последовательность наборов частиц по всему пути до конца управляемой траектории. Моделирование выполняется в строках с 5 по 9.

Вслед за этим вычисляется энтропия финального набора частиц. С помощью математического разложения, приведённого в Выражении 17.19, энтропия разбивается на два слагаемых: первое относится к оценке энтропии

положения робота в момент T, а второе – к ожидаемой энтропии карты. Первое слагаемое вычисляется в строках 11 12. Его правильность следует из правильности выведения энтропии гауссовой функции, приведённой в Таблице. 17.4.

Лемма. *Энтропия многомерного гауссиана* с d измерениями и Σ задана в виде
$H = \frac{d}{2}(1 + \log 2\pi) + \frac{1}{2}\log\det(\Sigma)$
Доказательство. Для
$p(x) = (2\pi)^{-\frac{d}{2}} \det(\Sigma)^{-\frac{1}{2}} \exp\{-\frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu - \frac{1}{2}\mu^T\Sigma^{-1}\mu\}$
получим
$H_p[x] = E[-\log p(x)]$
$= \frac{1}{2}(d\log 2\pi + \log\det(\Sigma) + E[x^T\Sigma^{-1}x] - 2E[x^T]\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu)$
Здесь$E[x^T] = \mu^T$, и $E[x^T\ \Sigma^{-1}\ x]$ разрешаются в следующем виде (где "$\cdot$" обозначает умножение)

$$\begin{aligned} E[x^T\ \Sigma^{-1}\ x] &= E[x\ x^T\ \cdot\ \Sigma^{-1}] \\ &= E[x\ x^T]\ \cdot\ \Sigma^{-1} \\ &= \mu\,\mu^T\ \cdot\ \Sigma^{-1} + \Sigma\ \cdot\ \Sigma^{-1} \\ &= \mu^T\Sigma^{-1}\mu + d \end{aligned}$$

Из этого следует, что

$$\begin{aligned} H_p[x] &= \tfrac{1}{2}(d\log 2\pi + \log\det(\Sigma) + \mu^T\Sigma^{-1}\mu + d - 2\mu^T\Sigma^{-1}\mu + \mu^T\Sigma^{-1}\mu) \\ &= \tfrac{d}{2}(1 + \log 2\pi) + \tfrac{1}{2}\log\det(\Sigma) \end{aligned}$$

Таблица 17.4 Энтропия многомерного гауссиана

Второй член выражения, означающий энтропию, вычисляется в строках с 13 по 16. Заметим, что вычисление включает энтропию карты m и для карт сетки занятости выполняется аналогично обсуждаемому в предыдущем разделе. В строках с 13 по 16 вычисляется средняя энтропия карты в виде среднего значения по всем частицам в момент времени T. Результатом является значение h, измеряющее ожидаемую энтропию в момент времени T, зависящий от предполагаемой последовательности управления. Затем в строках с 17 по 20 выбирается последовательность действий, минимизирующая ожидаемую энтропию, которая возвращается в строке 22.

Заметим, что в алгоритме для вычисления энтропии апостериорной вероятности по траекториям в строке 11 используется аппроксимация. Вместо выражения гауссиана по всем траекториям частиц $y_T^{[k]}$, вычисление производится на основании последних положений $x_T^{[k]}$. Это приближение хорошо работает на практике, и напоминает уже приведённый вид аппроксимации действия исследования.

В итоге, алгоритм исследования FastSLAM, в основном, является обобщением исследовательского алгоритма методом Монте-Карло, приведённого в Таблице 17.1, с двумя отличиями. Во-первых, он применяется к целым последовательностям управляющих воздействий, а не только к одному действию. Во-вторых, что более важно, алгоритм исследования FastSLAM вычисляет два типа энтропии, один для пути робота и один для карты.

17.5.3 Эмпирическое определение

Алгоритм исследования приводит к соответствующему поведению робота, особенно в циклических средах. На Рис. 17.15 показана ситуация, когда робот исследует *циклическую среду*, в которой имеется *петля*. Робот начинает в правом нижнем углу цикла, помеченной "Начало". На такте времени 35, предпринимаемые роботом действия снова приводят его на стартовую точку, рядом с неизвестной зоной слева от маркера. Общий прирост информации от возвращения на старт выше, чем продвижение вглубь неизвестной области, поскольку, вдобавок к новой информации на карте, будет уменьшена неопределённость положения. Поэтому робот, выполняющий активное исследование, решает замкнуть цикл и двигаться по уже обследованной территории.

На Рис. 17.16 более подробно показан компромисс между стоимостью и наградой. На ней показано 8 различных действий. Полезность действия 1 наивысшая и существенно превышает полезность действия 4 (и любых других действий без замыкания цикла).

На такте времени 45 робот замыкает цикл, как показано на Рис. 17.15. В этой точке неопределённость положения минимальна, и неопределённость карты начинает преобладать. В результате, действие замыкания цикла уже не выглядит привлекательным. В момент времени 88 робот выбирает исследования большой открытой области, к которой он и следует, что видно из Рис. 17.15.

На Рис. 17.17 показано изменение общей энтропии в эксперименте со временем. До такта времени 30, уменьшение неопределённости карты компенсировало увеличение неопределённости по траектории робота и энтропия оставалась более-менее постоянной. Хотя действие замыкания цикла уменьшает энтропию гипотезы по траектории робота, изменения неопределённости карты относительно мало. Это приводит к уменьшению общей энтропии. По мере того, как робот учитывает измерения, покрывающие доселе неизвестные области горизонтального коридора, изменения неопределённости карты и положения робота компенсируют друг друга. Падение общей энтропии около такта времени 90 вызвано наблюдениями в широкой части коридора. Это происходит потому, что карты сеток занятости уменьшают неопределённость карты, учитывая проход сканирования дальности, что почти линейно зависит от размера неизвестной зоны, покрываемой проходом сканирования.

Сложная взаимосвязь между неопределённостью карты и пути, и соответствующие понятия прироста информации играют важную роль в алгоритме исследования. Робот выбирает локализацию, когда это предпочтительно, но иногда движется в неизвестную область.

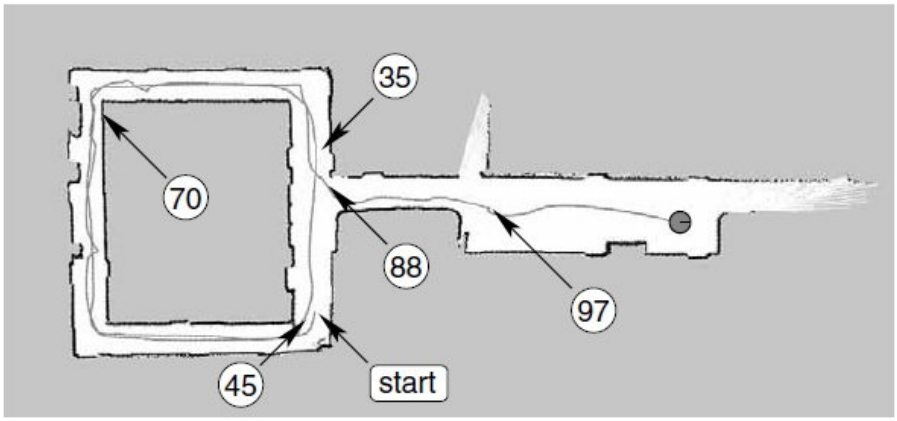


Рис. 17.15 Мобильный робот исследует циклическую среду. Робот начинает в правом нижнем углу цикла. После прохода по всему циклу, он решает повторить траекторию, чтобы уменьшить неопределённость, а затем продолжает исследование коридора. Рисунок принадлежит Сирилу Стачнису, университет Фрайбурга (Cyrill Stachniss, University of Freiburg).

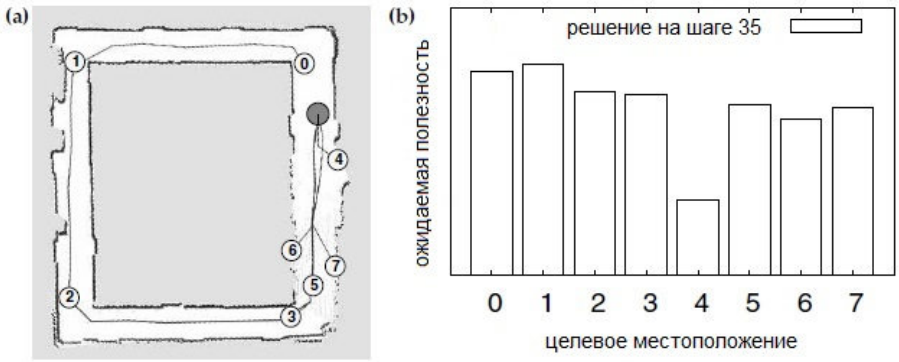


Рис. 17.16 В этой ситуации робот определяет ожидаемую полезность действий. действия исследования, рассматриваемые роботом(a), и ожидаемая полезность каждого действия (b). Действие 1 выбирается потому, что оно максимизирует ожидаемую полезность. Рисунок принадлежит Сирилу Стачнису, университет Фрайбурга (Cyrill Stachniss, University of Freiburg).

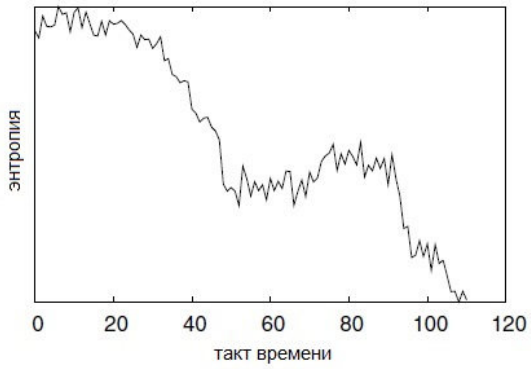


Рис. 17.17 Изменение энтропии в ходе исследовательского эксперимента, показанного на Рис. 17.15. Рисунок принадлежит Сирилу Стачнису, университет Фрайбурга (Cyrill Stachniss, University of Freiburg).

17.6 Выводы

В этой главе мы узнали об исследованиях с помощью роботов. Все представленные в главе алгоритмы преследуют единственную цель – максимизировать информацию, получаемую роботом. Распределяя действия управления так, чтобы они максимизировали прирост информации, робот способен эффективно выполнять задачи исследования.

Эта идея была использована для решения ряда задач исследования:

• В активной локализации робот стремится максимизировать осведомлённость о своём положении на известной карте. Был выведен алгоритм, вычисляющий ожидаемый прирост информации для перемещения в любое из возможных положений робота. В нём устанавливается компромисс между перемещением в точку, дающую максимальный прирост информации и стоимостью такого перемещения. Результирующий алгоритм хорошо выполняет отбор местоположений на основе прироста информации.

• В исследовании для построения карт робот знает своё положение с самого начала, но должен собирать данные об окружающей среде. После анализа идеи карты сетки занятости было замечено, что прирост информации можно вычислить отдельно для каждой ячейки на карте. После сравнения нескольких разных методов вычисления прироста информации была показана пригодность для практического использования даже самых простых методов, например, энтропии. Затем был выведен динамический алгоритм программирования перемещения в ближайшую точку с наилучшим балансом между приростом информации и стоимостью перемещения.

• Мы дополнили метод исследования на основе прироста информации для случая группы роботов. Это обобщение оказалось удивительно простым, поскольку очевидная модификация парадигмы динамического программирования сделала возможным вычисление стоимости перемещения в любое местоположение, сравнительно с приростом информации. На основании сравнения стоимости и прироста информации для разных местоположений, несколько роботов могут координировать цели для минимизации общего времени исследования.

• Наконец, обсуждались методы исследования для полной задачи SLAM, когда и положение робота, и карта окружающей среды неизвестны. Рассмотренный метод основан на разбиении энтропии на два компонента, первый из которых выражает неопределённость пути, а второй – неопределённость карты (усреднённую по всем возможным путям). Это соображение позволило создать алгоритм генерации и проверки для задачи исследования, в котором генерируются управляющие последовательности и вычисляются будущие оценки, позволяющие найти баланс между неопределённостями пути и карты при оценке последовательностей управления. Результатом стал метод исследования, в котором робот иногда двигается в неисследованные области для уменьшения неопределённости карты, а, иногда – возвращается на уже посещённые участки для улучшения оценок положения.

Большинство методов в этой главе – жадные, поскольку робот рассматривает единственный вариант выбора управляющего действия. Эта жадность стала результатом огромного коэффициента ветвления в большинстве задач исследования, что делает планирование на несколько шагов вперёд неприменимым. Однако, выбор правильного действия исследования потребовал некоторых размышлений.

Алгоритмы в этой главе рассматривают в качестве действия исследования робота перемещение в любую точку локальной координатной системы. Поэтому рассматриваемый базовый блок движения существенно превосходит общий блок движения, определённый в Главе 5. Но именно такое определение действия делает кажущиеся простыми методы применимыми для сложных многоэтапных задач исследования с помощью роботов.

Было замечено, что исследование можно сформулировать в виде общей задачи POMDP, используя функцию дохода, которая награждает за прирост информации. Однако, методы POMDP лучше использовать при малом коэффициенте ветвления и ограниченном числе возможных наблюдений. Задачи исследования характеризуются огромными пространствами состояний и наблюдений. В силу этого их лучше всего решать с помощью жадных методов, которые напрямую максимизируют прирост информации.

17.7 Библиографические примечания.

Исследование было основной областью разработки робототехнических систем, которые использовались в таких областях, как исследование вулканов (Bares and Wettergreen 1999; Wettergreen et al. 1999), лун и планет (Gat et al. 1994; Höllerer et al. 1999; Krotkov et al. 1999; Matthies et al. 1995; Li et al. 2000), поиск и спасение (Murphy 2004), составление карт заброшенных шахт (Madhavan et al. 1999; Thrun et al. 2004c; Baker et al. 2004), поиск метеоритов в Антарктике (Urmson et al. 2001; Apostolopoulos et al. 2001), исследование пустынь (Bapna et al. 1998) а также подводные исследования (Ballard 1987; Sandwell 1997; Smith and Dunn 1995; Whitcomb 2000).

Литература по разработке алгоритмов по исследованию с помощью роботов уходит корнями в различные области теории информации и принятия решений, описанных в двух предыдущих главах. Одним из ранних методов исследования с помощью роботов является алгоритм, описанный Куперсом и Буном (Kuipers and Byun, 1990, 1991), а также Пирсом и Куперсом (Pierce and Kuipers, 1994). В этом методе робот идентифицирует так называемые локально различимые места, что позволяет ему отличать уже посещённые и ещё неисследованные зоны. Похожая базовая работа принадлежит Дудеку

(Dudek et al., 1991), который разработал стратегию исследования неизвестной среды наподобие графа. Эти алгоритмы не учитывали расстояний и были специально разработаны для роботов с ограниченными способностями к восприятию.

Ранний метод исследования для изучения топологических карт мобильными роботами был предложен Кенигом и Симмонсом (Koenig and Simmons, 1993). Идея активного исследования для карт сеток занятости на основе динамического программирования восходит к работам Моравица и Труна (Moravec, 1988 and Thrun, 1993). Тейлор и Кригман (Tailor and Kriegman, 1993) описали метод посещения всех ориентиров в среде для построения карты на основе признаков. В этой системе робот сохраняет список всех не посещённых ориентиров среды. Идею максимизации информации в исследовании с помощью статистической формулировки можно найти в работе Келблинга (Kaelbling et al., 1996). Ямагучи (Yamauchi et al., 1999) представил метод исследования на основе границ, где границы между исследованными и неисследованными областями специально использовались для управления действиями робота. Позже Гонсалес-Банос и Латомбе (González-Baños and Latombe, 2001) предложили стратегию исследования, где для определения следующего действия учитывался размер невидимой зоны, которая может стать видимой с нескольких возможных точек наблюдения. Похожие методы исследования стали популярными в области моделирования трёхмерных объектов. Например, Уайти и Ферри (Whaite and Ferrie, 1997) изучили задачу сканирования объектов и учёта неопределённости параметров модели при определении следующей лучшей точки зрения.

Метод исследования также был обобщён для групп роботов, действующих совместно. Бургард (Burgard et al., 2000, 2004), и Симмонс (Simmons et al., 2000a) расширили жадную концепцию исследования для групп роботов, которые выполняли исследование совместно, пытаясь максимизировать информацию карты. Этот подход похож на метод инкрементного размещения, введённый Говардом (Howard et al., 2002) и на алгоритм, предложенный Страупом (Stroupe, 2004). Методы на основе рынка для координированного поиска были исследованы Злотом (Zlot et al., 2002). Диас (Dias et al., 2004) анализировал потенциальные отказы при координации нескольких роботов, предложив улучшенный алгоритм. Метод для работы с гетерогенными группами роботов был представлен Сингхом и Фуджимура (Singh and Fujimura, 1993). Обобщение для координированного исследования из нескольких, неизвестных стартовых локаций, было представлено Ко (Ko et al., 2003) и тщательно проверено Конолигом (Konolige et al., 2005). Этот метод использует структурную оценку среды вместе с модифицированной версией локализации MCL для оценки относительных местоположений робота (Fox et al. 2005). В работе Реклейтиса (Rekleitis et al., 2001b), авторы предложили метод исследования, в котором один робот наблюдал местоположение второго, выполняющего исследование, уменьшая, таким образом, его неопределённость местоположения. Некоторые из экспериментов с несколькими роботами, представленные в этой главе, были впервые описаны Труном (Thrun, 2001).

ЗАДАЧА ПОКРЫТИЯ

В некоторых публикациях задача исследования карты изучалась как *задача покрытия*, освещая проблему проектирования алгоритма для покрытия неизвестной среды. В новой работе Чосета (Choset, 2001) дан подробный обзор в этой области. Более новые методы (Acar and Choset 2002; Acar et al. 2003) используют для решения этой проблемы статистику, с алгоритмами не слишком отличными от представленных в книге.

В контексте SLAM несколько авторов вывели методы исследования, совместно выполняющие оптимизацию покрытия карты и активную локализацию. Макаренко (Makarenko et al., 2002) описал метод, определяющий необходимые действия на основе ожидаемого прироста информации, полученного при повторном наблюдении ориентиров (для более точного определения местоположения или положения робота) и при исследовании неизвестных мест. Похожим образом, Ньюман (Newman et al., 2003) описал метод исследования в контексте фреймворка Atlas (Bosse et al. 2003) для эффективного SLAM. В этом методе робот строит структуру наподобие графа для хранения информации об уже посещённых местах. Сим (Sim et al., 2004) специально обратил внимание на задачу планирования траектории в SLAM. В нем учитывался специальный параметрический класс политик спиральных траекторий в контексте метода на основе EKF для задачи SLAM. Метод исследования FastSLAM, описанный в этой главе, принадлежит Стачнису и Бургарду (Stachniss and Burgard, 2003, 2004). Исследование на базе SLAM, где робот выкладывает маркеры, чтобы облегчить локализацию, описана Баталиным и Сухатме (Batalin and Sukhatme, 2003).

Анализ эффективности методов исследования с помощью роботов также был объектом значительного интереса. Несколько авторов предоставили математический и эмпирический анализ сложности разных методов исследования (Albers and Henzinger 2000; Deng and Papadimitriou 1998; Koenig and Tovey 2003; Lumelsky et al. 1990; Rao et al. 1993). Например, Ли и Рис (Lee and Recce, 1997) представили экспериментальное исследование, в котором сравнили эффективность различных стратегий исследования для одиночных роботов.

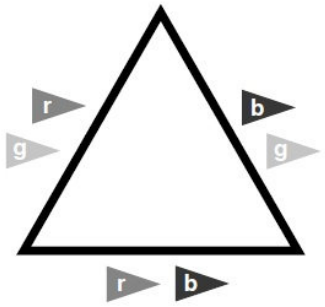
Описанный в книге метод активной локализации для мобильных роботов был впервые опубликован в работах Бургарда (Burgard et al., 1997) и Фокса (Fox et al., 1998). Чуть позже, Янсфельт и Кристенсен (Jensfelt and Christensen, 2001a) представили систему на основе смеси гауссиан для представления апостериорного положения робота и описали способ выполнения активной локализации на основе этого представления. Задача активной локализации была теоретически изучена, например, Келблингом (Kleinberg, 1994) для случая идеальных датчиков.

Несколько авторов разработали методы исследования для динамических сред. Особенный интерес представляют игры в преследование, описанные в обширной литературе по дифференциальным играм (Isaacs 1965; Bardi et al. 1999). Методы ухода от преследования в мобильной робототехнике принадлежат ЛаВалли (LaValle et al., 1997) и Гуйбасу (Guibas et al., 1999), и были недавно расширены Герки (Gerkey et al., 2004).

Наконец, исследование активно изучалось в теории автоматов. Парадигма последовательного принятия решений, в которой агент получает награду в ходе экспериментов изначально изучалась в контексте простых конечных автоматов, известных, как бандиты, см. Фельдман (Feldman, 1962), Берри и Фристедт (Berry and Fristedt, 1985), Роббинс (Robbins, 1952). Методы изучения структуры конечных автоматов восходят к работам Ривеста и Шапире (Rivest and Schapire, 1987a,b) а Мозер и Бахра (Mozer and Bachrach, 1989) разработали методы генерации последовательности проверок, различающие различные состояния в конечных автоматах. Границы сложности на основе состояний для исследуемых детерминированных сред были выведены Кенигом и Симмонсом (Koenig and Simmons, 1993) и Труном (Thrun, 1992), а позже – обобщены для стохастических сред Кирнсом и Сингхом (Kearns and Singh, 2003).

17.8 Упражнения

1. Пусть дан робот, действующий в треугольной среде с тремя типами ориентиров:



Каждое местоположение имеет два разных ориентира, отличающихся по цвету. Допустим, каждый раунд робот может определить наличие только одного типа ориентиров: или обозначенного “r”, или “g”, или “b”. Допустим, робот в начале выстреливает детектор ориентиров “b” и двигается по часовой стрелке до следующей стороны. Какой детектор оптимально использовать следующим? Как изменится ответ, если робот не двигается, или двигается против часовой стрелки до следующей стороны?

2. Допустим, дано K однонаправленных роботов, каждый из которых в этом упражнении может двигаться и выполнять измерения в любое время. В этом упражнении хотелось бы увидеть каждый видимый ориентир только единожды. О преимуществах повторного ориентира пока беспокоиться не нужно.

В тексте отмечалось, что использование нескольких роботов может ускорить исследование больше, чем на линейный коэффициент количества роботов (то есть K могут быть в более чем K раз быстрее, чем 1 робот).

(a) Насколько быстрее группа из K будет быстрее по сравнению с одним роботом?

(b) Дать пример среды, максимизирующей ускорение исследования для $K = 4$ роботов, и обсудить стратегию исследования, которая даст такое ускорение.

3. Допустим, выполняется преследование нарушителя в ограниченной известной среде. Можно ли изобразить среду, где K роботов могут преуспеть в обнаружении нарушителя за конечное время, но $K - 1$ роботов уже не могут. Нарисовать среду для $K = 2$, $K = 3$, и $K = 4$ роботов. Заметим, что результат не даёт никаких допущений относительно стратегии движения нарушителя кроме одного – если нарушитель попадает в поле зрения, он замечен.

ЗАДАЧА БАНДИТА

4. Очень упрощённая задача исследования известна как задача *K-рукого бандита* . В ней перед вами игральный автомат с K ручками. Рывок каждой ручки даёт награду \$1 с вероятностью p_K, где p_K фиксирована, но

неизвестна. Заданием является определение ручек, игра на которых принесёт максимальную награду.

(a) Доказать, что жадная стратегия исследования может быть неоптимальной, где «жадная» определяется как выбор действия относительно функции оценки максимального правдоподобия вероятностей p_k. (После n игр на ручке k, оценка максимального правдоподобия p_k задана как n_k/n, где n_k - количество раз, когда была получена награда \$1).

(b) Доказать, что оптимальная стратегия исследования никогда не будет исключать ни одну из ручек.

(c) Реализовать алгоритм K-рукого бандита для $K = 2$, с обоими вероятностями p_1 и p_2, выбранными равномерно на интервале $[0; 1]$. Реализовать самую лучшую стратегию, которую удастся найти. Стратегия исследования может зависеть только от переменных n_i для $i = 1, 2$. Объяснить стратегию, и измерить общий доход для 1000 игр из 100 шагов каждая.

5. В подразделе 17.4 мы эмпирически сравнили два метода вычисления прироста информации для ячейки сети: энтропию и ожидаемый прирост энтропии. Дать математическую границу ошибки между двумя величинами при допущениях, указанных в главе. Для каких значений занятости карты эта ошибка будет максимальна? Для каких значений она будет минимальна?

6. В тексте мы столкнулись с энтропией в виде гауссовой функции. Требуется вычислить ожидаемый прирост информации для простого обновления гауссиана. Допустим, оценивается неизвестная переменная состояния x, а текущие оценки максимального правдоподобия μ и Σ. Допустим, что датчик может измерить x, но измерение будет повреждено гауссовым шумом с ковариацией Q. Дать выражение для ожидаемого прироста информации через выполнение измерений. *Подсказка: Нужно сосредоточиться на ковариации, а не на математическом ожидании.*

18 Список литературы

Aberdeen, D. 2002. A survey of approximate methods for solving partially observable Markov decision processes. Technical report, Australia National University.

Acar, E.U., and H. Choset. 2002. Sensor-based coverage of unknown environments. *International Journal of Robotic Research* 21:345–366.

Acar, E.U., H. Choset, Y. Zhang, and M.J. Schervish. 2003. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *International Journal of Robotic Research* 22:441–466.

Albers, S., and M.R. Henzinger. 2000. Exploring unknown environments. *SIAM Journal on Computing* 29:1164–1188.

Anguelov, D., R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. 2002. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)*.

Anguelov, D., D. Koller, E. Parker, and S. Thrun. 2004. Detecting and modeling doors with mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Apostolopoulos, D., L. Pedersen, B. Shamah, K. Shillcutt, M.D. Wagner, and W.R. Whittaker. 2001. Robotic antarctic meteorite search: Outcomes. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 4174–4179.

Araneda, A. 2003. Statistical inference in mapping and localization for a mobile robot. In J. M. Bernardo, M.J. Bayarri, J.O. Berger, A. P. Dawid, D. Heckerman, A.F.M. Smith, and M.West (eds.), *Bayesian Statistics 7*. Oxford, UK: Oxford University Press.

Arkin, R. 1998. *Behavior-Based Robotics*. Cambridge, MA: MIT Press.

Arras, K.O., and S.J Vestli. 1998. Hybrid, high-precision localisation for the mail distributing mobile robot system MOPS. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Astrom, K.J. 1965. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications* 10:174–205.

Austin, D.J., and P. Jensfelt. 2000. Using multiple Gaussian hypotheses to represent probability-distributions for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Avots, D., E. Lim, R. Thibaux, and S. Thrun. 2002. A probabilistic technique for simultaneous localization and door state estimation with mobile robots in dynamic environments. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

B, Triggs, McLauchlan P, Hartley R, and Fitzgibbon A. 2000. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski (eds.), *Vision Algorithms: Theory and Practice*, LNCS, pp. 298–375. Springer Verlag.

Bagnell, J., and J. Schneider. 2001. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Bailey, T. 2002. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments.* PhD thesis, University of Sydney, Sydney, NSW, Australia.

Baker, C., A. Morris, D. Ferguson, S. Thayer, C. Whittaker, Z. Omohundro, C. Reverte, W. Whittaker, D. Hähnel, and S. Thrun. 2004. A campaign in autonomous mine mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Ballard, R.D. 1987. The *Discovery of the Titanic*. New York, NY:Warner/Madison Press.

Bapna, D., E. Rollins, J. Murphy, M. Maimone, W.L. Whittaker, and D. Wettergreen. 1998. The Atacama Desert trek: Outcomes. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 1, pp. 597–604.

Bar-Shalom, Y., and T.E. Fortmann. 1988. *Tracking and Data Association.* Academic Press.

Bar-Shalom, Y., and X.-R. Li. 1998. *Estimation and Tracking: Principles, Techniques, and Software.* Danvers, MA: YBS.

Bardi, M., Parthasarathym T., and T.E.S. Raghavan. 1999. *Stochastic and Differential Games: Theory and Numerical Methods.* Boston: Birkhauser.

Bares, J., and D.Wettergreen. 1999. Dante II: Technical description, results and lessons learned. *International Journal of Robotics Research* 18:621–649.

Barniv, Y. 1990. Dynamic programming algorithm for detecting dim moving targets. In Y. Bar-Shalom (ed.), *Multitarget-Multisensor Tracking: Advanced Applications*, pp. 85–154. Boston: Artech House.

Barto, A.G., S.J. Bradtke, and S.P. Singh. 1991. Real-time learning and control using asynchronous dynamic programming. Technical Report COINS 91-57, Department of Computer Science, University of Massachusetts, MA.

Batalin, M., and G. Sukhatme. 2003. Efficient exploration without localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Baxter, J., L. Weaver, and P. Bartlett. 2001. Infinite-horizon gradient-based policy search: II. Gradient ascent algorithms and experiments. *Journal of Artificial Intelligence Research.* To appear.

Bekker, G. 1956. *Theory of Land Locomotion.* University of Michigan.

Bekker, G. 1969. *Introduction to Terrain-Vehicle Systems.* University of Michigan.

Bellman, R.E. 1957. *Dynamic Programming.* Princeton, NJ: Princeton University Press.

Berry, D., and B. Fristedt. 1985. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall.

Bertsekas, Dimitri P., and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific.

Besl, P., and N. McKay. 1992. A method for registration of 3d shapes. *Transactions on Pattern Analysis and Machine Intelligence* 14:239–256.

Betgé-Brezetz, S., R. Chatila, and M. Devy. 1995. Object-based modelling and localization in natural environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Betgé-Brezetz, S., P. Hébert, R. Chatila, and M. Devy. 1996. Uncertain map making in natural environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis.

Betke, M., and K. Gurvits. 1994. Mobile robot localization using landmarks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 135–4142.

Biswas, R., B. Limketkai, S. Sanner, and S. Thrun. 2002. Towards object mapping in dynamic environments with mobile robots. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Blackwell, D. 1947. Conditional expectation and unbiased sequential estimation. *Annals of Mathematical Statistics* 18:105–110.

Blahut, R.E.,W. Miller, and C.H.Wilcox. 1991. *Radar and Sonar: Parts I&II.* New York, NY: Springer-Verlag.

Borenstein, J., B. Everett, and L. Feng. 1996. *Navigating Mobile Robots: Systems and Techniques. Wellesley,* MA: A.K. Peters, Ltd.

Borenstein, J., and Y. Koren. 1991. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation* 7:278–288.

Bosse, M., P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. 2004. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *International Journal of Robotics Research* 23:1113–1139.

Bosse, M., P. Newman, M. Soika, W. Feiten, J. Leonard, and S. Teller. 2003. An atlas framework for scalable mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Bouguet, J.-Y., and P. Perona. 1995. Visual navigation using a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 645–652.

Boutilier, C., R. Brafman, and C. Geib. 1998. Structured reachability analysis for Markov decision processes. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, pp. 24–32.

Brafman, R.I. 1997. A heuristic variable grid solution method for POMDPs. In *Proceedings of the AAAI National Conference on Artificial Intelligence.*

Brooks, R.A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2:14–23.

Brooks, R.A. 1990. Elephants don't play chess. *Autonomous Robots* 6:3–15.

Brooks, R.A., and T. Lozano-Perez. 1985. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics* 15:224–233.

Bryson, A.E., and H. Yu-Chi. 1975. *Applied Optimal Control.* Halsted Press, JohnWiley & Sons.

Bulata, H., and M. Devy. 1996. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, Minneapolis, USA.

Burgard, W., A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. 1999a. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence* 114:3–55.

Burgard,W., A. Derr, D. Fox, and A.B. Cremers. 1998. Integrating global position estimation and position tracking for mobile robots: the Dynamic Markov Localization approach. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Burgard, W., D. Fox, D. Hennig, and T. Schmidt. 1996. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence (AAAI).*

Burgard,W., D. Fox, H. Jans, C. Matenar, and S. Thrun. 1999b. Sonar-based mapping of large-scale mobile robot environments using EM. In *Proceedings of the International Conference on Machine Learning,* Bled, Slovenia.

Burgard, W., D. Fox, M. Moors, R.G. Simmons, and S. Thrun. 2000. Collaborative multi-robot exploration. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Burgard, W., D. Fox, and S. Thrun. 1997. Active mobile robot localization. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, San Mateo, CA. Morgan Kaufmann.

Burgard,W.,M. Moors, C. Stachniss, and F. Schneider. 2004. Coordinated multi-robot exploration. *IEEE Transactions on Robotics and Automation.* To appear.

Canny, J. 1987. *The Complexity of Robot Motion Planning.* Cambridge, MA: MIT Press.

Casella, G.C., and R.L. Berger. 1990. *Statistical Inference.* Pacific Grove, CA: Wadsworth & Brooks.

Cassandra, A.R., L.P. Kaelbling, and M.L. Littman. 1994. Acting optimally in partially observable stochastic domains. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 1023–1028.

Cassandra, A., M. Littman, and N. Zhang. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Conference on Uncertainty in AI (UAI).*

Castellanos, J.A., J.M.M. Montiel, J. Neira, and J.D. Tardós. 1999. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation* 15:948–953.

Castellanos, J.A., J. Neira, and J.D. Tardós. 2001. Multisensor fusion for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation* 17: 908–914.

Castellanos, J.A., J. Neira, and J.D. Tardós. 2004. Limits to the consistency of the EKFbased SLAM. In M.I. Ribeiro and J. Santos-Victor (eds.), *Proceedings of Intelligent Autonomous Vehicles (IAV-2004)*, Lisboa, PT. IFAC/EURON and IFAC/Elsevier.

Chatila, R., and J.-P. Laumond. 1985. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 138–145.

Cheeseman, P., and P. Smith. 1986. On the representation and estimation of spatial uncertainty. *International Journal of Robotics* 5:56 – 68.

Choset, H. 1996. *Sensor Based Motion Planning: The Hierarchical Generalized Voronoi Graph.* PhD thesis, California Institute of Technology.

Choset, H. 2001. Coverage for robotics—a survey of recent results. *Annals of Mathematical Artificial Intelligence* 31:113–126.

Choset, H., K. Lynch, S. Hutchinson, G. Kantor,W. Burgard, L. Kavraki, and S. Thrun. 2004. *Principles of Robotic Motion: Theory, Algorithms, and Implementation.* Cambridge, MA: MIT Press.

Chown, E., S. Kaplan, and D. Kortenkamp. 1995. Prototypes, location, and associative networks (plan): Towards a unified theory of cognitive mapping. *Cognitive Science* 19:1–51.

Chrisman, L. 1992. Reinforcement learning with perceptual aliasing: The perceptual distinction approach. In *Proceedings of 1992 AAAI Conference*, Menlo Park, CA. AAAI Press / The MIT Press.

Cid, R.M., C. Parra, and M. Devy. 2002. Visual navigation in natural environments: from range and color data to a landmark-based model. *Autonomous Robots* 13:143– 168.

Cohn, D. 1994. Queries and exploration using optimal experiment design. In J.D. Cowan, G. Tesauro, and J. Alspector (eds.), *Advances in Neural Information Processing Systems 6*, San Mateo, CA. Morgan Kaufmann.

Connell, J. 1990. *Minimalist Mobile Robotics.* Boston: Academic Press.

Coppersmith, D., and S. Winograd. 1990. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation* 9:251–280.

Cover, T.M., and J.A. Thomas. 1991. *Elements of Information Theory.* Wiley.

Cowell, R.G., A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems.* Berlin, New York: Springer Verlag.

Cox, I.J. 1991. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation* 7:193–204.

Cox, I.J., and J.J. Leonard. 1994. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence* 66:311–344.

Cox, I.J., and G.T. Wilfong (eds.). 1990. *Autonomous Robot Vehicles.* Springer Verlag.

Craig, J.J. 1989. *Introduction to Robotics: Mechanics and Control (2nd Edition).* Reading, MA: Addison-Wesley Publishing, Inc. 3rd edition.

Crowley, J. 1989. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 674–680.

Csorba, M. 1997. *Simultaneous Localisation and Map Building.* PhD thesis, University of Oxford.

Davison, A. 1998. *Mobile Robot Navigation Using Active Vision.* PhD thesis, University of Oxford, Oxford, UK.

Davison, A. 2003. Real time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Nice, France.

Dean, L.P. Kaelbling, J. Kirman, and A. Nicholson. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76:35–74.

Deans, M., and M. Hebert. 2000. Invariant filtering for simultaneous localization and mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1042–1047.

Deans, M.C., and M. Hebert. 2002. Experimental comparison of techniques for localization and mapping using a bearing-only sensor. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Sant'Angelo d'Ischia, Italy.

Dearden, R., and C. Boutilier. 1994. Integrating planning and execution in stochastic domains. In *Proceedings of the AAAI Spring Symposium on Decision Theoretic Planning,* pp. 55–61, Stanford, CA.

Dedeoglu, G., and G. Sukhatme. 2000. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS 2000)*, Knoxville, Tenneessee.

DeGroot, Morris H. 1975. *Probability and Statistics.* Reading, MA: Addison-Wesley.

Dellaert, F. 2005. Square root SAM. In S. Thrun, G. Sukhatme, and S. Schaal (eds.), *Proceedings of the Robotics Science and Systems Conference.* Cambridge, MA: MIT Press.

Dellaert, F., D. Fox,W. Burgard, and S. Thrun. 1999. Monte Carlo localization for mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Dellaert, F., S.M. Seitz, C. Thorpe, and S. Thrun. 2003. EM,MCMC, and chain flipping for structure from motion with unknown correspondence. *Machine Learning* 50:45– 71.

Dempster, A.P., A.N. Laird, and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39: 1–38.

Deng, X., and C. Papadimitriou. 1998. How to learn in an unknown environment: The rectilinear case. *Journal of the ACM* 45:215–245.

Devroye, L., L. Györfi, and G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition.* New York, NY: Springer-Verlag.

Devy, M., and H. Bulata. 1996. Multi-sensory perception and heterogeneous representations for the navigation of a mobile robot in a structured environment. In *Proceedings of the Symposium on Intelligent Robot Systems*, Lisboa.

Devy, M., and C. Parra. 1998. 3-d scene modelling and curve-based localization in natural environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Dias, M.B., M. Zinck, R. Zlot, and A. Stentz. 2004. Robust multirobot coordination in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Dickmanns, E.D. 2002. Vision for ground vehicles: history and prospects. *International Journal of Vehicle Autonomous Systems* 1:1–44.

Dickmanns, E.D., and V. Graefe. 1988. Application of monocular machine vision. *Machine Vision and Applications* 1:241–261.

Diebel, J., K. Reuterswärd, J. Davis, and S. Thrun. 2004. Simultaneous localization and mapping with active stereo vision. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Dietterich, T.G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

Dissanayake, G., P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. 2001. A solution to the simultaneous localisation and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation* 17:229–241.

Dissanayake, G., S.B. Williams, H. Durrant-Whyte, and T. Bailey. 2002. Map management for efficient simultaneous localization and mapping (SLAM). *Autonomous Robots* 12:267–286.

Dorf, R.C., and R.H. Bishop. 2001. *Modern Control Systems (Ninth Edition)*. Englewood Cliffs, NJ: Prentice Hall.

Doucet, A. 1998. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Cambridge University, Department of Engineering, Cambridge, UK.

Doucet, A., J.F.G. de Freitas, and N.J. Gordon (eds.). 2001. *Sequential Monte Carlo Methods In Practice*. New York: Springer Verlag.

Driankov, D., and A. Saffiotti (eds.). 2001. *Fuzzy Logic Techniques for Autonomous Vehicle Navigation*, volume 61 of *Studies in Fuzziness and Soft Computing*. Berlin, Germany: Springer-Verlag.

Duckett, T., S. Marsland, and J. Shapiro. 2000. Learning globally consistent maps by relaxation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 3841–3846.

Duckett, T., S. Marsland, and J. Shapiro. 2002. Fast, on-line learning of globally consistent maps. *Autonomous Robots* 12:287 – 300.

Duckett, T., and U. Nehmzow. 2001. Mobile robot self-localisation using occupancy histograms and a mixture of Gaussian location hypotheses. *Robotics and Autonomous Systems* 34:119–130.

Duda, R.O., P.E. Hart, and D. Stork. 2000. *Pattern classification and scene analysis (2nd edition)*. New York: JohnWiley and Sons.

Dudek, G., and D. Jegessur. 2000. Robust place recognition using local appearance based methods. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 466–474.

Dudek, G., and M. Jenkin. 2000. *Computational Principles of Mobile Robotics*. Cambridge CB2 2RU, UK: Cambridge University Press.

Dudek, G., M. Jenkin, E. Milios, and D. Wilkes. 1991. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation* 7:859–865.

Durrant-Whyte, H.F. 1988. Uncertain geometry in robotics. *IEEE Transactions on Robotics and Automation* 4:23 – 31.

Durrant-Whyte, H.F. 1996. Autonomous guided vehicle for cargo handling applications. *International Journal of Robotics Research* 15.

Durrant-Whyte, H., S. Majumder, S. Thrun, M. de Battista, and S. Scheding. 2001. A Bayesian algorithm for simultaneous localization and map building. In *Proceedings of the 10th International Symposium of Robotics Research (ISRR'01)*, Lorne, Australia.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Transactions on Robotics and Automation* pp. 249–265.

Eliazar, A., and R. Parr. 2003. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. IJCAI.

Eliazar, A., and R. Parr. 2004. DP-SLAM 2.0. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, New Orleans, USA.

Elliott, R.J., L.Aggoun, and J.B. Moore. 1995. *Hidden Markov Models: Estimation and Control.* New York, NY: Springer-Verlag.

Engelson, S., and D. McDermott. 1992. Error correction in mobile robot map learning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2555–2560.

Etter, P.C. 1996. *Underwater Acoustic Modeling: Principles, Techniques and Applications.* Amsterdam: Elsevier.

Featherstone, R. 1987. *Robot Dynamics Algorithms.* Boston, MA: Kluwer Academic Publishers.

Feder, H.J.S., J.J. Leonard, and C.M. Smith. 1999. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research* 18:650–668.

Feldman, D. 1962. Contributions to the two-armed bandit problem. *Ann. Math. Statist* 33:847–856.

Feller,W. 1968. *An Introduction To Probability Theory And Its Applications (3rd edition)x.* Quinn-Woodbine.

Feng, L., J. Borenstein, and H.R. Everett. 1994. "Where am I?" Sensors and methods for autonomous mobile robot positioning. Technical Report UM-MEAM-94-12, University of Michigan, Ann Arbor, MI.

Fenwick, J., P. Newman, and J. Leonard. 2002. Collaborative concurrent mapping and localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Ferguson, D., T. Stentz, and S. Thrun. 2004. PAO* for planning with hidden state. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Fischler, M.A., and R.C. Bolles. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24:381–395.

Folkesson, J., and H.I. Christensen. 2003. Outdoor exploration and SLAM using a compressed filter. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 419–427.

Folkesson, J., and H.I. Christensen. 2004a. Graphical SLAM: A self-correcting map. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Folkesson, J., and H.I. Christensen. 2004b. Robust SLAM. In *Proceedings of the International Symposium on Autonomous Vehicles*, Lisboa, PT.

Fox, D. 2003. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research* 22:985 – 1003.

Fox, D., W. Burgard, F. Dellaert, and S. Thrun. 1999a. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Orlando, FL. AAAI.

Fox, D., W. Burgard, H. Kruppa, and S. Thrun. 2000. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots* 8.

Fox, D., W. Burgard, and S. Thrun. 1998. Active Markov localization for mobile robots. *Robotics and Autonomous Systems* 25:195–207.

Fox, D., W. Burgard, and S. Thrun. 1999b. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research (JAIR)* 11:391–427.

Fox, D., W. Burgard, and S. Thrun. 1999c. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research* 11:391–427.

Fox, D., J. Ko, K. Konolige, and B. Stewart. 2005. A hierarchical Bayesian approach to mobile robot map structure learning. In P. Dario and R. Chatila (eds.), *Robotics Research: The Eleventh International Symposium, Springer Tracts in Advanced Robotics (STAR)*. Springer Verlag.

Freedman, D., and P. Diaconis. 1981. On this histogram as a density estimator: L_2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 57:453–476.

Frese, U. 2004. *An O(logn) Algorithm for Simultaneous Localization and Mapping of Mobile Robots in Indoor Environments.* PhD thesis, University of Erlangen-Nürnberg, Germany.

Frese, U., and G. Hirzinger. 2001. Simultaneous localization and mapping—a discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, pp. 17–26, Seattle,WA.

Frese, U., P. Larsson, and T. Duckett. 2005. A multigrid algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics.* To appear.

Frueh, C., and A. Zakhor. 2003. Constructing 3d city models by merging groundbased and airborne views. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, Wisconsin.

Gat, E. 1998. Three-layered architectures. In D. Kortenkamp, R.P. Bonasso, and R. Murphy (eds.), *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, pp. 195–210. Cambridge, MA: MIT Press.

Gat, E., R. Desai, R. Ivlev, J. Loch, and D.P. Miller. 1994. Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation* 10: 490–503.

Gauss, K.F. 1809. *Theoria Motus Corporum Coelestium (Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections).* Republished in 1857, and by Dover in 1963: Little, Brown, and Co.

Geffner, H., and B. Bonet. 1998. Solving large POMDPs by real time dynamic programming. In *Working Notes Fall AAAI Symposium on POMDPs*, Stanford, CA.

Gerkey, B., S. Thrun, and G. Gordon. 2004. Parallel stochastic hill-climbing with small teams. In L. Parker, F. Schneider, and A. Schultz (eds.), *Proceedings of the 3rd InternationalWorkshop on Multi-Robot Systems*, Amsterdam. NRL, Kluwer Publisher.

Gilks, W.R., S. Richardson, and D.J. Spiegelhalter (eds.). 1996. *Markov Chain Monte Carlo in Practice.* Chapman and Hall/CRC.

Goldberg, K. 1993. Orienting polygonal parts without sensors. *Algorithmica* 10:201– 225.

Golfarelli, M., D. Maio, and S. Rizzi. 1998. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 905–911.

Golub, G.H., and C.F. Van Loan. 1986. *Matrix Computations*. North Oxford Academic.

González-Baños, H.H., and J.C. Latombe. 2001. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research.*

Gordon, G. J. 1995. Stable function approximation in dynamic programming. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning.* Also appeared as Technical Report CMU-CS-95-103, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

Greiner, R., and R. Isukapalli. 1994. Learning to select useful landmarks. In *Proceedings of 1994 AAAI Conference*, pp. 1251–1256, Menlo Park, CA. AAAI Press / The MIT Press.

Grunbaum, F.A., M.Bernfeld, and R.E. Blahut (eds.). 1992. *Radar and Sonar: Part II.* New York, NY: Springer-Verlag.

Guibas, L.J., D.E. Knuth, and M. Sharir. 1992. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica* 7:381–413. *See also 17th Int. Coll. on Automata, Languages and Programming,* 1990, pp. 414–431.

Guibas, L.J., J.-C. Latombe, S.M. LaValle, D. Lin, and R. Motwani. 1999. A visibilitybased pursuit-evasion problem. *International Journal of Computational Geometry and Applications* 9:471–493.

Guivant, J., and E. Nebot. 2001. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation* 17:242–257. In press.

Guivant, J., and E. Nebot. 2002. Improving computational and memory requirements of simultaneous localization and map building algorithms. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2731–2736.

Guivant, J., E. Nebot, and S. Baiker. 2000. Autonomous navigation and map building using laser range sensors in outdoor applications. *Journal of Robotics Systems* 17: 565–583.

Guivant, J.E., E.M. Nebot, J. Nieto, and F. Masson. 2004. Navigation and mapping in large unstructured environments. *International Journal of Robotics Research* 23.

Gutmann, J.S., and D. Fox. 2002. An experimental comparison of localization methods continued. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Gutmann, J.-S.,W. Burgard, D. Fox, and K. Konolige. 1998. An experimental comparison of localization methods. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Gutmann, J.-S., and K. Konolige. 2000. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA).*

Gutmann, J.-S., and B. Nebel. 1997. Navigation mobiler roboter mit laserscans. In *Autonome Mobile Systeme.* Berlin: Springer Verlag. In German.

Gutmann, J.-S., and C. Schlegel. 1996. AMOS: Comparison of scan matching approaches for self-localization in indoor environments. In *Proc. of the 1st Euromicro Workshop on Advanced Mobile Robots.* IEEE Computer Society Press.

Hähnel, D.,W. Burgard, B.Wegbreit, and S. Thrun. 2003a. Towards lazy data association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy. Springer.

Hähnel, D., D. Fox, W. Burgard, and S. Thrun. 2003b. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Hähnel, D., D. Schulz, and W. Burgard. 2003c. Mobile robot mapping in populated environments. *Autonomous Robots* 17:579–598.

Hartley, R., and A. Zisserman. 2000. *Multiple View Geometry in Computer Vision.* Cambridge University Press.

Hauskrecht, M. 1997. Incremental methods for computing bounds in partially observable Markov decision processes. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 734–739, Providence, RI.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 13:33–94.

Hayet, J.B., F. Lerasle, and M. Devy. 2002. A visual landmark framework for indoor mobile robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*,Washington, DC.

Hertzberg, J., and F. Kirchner. 1996. Landmark-based autonomous navigation in sewerage pipes. In *Proc. of the First Euromicro Workshop on Advanced Mobile Robots.*

Hinkel, R., and T. Knieriemen. 1988. Environment perception with a laser radar in a fast moving robot. In *Proceedings of Symposium on Robot Control,* pp. 68.1–68.7, Karlsruhe, Germany.

Hoey, J., R. St-Aubin, A. Hu, and C. Boutilier. 1999. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, pp. 279–288.

Höllerer, T., S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. 1999. Exploring MARS: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers and Graphics* 23:779–785.

Howard, A. 2004. Multi-robot mapping using manifold representations. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 4198–4203.

Howard, A., M.J. Mataric, and G.S. Sukhatme. 2002. An incremental deployment algorithm for mobile robot teams. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Howard, A., M.J. Mataric, and G.S. Sukhatme. 2003. Cooperative relative localization for mobile robot teams: An ego-centric approach. In *Proceedings of the Naval Research Laboratory Workshop on Multi-Robot Systems,*Washington, D.C.

Howard, A., L.E. Parker, and G.S. Sukhatme. 2004. The SDR experience: Experiments with a large-scale heterogenous mobile robot team. In *Proceedings of the 9th International Symposium on Experimental Robotics 2004*, Singapore.

Howard, R.A. 1960. *Dynamic Programming and Markov Processes.* MIT Press andWiley.

Iagnemma, K., and S. Dubowsky. 2004. *Mobile Robots in Rough Terrain: Estimation, Motion Planning, and Control with Application to Planetary Rovers.* Springer.

Ilon, B.E., 1975. Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base. United States Patent #3,876,255.

Iocchi, L., K. Konolige, and M. Bajracharya. 2000. Visually realistic mapping of a planar environment with stereo. In *Proceesings of the 2000 International Symposium on Experimental Robotics,* Waikiki, Hawaii.

IRobots Inc., 2004. Roomba robotic floor vac. On the Web at http://www.irobot.com/consu

Isaacs, R. 1965. *Differential Games–A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization.* John Wiley and Sons, Inc.

Isard, M., and A. Blake. 1998. CONDENSATION: conditional density propagation for visual tracking. *International Journal of Computer Vision* 29:5–28.

Jaeger, H. 2000. Observable operator processes and conditioned continuation representations. *Neural Computation* 12:1371–1398.

James, M., and S. Singh. 2004. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*, pp. 417–424.

Jazwinsky, A.M. 1970. *Stochastic Processes and Filtering Theory.* New York: Academic.

Jensfelt, P., D. Austin, O. Wijk, and M. Andersson. 2000. Feature based condensation for mobile robot localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2531–2537.

Jensfelt, P., and H.I. Christensen. 2001a. Active global localisation for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation* 17:748–760.

Jensfelt, P., and H.I. Christensen. 2001b. Pose tracking using laser scanning and minimalistic environmental models. *IEEE Transactions on Robotics and Automation* 17:138–147.

Jensfelt, P., H.I. Christensen, and G. Zunino. 2002. Integrated systems for mapping and localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Julier, S., and J. Uhlmann. 1997. A new extension of the Kalman filter to nonlinear systems. In *International Symposium on Aerospace/Defense Sensing, Simulate and Controls,* Orlando, FL.

Julier, S.J., and J.K. Uhlmann. 2000. Building a million beacon map. In *Proceedings of the SPIE Sensor Fusion and Decentralized Control in Robotic Systems IV, Vol. #4571.*

Jung, I.K., and S. Lacroix. 2003. High resolution terrain mapping using low altitude aerial stereo imagery. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Nice, France.

Kaelbling, L.P., A.R. Cassandra, and J.A. Kurien. 1996. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Kaelbling, L.P., M.L. Littman, and A.R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Kaelbling, L. P., and S. J. Rosenschein. 1991. Action and planning in embedded agents. In *Designing Autonomous Agents*, pp. 35–48. Cambridge, MA: The MIT Press (and Elsevier).

Kalman, R.E. 1960. A new approach to linear filtering and prediction problems. *Trans. ASME, Journal of Basic Engineering* 82:35–45.

Kanazawa, K., D. Koller, and S.J. Russell. 1995. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the 11th Annual Conference on Uncertainty in AI,* Montreal, Canada.

Kavraki, L., and J.-C. Latombe. 1994. Randomized preprocessing of configuration space for fast path planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2138–2145.

Kavraki, L., P. Svestka, J.-C. Latombe, and M. Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12:566–580.

Kearns, M., and S. Singh. 2003. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49:209–232.

Khatib, O. 1986. Real-time obstacle avoidance for robot manipulator and mobile robots. *The International Journal of Robotics Research* 5:90–98.

Kirk, R.E., and P. Kirk. 1995. *Experimental Design: Procedures for the Behavioral Sciences.* Pacific Grove, CA: Brooks/Cole.

Kitagawa, G. 1996. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics* 5:1–25.

Kleinberg, J. 1994. The localization problem for mobile robots. In *Proc. of the 35th IEEE Symposium on Foundations of Computer Science.*

Ko, J., B. Stewart, D. Fox, K. Konolige, and B. Limketkai. 2003. A practical, decisiontheoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3232–3238.

Koditschek, D.E. 1987. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1–6.

Koenig, S., and R.G. Simmons. 1993. Exploration with and without a map. In *Proceedings of the AAAIWorkshop on Learning Action Models at the Eleventh National Conference on Artificial Intelligence (AAAI)*, pp. 28–32. Also available as AAAI Technical Report WS-93-06.

Koenig, S., and R. Simmons. 1998. A robot navigation architecture based on partially observable Markov decision process models. In Kortenkamp et al. (1998).

Koenig, S., and C. Tovey. 2003. Improved analysis of greedy mapping. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Konecny, G. 2002. *Geoinformation: Remote Sensing, Photogrammetry and Geographical Information Systems.* Taylor & Francis.

Konolige, K. 2004. Large-scale map-making. In *Proceedings of the AAAI National Conference on Artificial Intelligence,* pp. 457–463, San Jose, CA. AAAI.

Konolige, K., and K. Chou. 1999. Markov localization using correlation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).*

Konolige, K., D Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, B. Stewart, and R. Vincent. 2005. Centibots: Very large scale distributed robotic teams. In M. Ang and O. Khatib (eds.), *Experimental Robotics: The 9th International Symposium*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag.

Konolige, K., J.-S. Gutmann, D. Guzzoni, R. Ficklin, and K. Nicewarner. 1999. A mobile robot sense net. In *Proceedings of SPIE 3839 Sensor Fusion and Decentralized Control in Robotic Systmes II,* Boston.

Korf, R.E. 1988. Real-time heuristic search: New results. In *Proceedings of the sixth National Conference on Artificial Intelligence (AAAI-88)*, pp. 139–143, Los Angeles, CA 90024. Computer Science Department, University of California, AAAI Press/MIT Press.

Kortenkamp, D., R.P. Bonasso, and R. Murphy (eds.). 1998. *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.* Cambridge, MA:MIT/AAAI Press.

Kortenkamp, D., and T. Weymouth. 1994. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* pp. 979–984, Menlo Park. AAAI, AAAI Press/MIT Press.

Kröse, B., N. Vlassis, and R. Bunschoten. 2002. Omnidirectional vision for appearance-based robot localization. In G.D. Hagar, H.I. Cristensen, H. Bunke, and R. Klein (eds.), *Sensor Based Intelligent Robots (Lecture Notes in Computer Science #2238)*, pp. 39–50. Springer Verlag.

Krotkov, E., M. Hebert, L. Henriksen, P. Levin, M. Maimone, R.G. Simmons, and J. Teza. 1999. Evolution of a prototype lunar rover: Addition of laser-

based hazard detection, and results from field trials in lunar analog terrain. *Autonomous Robots* 7:119–130.

Kuipers, B., and Y.-T. Byun. 1990. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Technical report, Department of Computer Science, University of Texas at Austin, Austin, Texas 78712.

Kuipers, B., and Y.-T. Byun. 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems* pp. 47–63.

Kuipers, B.J., and T.S. Levitt. 1988. Navigation and mapping in large-scale space. *AI Magazine.*

Kuipers, B., J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. 2004. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Kushmerick, N., S. Hanks, and D.S. Weld. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76:239–286.

Kwok, C.T., D. Fox, and M. Meila. 2004. Real-time particle filters. *Proceedings of the IEEE* 92:469 – 484. Special Issue on Sequential State Estimation.

Latombe, J.-C. 1991. *Robot Motion Planning.* Boston, MA: Kluwer Academic Publishers.

LaValle, S.M., H. Gonzalez-Banos, C. Becker, and J.C. Latombe. 1997. Motion strategies for maintaining visibility of a moving target. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Lawler, E.L., and D.E.Wood. 1966. Branch-and-bound methods: A survey. *Operations Research* 14:699–719.

Lebeltel, O., P. Bessière, J. Diard, and E. Mazer. 2004. Bayesian robot programming. *Autonomous Robots* 16:49–97.

Lee, D., and M. Recce. 1997. Quantitative evaluation of the exploration strategies of a mobile robot. *International Journal of Robotics Research* 16:413–447.

Lenser, S., and M. Veloso. 2000. Sensor resetting localization for poorly modelled mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Leonard, J.J., and H.F. Durrant-Whyte. 1991. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation* 7:376–382.

Leonard, J.J., and H.J.S. Feder. 1999. A computationally efficient method for largescale concurrent mapping and localization. In J. Hollerbach and D. Koditschek (eds.), *Proceedings of the Ninth International Symposium on Robotics Research*, Salt Lake City, Utah.

Leonard, J.J., and H.J.S. Feder. 2001. Decoupled stochastic mapping. *IEEE Journal of Ocean Engineering* 26:561–571.

Leonard, J., and P. Newman. 2003. Consistent, convergent, and constant-time SLAM. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, Acapulco, Mexico.

Leonard, J.J., R.J. Rikoski, P.M. Newman, and M. Bosse. 2002a. Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research* 21:943–975.

Leonard, J., J.D. Tardós, S. Thrun, and H. Choset (eds.). 2002b. *Workshop Notes of the ICRAWorkshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4).* Washington, DC: ICRA Conference.

Levenberg, K. 1944. A method for the solution of certain problems in least squares. *Quarterly Applied Mathematics* 2:164–168.

Li, R., F. Ma, F. Xu, L. Matthies, C. Olson, and Y. Xiong. 2000. Large scale mars mapping and rover localization using descent and rover imagery. In *Proceedings of the ISPRS 19th Congress, IAPRS Vol. XXXIII,* Amsterdam.

Likhachev, M., G. Gordon, and S. Thrun. 2004. Planning for Markov decision processes with sparse stochasticity. In L. Saul, Y. Weiss, and L. Bottou (eds.), *Proceedings of Conference on Neural Information Processing Systems (NIPS).* Cambridge, MA: MIT Press.

Lin, L.-J., and T.M. Mitchell. 1992. Memory approaches to reinforcement learning in non-Markovian domains. Technical Report CMU-CS-92-138, Carnegie Mellon University, Pittsburgh, PA.

Littman, M.L., A.R. Cassandra, and L.P. Kaelbling. 1995. Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning.*

Littman, M.L., R.S. Sutton, and S. Singh. 2001. Predictive representations of state. In *Advances in Neural Information Processing Systems 14.*

Liu, J., and R. Chen. 1998. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association* 93:1032–1044.

Liu, Y., and S. Thrun. 2003. Results for outdoor-SLAM using sparse extended information filters. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Lovejoy,W.S. 1991. A survey of algorithmic methods for partially observable Markov decision processes. *Annals of Operations Research* 28:47–65.

Lozano-Perez, T. 1983. Spatial planning: A configuration space approach. *IEEE Transactions on Computers* pp. 108–120.

Lu, F., and E. Milios. 1994. Robot pose estimation in unknown environments by matching 2d range scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR).*

Lu, F., and E. Milios. 1997. Globally consistent range scan alignment for environment mapping. *Autonomous Robots* 4:333–349.

Lu, F., and E. Milios. 1998. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems* 18:249–275.

Lumelsky, S., S. Mukhopadhyay, and K. Sun. 1990. Dynamic path planning in sensorbased terrain acquisition. *IEEE Transactions on Robotics and Automation* 6:462–472.

MacDonald, I.L., andW. Zucchini. 1997. *Hidden Markov and Other Models for Discrete- Valued Time Series.* London, UK: Chapman and Hall.

Madhavan, R., G. Dissanayake, H. Durrant-Whyte, J. Roberts, P. Corke, and J. Cunningham. 1999. Issues in autonomous navigation of underground vehicles. *Journal of Mineral Resources Engineering* 8:313–324.

Mahadevan, S., and L. Kaelbling. 1996. The NSF workshop on reinforcement learning: Summary and observations. *AI Magazine Winter*:89–97.

Mahadevan, S., and N. Khaleeli. 1999. Robust mobile robot navigation using partially-observable semi-Markov decision processes. Internal report.

Makarenko, A.A., S.B.Williams, F. Bourgoult, and F. Durrant-Whyte. 2002. An experiment in integrated exploration. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Marquardt, D. 1963. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal of Applied Mathematics* 11:431–441.

Mason, M.T. 2001. *Mechanics of Robotic Manipulation.* Cambridge, MA: MIT Press.

Mataric, M.J. 1990. A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT, Cambridge, MA. Also available as MIT Artificial Intelligence Laboratory Tech Report AITR-1228.

Matthies, L., E. Gat, R. Harrison, B. Wilcox, R. Volpe, and T. Litwin. 1995. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous Robots* 2:291–311.

Maybeck, P.S. 1990. The Kalman filter: An introduction to concepts. In I.J. Cox and G.T. Wilfong (eds.), *Autonomous Robot Vehicles*. Springer Verlag.

Metropolis, N., and S. Ulam. 1949. The Monte Carlo method. *Journal of the American Statistical Association* 44:335–341.

Mikhail, E. M., J. S. Bethel, and J. C. McGlone. 2001. *Introduction to Modern Photogrammetry.* John Wiley and Sons, Inc.

Mine, H., and S. Osaki. 1970. *Markovian Decision Processes*. American Elsevier.

Minka, T. 2001. *A family of algorithms for approximate Bayesian inference.* PhD thesis, MIT Media Lab, Cambridge, MA.

Monahan, G.E. 1982. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science* 28:1–16.

Montemerlo, M., N. Roy, and S. Thrun. 2003a. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proceedings of the Conference on Intelligent Robots and Systems (IROS)*. Software package for download at www.cs.cmu.edu/carmen.

Montemerlo, M., and S. Thrun. 2004. Large-scale robotic 3-d mapping of urban structures. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Singapore. Springer Tracts in Advanced Robotics (STAR).

Montemerlo, M., S. Thrun, D. Koller, and B. Wegbreit. 2002a. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence,* Edmonton, Canada. AAAI.

Montemerlo, M., S. Thrun, D. Koller, and B. Wegbreit. 2003b. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. IJCAI.

Montemerlo, M., W. Whittaker, and S. Thrun. 2002b. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Moore, A.W. 1991. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the Eighth International Workshop on Machine Learning,* pp. 333–337.

Moravec, H.P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine* 9:61–74.

Moravec, H.P., and M.C. Martin, 1994. Robot navigation by 3D spatial evidence grids. Mobile Robot Laboratory, Robotics Institute, Carnegie Mellon University.

Moutarlier, P., and R. Chatila. 1989a. An experimental system for incremental environment modeling by an autonomous mobile robot. In *1st International Symposium on Experimental Robotics,* Montreal.

Moutarlier, P., and R. Chatila. 1989b. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th Int. Symposium on Robotics Research,* Tokyo.

Mozer,M.C., and J.R. Bachrach. 1989. Discovering the structure of a reactive environment by exploration. Technical Report CU-CS-451-89, Dept. of Computer Science, University of Colorado, Boulder.

Murphy, K. 2000a. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press.

Murphy, K. 2000b. A survey of POMDP solution techniques. Technical report, UC Berkeley, Berkeley, CA.

Murphy, K., and S. Russell. 2001. Rao-Blackwellized particle filtering for dynamic Bayesian networks. In A. Doucet, N. de Freitas, and N. Gordon (eds.), *Sequential Monte Carlo Methods in Practice*, pp. 499–516. Springer Verlag.

Murphy, R. 2000c. *Introduction to AI Robotics.* Cambridge, MA: MIT Press.

Murphy, R. 2004. Human-robot interaction in rescue robotics. *IEEE Systems, Man and Cybernetics Part C: Applications and Reviews* 34.

Narendra, P.M., and K. Fukunaga. 1977. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers* 26:914–922.

Neira, J., M.I. Ribeiro, and J.D. Tardós. 1997. Mobile robot localisation and map building using monocular vision. In *Proceedings of the International Symposium On Intelligent Robotics Systems,* Stockholm, Sweden.

Neira, J., and J.D. Tardós. 2001. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation* 17:890–897.

Neira, J., J.D. Tardós, and J.A. Castellanos. 2003. Linear time vehicle relocation in SLAM. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Nettleton, E.W., P.W. Gibbens, and H.F. Durrant-Whyte. 2000. Closed form solutions to the multiple platform simultaneous localisation and map building (slam) problem. In Bulur V. Dasarathy (ed.), *Sensor Fusion: Architectures, Algorithms, and Applications IV*, volume 4051, pp. 428–437, Bellingham.

Nettleton, E., S. Thrun, and H. Durrant-Whyte. 2003. Decentralised slam with lowbandwidth communication for teams of airborne vehicles. In *Proceedings of the International Conference on Field and Service Robotics,* Lake Yamanaka, Japan.

Newman, P. 2000. *On the Structure and Solution of the Simultaneous Localisation and Map Building Problem.* PhD thesis, Australian Centre for Field Robotics, University of Sydney, Sydney, Australia.

Newman, P.,M. Bosse, and J. Leonard. 2003. Autonomous feature-based exploration. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Newman, P.M., and H.F. Durrant-Whyte. 2001. A new solution to the simultaneous and map building (SLAM) problem. In *Proceedings of SPIE.*

Newman, P., and J.L. Rikoski. 2003. Towards constant-time slam on an autonomous underwater vehicle using synthetic aperture sonar. In *Proceedings of the International Symposium of Robotics Research,* Sienna, Italy.

Neyman, J. 1934. On the two different aspects of the representative model: the method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society* 97:558–606.

Ng, A.Y., A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. 2004. Autonomous inverted helicopter flight via reinforcement learning. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Singapore. Springer Tracts in Advanced Robotics (STAR).

Ng, A.Y., and M. Jordan. 2000. PEGASUS: a policy search method for large MDPs and POMDPs. In *Proceedings of Uncertainty in Artificial Intelligence.*

Ng, A.Y., J. Kim, M.I. Jordan, and S. Sastry. 2003. Autonomous helicopter flight via reinforcement learning. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Proceedings of Conference on Neural Information Processing Systems (NIPS).* Cambridge, MA: MIT Press.

Nieto, J., J.E. Guivant, and E.M. Nebot. 2004. The hybrid metric maps (HYMMs): A novel map representation for dense SLAM. In *Proceedings of the*

International Conference on Robotics and Automation (ICRA).

Nilsson, N.J. 1982. *Principles of Artificial Intelligence.* Berlin, New York: Springer Publisher.

Nilsson, N. 1984. Shakey the robot. Technical Report 323, SRI International, Menlo Park, CA.

Nourbakhsh, I. 1987. *Interleaving Planning and Execution for Autonomous Robots.* Boston, MA: Kluwer Academic Publishers.

Nourbakhsh, I., R. Powers, and S. Birchfield. 1995. DERVISH an office-navigating robot. *AI Magazine* 16.

Nüchter, A., H. Surmann, K. Lingemann, J. Hertzberg, and S. Thrun. 2004. 6D SLAM with application in autonomous mine mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Oore, S., G.E. Hinton, and G. Dudek. 1997. A mobile robot that learns its place. *Neural Computation* 9:683–699.

Ortin, D., J. Neira, and J.M. Montiel. 2004. Relocation using laser and vision. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, New Orleans.

Park, S., F. Pfenning, and S. Thrun. 2005. A probabilistic progamming language based upon sampling functions. In *Proceedings of the ACM Symposium on Principles of Programming Languages (POPL)*, Long Beach, CA. ACM SIGPLAN - SIGACT.

Parr, R., and S. Russell. 1998. Reinforcement learning with hierarchies of machines. In *Advances in Neural Information Processing Systems 10.* Cambridge, MA: MIT Press.

Paskin, M.A. 2003. Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. IJCAI.

Paul, R.P. 1981. *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: MIT Press.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* San Mateo, CA: Morgan Kaufmann.

Pierce, D., and B. Kuipers. 1994. Learning to explore and build maps. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* pp. 1264–1271, Menlo Park. AAAI, AAAI Press/MIT Press.

Pineau, J., G. Gordon, and S. Thrun. 2003a. Applying metric trees to belief-point POMDPs. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Proceedings of Conference on Neural Information Processing Systems (NIPS).* Cambridge, MA: MIT Press.

Pineau, J., G. Gordon, and S. Thrun. 2003b. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI),* Acapulco, Mexico. IJCAI.

Pineau, J., G. Gordon, and S. Thrun. 2003c. Policy-contingent abstraction for robust robot control. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, Acapulco, Mexico.

Pineau, J., M. Montemerlo, N. Roy, S. Thrun, and M. Pollack. 2003d. Towards robotic assistants in nursing homes: challenges and results. *Robotics and Autonomous Systems* 42:271–281.

Pitt, M., and N. Shephard. 1999. Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association* 94:590–599.

Poon, K.-M. 2001. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong Kong University of Science and Technology.

Poupart, P., and C. Boutilier. 2000. Value-directed belief state approximation for POMDPs. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, pp.

279–288.

Poupart, P., L.E. Ortiz, and C. Boutilier. 2001. Value-directed sampling methods for monitoring POMDPs. In *Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI).*

Procopiuc, O., P.K. Agarwal, L. Arge, and J.S. Vitter. 2003. Bkd-tree: A dynamic scalable kd-tree. In T. Hadzilacos, Y. Manolopoulos, J.F. Roddick, and Y. Theodoridis (eds.), *Advances in Spatial and Temporal Databases,* Santorini Island, Greece. Springer Verlag.

Rabiner, L.R., and B.H. Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine* 3:4–16.

Raibert, M.H. 1991. Trotting, pacing, and bounding by a quadruped robot. *Journal of Biomechanics* 23:79–98.

Raibert, M.H., M. Chepponis, and H.B. Brown Jr. 1986. Running on four legs as though they were one. *IEEE Transactions on Robotics and Automation* 2:70–82.

Rao, C.R. 1945. Information and accuracy obtainable in estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society* 37:81–91.

Rao, N., S. Hareti,W. Shi, and S. Iyengar. 1993. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory.

Reed, M.K., and P.K. Allen. 1997. A robotic system for 3-d model acquisition from multiple range images. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Rees,W.G. 2001. *Physical Principles of Remote Sensing (Topics in Remote Sensing).* Cambridge, UK: Cambridge University Press.

Reif, J.H. 1979. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science,* pp. 421–427.

Rekleitis, I.M., G. Dudek, and E.E. Milios. 2001a. Multi-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence* 31:7–40.

Rekleitis, I., R. Sim, G. Dudek, and E. Milios. 2001b. Collaborative exploration for map construction. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation.*

Rencken, W.D. 1993. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 2129–2197.

Reuter, J. 2000. Mobile robot self-localization using PDAB. In *Proceedings of the International Conference on Robotics and Automation (ICRA*).

Rikoski, R., J. Leonard, P. Newman, and H. Schmidt. 2004. Trajectory sonar perception in the ligurian sea. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, Singapore. Springer Tracts in Advanced Robotics (STAR).

Rimon, E., and D.E. Koditschek. 1992. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation* 8:501–518.

Rivest, R.L., and R.E. Schapire. 1987a. Diversity-based inference of finite automata. In *Proceedings of Foundations of Computer Science.*

Rivest, R.L., and R.E. Schapire. 1987b. A new approach to unsupervised learning in deterministic environments. In P. Langley (ed.), *Proceedings of the Fourth International Workshop on Machine Learning,* pp. 364–375, Irvine, California.

Robbins, H. 1952. Some aspects of the sequential design of experiments. *Bulletin of the American Mathemtical Society* 58:529–532.

Rosencrantz, M., G. Gordon, and S. Thrun. 2004. Learning low dimensional predictive representations. In *Proceedings of the Twenty-First International Conference*

on Machine Learning, Banff, Alberta, Canada.

Roumeliotis, S.I., and G.A. Bekey. 2000. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 2985–2992.

Rowat, P.F. 1979. *Representing the Spatial Experience and Solving Spatial problems in a Simulated Robot Environment.* PhD thesis, University of British Columbia, Vancouver, BC, Canada.

Roy, B.V., and J.N. Tsitsiklis. 1996. Stable linear approximations to dynamic programming for stochastic control problems with local transitions. In D. Touretzky, M. Mozer, and M.E. Hasselmo (eds.), *Advances in Neural Information Processing Systems 8.* Cambridge, MA: MIT Press.

Roy, N., W. Burgard, D. Fox, and S. Thrun. 1999. Coastal navigation: Robot navigation under uncertainty in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Roy, N., and G. Dudek. 2001. Collaborative exploration and rendezvous: Algorithms, performance bounds and observations. *Autonomous Robots* 11:117–136.

Roy, N., G. Gordon, and S. Thrun. 2004. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research.* To appear.

Roy, N., J. Pineau, and S. Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000),* Hong Kong.

Roy, N., and S. Thrun. 2002. Motion planning through policy search. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).*

Rubin, D.B. 1988. Using the SIR algorithm to simulate posterior distributions. In M.H. Bernardo, K.M. an DeGroot, D.V. Lindley, and A.F.M. Smith (eds.), *Bayesian Statistics 3.* Oxford, UK: Oxford University Press.

Rubinstein, R.Y. 1981. *Simulation and the Monte Carlo Method.* John Wiley and Sons, Inc.

Russell, S., and P. Norvig. 2002. *Artificial Intelligence: A Modern Approach.* Englewood Cliffs, NJ: Prentice Hall.

Saffiotti, A. 1997. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing* 1:180–197.

Sahin, E., P. Gaudiano, and R. Wagner. 1998. A comparison of visual looming and sonar as mobile robot range sensors. In *Proceedings of the Second International Conference on Cognitive And Neural Systems,* Boston, MA.

Salichs, M.A., J.M. Armingol, L. Moreno, and A. Escalera. 1999. Localization system for mobile robots in indoor environments. *Integrated Computer-Aided Engineering 6:* 303–318.

Salichs, M.A., and L. Moreno. 2000. Navigation of mobile robots: Open questions. *Robotica* 18:227–234.

Sandwell, D.T., 1997. Exploring the ocean basins with satellite altimeter data. http://julius.ngdc.noaa.gov/mgg/bathymetry/predicted/explore.HTML.

Saranli, U., and D.E. Koditschek. 2002. Back flips with a hexapedal robot. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, volume 3, pp. 128–134.

Schiele, B., and J.L. Crowley. 1994. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Schoppers, M.J. 1987. Universal plans for reactive robots in unpredictable environments. In J. McDermott (ed.), *Proceedings of the Tenth International*

Joint Conference on Artificial Intelligence (IJCAI-87), pp. 1039–1046, Milan, Italy. Morgan Kaufmann.

Schulz, D., W. Burgard, D. Fox, and A.B. Cremers. 2001a. Tracking multiple moving objects with a mobile robot. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Kauai, Hawaii.

Schulz, D., W. Burgard, D. Fox, and A.B. Cremers. 2001b. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Schulz, D., and D. Fox. 2004. Bayesian color estimation for adaptive vision-based robot localization. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Schwartz, J.T., M. Scharir, and J. Hopcroft. 1987. *Planning, Geometry and Complexity of Robot Motion.* Norwood, NJ: Ablex Publishing Corporation.

Scott, D.W. 1992. *Multivariate density estimation: theory, practice, and visualization.* John Wiley and Sons, Inc.

Shaffer, G., J. Gonzalez, and A. Stentz. 1992. Comparison of two range-based estimators for a mobile robot. In *SPIE Conf. on Mobile Robots VII*, pp. 661–667.

Sharma, R. 1992. Locally efficient path planning in an uncertain, dynamic environment using a probabilistic model. T-RA 8:105–110.

Shatkay, H, and L. Kaelbling. 1997. Learning topological maps with weak local odometric information. In *Proceedings of IJCAI-97.* IJCAI, Inc.

Siegwart, R., K.O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, R. Philippsen, R. Piguet, G. Ramel, G. Terrien, and N. Tomatis. 2003. A large scale installation of personal robots. Special issue on socially interactive robots. *Robotics and Autonomous Systems 42*: 203–222.

Siegwart, R., and I. Nourbakhsh. 2004. *Introduction to Autonomous Mobile Robots.* Cambridge, MA: MIT Press.

Sim, R., G. Dudek, and N. Roy. 2004. Online control policy optimization for minimizing map uncertainty during exploration. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Simmons, R.G., D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. 2000a. Coordination for multi-robot exploration and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Simmons, R.G., J. Fernandez, R. Goodwin, S. Koenig, and J. O'Sullivan. 2000b. Lessons learned from Xavier. *IEEE Robotics and Automation Magazine* 7:33–39.

Simmons, R.G., and S. Koenig. 1995. Probabilistic robot navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Simmons, R.G., S. Thrun, C. Athanassiou, J. Cheng, L. Chrisman, R. Goodwin, G.-T. Hsu, and H. Wan. 1992. Odysseus: An autonomous mobile robot. *AI Magazine.* extended abstract.

Singh, K., and K. Fujimura. 1993. Map making by cooperating mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 254– 259.

Smallwood, R.W., and E.J. Sondik. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21:1071–1088.

Smith, A.F.M., and A.E. Gelfand. 1992. Bayesian statistics without tears: a samplingresampling perspective. *American Statistician* 46:84–88.

Smith, R.C., and P. Cheeseman. 1986. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research* 5:56–68.

Smith, R.,M. Self, and P. Cheeseman. 1990. Estimating uncertain spatial relationships in robotics. In I.J. Cox and G.T. Wilfong (eds.), *Autonomous Robot Vehicles*, pp. 167– 193. Springer-Verlag.

Smith, S. M., and S. E. Dunn. 1995. The ocean explorer AUV: A modular platform for coastal sensor deployment. In *Proceedings of the Autonomous Vehicles in Mine Countermeasures Symposium*. Naval Postgraduate School.

Smith, T., and R.G. Simmons. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Annual Conference on Uncertainty in AI (UAI)*.

Soatto, S., and R. Brockett. 1998. Optimal structure from motion: Local ambiguities and global estimates. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 282–288.

Sondik, E. 1971. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University.

Stachniss, C., and W. Burgard. 2003. Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI),* Acapulco, Mexico. IJCAI.

Stachniss, C., and W. Burgard. 2004. Exploration with active loop-closing for Fast- SLAM. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Steels, L. 1991. Towards a theory of emergent functionality. In J-A. Meyer and R.Wilson (eds.), *Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

Stentz, A. 1995. The focussed D* algorithm for real-time replanning. In *Proceedings of IJCAI-95*.

Stewart, B., J. Ko, D. Fox, and K. Konolige. 2003. The revisiting problem in mobile robot map building: A hierarchical Bayesian approach. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, Acapulco, Mexico.

Strassen, V. 1969. Gaussian elimination is not optimal. *Numerische Mathematik* 13: 354–356.

Stroupe, A.W. 2004. Value-based action selection for exploration and mapping with robot teams. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Sturges, H. 1926. The choice of a class-interval. *Journal of the American Statistical Association* 21:65–66.

Subrahmaniam, K. 1979. *A Primer In Probability*. New York, NY: M. Dekker.

Swerling, P. 1958. A proposed stagewise differential correction procedure for satellite tracking and prediction. Technical Report P-1292, RAND Corporation.

Tailor, C.J., and D.J. Kriegman. 1993. Exloration strategies for mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 248– 253.

Tanner, M.A. 1996. *Tools for Statistical Inference*. New York: Springer Verlag. 3rd edition.

Tardós, J.D., J. Neira, P.M. Newman, and J.J. Leonard. 2002. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research* 21:311–330.

Teller, S., M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master. 2001. Calibrated, registered images of an extended urban area. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*.

Theocharous, G., K. Rohanimanesh, and S. Mahadevan. 2001. Learning hierarchical partially observed Markov decision process models for robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Thorp, E.O. 1966. *Elementary Probability*. R.E. Krieger.

Thrun, S. 1992. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA. Thrun, S. 1993. Exploration and model building in mobile robot domains. In E. Ruspini (ed.), *Proceedings of the IEEE International Conference on Neural Networks*, pp. 175–180, San Francisco, CA. IEEE Neural Network Council.

Thrun, S. 1998a. Bayesian landmark learning for mobile robot localization. *Machine Learning* 33.

Thrun, S. 1998b. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence* 99:21–71.

Thrun, S. 2000a. Monte Carlo POMDPs. In S.A. Solla, T.K. Leen, and K.-R. Müller (eds.), *Advances in Neural Information Processing Systems 12*, pp. 1064–1070. Cambridge, MA: MIT Press.

Thrun, S. 2000b. Towards programming tools for robots that integrate probabilistic computation and learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA. IEEE.

Thrun, S. 2001. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research* 20:335–363.

Thrun, S. 2002. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel (eds.), *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.

Thrun, S. 2003. Learning occupancy grids with forward sensor models. *Autonomous Robots* 15:111–127.

Thrun, S., M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. 2000a. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *International Journal of Robotics Research* 19:972–999.

Thrun, S., A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. 1998a. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy (eds.), *AI-based Mobile Robots: Case Studies of Successful Robot Systems*, pp. 21–52. Cambridge, MA: MIT Press.

Thrun, S., W. Burgard, and D. Fox. 2000b. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*.

Thrun, S., M. Diel, and D. Hähnel. 2003. Scan alignment and 3d surface modeling with a helicopter platform. In *Proceedings of the International Conference on Field and Service Robotics,* Lake Yamanaka, Japan.

Thrun, S., D. Fox, and W. Burgard. 1998b. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning* 31:29–53. Also appeared in Autonomous Robots 5, 253–271 (joint issue).

Thrun, S., D. Fox, and W. Burgard. 2000c. Monte Carlo localization with mixture proposal distribution. In *Proceedings of the AAAI National Conference on Artificial Intelligence,* Austin, TX. AAAI.

Thrun, S., D. Koller, Z. Ghahramani, H. Durrant-Whyte, and A.Y. Ng. 2002. Simultaneous mapping and localization with sparse extended information filters. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson (eds.), *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics,* Nice, France.

Thrun, S., and Y. Liu. 2003. Multi-robot SLAM with sparse extended information filers. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*, Sienna, Italy. Springer.

Thrun, S., Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. 2004a. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research* 23.

Thrun, S., C. Martin, Y. Liu, D. Hähnel, R. Emery-Montemerlo, D. Chakrabarti, and W. Burgard. 2004b. A real-time expectation maximization algorithm for acquiring multi-planar maps of indoor environments with mobile robots. *IEEE Transactions on Robotics* 20:433–443.

Thrun, S., S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hähnel, M. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker. 2004c. Autonomous exploration and mapping of abandoned mines. *IEEE Robotics and Automation Magazine.* Forthcoming.

Tomasi, C., and T. Kanade. 1992. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision 9:* 137–154.

Tomatis, N., I. Nourbakhsh, and R. Siegwart. 2002. Hybrid simultaneous localization and map building: closing the loop with multi-hypothesis tracking. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Uhlmann, J., M. Lanzagorta, and S. Julier. 1999. The NASA mars rover: A testbed for evaluating applications of covariance intersection. In *Proceedings of the SPIE 13th Annual Symposium in Aerospace/Defence Sensing, Simulation and Controls.*

United Nations, and International Federation of Robotics. 2004. *World Robotics 2004.* New York and Geneva: United Nations.

Urmson, C., B. Shamah, J. Teza, M.D. Wagner, D. Apostolopoulos, and W.R. Whittaker. 2001. A sensor arm for robotic antarctic meteorite search. In *Proceedings of the 3rd International Conference on Field and Service Robotics,* Helsinki, Finland.

Vaganay, J., J. Leonard, J.A. Curcio, and J.S.Willcox. 2004. Experimental validation of the moving long base-line navigation concept. In *Proceedings of the IEEE Conference on Autonomous Underwater Vehicles.*

van der Merwe, R. 2004. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* PhD thesis, OGI School of Science & Engineering.

van der Merwe, R., N. de Freitas, A. Doucet, and E.Wan. 2001. The unscented particle filter. In *Advances in Neural Information Processing Systems 13.*

Vlassis, N., B. Terwijn, and B. Kröse. 2002. Auxiliary particle filter robot localization from high-dimensional sensor observations. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Vukobratovic, M. 1989. *Introduction to Robotics.* Berlin, New York: Springer Publisher.

Wang, C.-C., C. Thorpe, and S. Thrun. 2003. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

Washington, R. 1997. BI-POMDP: Bounded, incremental, partially-observable Markov-model planning. In *Proceedings of the European Conference on Planning (ECP)*, Toulouse, France.

Watkins, C.J.C.H. 1989. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, England.

Weiss, G., C. Wetzler, and E. von Puttkamer. 1994. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 595–601.

Wesley, M.A., and T. Lozano-Perez. 1979. An algorithm for planning collision-free paths among polyhedral objects. *Communications of the ACM* 22:560–570.

West, M., and P.J. Harrison. 1997. *Bayesian Forecasting and Dynamic Models*, 2nd edition. New York: Springer-Verlag.

Wettergreen, D., D. Bapna, M. Maimone, and H. Thomas. 1999. Developing Nomad for robotic exploration of the Atacama Desert. *Robotics and Autonomous Systems* 26: 127–148.

Whaite, P., and F.P. Ferrie. 1997. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19:193–205.

Whitcomb, L. 2000. Underwater robotics: out of the research laboratory and into the field. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 85–90.

Williams, R.J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Williams, S.B. 2001. *Efficient Solutions to Autonomous Mapping and Navigation Problems.* PhD thesis, ACFR, University of Sydney, Sydney, Australia.

Williams, S.B., G. Dissanayake, and H.F. Durrant-Whyte. 2001. Constrained initialization of the simultaneous localization and mapping algorithm. In *Proceedings of the Symposium on Field and Service Robotics.* Springer Verlag.

Williams, S.B., G. Dissanayake, and H. Durrant-Whyte. 2002. An efficient approach to the simultaneous localisation and mapping problem. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 406–411.

Winer, B.J., D.R. Brown, and K.M. Michels. 1971. *Statistical Principles in Experimental Design.* New York: Mc-Graw-Hill.

Winkler, G. 1995. *Image Analysis, Random Fields, and Dynamic Monte Carlo Methods.* Berlin: Springer Verlag.

Wolf, D.F., and G.S. Sukhatme. 2004. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots.*

Wolf, J., W. Burgard, and H. Burkhardt. 2005. Robust vision-based localization by combining an image retrieval system with Monte Carlo localization. *IEEE Transactions on Robotics and Automation.*

Wong, J. 1989. *Terramechanics and Off-Road Vehicles.* Elsevier.

Yamauchi, B., and P. Langley. 1997. Place recognition in dynamic environments. *Journal of Robotic Systems* 14:107–120.

Yamauchi, B., A. Schultz, and W. Adams. 1999. Integrating exploration and localization for mobile robots. *Adaptive Systems 7.*

Yoshikawa, T. 1990. *Foundations of Robotics: Analysis and Control.* Cambridge, MA: MIT Press.

Zhang, N.L., and W. Zhang. 2001. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 14:29–51.

Zhao, H., and R. Shibasaki. 2001. A vehicle-borne system of generating textured CAD model of urban environment using laser range scanner and line cameras. In *Proc. InternationalWorkshop on Computer Vision Systems (ICVS)*, Vancouver, Canada.

Zlot, R., A.T. Stenz, M.B. Dias, and S. Thayer. 2002. Multi-robot exploration controlled by a market economy. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*