Название организации

Диссертация допущена к защите

зав. кафедрой

_____ ФИО зав. кафедрой

«____» _____ 2016 г.

ДИССЕРТАЦИЯ
на соискание ученой степени
МАГИСТРА

Тема: **Тема диссертации**

Направление: 111111 – Название направления

Магистерская программа: 111111 – Название программы

Выполнил студент гр. 1111/1 _____ ФИО автора

Научный руководитель,

д. ф.-м. н., ст. н. с. _____ ФИО руководителя

Рецензент,

д. ф.-м. н., в. н. с. _____ ФИО рецензента

Консультант по вопросам

охраны труда,

к. т. н., доц. _____ ФИО консультанта

Москва – 2016

Оглавление

Введение

Глава 1

Картирование с прямой и обратной моделью сенсора

В этой главе рассматриваются два ставших уже классическими подхода построения карт проходимости. TODO

1.1. Картирование с обратной моделью сенсора

Пусть m_i - клетка карты проходимости m. Будем считать, что каждая клетка m_i - бинарная случайная величина, принимающая два значения: {свободная, занятая}. *Наблюдением* сенсора z_t будем называть измерение и позу датчика в момент времени t, где это измерение было сделано. Вместо того, чтобы напрямую решать задачу картирования, будем искать вероятность занятости некоторой карты m при наблюдениях $z_1, ..., z_T$

$$p(m|z_1, ..., z_T) \equiv p(m|z_{1,T})$$

Основная проблема в том, что карта проходимости m принадлежит пространству большой размерности. Чтобы обойти эту проблему при оценке $p(m|z_{1,T})$, вводится предположение о том, что клетки карты m_i - случайные, независимые величины. Тогда

$$p(m|z_{1,T}) = \prod_i p(m_i|z_{1,T})$$

Таким образом, достаточно понять, как можно оценить вероятность занятости клетки i при известных наблюдениях $z_{1,T}$. Разложим $p(m_i|z_t)$ по правилу Байеса:

$$p(m_i|z_{1,t}) = \frac{p(z_t|m_i, z_{1,t-1})p(m_i|z_{1,t-1})}{p(z_t|z_{1,t-1})} \tag{1.1}$$

В предположении статичности окружения, ясно, что наблюдение z_t не зависит от предыдущих наблюдений, при условии заданной карты проходимости m:

$$p(z_t|m, z_{1,t-1}) = p(z_t|m)$$

Это действительно верно в предположении о статичности окружения. Однако, в этом методе делается более сильное утверждение: *наблюдение z_t не зависит от предыдущих измерений при заданном состоянии клетки m_i, в независимости от состояний соседних клеток.*

$$p(z_t|m_i, z_{1,t-1}) = p(z_t|m_i) \tag{1.2}$$

Подставив в (1.1) формулу выше, снова воспользуемся правилом Байеса

$$p(m_i|z_{1,t}) = \frac{p(z_t|m_i)p(m_i|z_{1,t-1})}{p(z_t|z_{1,t-1})} = \frac{p(m_i|z_t)p(z_t)p(m_i|z_{1,t-1})}{p(m_i)p(z_t|z_{1,t-1})} \tag{1.3}$$

Напомним, что эта формула написана для случая, когда m_i занята. Похожую формулу можно получить для свободной m_i:

$$p(\overline{m_i}|z_{1,t}) = \frac{p(\overline{m_i}|z_t)p(z_t)p(\overline{m_i}|z_{1,t-1})}{p(\overline{m_i})p(z_t|z_{1,t-1})} \tag{1.4}$$

Поделив (1.3) на (1.4) получим

$$\frac{p(m_i|z_{1,t})}{p(\overline{m_i}|z_{1,t})} = \frac{p(m_i|z_t)}{p(\overline{m_i}|z_t)}\frac{p(\overline{m_i})}{p(m_i)}\frac{p(m_i|z_{1,t-1})}{p(\overline{m_i}|z_{1,t-1})} \tag{1.5}$$

Заметим, что $p(\overline{m_i}) = 1 - p(m_i)$. Поэтому, переписав (1.5) в виде log-odds $l(p(m_i)) = \log \frac{p(m_i)}{1-p(m_i)}$, окончательно получаем формулу позволяющую рекурсивно вычислять $l(m_i|z_{1,t})$

$$l(m_i|z_{1,t}) = l(m_i|z_t) - l(m_i) + l(m_i|z_{1,t-1}) \tag{1.6}$$

Алгоритм 1: Картирование с обратной моделью сенсора

Инициализация

for *all* m_i *in* m **do**
  $l_i = \log \frac{p(m_i)}{1-p(m_i)}$
end

Рекурсивное обновление log-odds

for *all* z_t **do**
  for *all* m_i *in* m **do**
    $l_i += \log \frac{p(m_i|z_t)}{1-p(m_i|z_t)} - \log \frac{p(m_i)}{1-p(m_i)}$
  end
end

Получение вероятностей из log-odds

for *all* m_i *in* m **do**
  $p(m_i|z_{1,T}) = 1 - e^{-l_i}$
end

В (1.5) вероятность $p(m_i)$ выражает наши априорные представления о карте, обычно её полагают равной 0.5, считая что какой-либо информации о занятости всей карты в целом нам ничего определенного неизвестно.

1.1.1. Обратная модель сенсора

Величину $p(m_i|z_t)$ называют *обратной моделью сенсора (inverse sensor model)*, выражающую вероятность занятости клетки m_i при известном наблюдении z_t. Эта модель называется обратной, так как она обратна процессу измерения – расстояние до объекта в такой модели определяется показанием сонара, хотя в реальности наблюдение сонара определяется расстоянием до препятствия. Пример того как выглядит эта вероятность можно увидеть на рисунке TODOInverseModelExample.

Заметим, что обратная модель *напрямую не содержит в себе зависимость от соседних клеток*. Это очень важное допущение, которое предпола-

гает, что о состоянии клетки можно сделать выводы основываясь только на *наблюдениях*, независимо от соседних клеток карты. В этом заключается основной проблема этого метода, когда гипотеза о независимости клеток не работает.

1.1.2. Недостатки метода картирования с обратной моделью

Важное предположение о независимости клеток, необходимое для разложения вероятности $p(m)$ на произведение всех $p(m_i)$, является в некоторых случаях существенным. Рассмотрим в качестве примера ситуацию, когда для картирования используется идеальные сонары (без ошибки измерений). В отличие от лазерного дальномера, сонар имеет достаточно широкую область видимости, которая часто представляется в виде конуса, пересекающий множество клеток (см рис TODOInverseSonarsExample). Измерение сонара говорит о следующем - на конце конуса должно находится препятствие, которое должно хорошо объяснять полученное измерение.

На рисунке TODOInverseSonarsExample изображены два сонара, области видимости которых пересекаются в нескольких клетках. Для левого сонара эти клетки принадлежат области препятствия, а для правого - области свободной от препятствия. В результате работы алгоритма мы получим, противоречивую информацию о занятости этих клеток: одно измерение говорит о том, что эти клетки должны быть заняты, другое - что свободны. Легко понять, что эти клетки должны быть свободны, так как есть другие клетки хорошо объясняющие эти 2 измерения (Рис TODOInverseSonarsExample). Однако эта важная дополнительная информация не используется методом, в силу предположения о независимости клеток.

В случае идеальных лазерных дальномеров, у которых очень узкая область видимости, эта проблема практически не касается.

Таким образом, можно сделать вывод, что по крайней мере в случае сонаров, использовать этот метод с предположением о независимости клеток нельзя. Но напрямую вычислить вероятность $p(m|z_{1-t})$ не представляется возможным,

так как пространство всевозможных карт огромно.

Себастьян Трун (Sebastian Thrun) в работе TODOThrun предложил другой метод картирования, лишенный проблем выше.

1.2. Картирование с прямой моделью сенсора

Величина $p(z|m)$ представляет собой вероятностное распределение наблюдения сенсора z при некоторой заданной карте проходимости m, которую будем называть *прямой моделью (forward model)* по аналогии с обратной моделью $p(m|z)$. Интуитивно прямая модель показывает на сколько хорошо наблюдение z объясняет карту проходимости m. Далее приведено подробное описание прямой модели сонара Себастьяна Труна, так как в дальнейшем оно будет использовано в работе.

1.2.1. Прямая модель сонара Труна

Предполагается, что сонар выдает измерения r принадлежащие $[R_{min}, R_{max}]$. Измерение r может быть получено в результате двух сценариев:

1. **Случайный выброс.** С вероятностью p_{rand} сонар выдает случайное значение дальности, распределенное равномерно на $[R_{min}, R_{max}]$. Этот случай описывает возможные ошибочные измерения сенсора, которые могут получится в результате переотражений, зашумлений другими сонарами и т.д.

2. **Обычный случай.** С вероятностью p_{hit} некоторое препятствие, которое находится в области видимости сонара, может отразить волну, таким образом сонар измерит дистанцию до этого препятствия с некоторой гауссовой ошибкой. С вероятностью $1 - p_{hit}$ препятствие волну не отразит, но волна может отразиться от либо следующего препятствия, либо сонар в качестве измерения вернет максимальное R_{max}.

В качестве примера рассмотрим случай на Рис Сонар3Препятсвия. Видно что самое близкое препятствие не лежит в конусе видимости сонара, поэтому при обычном сценарии работы сонара оно не влияет на $p(z|m)$. С вероятностью p_{rand} сонар выдаст ошибочное изменение. Пусть d_1 и d_2 расстояния до первого и второго препятствия в области видимости сонара соответственно. С вероятностью $(1 - p_{rand})p_{hit}$ сонар обнаружит первое препятствие и вернёт $d_1 + e$, где e - гауссова ошибка. Однако с вероятностью $(1 - p_{rand})(1 - p_{hit})$ первое препятствие не будет замечено сенсором. Аналогично с вероятностью $(1 - p_{rand})(1 - p_{hit})p_{hit}$ будет обнаружено второе препятствие. С вероятностью $(1 - p_{rand})(1 - p_{hit})(1 - p_{hit})$ сенсор вернет максимальное измерение R_{max}.

Теперь опишем эту модель формально. Пусть внутри области видимости сонара находятся K препятствий, отсортированных в порядке возрастания дистанции d_k. Через $\{c_*, c_0, c_{1,K}\}$ будем обозначать множество различных сценариев работы сонара, через c_* - случайный выброс, c_0 - измерение R_{max}.

1. Пусть реализовался случай, когда измерение было порождено событием c_k, $k \in \{0, .., K\}$

$$p(z|m, c_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(z-d_k)^2}{\sigma^2}} \tag{1.7}$$

2. Если реализовался случай c_*, то

$$p(z|m, c_*) = \frac{1}{R_{max}} \tag{1.8}$$

Таким образом, распределение $p(z|m)$ является смесью распределений

$$p(z|m) = \sum_{c_i \in \{c_*, c_{0,K}\}} p(z|m, c_k)p(c_k) \tag{1.9}$$

Из рассуждений выше запишем априорную вероятность $p(c_k)$

$$p(c_k) = \begin{cases} p_{rand}, & k = * \\ (1 - p_{rand})(1 - p_{hit})^K, & k = 0 \\ (1 - p_{rand})(1 - p_{hit})^{k-1} p_{hit}, & k > 0 \end{cases} \tag{1.10}$$

Окончательно получаем

$$\begin{aligned} p(z|m) = & \frac{1}{R_{max}} p_{rand} \\ & + \sum_{i \in \{1,\dots,K\}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(z-d_k)^2}{\sigma^2}} (1 - p_{rand})(1 - p_{hit})^{k-1} p_{hit} \\ & + \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(z-R_{max})^2}{\sigma^2}} (1 - p_{rand})(1 - p_{hit})^K \end{aligned} \tag{1.11}$$

1.2.2. Генерация синтетических данных с помощью прямой модели

В процессе разработки алгоритма картирования необходим некоторый массив данных, который позволит эффективно исследовать свойства алгоритма и сравнивать его с другими методами в одних и тех же сценариях. Однако сбор таких данных с реального робота представляет большие трудности:

- Необходимо знать положение робота в пространстве в каждый момент времени, и следовательно определие положения робота каждый раз при измерении, требует либо специального полигона, либо наличие инструментов, которые позволят быстро и точно это сделать. Ясно, что это не всегда возможно.

- Свойства сонара сильно отличаются в зависимости от окружения, поэтому для полноты исследования того или иного метода картирования требуется возможность проводить эксперименты в различных условиях. На реальном роботе это далеко не всегда возможно.

В этой работе предлагается простой метод, который позволяет быстро получать эксперементальные синтетические данные. Ядром этого метода является

прямая модель сонара (1.11).

На вход алгоритму подаётся некоторая карта проходимости, например в формате картинки, с заданным масштабом. Затем можно задать траекторию движения робота и параметры прямой модели сонара, которые позволяют простым образом моделировать различные сценарии работы сонара. Каждая клетка на входной картинке является препятствием определенного размера. Задавая положение робота на этой карте, мы имитируем измерение сонара следующим образом:

1. С вероятностью p_{rand} сонар выдает случайное значение дальности, распределенное равномерно на $[R_{min}, R_{max}]$.

2. Находим в области видимости ближайшее не рассмотренное препятствие. С вероятностью p_{hit} препятствие считается замеченным сонаром, дистанция от сонара до этой клетки с нормальным шумом выдается в качестве измерения.

3. С вероятностью $1-p_{hit}$ препятствие считается незамеченным сонаром, оно помечается как рассмотренное и алгоритм возвращается к предыдущему шагу.

4. Если все препятствия остались незамеченными, то выдается максимальное измерение R_{max}.

Таким образом, наш метод решает указанные выше проблемы и позволяет быстро создать необходимый массив различных данных.

1.2.3. Картирование с прямой моделью

Придумать какие-то слова про картирование с помощью EM алгоритма и какой-нибудь ещё метод. БЛАБЛА

По сравнению с методами картирования, которые используют обратную модель, алгоритмы с прямой моделью сохраняют зависимости между клетками карты, что позволяет лучше восстанавливать карту проходимости. На Рис ОбрПрямДверь видно, что метод из работы РаботаТрун восстановила дверной проем, в отличии от метода обратной модели.

Основной недостаток этих методов заключается в том, что они не работают в режиме реального времени и требуют больших вычислительных ресурсов для поиска оптимальной карты.

Глава 2

Картирование методом стохастического градиента

Используя прямую модель Труна, мы предлагаем метод картирования, который использует преимущества прямой модели, при этом допускает реализацию, работающую в режиме реального времени. Как и раньше, через m будем обозначать карту проходимости. Через S - множество сонаров s. Через $o(m_i)$ будем обозначать занятость клетки m_i: $o(m_i) = 0$ - клетка проходима, $o(m_i) = 1$ - клетка непроходима.

Вначале введем функционал, состоящий из прямой модели сенсора и априорных представлениях об окружении. Затем случайным градиентным спуском будем максимизировать величину этого функционала.

2.1. Функционал правдоподобия карты проходимости

Введем следующий функционал от m при заданных наблюдениях S:

$$\Phi(m, S) = \phi_{sonars}(m, S) + \phi_{occupancy}(m) + \phi_{borders}(m), \tag{2.1}$$

Рассмотрим составляющие функционала (2.1)

1. ϕ_{sonars} - распределение наблюдений сонаров z_s при заданной карте проходимости m.

$$\phi_{sonars}(m, S) = p(z_1, ..., z_S|m) = \prod_{s \in S} p(z_s|m) \tag{2.2}$$

Эта часть функционала (2.1) показывает на сколько хорошо карта m объясняет показание сонаров $s \in S$. В работе в качестве модели сонара

$p(z_s|m)$ используется прямая модель Труна. Вместо (2.2) в окончательной формуле используется логарифм правдоподобия:

$$\phi_{sonars}(m, S) = \log p(z_1, ..., z_S|m) = \sum_{s \in S} \log p(z_s|m) \tag{2.3}$$

2. $\phi_{occupancy}(m)$ отвечает за априорные знания о проходимости карты

$$\phi_{occupancy}(m) = \sum_{m_i} w_o o(m_i) \tag{2.4}$$

В зависимости от значения весового коэффициента w_o можно регулировать наше первоначальное представление о карте, без учета наблюдений сонаров. Например, при $w_o < 0$ и $\phi_{sonars}(m, S) = const$ пустая карта будет максимизировать $\Phi(m, S)$.

3. $\phi_{borders}(m)$ является суммой штрафов для каждой клетки, пропорциональный квадрату числа границ между проходимой и непроходимой областью этой ячейки

$$\phi_{borders}(m) = \sum_{m_i} w_b n^2(m_i) \tag{2.5}$$

Функционал (2.5), как и (2.4), отвечает за наши априорные знания о окружении и имеет простую интерпретацию. Естественно считать что, если большинство соседей заняты, то и рассматриваемая клетка, скорее всего, непроходима. Аналогичную гипотезу можно сформулировать и для незанятых клеток. Поэтому, считая $w_b < 0$, за каждого соседа, который не согласуются с проходимостью, мы штрафуем. Абсолютная величина коэффициента w_b позволяет регулировать относительный вклад в функционал (2.1).

Таким образом задача картирования сводится к задаче максимизации (2.1) по всем возможным картам проходимости

$$m^*(S) = \underset{m}{\operatorname{argmax}} \left(\phi_{sonars}(m, S) + \phi_{occupancy}(m) + \phi_{borders}(m) \right) \tag{2.6}$$

2.2. Алгоритм картирования стохастическим градиента

Оптимизационная задача (2.6) решается методом стохастического градиента. Каждый оптимизационный шаг состоит из следующих действий:

1. Случайным образом выбирается клетка m_i и значение проходимости $o(m_i)$ инвертируется

   $$o^*(m_i) = 1 - o(m_i)$$

2. Для нового значения проходимости клетки m_i пересчитываются $\phi_{sonars}(m, S)$, $\phi_{occupancy}(m)$ и $\phi_{borders}(m)$.

   В слагаемом $\phi_{sonars}(m, S) = \prod_{s \in S} p(z_s|m)$ меняются только те члены произведения, для которых инвертированная клетка лежит в области видимости сенсора. Поэтому можно достаточно быстро пересчитать новое значение $\phi^*_{sonars}(m, S)$.

   В силу того, что слагаемые $\phi_{occupancy}(m)$ и $\phi_{borders}(m)$ являются суммами слагаемых, величина которых зависит только значения проходимости самой клетки и её ближайших соседей, поэтому ясно, что при инвертировании одной клетки можно быстро и понятным способом пересчитать новые значения $\phi^*_{occupancy}(m)$ и $\phi^*_{borders}(m)$.

3. Если $\Phi_{new}(m, S) = \phi^*_{sonars}(m, S) + \phi^*_{occupancy}(m) + \phi^*_{borders}(m) > \Phi(m, S)$, то сохраняем новое значение $o^*(m_i)$ сохраняя новое состояние, иначе возвращаемся в предыдущее состояние.

   Для того чтобы избежать застревания в локальных минимумах, добавляется рандомизация сохранения нового состояния: инвертирование сохраняется с вероятностью p_{rand}, вне зависимости от величины $\Phi_{new}(m, S)$.

Алгоритм 2: Оффлайн версия картирования методом стохастического градиента

Data: m^0- начальное состояние карты, S - множество наблюдений сонаров, k - число оптимизационных шагов

Result: m - карта проходимости

инициализация;

begin

$m = m^0$;

$\phi_{sonars}(m, S), \phi_{occupancy}(m), \phi_{borders}(m)$;

$\Phi(m, S) = \phi_{sonars}(m, S) + \phi_{occupancy}(m) + \phi_{borders}(m)$;

end

for $i = 0; i < k; i = i + 1$ **do**

случайным образом выбирается клетка карты m_i;

$o^*(m_i) = 1 - o(m_i)$;

пересчитать $\phi^*_{sonars}(m, S), \phi^*_{occupancy}(m), \phi^*_{borders}(m)$;

$\Phi^*_(m, S) = \phi^*_{sonars}(m, S) + \phi^*_{occupancy}(m) + \phi^*_{borders}(m) > \Phi(m, S)$;

if $\Phi^*_(m, S) > \Phi_(m, S)$ *или* $rand(0, 1) < p_{rand}$ **then**

$m = m^*$ $\phi_{sonars}(m, S) = \phi^*_{sonars}(m, S)$;

$\phi_{occupancy}(m) = \phi^*_{occupancy}(m)$;

$\phi_{borders}(m) = \phi^*_{borders}(m)$;

$\Phi(m, S) = \Phi^*(m, S)$;

end

end

В этой работе имплементирована оффлайн версия алгоритма 2, которая на вход получает сразу все наблюдения сонаров, и затем оптимизирует карту проходимости.

2.3. Работа в режиме реального времени

Как уже говорилась ранее можно реализовать алгоритм, который работает в режиме реального времени. Далее предлагаются способы того, каким образом это можно сделать.

В режиме реального времени оптимизация ведется одновременно с получением новых данных. Ясно, что если при долгом и непрерывном сборе данных, в определенный момент времени количество наблюдений превысит тот их объем, который возможно обрабатывать в режиме реального времени. Поэтому необходимо выбирать лишь ту часть данных, которые будут алгоритмом в процессе оптимизации. Здесь предлагаются следующие способы выбора наблюдений для использования в оптимизации:

- Использовать скользящее окно и рассматривать последние N_{max} измерений - таким образом мы гарантируем, что каждый пересчет $\phi_{sonars}(m, S)$ не превысит $T_{max} = \Delta_t N_{max}$, где Δ_t - время пересчета одного сонара.

- Для каждой клетки хранить номера измерений сонаров, которые содержат её в поле зрения. Ограничение числа привязанных к каждой ячейке измерений значением N_{max} гарантирует, что для каждой ячейки пересчет $\phi_{sonars}(m, S)$ будет занимать времени не более $T_{max} = \Delta_t N_{max}$. Предлагается выбирать N_{max} последних наблюдений.

- Будем использовать трехмерную сетку пространства (x, y, ϕ), каждый узел этой сетки хранит список наблюдений, которые принадлежат области пространства $(\Delta x, \Delta y, \Delta\phi)$, соответствующей этому узлу. Тогда в каждом узле можно хранить N_{max} последних измерений или даже эти измерения некоторым образом фильтровать. При этом масштаб этой сетки, может не совпадать с масштабом карты проходимости.

Глава 3

Картирование методом градиентного спуска

Заключение

Список литературы

Приложение А

Заголовок приложения