Занятие № 4. SLAM

Одной из наиболее востребованных проблем в робототехнике является локация робота. У этой задачи есть множество различных вариантов: локация на известной карте или на неизвестной; с использованием GPS или без него; с оснащением робота датчиками, считывающими информацию об окружающей среде, или разрешив ему пользоваться только данными, поступающими от его колёс, и так далее.
Разберёмся в одной из задач, связанных с локализацией: Одновременная Локализаци и Построение Карты (Simultanious Localization And Mapping - SLAM). То есть задача формулируется таким образом, что строить карту местности и определять собственное положение на ней необходимо одновременно. Будем рассматривать роботов, которые оснащены датчиками, считывающими информацию об окружающей среде (условимся, что это будет лазерный дальномер).
Такая задача может вставать в различных областях, где применяются роботы: начиная от роботов-пылесосов и заканчивая автономными марсоходами или роботами-исследователями морского дна. Итак, опишем подробно и чётко задачу.
Робот с некоторой периодичностью считывает данные с датчиков лазерных дальномеров. Условимся, что у робота есть определённый “угол обзора” (который может равняться 360 градусам), внутри которого он сканирует окружения, посылая несколько сотен лучей, отстоящих друг от друга на равные углы так, чтобы покрывать весь угол обзора. Таким образом, в каждый период робот получает набор значений, на каком расстоянии от него находятся некоторые препятствия внутри угла обзора

По приходящим “сканам” необходимо строить карту и определять своё положение на ней. Классическим считается алгоритм, использующий расширенный фильтр Калмана.
Чтобы не вдаваться в глубокое математическое описание алгоритма, опишем кратко, как он работает: - Робот только начинает движение. Никаких знаний о карте у него нет. - Робот замечает и определяет первое “препятствие”. В контексте задачи EKF SLAM мы оперируем не со всем сканом, а выделяем из него особые точки (препятствия или фичи).
Погрешность определения препятствия A связана с погрешностью измерительного датчика.

- Робот движется относительно этого препятствия. Во время движения у робота накапливается погрешность одометрии (одометрия - это оценка траектории робота, основанная на вращении колёс робота). Такая оценка не может быть точной, поскольку, например, заранее не известна прочность сцепления робота с поверхностью, по которой он перемещается. Но на данном шаге у робота есть только данные одометрии и положение (вероятнее всего не точное) препятствия A.

- Робот замечает препятствия B и C. Их положение очень неточно в силу накопившейся погрешности. На этом этапе строится матрица ковариации между всеми найденными препятствиями.

- Робот возвращается и вновь определяет препятствие A. Погрешность собственного расположения заметно уменьшается. Немного уменьшается погрешность расположения препятствий B и C (поскольку в матрице ковариаций указана оценка ковариаций всех препятствий)

- Робот проводит измерение препятствия B. Теперь погрешность расположения этого препятствия очень мала. Одновременно с этим уменьшается погрешность препятствия C. Теперь карта расположений препятствий A, B и C достаточно точная (если продолжать проделывать этот алгоритм точность будет возрастать). На точной карте можно точно определять собственное положение.

Самый главный недостаток этого подхода - алгоритмическая сложность. Необходимо строить матрицу ковариации между всеми препятствиями. Логично, что если на местности будет N препятствий, то размер матрицы будет N x N. На практике такой алгоритм не применим, поскольку для его использования необходимо обладать очень высокими вычислительными мощностями (что очень сложно реализовать для небольших роботов). Поэтому существует множество вариантов реализации SLAM. Мы познакомимся ближе с реализацией, которая относится к области SLAM, использующего фильтр частиц и носит название tiny SLAM.
Особенность этого алгоритма в том, что когда приходит новый скан, начинает свою работу скан-матчер, который, использует новый скан и уже построенную карту. Скан-матчер пытается наложить новый скан на уже сформированную карту так, чтобы в результате наложения получилось как можно больше совпадений.
Допустим, на каком-то шаге было предположено, что карта имеет такой вид, и мы находимся в таком положении, как показано на рисунке

На следующем шаге, допустим, мы получаем такой скан:

Далее два положения накладываются друг на друга таким образом, чтобы эти два изображения были наиболее “похожими”

И далее модифицируется представление о карте

Полное представление о существующей реализации алгоритма tiny SLAM в ROS можно увидеть по ссылке http://wiki.ros.org/tiny_slam. Там же есть ссылка на github-репозиторий, где можно ознакомиться с исходным кодом. Настоятельно рекомендуется познакомиться с этим проектом, поскольку в некоторых главах мы будем ссылаться на решения, реализованные в этом проекте.

Links

- Cyrill Stachniss course