SLAM: Map My World

Shane Reynolds

February 16, 2019

Abstract

1 Introduction

Consider a robot in an unknown environment, with no known map. The robot takes sensor readings and experiences control actions. Based on these observations and actions, the robot must construct a map and localise itself within the map. In robotics, this scenario is known as the *simultaneous localisation and mapping problem*, or SLAM. In simpler terms, using sensor readings and control data, SLAM concurrently constructs an environment map, and determines the robots location and orientation within that map. This is an important problem since odometric data is subject to small perturbations which are introduced from wheel slippage and sensor noise - often refered to as odometric drift. Mapping allows a robot to revisit previously mapped terrain and reset any localisation error. Also, location and orientation within a given map are normally used as inputs for higher order functions like path planning. The problem is more difficult to solve than localisation with known poses since high dimensionality of map spaces can often lead to computational intractabilityy. This paper explores two approaches to solving SLAM. The first of these solutions is called FastSLAM which employs a combination of Extended Kalman Filters(EKF) and Monte Carlo Localisation (MCL) to solve the problem. The second approach is called GraphSLAM, which solves the problem by optimising a graph structure built by the algorithm. The paper concludes with an application of GraphSLAM in a Gazebo simulation using an off-the-shelf implementation called RTAB-Map in ROS. The GraphSLAM implementation is tested across two different environments providing opportunities for discussion of the algorithm robustness.

2 Background

Robot localisation aims to determine, for some discrete time step t, a distribution of the robot's pose, x_t, given a series of observations, $z_{1:t}$, control actions, $u_{1:t}$, and a map, m. The localisation problem is often expressed, using conditional probability notation, as follows:

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

Mapping of an environment is the problem of determining a distribution over all possible map configurations, m, given a series of observations, $z_{1:t}$, and known robot poses, $x_{1:t}$. The mapping problem is often expressed, using conditional probability notation, as follows:

$$p(m|z_{1:t}, x_{1:t}) \tag{2}$$

Generally, a robot has neither the map, nor known poses, meaning that posteriors for both the map space and the robot's pose need to be determined. Equations (1) and (2) show these distributions are dependent on each other - evaluation of pose posterior requires the map, and evaluation of map posterior requires pose. This is often referred to as the chicken and egg problem. The main implication is that approaches designed to solve equations (1) and (2) cannot be readily applied in their current forms. Further, the problem can no longer be sufficiently expressed using these

equations. SLAM, as the name suggests, determines these the map, m, and pose, x_t, simultaneously given sensor observations, $z_{1:t}$, and control actions, $u_{1:t}$. Mathematically, this is expressed as:

$$p(x_t, m|z_{1:t}, u_{1:t}) \tag{3}$$

Equation (3) is the equation for the *online* SLAM problem, which is only concerned with determining the robot's current location. Another variation of the SLAM problem, which is more difficult to solve, is the *full* SLAM problem. The *full* SLAM problem seeks to determine the complete pose history of the robot, $x_{1:t}$. This is often expressed as follows:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) \tag{4}$$

Typically, *online* SLAM is used to dynamically localise the robot, whilst *full* SLAM is determine offline and is used to determine where the robot has been. It is not hard to see that the solution to the *online* SLAM problem can be determined by solving the *full* SLAM problem and integrating - the relationship between the two problems can be seen in equation ():

$$p(x_t, m|z_{1:t}, u_{1:t}) = \int_{x_{t-1}} \int_{x_{t-2}} \cdots \int_{x_1} p(x_{1:t}, m|z_{1:t}, u_{1:t}) dx_1 dx_2 \dots dx_{t-1} \tag{5}$$

NEED TO TALK ABOUT CORRESPONDENCES

2.1 Grid Based FastSLAM

FastSLAM was first proposed by XXXX. The central idea was an application of Rao-Blackwellized particle filters to estimate the joint posterior shown in equation (4) (reference to Murphy). The key assumption made in this paper was that if the robot's path, $x_{1:t}$, is known, then landmark locations are conditionally independent of pose (AND CORRESPONDENCES). This assumption allows us to factorise equation (4) as follows:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}) \cdot p(m|x_{1:t}, z_{1:t}) \tag{6}$$

Factorisation effectively breaks the problem up into two smaller problems: estimation of the robot path $p(x_{1:t}|z_{1:t}, u_{1:t})$; and estimation of the robot map $p(m|x_{1:t}, z_{1:t})$. These smaller problems are known as the *localisation* problem, and *mapping* problem, respectively. Considering the problem in this light allows for an iterative algorithm for FastSLAM consisting of two main steps: the robot position is calculated; and then the map is updated based on that position.

Grid based FastSLAM uses the occupancy grid mapping algorithm, which is discussed in XXXX. The localisation problem can be solved using a particle filter, such as Monte Carlo Localisation (MCL), where each particle represents a potential trajectory of the robot. The mapping problem can be computed analytically, assuming $x_{1:t}$ and $z_{1:t}$ are known.

TALK ABOUT HOW THE FASTSLAM ALGORITHM ACTUALLY WORKS.

TALK ABOUT THE NEED FOR LANDMARKS IN VANILLA FASTSLAM, TALK ABOUT THE USE OF OCCUPANCY GRID MAPPING AND WHY IT IS OKAY TO USE THIS IN

TALK ABOUT TWO MAIN APPROACHES TO SOLVING SLAM PROBLEM (THEY ARE SPECIFICALLY NAMED - NEED TO CHECK THIS IN PAPER).

2.1.1 Occupancy Grid Mapping

A common technique for map representation is to decompose a continuous environment, m, into a discrete grid representation such that:

$$m = \sum_i m_i \tag{7}$$

Each cell, m_i, is either occupied or unoccupied. This representation, first introduced by XXXX, is referred to as an occupancy grid map. SHOW FIGURE FOR OCCUPANCY GRID MAP
As previously mentioned, the mapping problem requires estimation of the posterior $p(m|x_{1:t}, z_{1:t})$. The main problem with estimating this posterior using a discrete grid cell representation is the high dimensionality. An example of this can be seen by considering a map broken down into 500 discrete cells. If each cell is either occupied or unoccupied, then the total number of different maps that exist are 2^{500}. Estimating $p(m|x_{1:t}, z_{1:t})$ requires determining the probability for each of the 2^{500} maps. This is computationally intractable for a system which needs to operate in real time. Instead of attempting this, occupancy grid algorithms estimate $p(m_i|x_{1:t}, z_{1:t})$ for each grid cell m_i. Considering the occupancy of each grid cell as independent from other grid cells, whilst not strictly true, is a convenient assumption allowing the expression expression of the posterior $p(m|x_{1:t}, z_{1:t})$ as a product of it's marginals:

$$p(m|x_{1:t}, z_{1:t}) = \prod_i p(m_i|x_{1:t}, z_{1:t}) \tag{8}$$

2.1.2 Grid Based FastSLAM Algorithm

2.2 GraphSLAM

3 Scene and robot configuration

explains how the gazebo world was created by providing an overview of the layout of items in his/her customized Gazebo world. Student also describes the robot's parameters, sensor features, and reasoning on the package structure.

4 Results

Results - The student should include the images for mapping process, final map (2D/3D) for both Gazebo worlds.

5 Discussion

Discussion - The student explains how the procedure went and methodologies to improve it. The student should compare and contrast the performance of RTAB Mapping in different worlds.

6 Future Work