SLAM: Map My World

Shane Reynolds

March 6, 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 ():

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

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:

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

We note that $p(m_i|x_{1:t}, z_{1:t})$ in equation (XXXX) is estimating a fixed binary quantity from a sequence of sensor measurements. According to XXXX beliefs of this type are commonly implemented as *log odds ratios*. We note that the *odds* of a state is defined as the ratio of the probability of the binary event divided by the probability of its negate. Using Bayes' theorem, we express $p(m_i|x_{1:t}, z_{1:t})$ as follows:

$$p(m_i|x_{1:t}, z_{1:t}) = \frac{p(z_t|m_i, z_{1:t-1}, x_{1:t}) \cdot p(m_i|z_{1:t}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \tag{9}$$

Applying the Markov assumption for the first term in the numerator yields:

$$p(m_i|x_{1:t}, z_{1:t}) = \frac{p(z_t|m_i, x_t) \cdot p(m_i|z_{1:t}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \tag{10}$$

We note, applying Bayes' theorem, that:

$$p(z_t|m_i, x_t) = \frac{p(m_i|z_t, x_t) \cdot p(z_t|x_t)}{p(m_i|x_t)} \tag{11}$$

Using equation (XXXX), we can re-express equation (XXXX) as:

$$p(m_i|x_{1:t}, z_{1:t}) = \frac{p(m_i|z_t, x_t) \cdot p(z_t|x_t) \cdot p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i|x_t) \cdot p(z_t|z_{1:t-1}, x_{1:t})} \tag{12}$$

A further application of the Markov assumption allows us to express equation (XXXX) as:

$$p(m_i|x_{1:t}, z_{1:t}) = \frac{p(m_i|z_t, x_t) \cdot p(z_t|x_t) \cdot p(m_i|z_{1:t-1}, x_{1:t-1})}{p(m_i) \cdot p(z_t|z_{1:t-1}, x_{1:t})} \tag{13}$$

Similarly, we can follow the same process for the negate of m_i, $\neg m_i$. This yields the following expression:

$$p(\neg m_i|x_{1:t}, z_{1:t}) = \frac{p(\neg m_i|z_t, x_t) \cdot p(z_t|x_t) \cdot p(\neg m_i|z_{1:t-1}, x_{1:t-1})}{p(\neg m_i) \cdot p(z_t|z_{1:t-1}, x_{1:t})} \tag{14}$$

Hence, after some manipulation, the *odds* can be expressed as:

$$\frac{p(m_i|z_{1:t}, u_{1:t})}{p(\neg m_i|z_{1:t}, u_{1:t})} = \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \cdot \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \cdot \frac{1 - p(m_i)}{p(m_i)} \tag{15}$$

Taking the logarithm of this yields the the *log odds* calculation, l_t, for single grid cell:

$$l_t = \log\left(\frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)}\right) + \log\left(\frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})}\right) + \log\left(\frac{1 - p(m_i)}{p(m_i)}\right) \tag{16}$$

We note that equation (XXXX) can be expressed recursively since the middle term is just l_{t-1}. Further, the inverted final term is simply the inital belief l_0 Hence, we get:

$$l_t = l_{t-1} + \log\left(\frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)}\right) - l_0 \tag{17}$$

The final term in equation (XXXX) is just the initial belief for the grid cell, and the second to last term is referred to as the *inverse sensor model.* The occupancy grid mapping algorithm takes the current pose x_t and current sensor measurements z_t, along with the array of log odds ratios calculated in a previous iteration for each grid cell. This array is denoted as $\{l_{t-1,i}\}$. The algorithm simply loops through all of the grid cells which comprise an environmental map, updating the log odds ratio for each cell. The pseudo code for the occupancy grid algorithm can be seen in XXXX.

2.1.2 Grid Based FastSLAM Algorithm

The grid based FastSLAM algorithm uses a particle filter to

2.2 GraphSLAM

GraphSLAM, in comparison to GBFS, takes a fundamentally different approach to solving the SLAM problem. The main idea behind GraphSLAM is to build up a graph that consists of robot poses, x_t, and map features, m_t. The graph edges represent a series of constraints. The constraints are used to develop a likelihood cost function - minimising this cost function is analogous to determining pose and map features that maximise the likelihood of measurement observations. This is achieved using optimisation techniques, and is performed offline once the graph has been constructed. Picture of the basic set up of GraphSLAM talk about front end and back end.

Figure 1: text

2.2.1 GraphSLAM Front End

The front end of GraphSLAM is concerned with constructing the graph and determining the constraints. A robot nagivating some 2D topology typically has a pose comprised of position coordinates x, and y, as well as orientation θ. To make things simple, we will encapsulate these three parameters in a single pose vector $x_t = \begin{bmatrix} x & y & \theta \end{bmatrix}$. To simplify things further often a single parameter is used to represent state variables that contain one or more poses x with the map m. Thrun and Montemerlo define $y_{0:t}$ to be a vector composed of the path $x_{0:t}$ and the map m, whereas y_t is simply composed as the momentary pose at time t and the map m:

$$y_{0:t} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_t \\ m \end{bmatrix} \quad \text{and} \quad y_t = \begin{bmatrix} x_t \\ m \end{bmatrix} \tag{18}$$

As the robot moves around the environment, it creates pose nodes x_t, and observes map features creating feature nodes m_t.
Edges between two corresponding pose nodes are referred to as motion constraints, and are derived from odometry sensor data - these are represented using notation u_t. Edges between poses and map features are referred to as measurement constraints, and are derived from exteroceptive sensors - these are represented using notation z_t. An example of a graph composed of pose nodes and map feature nodes can be seen in Figure XXXX. The robot moves from pose x_0 to pose x_1, and so on. As the robot moves, sensors detect map features m_1 and m_2. Odometric measurements are captured

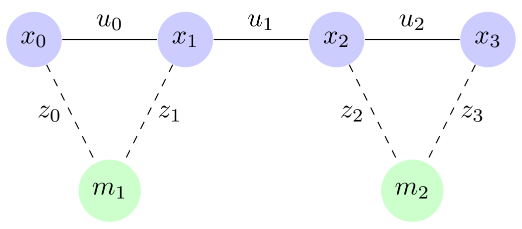

Figure 2: text

and used to create edges u_0, u_1, and u_2. Additionally, exteroceptive sensors capture the distance from the robot location to the map features resulting in dashed line edges z_0, z_1, z_2, and z_3. The edges form a set of *information constraints*. To compute a map posterior, GraphSLAM linearises the set of constraints which results in a sparse information matrix and an information vector.

2.2.2 GraphSlam Back End

2.2.3 RTAB Map Impementation

3 Scene and robot configuration

3.1 Robot Model

A robot model with simple geometries was selected for use in the SLAM simulation. The chassis is a rectangular prisim, and two wheels are located either side of the chassis centre forming a differential drive configuration - the robot model can be seein in Figure XXXX. An RGBD camera, front mounted on the chassis, provides 3D point cloud data of the environment and is represented by a red cube. Finally, a laser range finder is mounted on top of the robot providing 2D planar depth information.

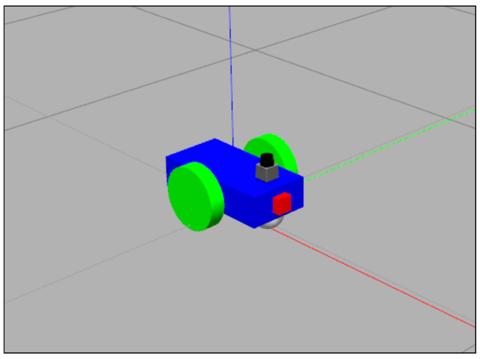

Figure 3: text

Table 1: heading

Link	Parent Link
robot_footprint	odom
chassis	robot_footprint
camera	chassis
hokuyo	chassis
left_wheel	chassis
right_wheel	chassis
camera_link_optical	camera

A full list of the links used to construct the robot model in the `graph_slam_bot.xacro` xml file can be seen in Table XXXX. Pictorially, the 3D coordinate frame hierachy is shown in Figure XXXX. This shows the coordinate frame attached to each link, in addition to the coordinate fram used for odometry. Note that arrows go from parent frames to child frames.

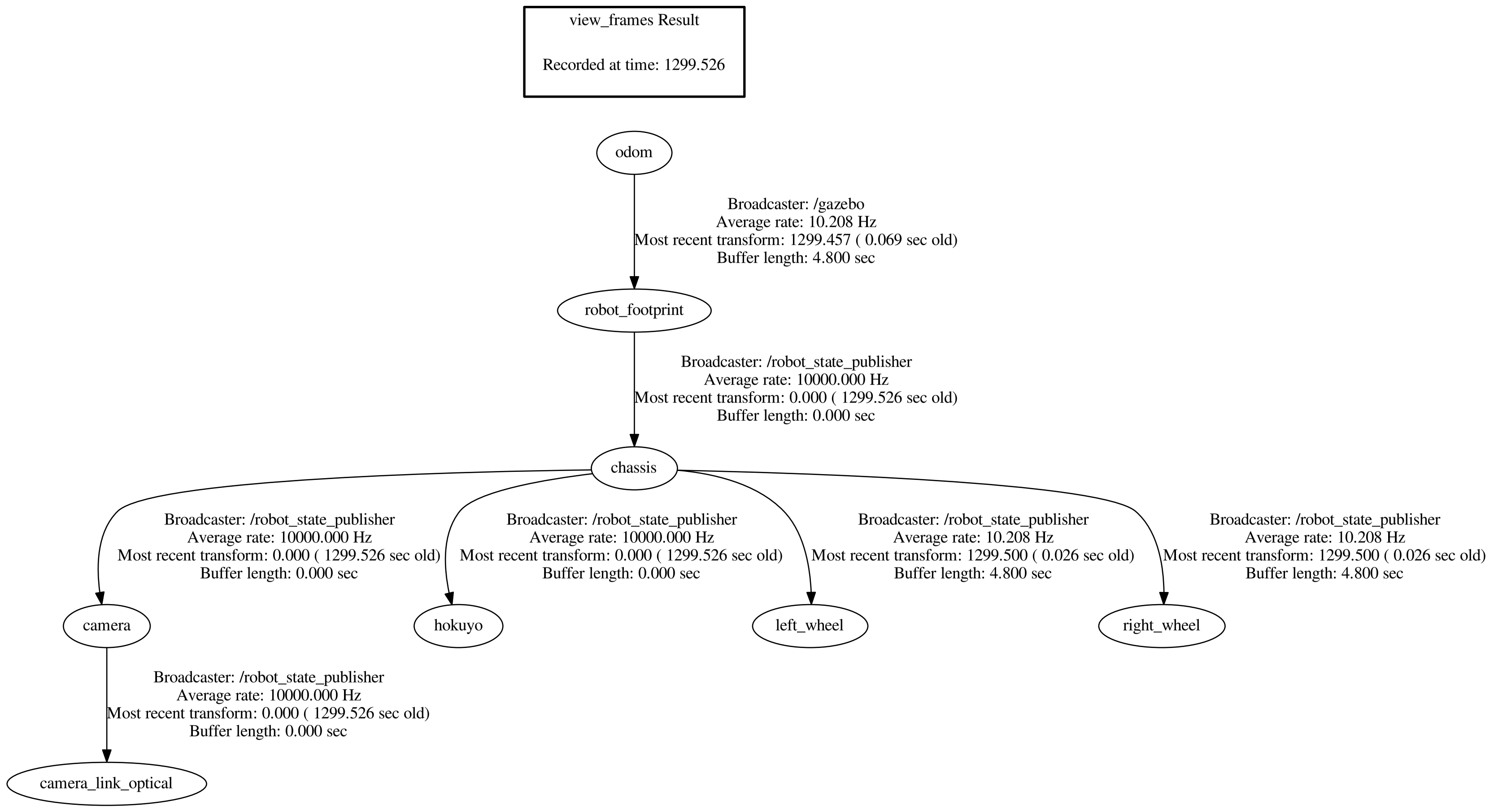
view_frames Result
Recorded at time: 1299.526
odom
Broadcaster: /gazebo
Average rate: 10.208 Hz
Most recent transform: 1299.457 (0.069 sec old)
Buffer length: 4.800 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1299.526 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1299.526 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1299.526 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.208 Hz
Most recent transform: 1299.500 (0.026 sec old)
Buffer length: 4.800 sec
Broadcaster: /robot_state_publisher
Average rate: 10.208 Hz
Most recent transform: 1299.500 (0.026 sec old)
Buffer length: 4.800 sec
camera
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (1299.526 sec old)
Buffer length: 0.000 sec
camera_link_optical

Figure 4: text

RTAB-Map is the package used to equip the robot model with GraphSLAM capabilities. The package uses RGB-D or stereo cameras and Lidar, meaning that there is some configuration of the RTAB-Map node to ensure that it is correctly subscribed to the correct topics published from the robot model sensors. A block diagram of the suggested configuration for robot models employing both RGB-D and Lidar can be seen in Figure XXXX. Notably, there are three topics for subsctiption from the RGB-D camera: RGB Image, Camera Info, and Image Depth. The Laser Scan and Odometry need to be subscribed from the published Lidar and rotary encoders topics, respectively.

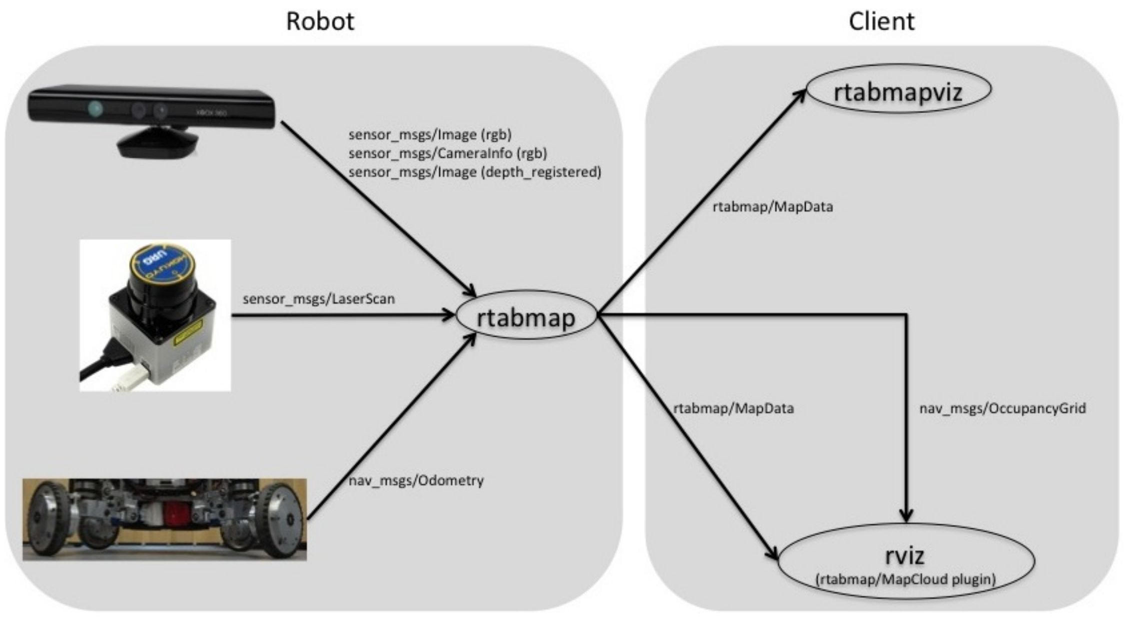

Figure 5: text

Table 2: text

Sensor Topic	Topic Path
RGBD Camera Image	`/camera_ir/rgb/image_raw`
RGBD Camera Depth	`/camera_ir/depth/image_raw`
RGBD Camera Info	`/camera_ir/rgb/camera_info`
Laser Scan	`/graph_slam_bot/laser/scan`
Odometry	

text

Table 3: text

Parameter	Value
`Grid/FromDepth`	`false`
`subscribe_scan`	`true`
`subscribe_depth`	`true`

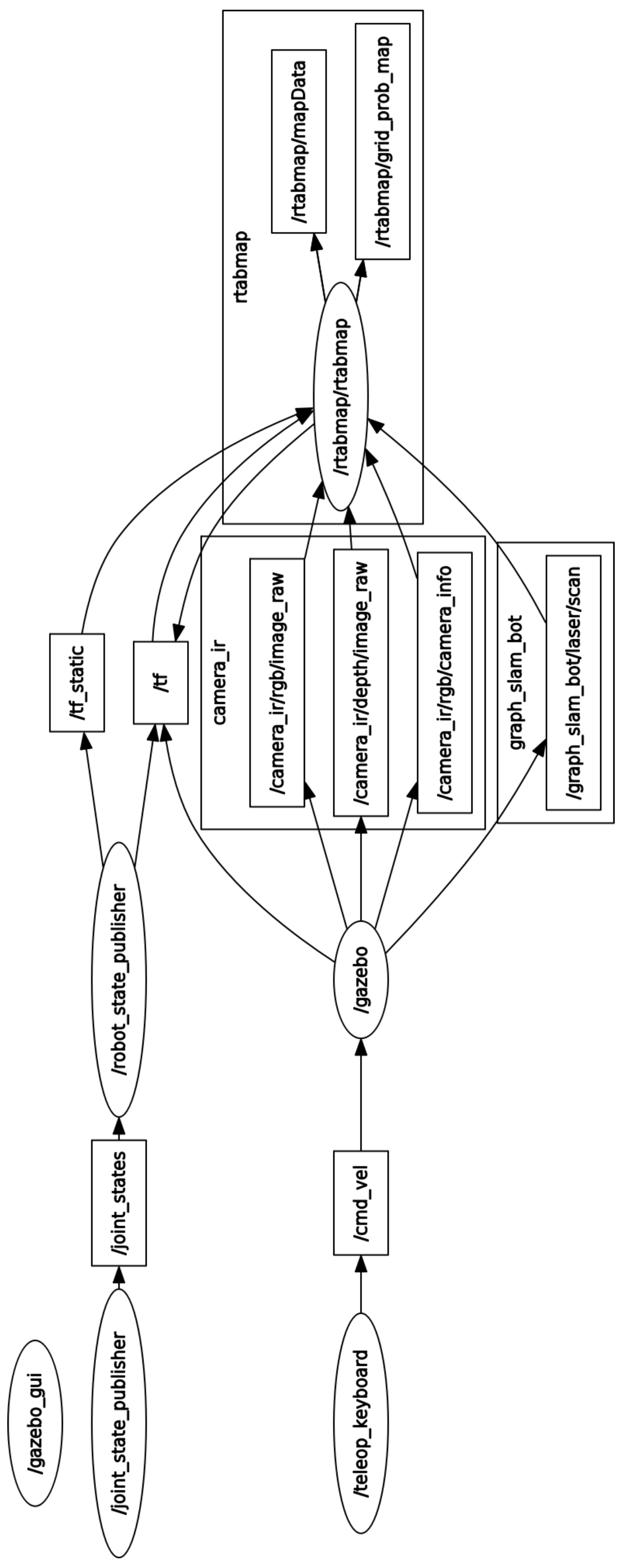

Figure 6: text

WHAT ARE THE MAIN FILES THAT WERE REQUIRED TO CONFIGURE THE ROBOT TALK ABOUT THE LAUNCH FILES THAT WERE CREATED TO LAUNCH EACH OF THE RESPECTIVE NODES

3.2 Environment Model

Rtab-Map was tested in two different environment models: a benchmark model, and an alternative model. Models have differing topologies, sizes, and features.

3.2.1 Benchmark Environment

The benchmark model is of a small apartment mock up, featuring a kitchen and lounge room. The model is highly textured and contains many features - a good representation of real world environments. Rtab-Map should perform well in this environment because it is visually rich providing many opportunities for loop closure. The robot spawn point is in the kitchen. The benchmark environment model can be seen in Figure XXXX.

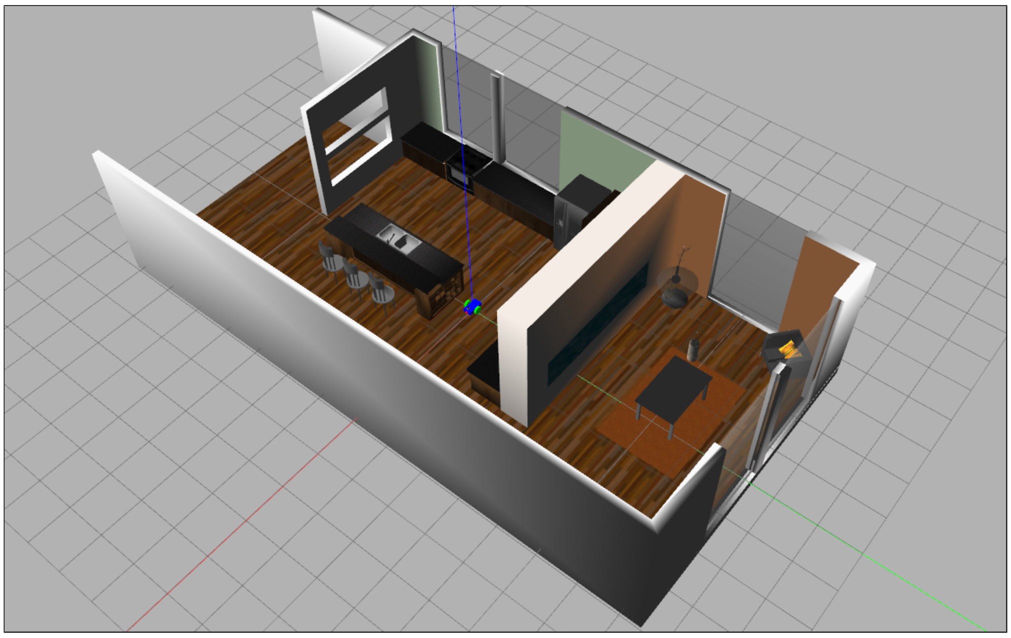

Figure 7: text

3.2.2 Alternative Environment

The alternative model was designed to be very simple. The topology is a basic rectangular room. An important design feature is the variation of features in the environment: some areas have many features, and other areas are featureless. This design choice was made to help evaluate Rtab-Map's performance in sections of the environment that are visually bland, or that have repetitive textures. The environment model was developed using the building editor embedded in Gazebo. The alternative environment model can be seen in Figure XXXX.

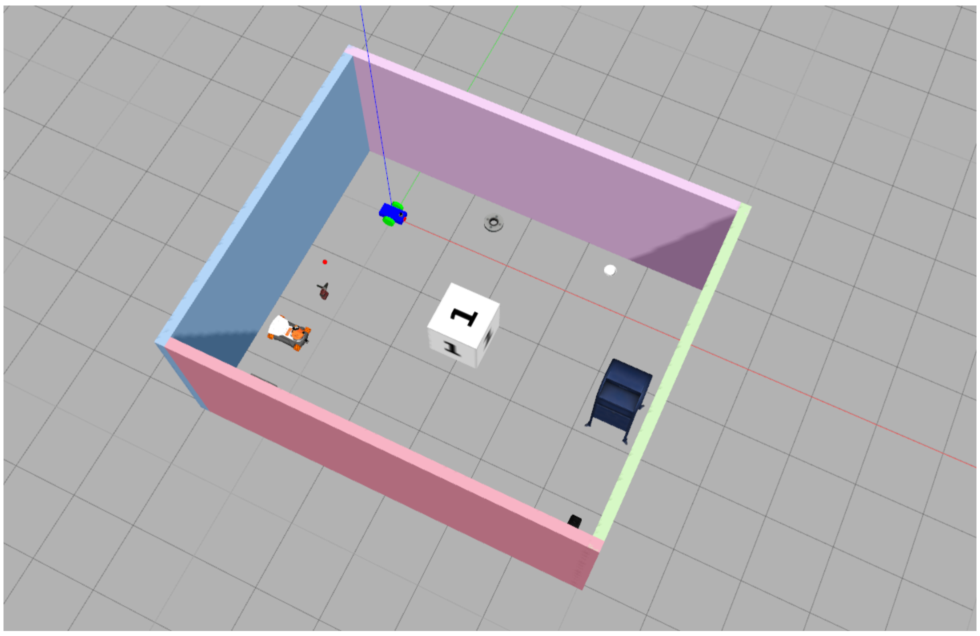

Figure 8: text

4 Results

4.1 Benchmark Environment

A teleop node was set up to manoeuvre the robot around the benchmark environment using keyboard input control. Approximately three laps of the kitchen, balcony, and lounge room areas were performed. Figure XXXX shows a final iteration of the 2D occupancy grid map, in addition to blue line which represents the robot's node graph through the environment, which is optimised by GraphSLAM in Rtab-Map. The red lines connecting nodes show loop closure activity. The red and green coordinate axes show the robot's starting location.

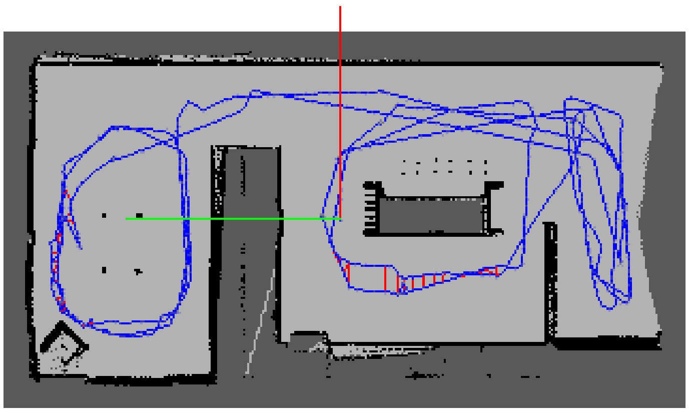

Figure 9: text

As the robot traverses the environment Rtab-Map identifies features from images using the `SURF` detection algorithm which was specified in the mapping node configuration. Figure XXXX shows a single image along the robot path, taken from the Rtab-Map database. The yellow highlights identify features that `SURF` has identified. As images are collected along the robot path, features in new images are compared to features from old images. The pink highlights represent instances where two images have features in common. These common features are used to create neighbouring links between images and loop closures. Figure XXXX shows the constraint view for the same point in time as Figure XXXX - this shows where the neighbouring links and loop closures were created.

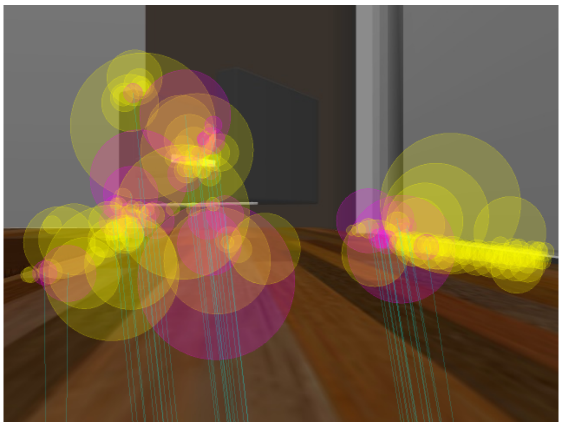
Figure 10: text

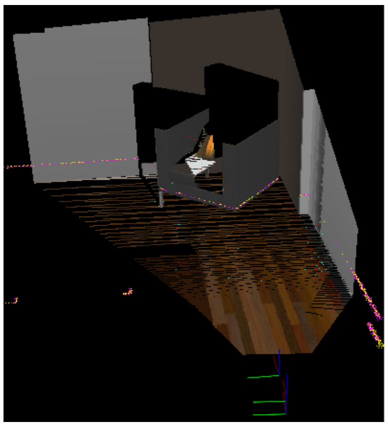
Figure 11: text

NEED TO TALK MORE ABOUT THE DIFFERENT LOOP CLOSURE ACTIVITY AND ACTIVITY WHERE THERE WAS NO LOOP CLOSURE.
Rtab-Map successfully created a 3D map from captured point cloud data captured by the RGBD camera. The 3D environment model was of reasonable fidelity, and well aligned with the 2D occupancy grid. Figure XXXX shows the 3D environment model from Rviz.

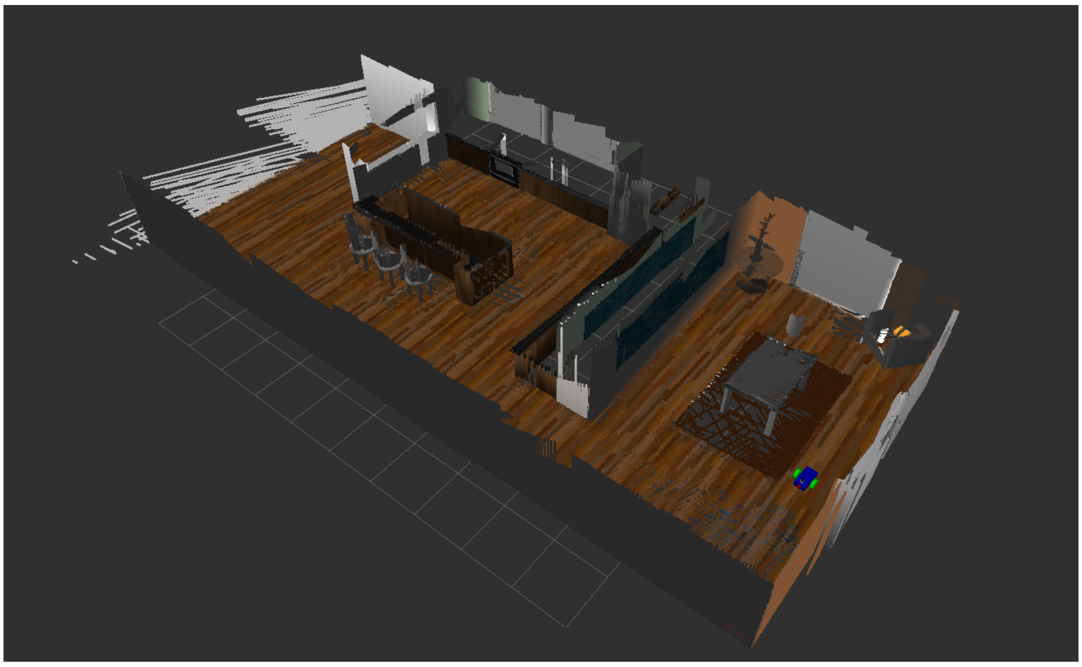
Figure 12: text

The robot was able to successfully localise within the mapped environment, making small corrections to the odometry upon detection of loop closure.

4.2 Alternative Environment

As with the benchmark model, the robot was manoeuvred through the alternative environment using keyboard commands issued to a teleop node. Three laps of the basic environment were undertaken. The final iteration of the 2D occupancy grid map can be seen in Figure XXXX. The node graph used by GraphSLAM in Rtab-Map is represented by the blue line, and loop closure activity is shown in red.

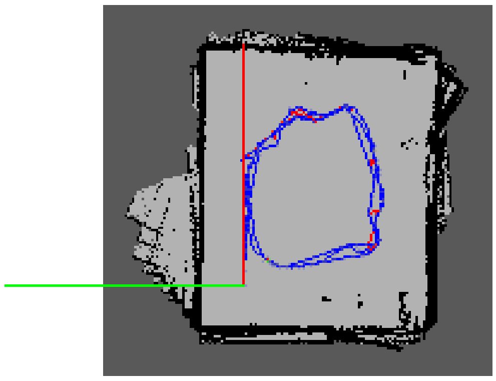

Figure 13: text

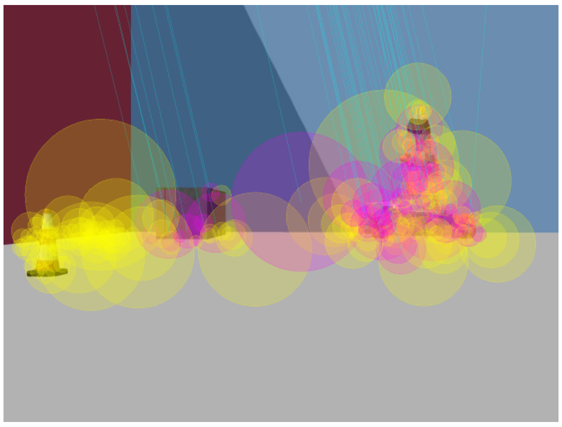

Figure 14: text

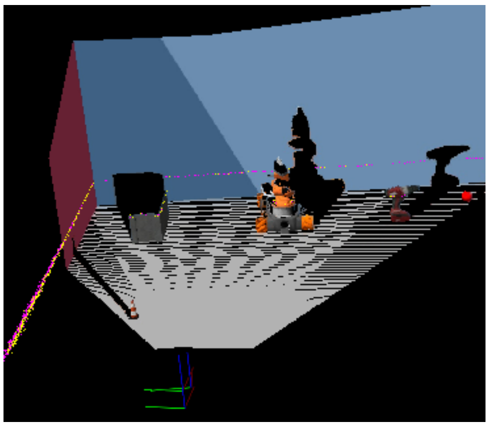

Figure 15: text

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

References

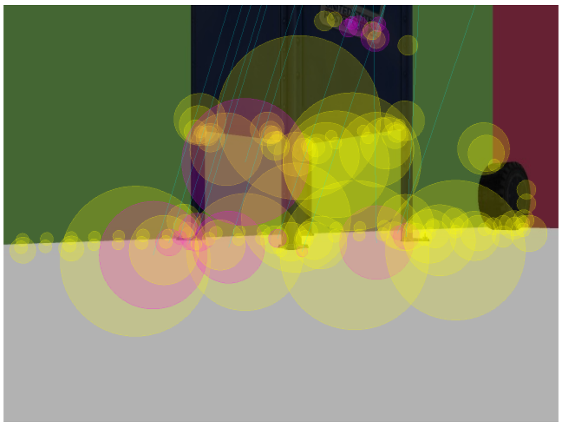

Figure 16: text

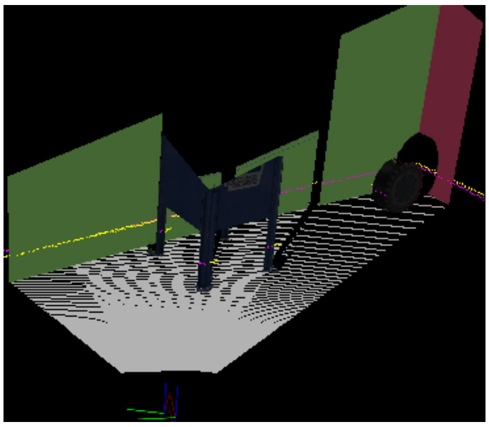

Figure 17: text

Figure 18: text

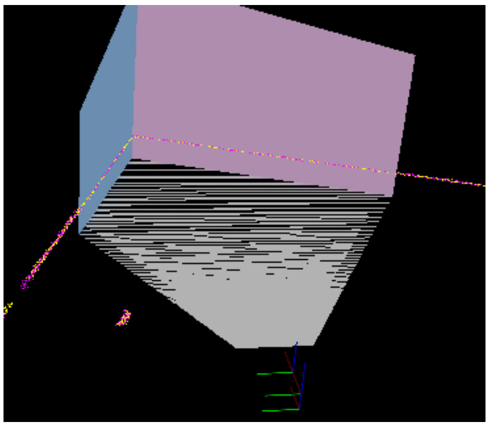

Figure 19: text