Project : Map My World ?

Goutam Gupta

Abstract—In localisation, the robot is provided with the map of its environment. The robot also has access to its movement and sensor data. Using the three of these together, the robot estimates its pose. But, there are many applications where there isn't a known map either because the area is unexplored or because the surrounding changes often and the map may not be up-to-date. In such a case, the robot Will have to construct a map of its environment which leads us to the problem of robotic mapping. SLAM (simultaneous localisation and mapping) helps to simultaneously estimate the robot's pose and produce a map of its environment. There are many challenges associated with SLAM because the map generally lies in continuous space and as a result there are infinitely many variables to describe it. Geometry and nature of space also poses great challenge to mapping problem. In this project we will apply occupancy grid mapping to solve the mapping problem and use it in Graph SLAM methods for simultaneous localisation and mapping. We will also apply real time appearance based mapping (RTAB-MAP) to solve 3D SLAM problems by using the data collected from vision sensors to localise the robot and map its environment.

Index Terms—Robot, IEEEtran, Udacity, Localization, Mapping, SLAM

1 Introduction

The robot mapping problem involves producing a map of the environment given the robot's poses. Mapping assumes that a robot knows its poses and as usual has access to its movement an sensor data. Robot must produce a map of the environment using the known trajectory and measurement data. However, even such a case can be quite uncommon in real world. Often we neither have a map nor we know the robot's pose. This is where SLAM helps.
Mapping involves continuous space and hence it has to deal with infinite number of variables. When this couples with the uncertainties present in the perception, the mapping task becomes even more challenging. Occupancy grid mapping can be used to map any environment by dividing it into finite number of grid cells. By estimating the sets of each individual cells, the map of the environment is estimated.
In Slam, a robot must construct a map of the environment while simultaneously localising itself relative to its map. This problem is more challenging than localisation and mapping, since neither the map nor the robot's poses are provided. With noise in robot's motions and measurements, the map and robot's pose will be uncertain and errors in robot's pose estimate and map will be correlated.The accuracy if the map depends on the accuracy of the localisation and vice versa.Some of the application areas where SLAM plays a critical role are home service robots, self driving vehicle,underground mining and aerial surveillance etc.
SLAM algorithm generally falls into five categories:

- Extended Kalman Filter (EKF) .
- Sparse extended information filter (SEIF)
- Fast SLAM
- Graph SLAM

Fast SLAM uses particle filter approach along with low dimensional Kalman filter to solve the SLAM problem. Adapting this algorithm to grid maps results in a grid based fast SLAM algorithm. Graph SLAM uses constraints to represent relationship between robot's pose and its environment,and then tries to resolve all of these constraints to create most likely map given the data.

In this project, we will learn to utilise RTAB-MAP to solve the problem of 3D SLAM. This algorithm uses data collected form vision sensors to localise the robot and map the environment.A process called loop closure is used to determine whether the robot has seen a location before. RTAB MAP is optimised for large-scale and long term SLAM, by using multiple strategies to allow for loop closures to be done in real time. Major task in this project includes the following:

- Create a ROS package that is able to launch the robot created in localisation project and have it map the surrounding area.
- Use Debug tools and other checks mentioned in the lessons to verify that the robot is mapping correctly
- Create a 3D map for the supplied environment

2 Background

In localisation problem, the goal was to estimate robot's pose in a known environment. In mapping problem, the goal has changed from estimating the robot's pose in a known map to estimating the map itself.Mapping is required in undisclosed areas since the priory map is unavailable.In changing environment,robot should always perform instantaneous mapping in both discovered and undiscovered areas even if the priory map is available. Mapping is also crucial in static environment and robot should still perform instantaneous mapping even if a priory map is available. Sensor measurements are always noisy and accumulate errors and may at times produces a low accuracy map.It is always better to map a static environment and aim to correct the noise to obtain a high accuracy map.

Mapping with mobile robots is a challenging problem for two main reasons:

- Unknown map and poses: We either have to assume the known poses and estimate the map or assume the unknown poses and estimate the map and poses relative to it. Estimating the map in a known poses is a challenging problem, because of the large number of variables. But this can be solved by Occupancy grid mapping algorithm. Estimating the map and poses relative to each other is a more challenging problem which can be solved by SLAM algorithms.

- Huge hypothesis space : Hypothesis space is the space of all possible maps that can be formed during mapping. This space is highly dimensional since maps are defined over a continuous space,especially when robots are deployed in open environment where they have to sense infinite numbers of objects. Occupancy grid mapping provides a discrete approximation of map. But even in this discrete approximation, the space of all possible maps will still be higher. The challenge is to estimate the full posterior map for maps with high dimensional spaces. Moreover, the Bayes filtering approach that is used in localisation to estimate posterior pose diverges this case. Extension of Bayes filtering should be used in mapping to accommodate a huge hypothesis space.

Robot has to combine several instantaneous maps in order to estimate actual map. It encounters following difficulties:

- Size of Environment: Mapping large area is hard because of large amount of data to be processed. It becomes even more difficult when the size of the map becomes bigger than the Robot's perceptual range.
- Noise : Noise always exist in perceptual sensors, odometer sensors and actuators. During mapping noise should e filtered from these sensors and actuators. Moreover, with large noise,mapping becomes more challenging as these noise add up.
- Perceptual ambiguity: The ambiguity occurs when two places look alike. Robot must co relate between those two places which the robot travelled through at different point of time.
- Cycles : When the robot travels in a cyclic manner, its odometry incrementally accumulate error. The error is quite large at the end of the cycle.

2.1 Occupancy grid mapping

Occupancy grid mapping algorithm can estimate the robot's map given known poses and noisy measurement. In most application,odometry data is noisy and robot's poses are unknown to us. Under such situation,mapping usually happens after SLAM. During SLAM, robot builds a map of the environment and localise itself relative to it. After SLAM, occupancy grid mapping algorithm uses the exact robot poses filtered from SLAM. With the known poses from SLAM and noisy measurement, the mapping algorithm generates a map for path planning and navigation. Hence, in mapping, we estimate a posterior over the map given the robot's poses and measurements. 3D maps can also be estimated through the occupancy grid algorithm, but at much higher computational memory because of large number of noisy 3D measurements that need to be filtered out. In this algorithm, the space is uniformly portioned into finite number of grid cells. Each of these grid cells will hold a binary random value that corresponds to the location it covered. Based on the measurement data, these grid space will be filled with 0s(free space) or 1s (occupied space).This algorithm implements binary Bayes filter to estimate the occupancy value of each cell.
The algorithm workflow is summarised below:

- The algorithm takes previous occupancy values of the cell,the poses and the measurements as parameters.
- It loops through the grid cells.
- For each of the cells, the algorithm test if the cells are currently being sensed by the range finder sensors. This will be determined by the cells that fall under the measurement cone (perceptual field of measurement)
- The algorithm will update the cell that fall under the measurement cone by computing the new belief using the binary bayes formula.
- For the cells that do not fall under measurement cone, their occupancy value remain unchanged
- Algorithm returns updated occupancy values of the cell
- Another cycle of iteration happens.

Sometimes,mobile robots might have a combination of various sensors. Mapping with combination of sensors lead to a more precise map. Intuitive approach will be to implement the occupancy grid mapping algorithm and solve for each sensor model. But this will fail, since each and every sensor has different characteristic. In addition to this, the sensors are sensitive to different obstacles. e.g RGBD camera might detect an obstacle at a particular location but this object might not be seen by a Laser LIDAR, and thus LIDAR will process it as a free space. The best approach to a multi sensor fusion problem is to build separate map for each sensor type independent of each other and then integrate them. If the measurement is independent of each other, we can easily combine the maps using Demorgan's law. The resultant map now combines the estimate of occupancy values of each cell. In order to obtain the most likely map, we need to compute the max value of each cell. Another approach would be to perform OR operation between values of each cell. Resulting map,now perfectly combines the measurement from different sensors.

2.2 Grid based FAST SLAM

In SLAM, the input is measurement and control data and the output is map and the trajectory that the robot follows. Online SLAM refers to estimating the current pose and update the map given the current measurement and controls. In online SLAM, previous measurements and controls are not taken into consideration when estimating its current pose and map.Hence, we solve instantaneous poses independently from previous measurements and controls data.
In FULL SLAM, we estimate the entire trajectory and map given all the measurements and controls. With FULL SLAM problem, we estimate all the variables that occur throughout the robot travel time. FULL SLAM is represented by posterior over the entire path along the map given all

the measurements and control up to time t. Online SLAM problem can be obtained by integrating all the previous poses from FULL SLAM problem .
SLAM algorithm have to identify if a relation exists between any newly detected object and previously detected objects. It helps robot to understand if it has been in the same location before. This discrete relation between objects is known by correspondence. The posterior representing the SLAM problem should include the correspondence values. The advantage of adding correspondence values to the posterior is to have the robot better understand where it is located by establishing a relation between objects.
Main challenges to SLAM problem is attributed due to high dimensional continuous parameter space composed of the robot poses and the location of the objects. The discrete parameter space of SLAM problem which is composed out of the correspondence values, is also highly dimensional due to large number of correspondence variables. Correspondence values increases exponentially over time since the robot will keep sensing the environment and relating the newly detected object to the previously detected ones. Thus SLAM algorithm will have to rely on approximation while estimating a posterior in order to conserve memory.
Fast SLAM algorithm solves the full SLAM problem with known correspondences. Fast SLAM estimates a posterior over the trajectory using a particle filter approach. At the same time, Fast SLAM uses a low dimensional extended Kalman filter to solve independent features of the map which are modelled with local Gaussian.Hence in Fast SLAM, MCL algorithm estimates the robot's trajectory whereas EKF estimates features of the map. While Fast SLAM solves FULL SLAM problem by estimating the full robot path, each particle in fast SLAM estimates instantaneous poses and thus fast SLAM also solves Online SLAM problem.
Fast SLAM 1.0 and 2.0 assumes that there are known landmarks, thus modelling an arbitrary environment is not possible. Grid-based fast SLAM helps in robot mapping when landmark positions are unavailable to us. Grid based fast SLAM algorithm will update each particle by solving the mapping with known poses problem using occupancy grid mapping. Grid based fast SLAM involves three techniques: sampling motion,map estimation and importance weight. In sampling motion step, we estimate the current pose given the k-th particle's previous pose and the current control data. In map estimation step, the current map is estimated given the current measurements, the current kth particle's pose and the previous kth particle's map. Importance weight technique involves estimating the current likelihood of the measurement given the current kth particle's pose and the current kth particle's map.The sampling motion and importance weight is solved with MCL algorithm whereas the map estimation technique is solved with the occupancy grid mapping algorithm.

2.3 Graph SLAM

Graph SLAM is a SLAM algorithm that solves the full SLAM problem. This algorithm covers the entire path and map instead of just the most recent map. Graph SLAM algorithm reduces the need for on-board processing capability while improving accuracy over fast SLAM. It uses graphs to represent robot's poses and the error. The soft constraints associated with graph SLAM come sin two forms: Motion constraints and the measurement constraint. Motion constraint work between two successive robot poses whereas measurement constraint work between the robot's pose and a feature in the environment.
As the robot moves around, more and more nodes are added to the graph and over time the graph constructed by the mobile robot becomes very large in size. Every motion and measurement constraint pose the system closer to that constraint's desire state. Since, the measurement and motion are uncertain,constraints will conflict with each other and there will always be some error present. The goal in this approach is to find the node configuration that minimises the overall error present in all the constraints.
The goal of the graph SLAM is to create a graph of all Robot pose and features accounted in the environment and find the most likely robot's path and map of the environment. This task can be broken into two sections: Front end and the Back end. Front end deals with constructing the graph using odometry and sensory measurements collected by the robot. This involves interpreting the sensory data,creating a graph and continuing to add nodes and edges to it as the robot traverses the environment. Front end can differ greatly from application to application depending on the desired goal including accuracy, sensor used and the other factors. Front end also deals with the problem of solving the data association problem i.e accurately identifying whether features in the environment have been previously seen.
The input to the back end is the completed graph with all of the constraints. The output is the most probable configuration of robot poses and map features. Back end is an optimisation process that takes all of the constraints and finds the system configuration that produces the smallest error. Back end is lot more consistent across applications.Back end and front end performs iteratively. At the core of the Graph SLAM is graph optimisation which is the process of minimising the error present in all of the constraints in the graph. Likelihood tries to estimate the most likely configuration of state and features locations given the motion and measurement observation.

Optimisation problem in Graph SLAM can be solved by constructing an information matrix and information vector which is defined over the poses and features,containing the best estimate of each. Most motion and measurement constraints are non-linear and must be linearized before they can be added to the information matrix and information vector. The linearization process is repeated several times, each time using better and better estimate to linearize the constraints.

2.4 3D SLAM with RTAB-MAP

Real time appearance based mapping uses data collected form vision sensors to localise the robot and map the environment. A process called loop closure is used to determine whether the robot has seen a location before. Front end of RTAB-MAP focuses on sensor data used to obtain the constraints that are used for feature optimisation approaches. Although landmark constraints are used for other graphs

SLAM methods like 2D graphs SLAM, RTAB-Map doesn't use them. This algorithm considers only odometry constraints and loop closure constraints. Front end also involves graph management, which includes node creation and loop closure detection using bag-of-worlds.
Back end of RTAB-Map includes graph optimisation and an assembly of an occupancy grid from the data of the graph. Loop closure is the process of finding a match between the current and previously visited location in SLAM. There are two types of loop closures : Local and Global. In local loop closure, SLAM methods find match between a new observation and a limited map region. The size and location of the limited map region is determined by the uncertainty associated with the robot's position. This type of approach fails if the estimated position is incorrect.
In Global loop closure, a new location is compared with previously viewed locations. If no match is found, the new location is added to the memory. As the map grows and more location are added, the amount of time to check whether the location has been previously seen increases linearly. If the time it takes to search and compare the new image to the one stored in memory location becomes larger than the acquisition time, the map becomes ineffective. RTAB-Map uses a global loop closure combined with other techniques to ensure that the loop closure process happens in real time.In RTAB-Mapping loop closure is detected using a bag-of-words approach which is commonly used in vision mapping.
In RTAB-Map,whenever a new image is acquired a new node is created in STM. When creating node,features are extracted and compared to the vocabulary to find all of the words in the image,creating a bag of words for the node. Nodes are assigned a weight in the STM based on how long the robot spent in the location. When STM exceeds its size, the oldest node is moved to the WM(working memory) to be considered for loop closure detection. Loop closure happens in WM. When time required to process new data reaches a threshold, some nodes are transferred from WM to LTM(long-time memory). If a loop closure is detected ,neighbours in LTM of an old node can be transferred back to the WM.
RTAB-Map optimisation is done by adding a new constraints to the map's graph when a loop closure hypothesis is accepted. The graph optimiser minimises the errors in the map.

3 Scene and Robot Configuration

3.1 Scene Configuration

Robot mapping and simulation is tried with the given kitchen environment. A snapshot of the kitchen environment with the bot developed in previous project is shown in the figure 1 below:

3.2 Robot Configuration

The bot developed in the previous project is being used for mapping simulation. This bot has four wheel, a depth camera sensor and a LIDAR sensor. The bot used in previous project had only RGB camera.The bot and the rqt graph showing all the links and joints are shown in the figure 2 and figure 3 respectively:

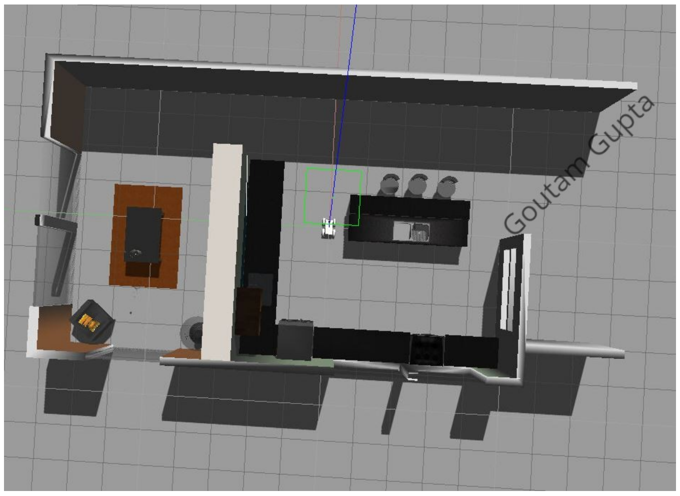

Fig. 1. My bot in the kitchen environment.

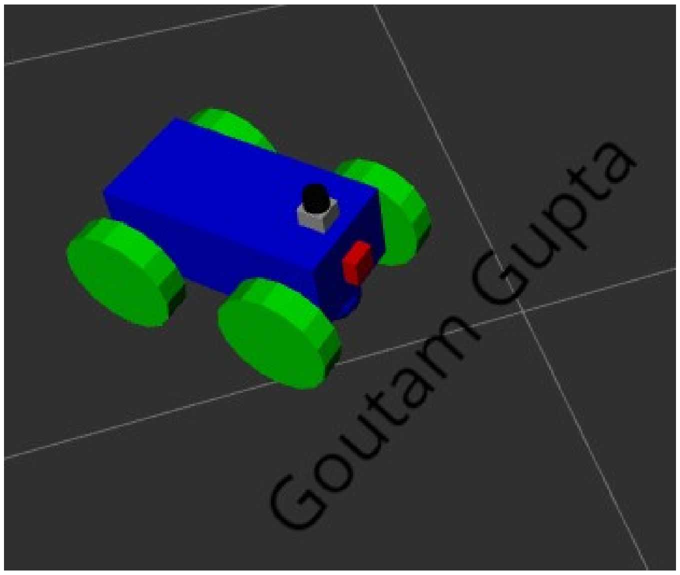

Fig. 2. My robot model.

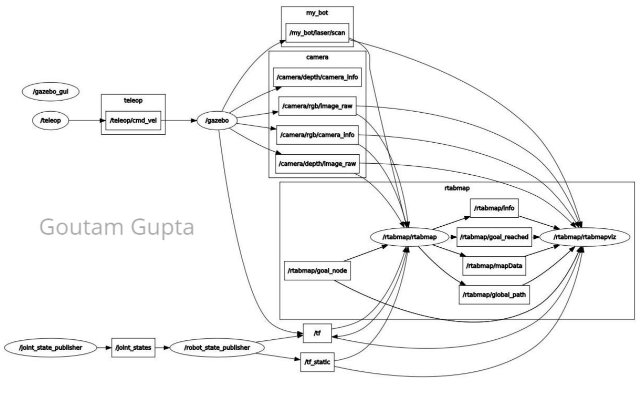

Fig. 3. rqt graph of the robot model.

4 Result

4.1 RTAB MAP loop detection

The rtabmap-databaseViewer is a great tool for exploring the database when we are done generating it. It is isolated from ROS and allows for a complete analysis of the mapping session.This is how we check for loop closures, generate 3D maps for viewing, extract images, check feature mapping rich zones, and much more. Another tool that can be used is rtabmapviz, which is an additional node for real time visualization of feature mapping, loop closures, and more.Rtabmapviz is great to deploy on a real robot during live mapping to ensure that we are getting the necessary features to complete loop closures.

Following figure shows the RTAB map view generated from rtabmapviz tool.

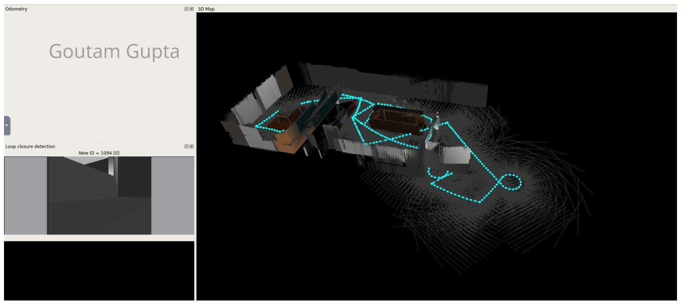

Fig. 4. RTAB map of the bot in kitchen environment.

4.2 Mapping accuracy

As it can be seen from above figure that the robot mapping is quite accurate. The dining table and other prominent objects can be easily distinguishable. Mapping accuracy can be further improved by tuning parameters inside the mapping.launch file. Parameters like minimum inliers to accept loop closures, Hessian threshold, detection rate (rate at which new nodes are added to the map) and maximum visual words per bag etc. can be tuned to obtain most accurate map of the environment.

5 Discussion

Robot mapping in an kitchen environment is performed. As discussed, RTAB mapping uses data captured by depth camera sensors to localise the robot and map the environment. The process uses loop closure technique to determine if location was seen before. It uses a global loop closure approach by feature comparison and bag-of-words approach to ensure that the loop closure happens in real time.

6 Conclusion / Future work

Vision mapping algorithm used for detecting loop closure can be improved further. It can also be improved by adding larger set of feature descriptors to form bag of words. We can also seek to apply sensor fusion technique in order to increase the accuracy of the map. More number of depth cameras or many different kind of vision sensors can be mounted on the bot for sensor fusion. The placement of the sensors can be oriented further to capture the obstacles appropriately, so that occupancy grid mapping becomes more effective. Finally, we can perform the simulation in a more compute capable system to perform robot mapping in a real time environment, where sensor fusion and a more effective technique for loop closure and memory management can be implemented in an efficient manner.