A research on GraphSLAM based on RTAB-MAP-ROS

Ming Lin

Abstract—SLAM is a key problem in robotics. Because of the infinite states and variables, SLAM is a difficult problem. What is more, environment can be dynamic. Once the environment turn in to dynamic, time is also should be considered into algorithm. GraphSLAM is a one of the powerful solutions. It makes a sparse matrix for mapping features and states. The sparse matrix is used for construct constraints for each connections. The reason why use sparse is that system can't save and do not have to save all the connections relationship. It just keep the important weight about the connections which is directly connected. Actually, the connections are formulated into Gaussian distribution. GraphSLAM has the capability for handling full SLAM problem with the constraints sparse matrix's help. The simulation system is constructed based on gazebo and rviz.

Index Terms—Robot, ROS, Udacity, LaTeX, Localization, GraphSLAM, AMCL.

1 Introduction

GRaphSLAM is a powerful algorithm for handling the fullSLAM algorithm. It has two main part. One is front end and the other one is back end. In front end of algorithm, GraphSLAM receives sensor data and odometry data from simulation or hardware. In this part, algorithm construct constraints between robot pose, odometry and features. In this part, algorithm performs a online characteristic and accuracy of the map is not good enough. Back end part is for solving fullSLAM problem. This part is offline process. This process based on the history data include all trajectory and map, then calculating optimal posterior about the robot's trajectory and map. The mapping method is based on binary bayes filter algorithm. [1]

2 Background

2.1 FastSLAM

Before GraphSLAM, the algorithm FastSLAM should be introduced. FastSLAM is a online SLAM algorithm which can estimate the robot pose and map features. It uses monte carlo localization like particle filter to localize the robot and implements low dimensional extended kalman filter to estimate the map's state. FastSLAM can divided into 3 version. One is FastSLAM 1.0, this version is simple and easy to implement, but this version is known to be inefficient since particle filters generate sample inefficiency. FastSLAM 2.0 algorithm overcomes the inefficiency of FastSLAM 1.0 by imposing a different distribution, which results in a low number of particles. Both of FastSLAM 1.0 and FastSLAM 2.0 uses a low dimensional Extended Kalman Filter to estimate the posterior over the map features. It means Both of them should work based on the landmark consistency. The third version of FastSLAM is called Grid-based FastSLAM which adapts FastSLAM to grid maps. Grid-based FastSLAM is non landmark-based algorithm. It means, it don't need any landmarks.FastSLAM uses MCL for localization and estimation. But the key different with previous version is that FastSLAM let each particle holds a full grid map about environment. It just like ordinary particle filter, but the main difference is the occupancy grid map become one of the parameter when resampling. So when resampling step, the particles which have high weight will be selected for next iteration. As a result, the occupancy grids that only appropriate with measurement are remained. ROS package gmapping is based on the Grid-based FastSLAM algorithm.

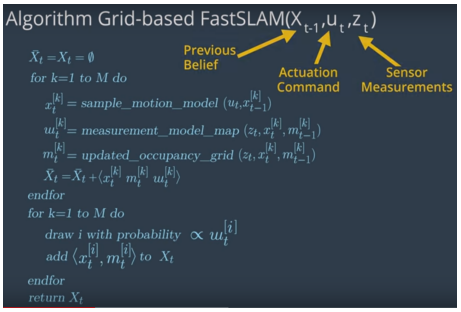

Fig. 1. GridBased FastSLAM

2.2 GraphSLAM

Graph SLAM is a graph based SLAM algorithm. The reason why call GraphSLAM because the algorithm can be very easy to understand with graph.

GraphSLAM has several advantages against to FastSLAM. GraphSLAM performs better accuracy than the FastSLAM. What is more, GraphSLAM uses a sparse matrix for calculating, so it has a better on board processing efficiency. GraphSLAM can divided into two parts. One it front end, another one is back end. The front-end of GraphSLAM looks at how to construct the graph using the odometry and sensory measurements collected by the robot. This includes interpreting sensory data, creating graph and continuing to add nodes and edges to it as the robot traverses the environment. So if a node should be added to graph is a pretty important problem. This is important. Imagine a

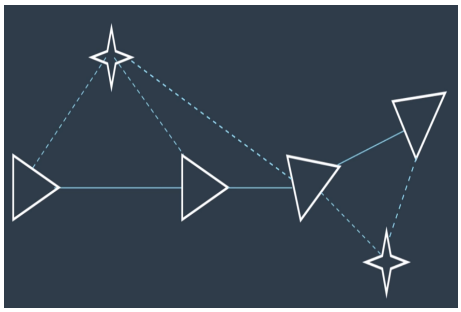

Fig. 2. why called GraphSLAM

robot travels a loop , however if robot don't know which scene is same with previous situation, then it will add a new node to sparse matrix. If robot do this, the accuracy of the map becomes worse since robot add a constraint which is not exist. So the scene understanding is also a important work. With LIDAR sensor or camera sensor, robot should understand where to add a node and where not to add. The core of GraphSLAM is construct sparse matrix and find the optimal constraints about each link. Sparse matrix also called information matrix.

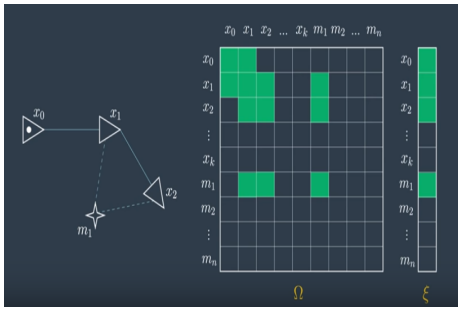

Fig. 3. Sparse Matrix construct

When robot moves to a environment, if the environment is new to robot, then it starts to recognize the environment's feature. For example, the feature extraction algorithm in computer vision called 'SURF' is used. Then each SURF features is m1,m2,m3.....mn in above figure. The robot poses is represented in above figure is x0,x1,x2....xn. When robot moving, a pose difference between k and k-1 time step should becoming a constraint. So the constraints between each m1,m2.. and x1,x2.. can be generated. If there is no direct constraint, then the position's constraint value should be zero. As a result, the sparse matrix can be very large because of map size, but the most of matrix value is zero. This is why GraphSLAM performs better on-board computing characteristic. Since robot moves, it can't be move only in linear way. Most of robot moves in a nonliear way. So GraphSLAM uses taylor series to linearize the non linear part. GraphSLAM depends on the topology of the system. If the robot moves through without turns, then the topology of the system is the linear topology. It is a simple case. But when robot moves with cyclical graph, then one pose can be connect to many features and features will be connect to many other poses. This increases robot pose and map's inaccuracy. As a solution, variable elimination process is given. Variable elimination can be applied iteratively to remove all cyclical constraints. Just like it sounds, variable elimination entails removing a variable (ex. feature) entirely from the graph and matrix. This can be done by adjusting existing links or adding new links to accommodate for those links that will be removed. [2]

3 GRAPHSLAM IMPLEMENTATION IN ROS

GraphSLAM is installed using the native ROS system. In order to use GraphSLAM packge, some of the code should be modified. [3]

1) Install RTAB-MAP ros [4]
2) Define new URDF model. The robot rgbd frame should be added.
3) Make a launch file for mapping
4) Add rviz for mapping.launch
5) Make a new gazebo world for test
6) Modify the gazebo plugin
7) Configure appropriate interface about gazebo plugin and topics

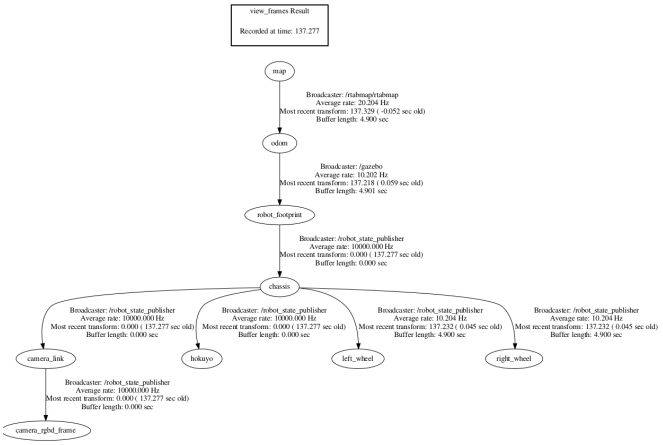

Fig. 4. TF tree for robot

4 SIMULATION

The simulation is implemented based on GAZEBO and RVIZ. The sensor is depth camera and hokuyo laser scanner. Depth camera provided raw image and depth image, and the range is provided from hokuyo laser scanner. The simulation is implemented with two part. One is udacity provided dining kitchen world and the other world is defined by myself.

4.1 Dining kitchen world

This world simulation is divided in to two part. One part is robot only move one cycle, the other one part is robot move 3 cycles. The initialize view in RVIZ is shown as below.

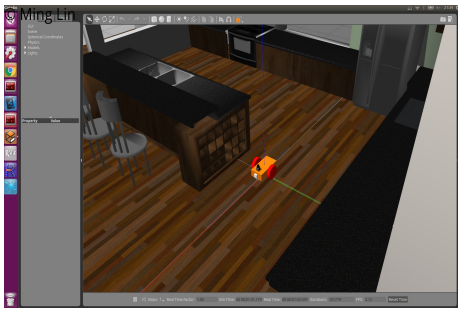

Fig. 5. Initialization in GAZEBO

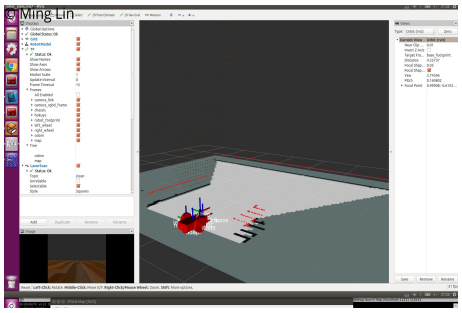

Fig. 6. Initialization in RVIZ

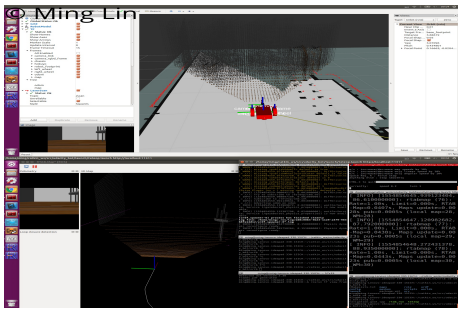

Fig. 7. One loop working

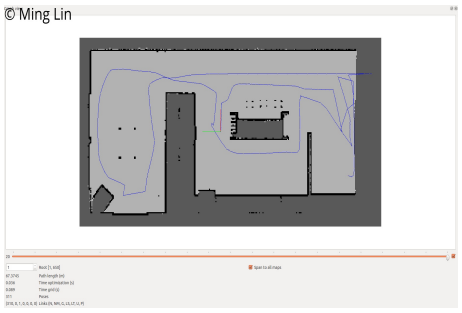

Fig. 8. One loop map

4.1.1 *One loop*

The mapping results are shown as below. The working figure is shown as below. RTAB-MAP viewer is opened and it was logging robot's pose.

Finnaly, generated map looks like tough but it gives relative accurate result.

However the cloud points of environment results is not so good. It just gives a tough outline about the item. Furthermore, some of the environment robot even never sensed it.

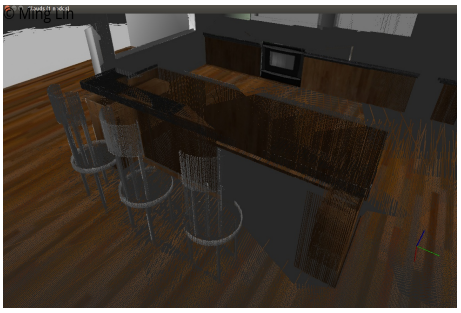

Fig. 9. One loop mapping problem

4.1.2 *Three loops*

The working figure is shown as below. RTAB-MAP viewer is opened and it was logging robot's pose. The map generated is shown as below. After three loops traveled, robot generated a new map which with a little change. It seems like change the map's coordinate and make a more accurate map about environment. The right angle was mapped as world and the horizon lines were also mapped correctly.

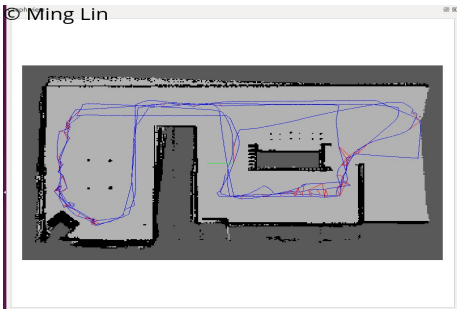

Fig. 10. Map generated with 3 loops

Since robot moved three loops, the point cloud of items are increased. However since robot moved three loops, each items position occured a little displacement. It makes map become confusing.

When robot moved to sofa, the map started occur displacement. I think the problem is at sofa environment, robot detected features about scene, however since the environment becomes complex, the features detected were always changed. So the relationship between poses and features become complex and confusing. Although it is optimized with RTAB-MAP viewer, however the results still not good enough.

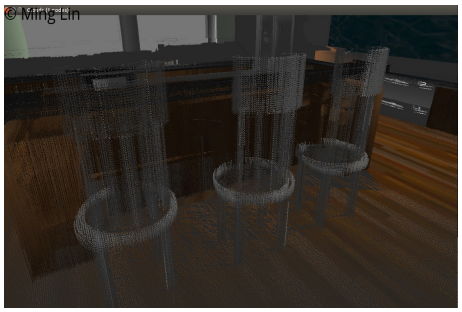

Fig. 11. Three loops problem

4.2 Ming world

This part is no differences between dining kitchen simulation. The only changes are modified world. In this world, open environment is generated. Not like the dining kitchen, it is opened world. Each items are different. This setting is for eliminate the consistancy about items and features. So the robot can match every features without repetition.

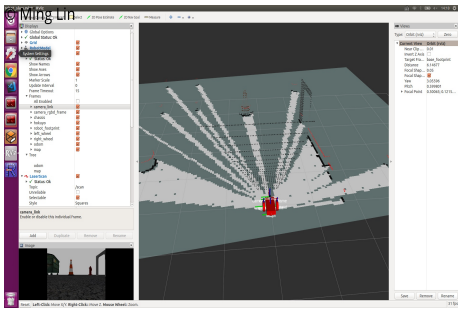

Fig. 12. Ming world initialization

From the RTAB-MAP viewer, it can be found that SURF feature can be extracted correctly since the item is unique. There are no similar features. Every feature is unique and special. So the constraints are all unique.

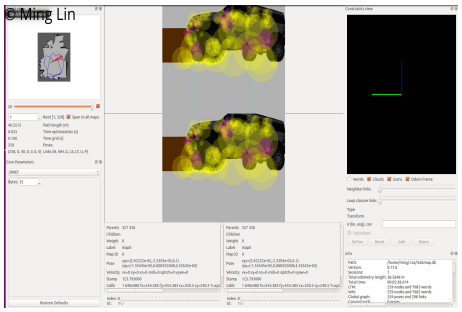

Fig. 13. RTAB MAP Viewer

The point cloud of result(fig.14) shows there are no obvious displacement happened.

The final generated map(fig. 15) extract exact occupancy grid which are obstacles and unknown area.

5 Summary

In this research, RTAB-map ros package is introduced. From the simulation results, it can be confirmed that when robot runs several times in the environment, algorithm can generates more accurate and feature rich map. Although one loop can generates map about environment, the accuracy is not good enough. After online processing, offline processing in RTAB-MAP viewer is also very important. It can optimizes the constraints about every pose and every feature based on the robot trajectory and map information.

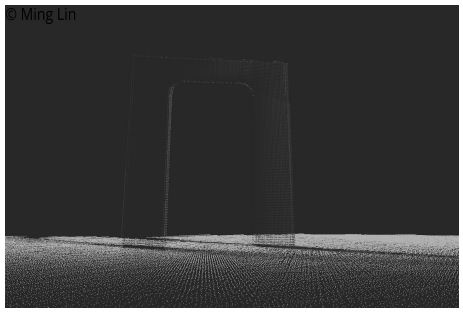

Fig. 14. Ming world performance

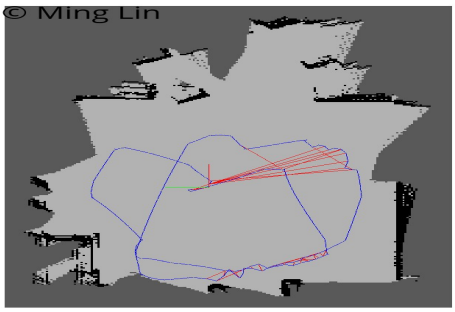

Fig. 15. Map generation result

References

[1] “https://www.udacity.com,”
[2] S. Thrun., *Probabilistic robotics*.
[3] “http://gazebosim.org/tutorials?tut=ros-overview,”
[4] “http://wiki.ros.org/rtabmap-ros,”