Exploring SLAM

Y. Khaky

Abstract: The purpose of this paper is to introduce and explain a specific kind of a SLAM algorithm called RTAB-Map. Simultaneous Localization and Mapping or SLAM for short is the problem at which the robot attempts to construct a map of the environment while simultaneously localizing itself within that map. Neither the map nor the location is known, however, each one of them is needed to define the other. SLAM is an important problem to solve as it is a problem that shows up very frequently in many applications, range from autonomous vacuum cleaning to surveilling construction sites.

***Index Terms*— SLAM, Graph SLAM, RTAB-Map, mapping**

I. Introduction

There are several algorithms to implement SLAM, namely: Extended Kalman filter, Sparse Extended Information Filter, Extended information form, FAST Slam and Graph Slam. The one implemented in this lab is a version of the Graph slam referred to as RTAB-Map.

II. Background

A. The Theory

Real-Time appearance Based Mapping (RTAB-Map) implies that the data is collected in the form of images, these steam of images from the camera on the robot will be used to perform graph SLAM, localizing the robot and mapping the environment at the same time.

There are two sections to graph mapping, the front and the back end. The front end is concerned about sensor data. The image and odometry data are collected. They are used in calculating loop closure. Loop closure is the process of classifying a the current location as – previously visited. This is done by comparing the current image with the previously captured images. The back end is concerned about graph optimization and outputting the map. In a perfect world, if the robot visits the same position again, the images from that place would perfectly overlap with the previously captured images, however due to the error accumulated from the noisy robot motion and the noisy sensors, this would never happen, thus the usage of the Bag of Words algorithm. When an image looks similar enough (a certain threshold is reached) to a previously detected image, the robot concludes it is at the same position as the earlier image. When a positive match happens, and two images are linked, a loop closure takes place. When a loop closure occurs, the back end is given sufficient input to reduce the error in the graph and the output map. This results in a non-choppy image as the image isn't recorded as a new image in an unvisited new location. The figure below shows the effect of incorporation loop closure detection algorithm to the output map produced.

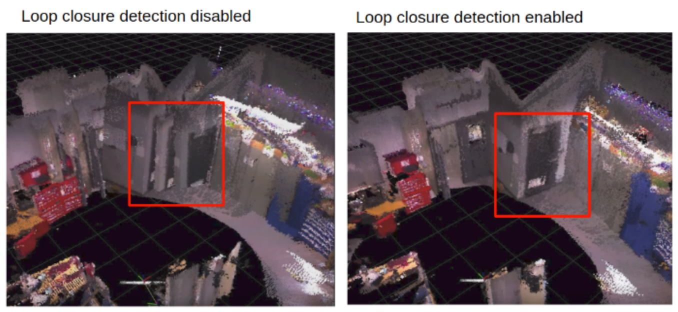

Loop closure detection disabled
Loop closure detection enabled

Fig. 1 Showing the difference between two maps one done with loop closure and one without

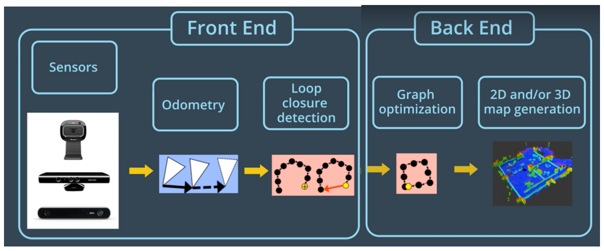


Fig. 2 Showing the two components of Graph Mapping

III. Robot Configuration

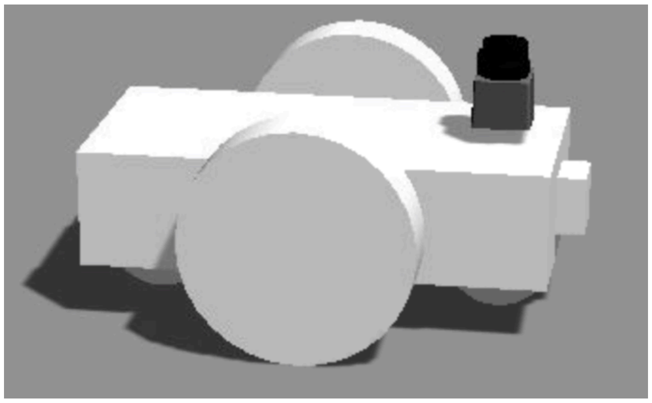


Fig. 3 Showing a picture of the robot

The robot has two motorized wheels and two caster wheels. The robot has two sensors, a LIDAR and Kinect sensor. The Kinect captures images that are going to be used in the Bag of Words algorithm. The resolution of the camera is set to 640X480. The wheels of the robot have encoders that are used for odometry. The reason for choosing this robot model was due to simplicity and ease of creation.

IV. World Creation

There are 3 maps used in this project, 2 of them produced a successful mapping, but the last one didn't result in any loop closure. That observation will be discussed under the "Failed Experiments" section.

The first world used is the Kitchen world which is available as an item to be inserted from Gazebo. The world has two different rooms with several features that would help in loop closure. The figure below shows the kitchen map.

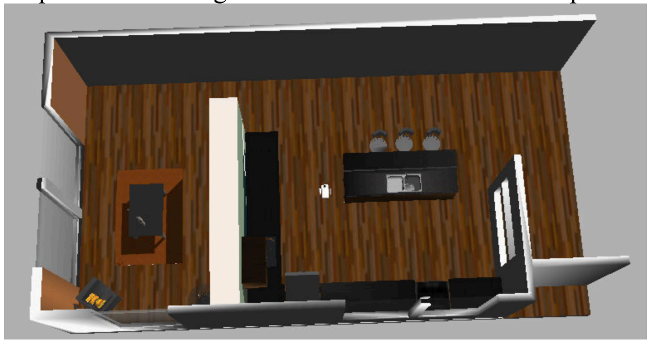


Fig. 4 Showing the kitchen map

The second world was created using the building editor in Gazebo. A square is divided into 4 triangular compartments by constructing two walls across the two diagonals. Random small items were placed in the environment to increase the number of features thus facilitating loop closure. The figure below shows the "4 compartments" map.

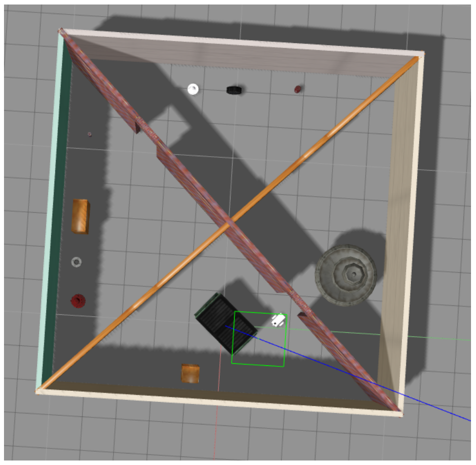


Fig. 5 Showing the "4 compartments" map

Further discussion on the 3rd (failed map) will be done in one of the following sections.

V. Procedure Summary

Gazebo was launched simulating both the environment and the robot using "roslaunch mapping udacity_world.launch". Then the mapping node was run using "roslaunch mapping mapping.launch". Then the robot was navigated around the environment using a teleop node which was launched using "roslaunch mapping teleop.launch". The robot was circulated around the environment more than once to enable loop closure detection. For the kitchen map, robot was circulated 3 times and for the 4-compartments some rooms were visited up to 5 times.

VI. Results

The two mapping jobs had good results. Unfortunately, a method to evaluate the accuracy of a map after a mapping procedure was not used (more about that in the future works section) so there is no way to know the accuracy of a map and compare it to another map. A metric that we can use is the number of loop closures, this doesn't tell us directly if the map result is correct, however the more loop closures there are the more likely a map is accurate because with every loop closure we get an added constraint to the graph that needs to be optimized for, so in theory, the more loop closures/ graph constraints, the more the chances a robot got to correct for errors in odometry.

The kitchen map resulted in 11 loop closures. The figure below shows the map obtained from mapping.

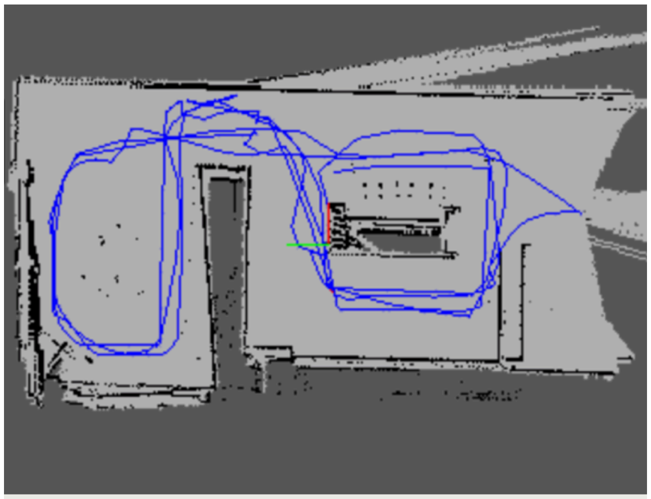

(475, 0, 11, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Fig. 6 Kitchen map

The 4-compartments map resulted in 17 loop closures and the map as viewed from the RTAB-Map database viewer is presented below.

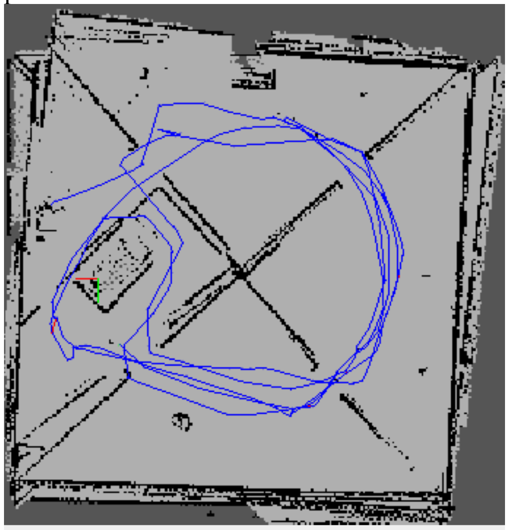

(247, 0, 17, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

Fig. 7 The 4-compartments map

VII. Discussion

It can be seen from visually inspecting the maps generated they are not very accurate. The first environment shows white space beyond the north wall which could have not been mapped because the wall is opaque. The map of the second environment looks like two maps on top of each other with some rotational offset between them.

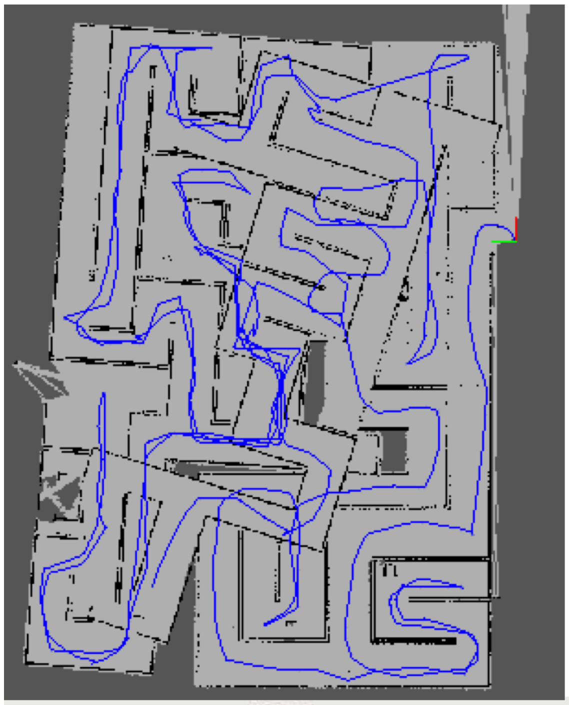
The most likely reason for second problem mentioned may be due to the robot colliding with the wall, causing the wheels to rotate without the robot actually moving. This would cause the wheel encoders to count wheel rotations without the robot actually moving. These wheel rotations, that are not paired with a robot movement, would cause an error to be introduced in the odometry calculations.

Another problem that is visible by inspection is the high amount of noise. The walls of the first map should be presented as a solid line all the way around the environment, however some walls are not a solid line. The top left corner of the map is a dotted line, one might assume that there is an actual gap in the wall that a smaller robot can go through, however that is not the case.

The possible reason for this problem may be due to a low resolution of a LIDAR sensor. As seen from the trajectory of the robot, the robot never approached the top left corner of the map. The further the robot is from a wall the more the spaces between the LIDAR ways when they hit a surface as the laser rays are diverging from the source.

VIII. Failed Experiments

It is also worth noting that several other maps were tried, however mapping them was not successful. The main issue found was that the capture rate of the camera was not frequent enough. Therefore, the chances of two pictures having similar features is significantly reduced especially if the map is big. With this slow capture rate, the mapping had to be kept to small maps to increase the chances of loop closure. Since no lop closures were detected, errors in mapping were never corrected. The probable cause of the bad mapping is due to hitting the walls which may lead the wheels to slip and results in the wheel encoder counting wheel rotation while in fact is has not moves from its place.


(994, 0, 0, 0, 0, 0, 0) Links (N, NM, G, LS, LT, U, P)

IX. Future Work

There are a couple of problems discussed in the previous sections. Here is a list:

1. Walls are not solid black color in some places, they have some white in them.
2. Two overlapping maps onto of each other with a rotational offset.
3. White area beyond walls that could have not been seen by the robot.
4. No loop closure when map is huge.
5. Noisy/ grainy map. (some black dots appear randomly)

The first problem can be solved in multiple of ways, the resolution of the LIDAR can be increased by spinning it faster or using a higher resolution LIDAR. Another way to solve this is to make sure that the robot moves close to a wall so the divergence between two LIDAR laser lays is minimal. A third way to solve this is to process the map and filter out those apparent gaps in the wall and replace them by a black dot. The filter can be made such that it fills in the gaps in straight lines only and not curving surfaces.

The second issue seems to be the most significant especially for the second map. Some of that effect can also be seen in the left room on the kitchen map. It seems that reason for this kind of behavior is due to a problem in the odometry. We see can distinctly see two maps on top of each other as opposed to many maps. This suggests that an error in the odometry was introduced abruptly and suddenly. This implies that a wheel is slipping (wheel encoder count rotations without the robot actually moving). To minimize wheel slippage, when the robot is operated, it's motion should always be smooth and continuous. The operator should try not hit walls or decelerate or accelerate fast as all these will cause wheel to slip. The top speed at the teleop node can be limited so when the robot breaks the deceleration doesn't cause too much slippage. Another thing that can be done is to imbed an accelerometer on the robot that stops updating wheel encoders when the acceleration gets too high. Another possible solution to reduce slippage is increasing the friction of wheels.

The third problem, seeing map indicating available space beyond a wall, is most visible with the first map. We can see the upper area of first map showing light colored pixels beyond a wall, which would have been impossible to the robot to see given it's outlines trajectory. The reason this occurred was because the robot rolled up the wall and looked upwards for a small amount of time making the robot registering the space as available. Similar to the solution to the previous problem, a gyroscope can be attached to the robot, whenever the yaw is more than a certain threshold (whenever the robot is not horizontal to the floor) the mapping node would reject the sensor input from the LIDAR and treat them as invalid.

The fourth problem is very problematic if we use this algorithm for big spaces. As discussed in the "FAILED EXPERIMENTS" section, the reason for this problem is the low capturing frequency of the camera. The "DetectionRate" is an argument used in the mapping node to set the capturing frequency rate. For these experiments that variable was set to 1Hz. The higher this value is better because the chances to skip an important feature is reduced, however there is a tradeoff. The time between the two captures is the maximum time allowed for comparing bags of images with each other inside the Working Memory. If we increase the capturing frequency we reduce the time period thus reducing the size of our working memory. The reduction in working memory would result in reduction in loop closures because less features are being compared with one another. There is an optimal amount of image capturing and that was not looked into in the experiments explained in this paper. In the future, a study can be made to formulate the optimal frequency to result in most accurate loop closures.

The fifth problem ties in to the first problem discussed. The solution for that would be the same solution implemented for the first problem.

In conclusion of the Future Work section, the parameters need to be explored more. The mapping node has many parameters and exploring them in one single paper is not feasible. From different algorithms to parameter tuning, the possibilities of the node parameters are very huge.

X. Appendix

A. *Memory Allocation*

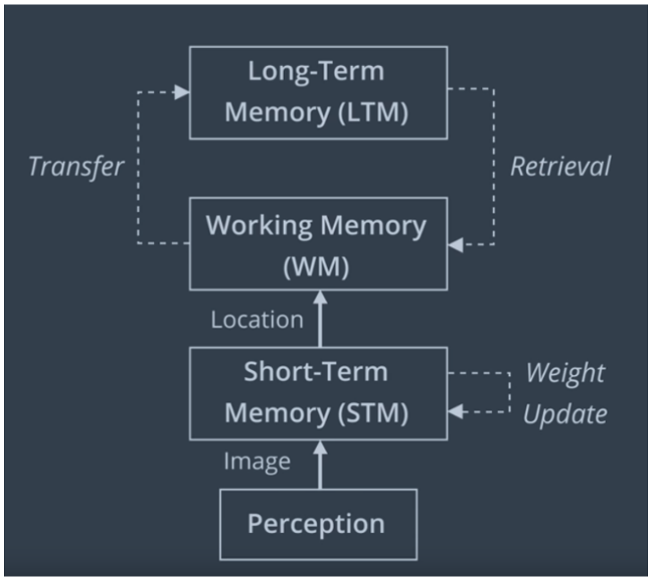


Fig. 8 Showing memory allocation flowchart

There are three stages/ locations an image can go to. The Short-Term memory, Working Memory and Long-Term Memory. Working memory is where all the images that will be compared to the incoming image will be stored. As mentioned earlier, we need to keep the computation period be lower than the capturing period. To achieve a constant computation time, we limit the number of images in the working memory. Only the images in the working memory are considered for loop closure algorithm. Images have weights in the WM, these weights depend on the amount of time the robot spends around that location of the image. An image would be weighted highly if the robot spends more time around that location. Highly weighted images are more likely to remain inside the working memory because the ones with low weight are transferred out to the Long-Term Memory.

The Short-Term memory is where incoming images get stored. The STM collects new images, the weight of these images is what decides if this image should get transferred to the Working Memory. After every new image capture, the weights are updated, the images with the highest weights are moved to the WM.

In the event of a loop closure, all the images in the LTM around the location where the loop closure happened will be called into the WM from the LTM thus increasing the chances of a additional loop closures in the near future because the robot is currently at a location where it has visited before, therefore previous images from around this location have high chance of loop closure.