Map My World

Balaveera Thatikonda

Abstract—Generated a ROS package with 2D occupancy grid and 3D octomap for a robot to map environments using the RTAB-Map SLAM.

Index Terms—Robotics, Map My World, SLAM, Occupancy Grid, Udacity.

1 Introduction

TRYING to develop a deployable package in a know environment, so that robot can maneuver in the environment by localizing and mapping the environment. Started taking the robot developed in the Localization project and added a sensor RGB-D camera to get the depth of the image. Based on the classroom exercises to do a SLAM, applied the RTAB-Map as it has advantage of speed and memory management when compared to other SLAM methods such as FastSLAM, Grid-based FastSLAM

2 Background / Formulation

By comparing these FastSLAM, Grid-Based FastSLAM, Graph SLAM methods, the Graph SLAM solves the FastSLAM problem of covering the entire path and map instead of most recent pose and map. RTAB (RealTime Appearance Based Mapping) is an algorithm which is based on the GraphSLAM method. This RTAB localizes the robot and maps the environment by using the data collected through the visual sensors and this is efficient in speed and memory utilization. The reason to choose the RTAB is that the FastSLAM has particle filter inefficiency and uses the low dimensional EKF to estimate the posterior over the map as the EKF takes more time in processing the non-linear movements.

2.1 FastSLAM

FastSLAM (1.0 2.0) solves the full SLAM problems with known correspondences by estimating the trajectory and estimating the map. FastSLAM 2.0 overcomes the inefficiency of the FastSLAM 1.0 by imposiing a different distribution, which results in a lower number of particles. FastSLAM (1.0 2.0) estimates a posterior over the trajectory using a particle filter and uses a low dimensional EKF to solve independent features of the map which are modeled with local Gaussian. But one disadvantage with this algorithm is that it is always assumes there are known landmark positions, and so with this we are not able to model an arbitrary environment.

2.2 Grid-Based FastSLAM

Grid-based FastSLAM, is an extension to FastSLAM, models the environment using grid maps without predefining any landmark position by extending the FastSLAM algorithm to occupancy grid maps. In this each particle holds a guess of the robot trajectory and each particle maintains its own map. This algorithm solves the mapping with known poses using the occupancy grid mapping by each particle. Inorder to adapt the FastSLAM to grid mapping, it needs three techniques:

1) Sampling Motion - Estimates the current pose given the k-th particle previous pose and current controls u.
2) Map Estimation - Estimates the current map given current measurements, the current k-th particle pose, and the previous k-th particle map
3) Importance weight - Estimates the current likelihood of the measurement given the current k-th particle pose and the current k-th particle map.

2.3 Graph SLAM

Graph SLAM solves the full SLAM problems, covers the entire path and map instead of just most recent pose and map allows to cover the dependencies in current and previous poses. This solves the problem by creating all the poses and features encountered in the environment and find the most likely path and map of the environment. That is achieved by dividing the tasks into Front-end and Back-end. In the Front-end, algorithm constructs the graph using the odometery and sensory measurements collected by the robot. In the Back-end task, it takes the input from Front-end task and optimizes the graph and generates a system configuration that produces the smallest error. The Graph optimization is done by applying a principle called MLE-Maximum Likelihood Estimation. This MLE is solved numerically by applying an optimization algorithm such as gradient descent.

2.4 Graph Based SLAM(RTAB)

RTAB -(Real Time Appearance Based)- Mapping. In this the algorithm uses the data collected from the vision sensors to localize the robot and map the environment. It uses a process called loop closure detection to determine whether the robot has seen a location before. As the robot moves in the environments, the map is expanded and the number of images that each new image must be compared increases linearly. This makes the loop closure to take longer with the complexity increasing linearly. In the front-end process, the RTAB collects data from the sensors(imu, decoders,camera),

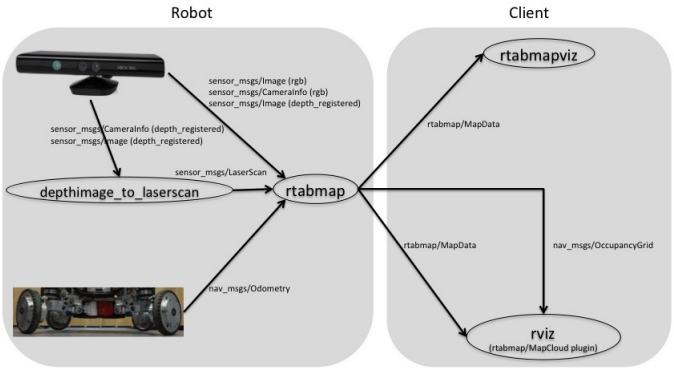


Fig. 1. Sensor Mappings

odometery and constructs the metric graph with loop closure detection with the images from the RGB-D camera. Uses the global closure approach as it is more accurate in an unknown environment. Loop closure uses the visual bag-of-words approach to match the new images with already seen images. In back-end, the graph optimizer minimizes the errors in the graph using graph optimization techniques like Tree-based network optimizer, General Graph Optimization and GTSAM(smoothin and mapping).

2.5 Comparison/Contrast

In a real world, robot needs to estimate its position and has to map the dynamically changing environment. When the particles approach(FastSLAM) is uses they are prone to errors in estimating the position as it is uses the landmark approach, assumes there are certain fixed position, but in reality the items in the environment will change like in the mining. The (RTAB-Graph SLAM) visual comparison approach seems to be promising where it uses the loop closure detection process and clearly constructs the map and its position.

3 Simulations

Following are the images of the robots in two different environments(refer to Table 1). Initially started with the provided environment - Kitchen and Dining. Used the robot developed in the previous project and extended it to have the RGBD camera to capture the visual odometery. Here we are simulating the Kinect camera and for that replaced the existing camera and its shared object file to that off the Kinect shared object file and updated some additional parameters as required and sensor mappings will look as in the Fig.1 Refer to the Table 2 for the environments mapped in Rviz and to the Table 3 for the RTAB images of the both the environments.

4 Set up

4.0.1 Robot

Used the robot designed in the previous project which has a base, two actuators, a camera , two wheels, two coasters. And extended it to have a RGBD camera to get the visual odometery.

TABLE 1
Robots in Environment Images

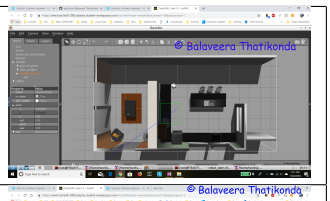
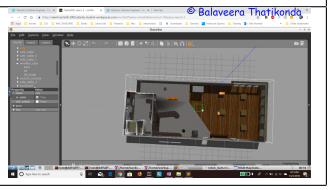
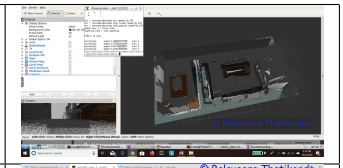
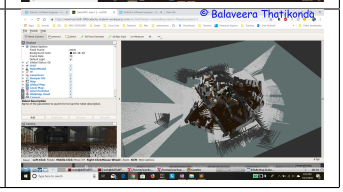
Kitchen and Dining	
Custom Environment	

TABLE 2
Environment Mapped Images

Kitchen and Dining	
Custom Environment	

4.0.2 Packages used

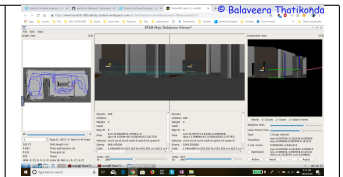
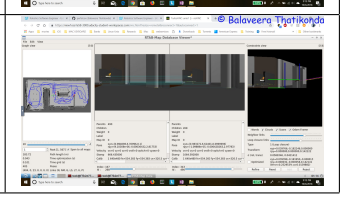
Packages - move_base, teleop, RTAB,Rviz, Gazebo.
Topics received - camera, depthimage sensor, map, odom
Topic published - mapdata, Occupancy Grid, Map cloud.
Environments - Kitchen and Dining, Custom Environment.

Updated the robot's URDF to properly adjust the links for replacing the camera with kinect camera and we can see the links visually in this image (refer to Fig 2).

4.0.3 Execution

Launched the SLAM Project with the following in separate windows

TABLE 3
RTAB Images

Kitchen and Dining	
Custom Environment	

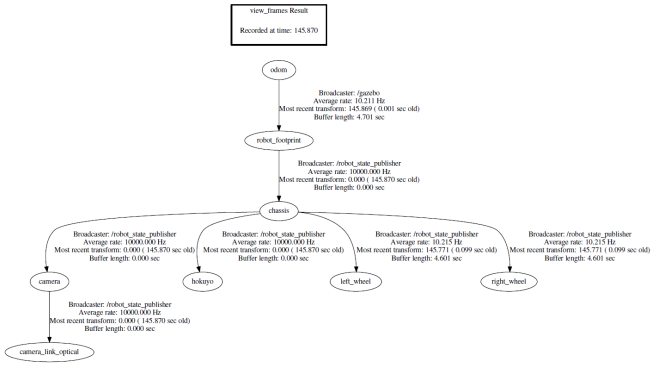


Fig. 2. Robot Links

1) Robot in the Kitchen and Dining in Gazebo in the first run and in the Custom Environment in the second run.
2) Teleop - keyboard for navigating the robot in the environment.
3) mapping - which loads the RTAB package.
4) rviz - where we can see how the robot is mapping in environment.

Once all the packages are loaded, the images generated from the Robot's camera are projecting to top, this is corrected by adjusting the parameters in the robot's link in the urdf to display in front.

5 Data Acquisition

In each execution, the Robot is navigated 3 times whole the environment to map the environment to get atleast 3 loop closure detections which provides a good mapping of the environment, refer to the images in the Table 2

After each exection, the RTAB generates rtabmap.db which contains information about the map and can be used for further analysis.

6 Results

Once the RTAB generates the database, it is analysed using the rtabmap-databaseViewer. It is evident from the images (refer to Table 3), that the robot has detected and mapped the environment properly which has got 23 global loop closures for Kitchen-Dining and 35 global loop closures for custom environment and by this it is clear that robot has mapped environment properly and the package can be deployed in the real time.

7 Discussion

As it is clearly evident from the number of loop closures obtained and the mapping of the environment, that the package is deployable in the real time environment.

8 Conclusion / Future work

Achieved the required goal. In future, would like to deploy on a real device and improve iteratively.

9 References

1) https://github.com/lucasw/gazebo_ros_demos/commit/9c52
2) Common Questions in Project