RTAB SLAM Project

Muthanna A. Attyah

Abstract—RTAB-Map is considered to be the best solution for SLAM to develop robots that can map environments in 3D. These considerations come from RTAB-Map's speed, memory management, and its software tools that are available for wide range of environments. In this project **rtabmap_ros** package, which is a ROS wrapper (API) for interacting with RTAB-Map will be configured in ROS on a simulated robot with RGBD camera and laser scanner. The algorithm will be used to SLAM in two different simulated worlds to generate a 2-D and 3-D occupancy maps for both worlds.

Index Terms—Robot, IEEEtran, Udacity, Localization, SLAM, RTAB Map, Octomap, ROS

1 Introduction

RTAB-MAP (Real-Time Appearance-Based Mapping) is a RGB-D Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the maps graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected. RTAB-Map can be used alone with a hand-held Kinect or stereo camera for 6DoF RGB-D mapping, or on a robot equipped with a laser range-finder for 3DoF mapping. RTAB-Map tools are available for multiple environments including ROS, Ubuntu, MacOS, Windows, Android, Raspberry Pi, and Docker. [1]

2 Background

Mapping is one of the most important capabilities for a mobile robot to enable it to achieve its tasks. It consists of creating a representation of obstacles in its environment based on data obtained from robot sensors. Mapping task is harder when considering the inherent noise in the data obtained from most of the sensors. There are many mapping algorithms that are designed to handle the inherent noise and operate either in small indoor structures or complex outdoor environments.

Algorithms that are designed for mapping indoor spaces usually generates two dimensional floor plan-like maps, which can fairly represent walls and other features. This type of representation becomes very poor when we try to model more complex environments, which usually have much richer features to be represented (e.g. buildings, trees, and cars on outdoors). A natural extension for 2D maps is to use three-dimensional representations. [2]

When a robot is constructing or updating a map of its unknown environment while simultaneously keeping track of its location within it, the robot is doing a Simultaneous localization and mapping algorithm or **SLAM**. SLAM process has multiple challenges including unknown map, big size of environment, huge hypothesis space, noise from sensors, perceptual ambiguity, and number of cycles taken by robot. Following are some of the popular approximate solution methods that will do SLAM and handle some of the challenges.

- **EKF SLAM** algorithms are feature based, and use the maximum likelihood algorithm for data association. For the past decade, the EKF SLAM has been the de facto method for SLAM, until the introduction of FastSLAM. Associated with the EKF is the Gaussian noise assumption, which significantly impairs EKF SLAM's ability to deal with uncertainty. With greater amount of uncertainty in the posterior, the linearizion in the EKF fails. [3]

- **GraphSLAM** algorithm uses sparse information matrices produced by generating a factor graph of observation interdependencies (two observations are related if they contain data about the same landmark) [3]

- **Occupancy Grid** to represent a map of the environment as an evenly spaced field of binary random variables each representing the presence of an obstacle at that location in the environment. Occupancy grid algorithms compute approximate posterior estimates for these random variables. The most common type of occupancy grid maps are 2d maps that describe a slice of the 3d world.

- **FastSLAM 1 and 2** estimates a posterior over the trajectory using a particle filter approach. This will give an advantage to SLAM to solve the problem of mapping with known poses. It uses a low dimensional Extended Kalman Filter to solve independent features of the map which are modeled with local Gaussian.

- **ORB-SLAM 1 and 2** a real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and re-localize the camera in real time. [4]

- **CoSLAM** a visual SLAM software that aims to use multiple freely moving cameras to simultaneously compute their egomotion and the 3D map of the surrounding scenes in a highly dynamic environment. [5]

In this project RTAB-MAP which is a Graph-Based SLAM approach based on an incremental appearance-based loop closure detector will be used to test SLAM algorithm in simulated indoor small size environment.

3 Robot Model Configuration

Simulated robot model was created in Gazebo and equipped with sensors that are required to solve SLAM problem. Microsoft XBOX 360 Kinect sensor was added as the RGBD camera that will provide both RGB image and Depth image. Hokuyo lidar was added as a laser scanner that will provide 2D laser scan. Robot is having two wheels with differential driving mechanism. Robot Shape is as illustrated below:

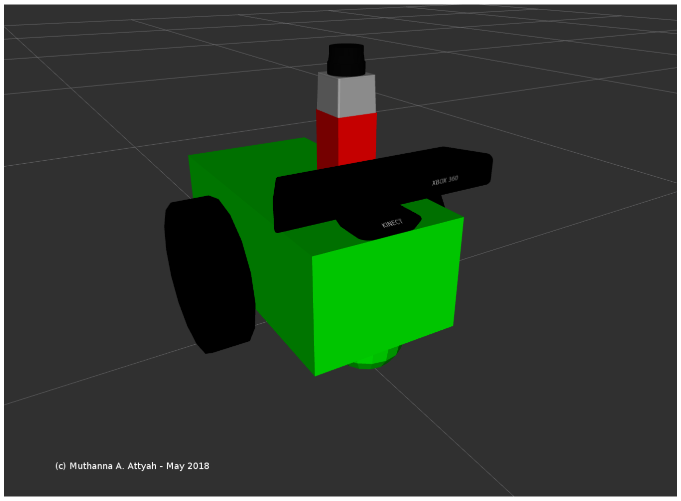

Fig. 1. robot

The tf frames map of the different robot joints are as showing in below tf map. One additional joint was added to the RGBD camera to fix the orientation of the generated depth point cloud which is by default is generated along the Z axis.

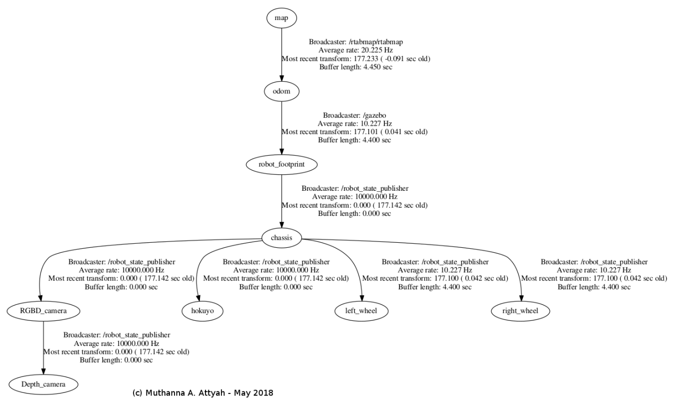
map
Broadcaster: /rtabmap/rtabmap
Average rate: 20.225 Hz
Most recent transform: 177.233 (-0.091 sec old)
Buffer length: 4.450 sec
odom
Broadcaster: /gazebo
Average rate: 10.227 Hz
Most recent transform: 177.101 (0.041 sec old)
Buffer length: 4.400 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (177.142 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (177.142 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (177.142 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.227 Hz
Most recent transform: 177.100 (0.042 sec old)
Buffer length: 4.400 sec
Broadcaster: /robot_state_publisher
Average rate: 10.227 Hz
Most recent transform: 177.100 (0.042 sec old)
Buffer length: 4.400 sec
RGBD_camera
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (177.142 sec old)
Buffer length: 0.000 sec
Depth_camera
(c) Muthanna A. Attyah - May 2018

Fig. 2. TF Frames Map

3.1 Gazebo Plug-ins

The following plug-ins were used in the robot model:

- **libgazebo_ros_diff_drive.so** : to simulated the differential wheel driving mechanisim.
- **libgazebo_ros_openni_kinect.so** : to simulated Microsoft XBOX 360 Kinect RGBD camera.
- **libgazebo_ros_laser.so** : to simulate Hokuyo laser scanner sensor.

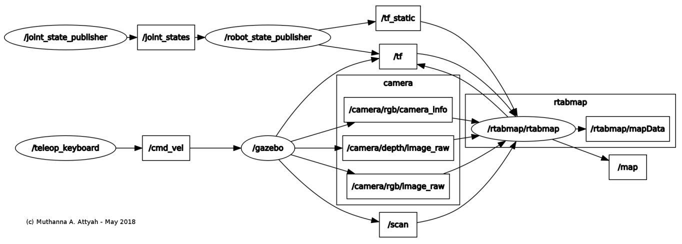

Fig. 3. ROS Graph

Scene and robot configuration - Explain how your personal Gazebo world was created and what is the layout of it. Justify your choice of robot parameters, sensor location, and how you decided to configure your package structure. Scene and robot configuration - Student explains how the gazebo world was created by providing an overview of the layout of items in his/her customized Gazebo world. Student also describes the robot's parameters, sensor features, and reasoning on the package structure

4 World Creation

4.1 Supplied Gazebo World

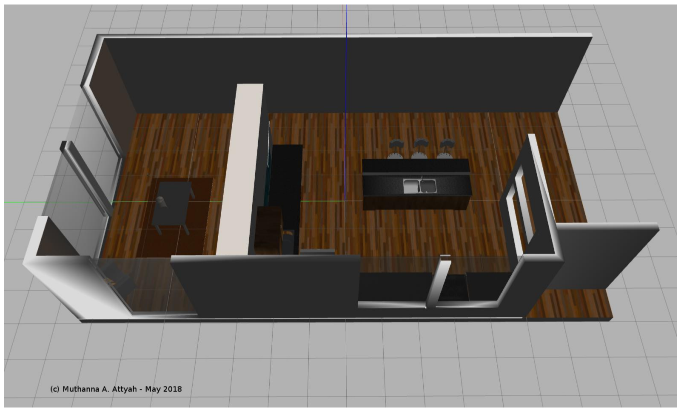

Fig. 4. Kitchen World

4.2 Personal Gazebo World

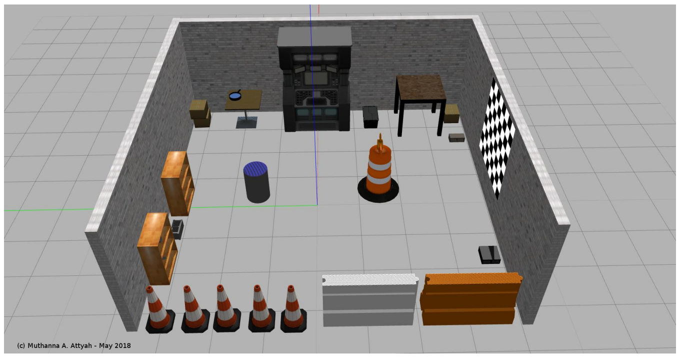

Fig. 5. Personal World

5 Results

Results - Show the results of both occupancy grid and 3D map. The student should include the images for mapping process, final map (2D/3D) for both Gazebo worlds.

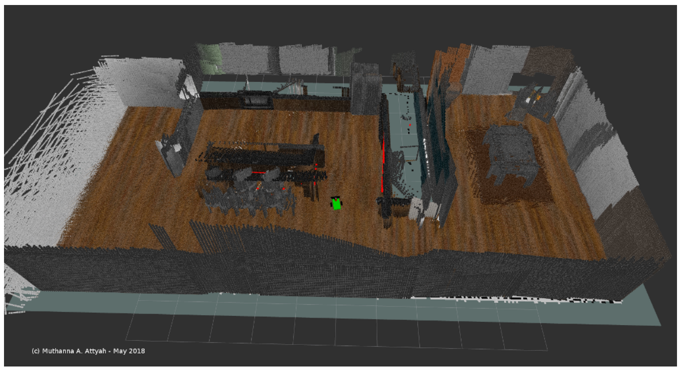

Fig. 6. Kitchen World 3D Map

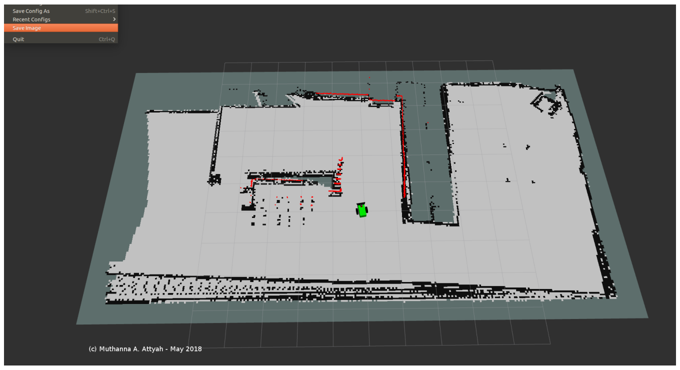

Fig. 7. Kitchen 2D Map

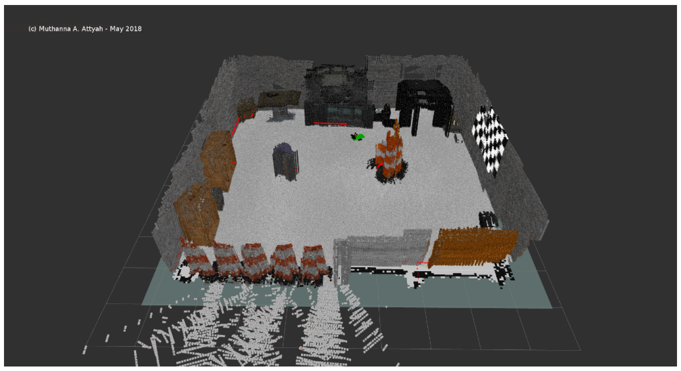

Fig. 8. Personal World 3D Map

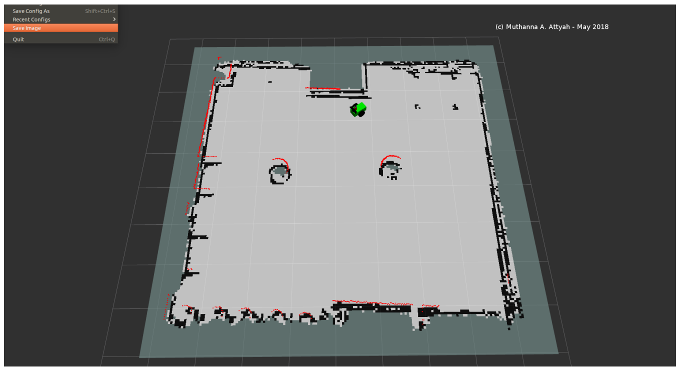

Fig. 9. Personal 2D Map

6 Discussion

What went well, what went wrong. Reflect upon the results of your robot's performance, and the performance of mapping in both worlds. Justify your answers with facts. 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.

7 Future work

Student discusses future desires with RTAB-Map. Talk about any robots and environment they applied this too. ”Future Work - The student can discuss how they would like to leverage this tool in robotics. The student identifies other areas where mapping could be done and for what reason. Such as simulated room or physical place.

References

[1] M. Labb and F. Michaud, "Real-time appearance-based mapping github repo,"
[2] G. S. S. Denis F. Wolf, "Localization and mapping in urban environments using mobile robots,"
[3] t. f. e. Wikipedia, "Simultaneous localization and mapping,"
[4] J. M. M. M. D. G.-L. Raul Mur-Artal, Juan D. Tardos, "Orb-slam2 github repo,"
[5] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2013.