SLAM-Project

Abstract

This project explores the challenges of robot mapping in 2D and 3D spaces. We discuss the pros and cons of different mapping algorithms and how they compare. We also compare different robot configurations and their effectiveness is various environments. The robot is launched in an initially unmapped environment, and RTab mapping is used to map out the new environment while the robot is manually controlled.

Introduction

The goal of this project is to create a robot that can map a given environment. The robot will have onboard a RGB-d camera and a laser scanner to sense the world around it. The robot is launched in a simulated environment and is manually navigated across the map through multiple iterations until it is done mapping.

Background

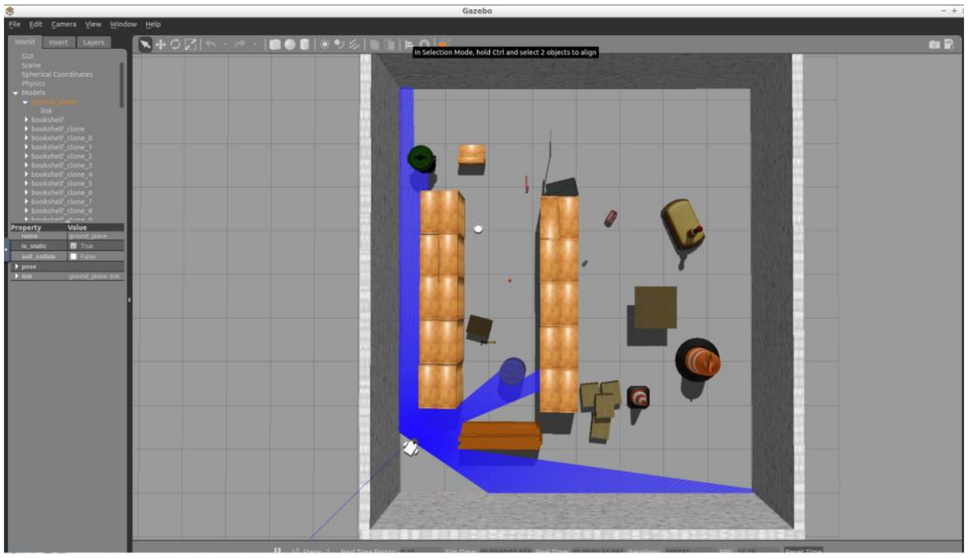
Mapping in two dimensional space is important because it describes the movement space of a large class of robots that are constrained to the ground. It is much quicker and cost efficient to compute making it a good candidate for input into traditional SLAM algorithms. It is also a necessity for robots operating with constrained resources. Mapping in 3D although more costly, provides a lot more data and accuracy than in 2D. This is crucial for flying robots whose movement space is not constrained by the ground and need to navigate and dodge obstacles in 3D.

The main problem is to map an environment with a reasonable degree of accuracy. Mapping is important especially in dynamic environments where obstacles are constantly changing and a static map becomes unreliable. One of the core challenges of mapping is the huge hypothesis space. Mapping by nature is highly dimensional making it computationally challenging especially in mobile robotics. The space of all possible maps is large especially in open environments where there are potentially an infinite number of objects to map.

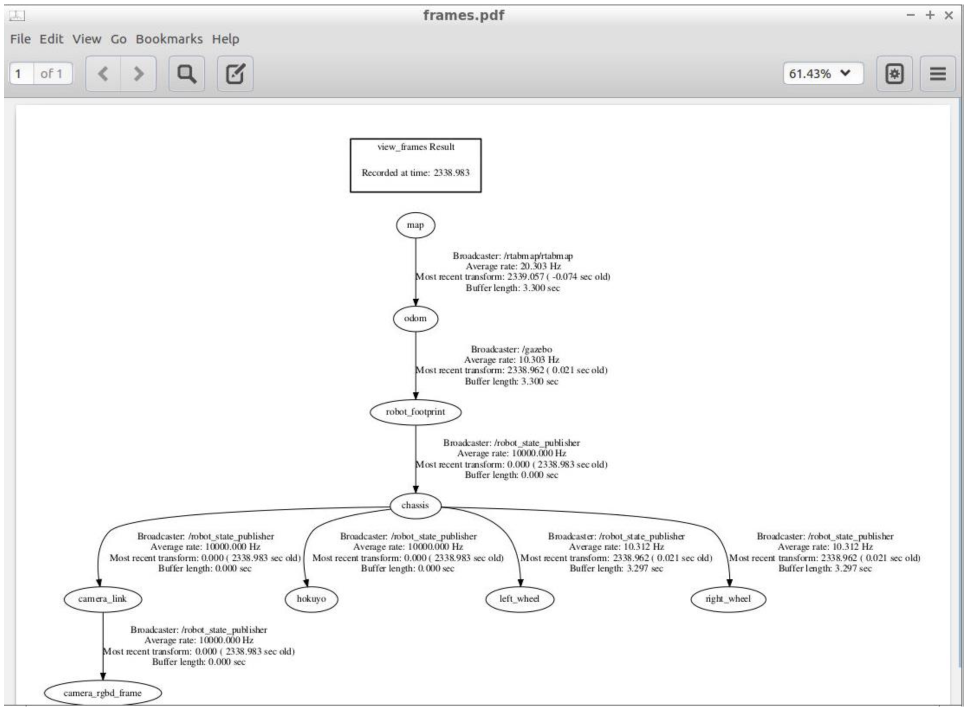
Occupancy grid mapping is a mapping algorithm that uses grids to dictate occupied vs free space. The environment is reduced into squares of a specified size and the value of that square dictates the state of all pixels captures in that cell. The size of the cells can be adjusted for performance and accuracy. The nature of this mapping algorithm makes it an ideal candidate for input to navigational systems.

Grid based FastSLAM applies the concepts of FASTSlam to occupancy grid maps to get a robust SLAM solution to any environment. It addresses a large assumption in FastSLAM which is that there are landmarks available and their position are known. GraphSLAM uses a graph of motion and sensor constraints to define and relate each new landmark. It stores the constraints in information vectors which can then be solved with a linear system of equations. It faces similar issues to the Kalman filter in that systems in the real world are rarely linear. It also uses the Taylor series to approximate nonlinear systems.

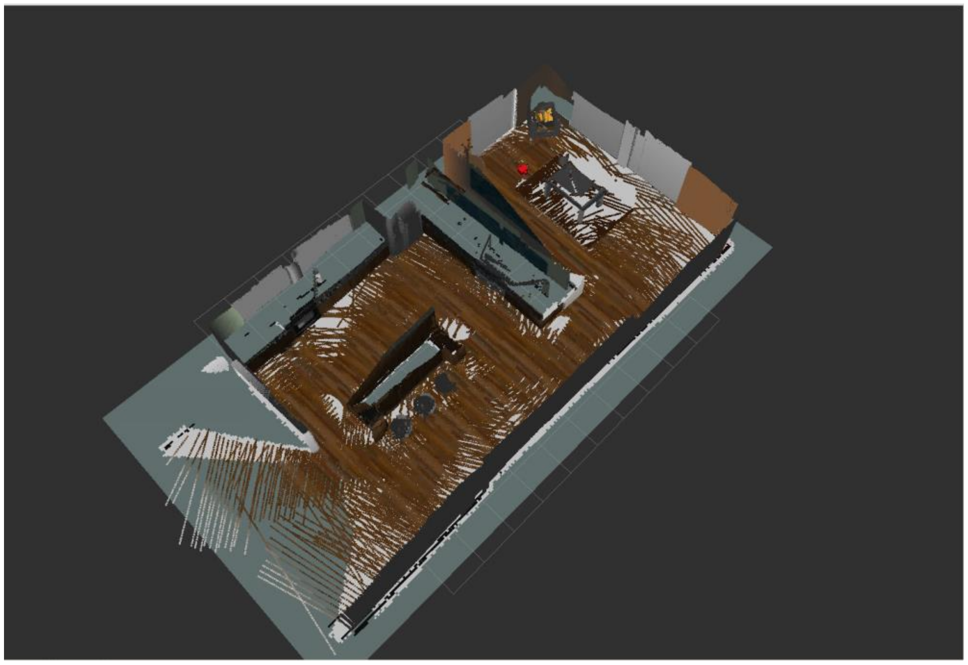
The custom world was modeled after a warehouse with multiple shelves and aisle. Tools and objects are scattered around the aisles to create obstacles for the robot. Boxes and containers are lazily stacked in the corner to further emulate a messy warehouse setting.

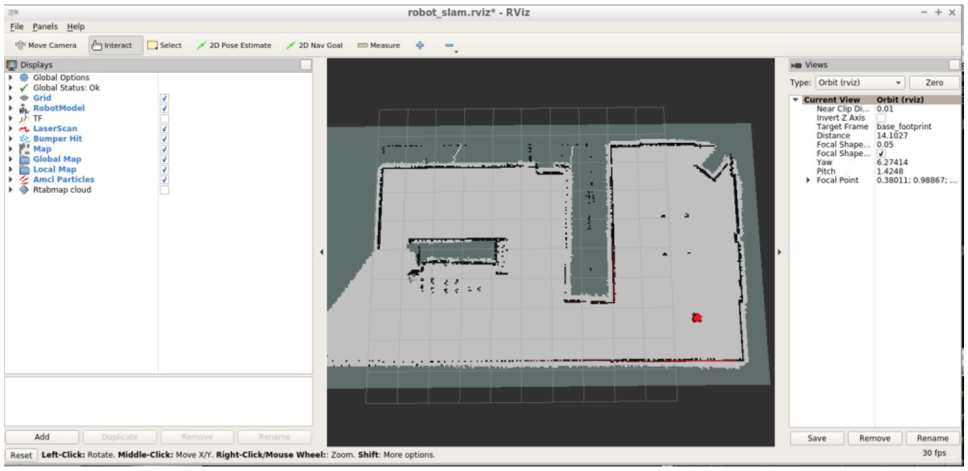

Gazebo
File Edit Camera View Window Help
World Insert Layers
In Selection Mode, hold Ctrl and select 2 objects to align

The robot was kept relatively small and nimble to be able to navigate the many tight gaps of the chaotic warehouse environment. The RGBD camera and the laser scanner are both front and center on the robot since the newest information is likely to be coming from the front when moving forward. The packages are separated by their functions. World.launch is only responsible for launching both the world and the robot. Teleop.launch is responsible for translating keyboard inputs into controls for the differential drive. Mapping.launch is responsible for taking care of the mapping logic and parameter routing involved.


frames.pdf
File Edit View Go Bookmarks Help
1 of 1
61.43%
view_frames Result
Recorded at time: 2338.983
map
Broadcaster: /rtabmap/rtabmap
Average rate: 20.303 Hz
Most recent transform: 2339.057 (-0.074 sec old)
Buffer length: 3.300 sec
odom
Broadcaster: /gazebo
Average rate: 10.303 Hz
Most recent transform: 2338.962 (0.021 sec old)
Buffer length: 3.300 sec
robot_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (2338.983 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (2338.983 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (2338.983 sec old)
Buffer length: 0.000 sec
Broadcaster: /robot_state_publisher
Average rate: 10.312 Hz
Most recent transform: 2338.962 (0.021 sec old)
Buffer length: 3.297 sec
Broadcaster: /robot_state_publisher
Average rate: 10.312 Hz
Most recent transform: 2338.962 (0.021 sec old)
Buffer length: 3.297 sec
camera_link
hokuyo
left_wheel
right_wheel
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (2338.983 sec old)
Buffer length: 0.000 sec
camera_rgbd_frame

Results



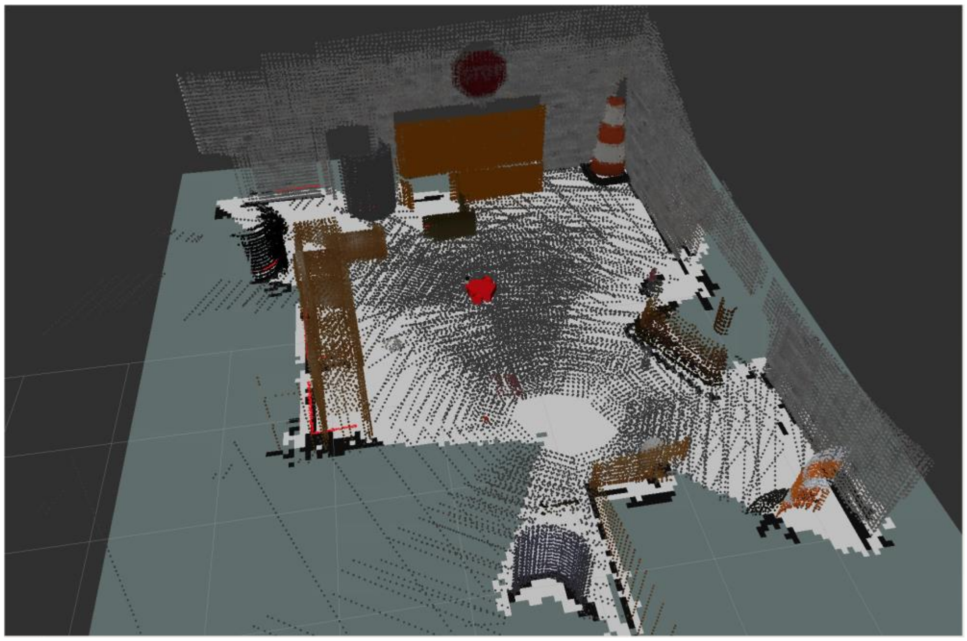


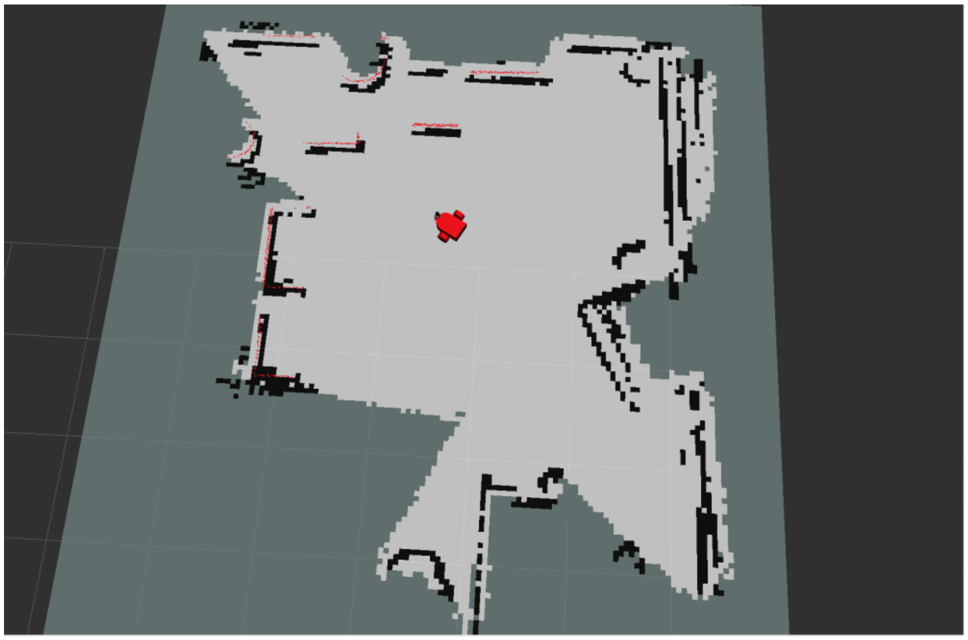
Global Options	
Global Status: Ok	
Grid	✓
RobotModel	✓
TF	☐
LaserScan	✓
Bumper Hit	✓
Map	✓
Global Map	✓
Local Map	✓
Amcl Particles	✓
Rtabmap cloud	☐

Type: Orbit (rviz) Zero

Current View	Orbit (rviz)
Near Clip Di...	0.01
Invert Z Axis	☐
Target Frame	base_footprint
Distance	14.1027
Focal Shape...	0.05
Focal Shape...	✓
Yaw	6.27414
Pitch	1.4248
Focal Point	0.38011; 0.98867; ...

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options.





Although there are a couple cracks and spaces not entirely defined because the robot couldn't squeeze through all the tight spaces, the robot was successfully able to generate the map and occupancy grid for both worlds.

Discussion

The initial roadblock for me was just understanding the structure of ROS and how each part interconnects and plays off each other. The first thing I got stuck on was the teleop feature. I looked at the turtlebot's teleop package and tried to emulate as much as possible, but I didn't have a firm understanding of ROS packages and namespaces which lead to a lot of debugging. The biggest roadblocks in this project for me was definitely just debugging urdf frames and topic subscribers and publishers. ROS is a lot less "magical" thanks to the hours and hours of debugging poured into this project.

RTAB-mapping was easier in the kitchen dining world because unique objects and locations were more dense throughout the area. However, the custom world had fewer tall objects which the robot seemed to struggle with in the kitchen. I think it is a good idea to choose a chaotic environment since it exacerbates the number of unique features and vocabulary allowing for better loop closure detection. It's possible that the reason it struggled with taller features is because of camera placement. The structure of the robot itself is not very tall to emphasize structural stability and lower center of gravity allowing for quicker maneuvers.

Future Work

In terms of this project, I think a good next step is to implement localization and test a taller frame for the robot. I would love to extend this project to a real world robot to map out areas that are hard to explore on foot. I think attaching this to a drone and mapping remote and unexplored locations could have a big application in terms of reconnaisance and mapping. One example is mapping out a path for rock climbers. Since the cliff faces are constantly changing due to rock slides, erosion, and natural cause, a drone can be sent up to map all the geographical features to determine the best possible path.