Mapping - Map My World Robot

Jason, Yu-Chieh Huang

Abstract—A mapping problem is crucial for a robot in an unknown environment or in a dynamic environment which locations of objects are changing. With the latest map information, the robot can plan the correct path to reach goals. A robot with a laser-range finder and a Kinect navigates two worlds, and 3D mappings and occupancy grids are constructed. It adopts the Real-Time Appearance-Based Mapping (RTAB-Map) to map worlds. RTAB-Map have loop closures and memory management functions to correct odometry errors in a real-time performance. The result shows that 3D mappings and occupancy grids are similar to worlds, and obstacles are correct identified.

Index Terms—Robot, Mapping, RTAB-Map.

1 Introduction

A mapping problem is a task to build maps given a robot's pose, and it is an inverse problem with respect to a localization problem, which the robot estimates its poses base on maps. With maps information, the robot can execute the path planning to reach goals.

However, there are many challenges to solve a mapping problem. For example, the three dimension world is a continuous space, and there are a large amount of variables. Data collected by sensors are often noisy, and it is likely to accumulate to a huge deviation even noise is small. Moreover, the robot is likely to be confused by similar scenes at different locations.

In a static environment, the robot continues to update maps information to increase maps accuracy from noisy measurements. On the other hand, in a dynamic environment, even the robot has original maps information, it still needs to update maps data, since the environment is likely to change, and this can avoid collisions against moving obstacles.

Moreover, a real-time processing is required on the robot. In many cases, the computation resource on the robot is limited. A method to map with acceptable computation requirements is demanded.

2 Background

The mapping problem can be considered as 2D or 3D problems. A 2D mapping is a simpler case, since it has less variable compared to the 3D case. The occupancy grid discrete the continuous space, and reduce complexity of the problem. Since measurement errors is difficult to remove, multiple sensors fusion with De Morgan's law is a technique to increase the belief whether grids are occupied or not.

As to 3D mapping, the way to represent space has high impact to the system performance. Point clouds and Voxels require high memory and computation resource, and Octrees is one of the memory efficient and tree based data representation. The main concept is to use different resolution base on local regions complexity.

The Real-Time Appearance-Based Mapping (RTAB-Map) is adopted to solve the mapping problem in this paper. Appearance-Based means that data are collected from vision sensors, and data are processed to correct error from the odometry. As the robot moves, it continues to calculate the motion from odometry measurements. However, errors from the odometry are accumulated when the robot moves. New images from previous visited locations are registered as images in different locations, not original ones. Therefore, maps are usually blur and inconsistent to the real scene.

A loop closure mechanism is adopted to solve this problem. Features from images are extracted and compare to previous ones. If features are high correlated, the robot will consider them from the same location. Base on a features comparison, the robot can adjust errors from the odometry.

The image database is increased linearly as time goes by. Comparisons on a large amount of images consume lots of resource, and the real-time performance is violated. RTAB-Map choices a memory management mechanism to keep the database nearly constant and the process is real time.

The short-term memory subsystem is designed to extract features from recent images, and create nodes. The weighting of nodes are updated if new images have high correlation and are likely from the same place. RTAB-Map keeps moving the oldest nodes to the working memory if nodes in the short-term memory exceed a fixed size. The loop closure compares the latest image to images in the working memory. When there are too many images in the working memory and the real-time performance is going to break, the oldest and lower weighting nodes are transferred to the long-term memory. If a loop closure is detected, neighbors in the long-term memory is retrieved back to the working memory.

Therefore, nearly constant nodes size is maintained and the real-time performance is kept.

3 Simulations

3.1 Robot Model

The robot model has a 0.4 x 0.2 x 0.1 chassis, and there are two side wheels with two spherical slippery casters under the chassis. A Hokuyo laser range finder is mounted on the top of the chassis, while a Kinect, a RGBD camera, is attached to a rod on the top of the chassis. In fig.1, the

robot model in Gazebo is shown; in fig.2 shows the robot's transformation graph, and it clearly illustrates the hierarchy of links as expected connections.

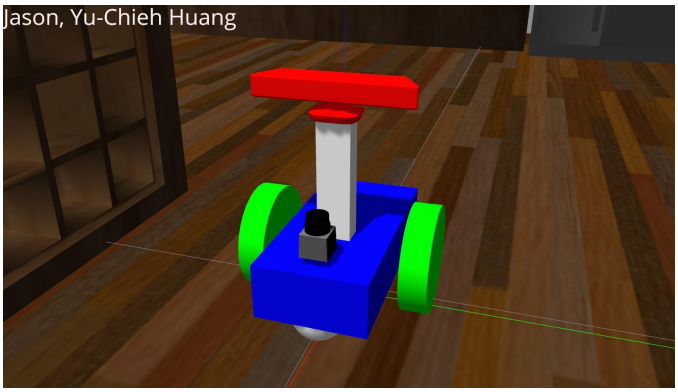

Fig. 1. The robot model.

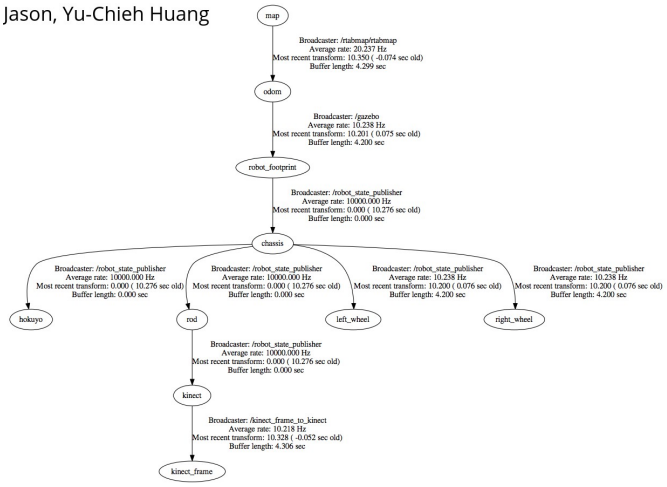

Fig. 2. The robot transformation graph.

3.2 Benchmark World

The benchmark world is provided by Udacity, and is shown in fig.3.

Fig. 3. The benchmark world.

3.3 Mine World

The customized world is built base on a Gazebo cafe scene, but it is modified to a small factory which re-engineering cars, maybe self-driving cars. Many small items are added to increase mapping performance, including packages, cokes and biers, as shown in fig.4.

Fig. 4. The mine world.

4 Results

4.1 Benchmark World

The 3D mapping and the occupancy grid of the benchmark world are shown in fig.5 and fig.6. The 3D mapping illustrates good representations of the real model in both RGB colors and depth information. In the occupancy grid, locations of scene components are depicted, and there are 14 loop closures.

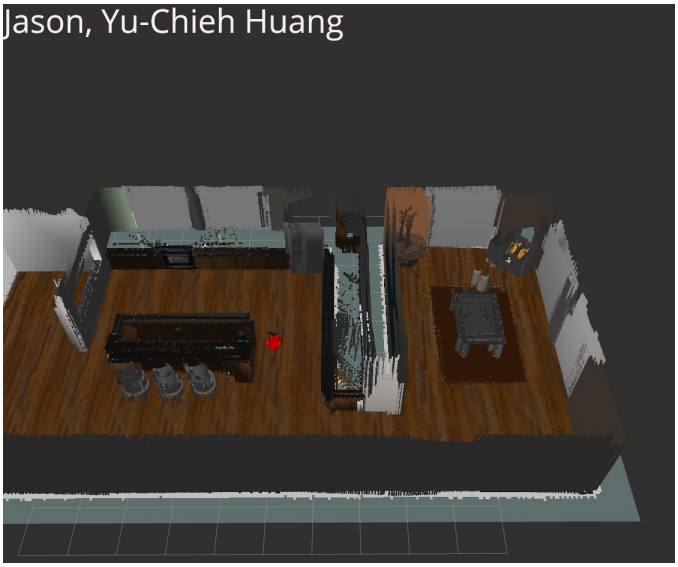

Fig. 5. The 3D mapping of the benchmark world.

4.2 Mine World

The 3D mapping and the occupancy grid of the Mine world are shown in fig.7 and fig.8. The 3D mapping illustrates good representations of the real model in both RGB colors and depth. In the occupancy grid, locations of scene components are depicted, and there are 25 loop closures.

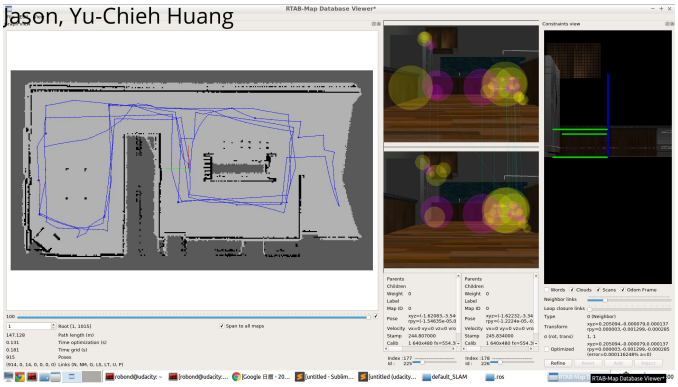

Fig. 6. The occupancy grid of the benchmark world.

Fig. 7. The 3D mapping of the Mine world.

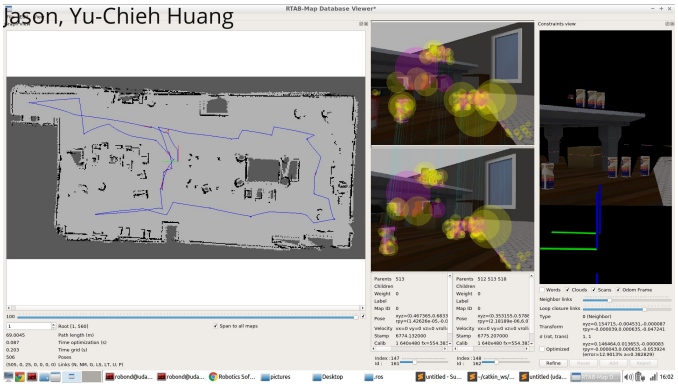

Fig. 8. The occupancy grid of the Mine world.

5 Discussion

5.1 Rtab mapping on different worlds

Rtabmap relies on loop closures to adjust error between each frames. Loop closures are more than 10 for both worlds, and 3D worlds and 2D occupancy grids are good compared to the original simulation environments.

5.2 ROS data flow

Fig.9 illustrates data flow between ROS nodes and topics. The *teleop* node receives user inputs and send to the *cmd_vel* topic to control the robot movement inside Gazebo.

The *rtabmap* package requires data of laser scan from the Hokuyo laser range finder and RGBD information from Kinect. Base on these sources, the *rtabmap* package can reconstruct 3D worlds and 2D occupancy grid.

Tf messages are collected from the *joint_state_publisher*, *robot_state_publisher* and *kinect_frame_to_kinect*. The *kinect_frame_to_kinect* is introduced to solve the inconsistency of the Kinect 3D model and signal, and it will be discussed in section 5.2.

This graph is a useful tool when connecting nodes and topics. We can verify whether connections are correct or not by reviewing it. In fig.10, it is an example when *teleop/cmd_vel* is not remapped to *cmd_vel* properly, that is this line of code is not added. The *teleop* node, a red encircled part, is not connected to *gazzebo*, then the robot can not be controlled by keyboard inputs. Therefore, correct connections between nodes and topics can be achieved after reviewing the rqt graph.

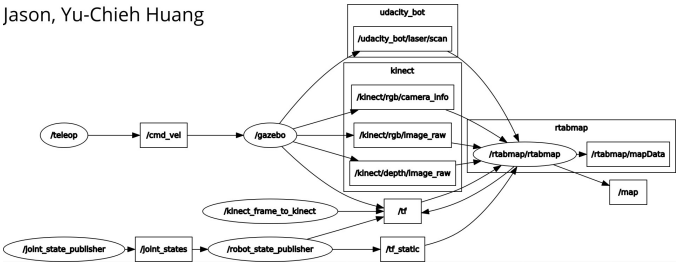

Fig. 9. The rqt graph.

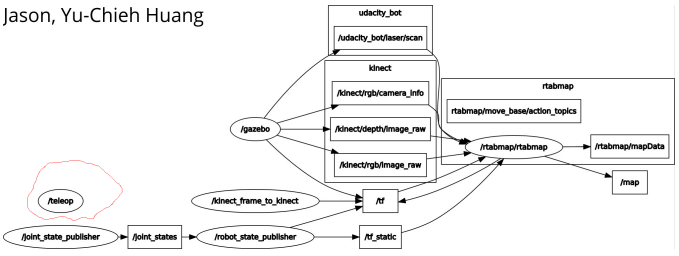

Fig. 10. A wrong-connection example.

5.3 Kinect 3D model and signal adjustment

When launching default configurations, the Hokuyo laser range finder shows that red laser aligns with the scene structure. However, the Kinect is facing on the right side of the robot, but the receiving signal is a front-side depth

image, as shown in fig.11. This shows that the Kinect visual is inconsistent with the real sensing direction. Therefore, the Kinect visual is rotated to align with the sensing direction, facing front, as shown in fig 12.

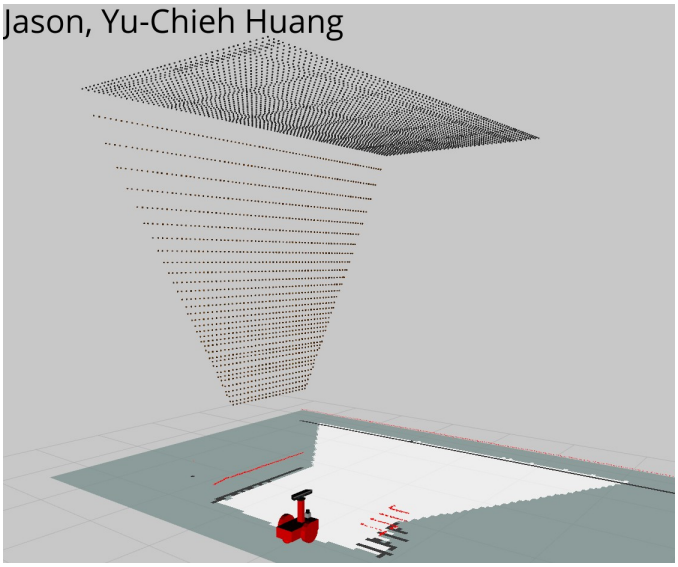

Fig. 11. Original Kinect sensing in rviz.

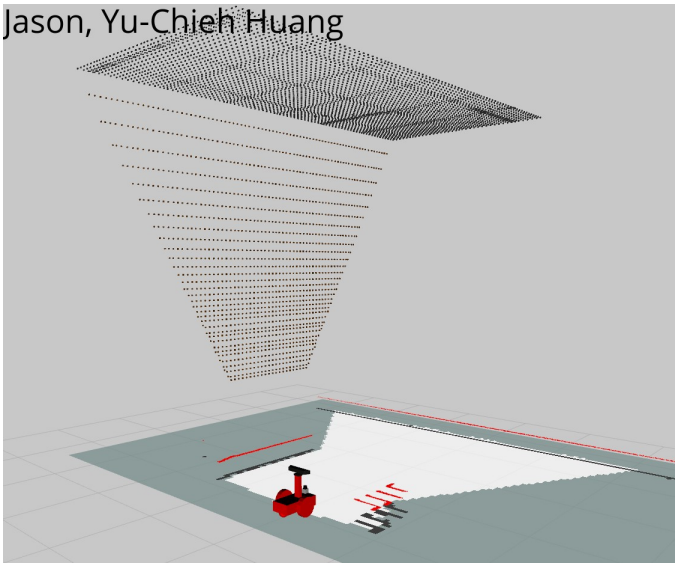

Fig. 12. Kinect sensing in rviz after changing xacro visual params.

Nonetheless, point clouds is on the top of the robot, and this denotes the Kinect frame should be rotated. After rotating Kinect frame to Kinect tf, point clouds are aligned with the scene structure, as shown in fig.13.

6 Conclusion / Future work

In this work, robot mappings in two different worlds are present. Both 3D mappings and 2D occupancy grids are success. Using mapped worlds to localize the robot will be an interesting task.

Moreover, the Kinect sensor provides RGBD signal, and the laser input for the *rtabmap* package can be replaced by the depth information from the Kinect. To achieve this, the *depthimage_to_laserscan* [1] package can extract 2D depth information of a plane from Kinect 3D clouds. Therefore, remoing the Hokuyo laser-range finder, and adopting only the Kinect for both RGBD image and laser depth information will be the future work.

The system can be deployed to a real robot, and let it to move in a building. We can estimate the real-time performance by monitoring whether a robot can avoid collisions with moving obstacles, like pedestrians, or not.

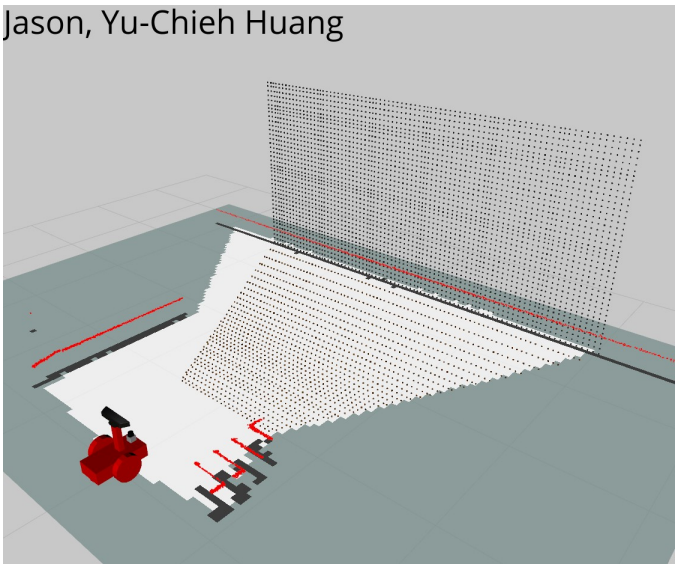

Fig. 13. Final Kinect sensing in rviz after changing xacro visual and tf params.

References

[1] *ROS wiki - Depthimage to Laserscan*.