Map My World Robot

Abstract

The third project in term 2 of the Udacity Robotics Nano Degree program requires students to use ROS and Gazebo along with RTAB-Map, to create a 2D occupancy grid and a 3D octomap of two environments - one supplied and the other student created.

Students extend a previous robot creation to upgrade sensors to supply the necessary sensor messages for RTAB-Map. This leverages the laser scanner, IMU/Wheel Encoder but replaces the camera with a RGB-D camera (ie kinect).

Further the ROS project is created with all links connected with appropriate naming and mapping.

The robot is launched and teleoped around the room to generate a map of the environment.

After successfully mapping the supplied environment, a student defined environment is created and mapped using the same technique.

Introduction

In this project a robot model uses a Simultaneous Localisation and Mapping (SLAM) technique called RTAB-Map (Real-Time Appearance-Based Mapping). It is a RGB-D Graph Based SLAM approach that uses incremental appearance based loop closure detection.

The RTAB-Map ROS wrapper is leveraged with visual representation in real time via rtabmapviz. The resultant map is stored in local database that be later interrogated via rtabmap-databaseViewer.

Background

When a robot encounters a new environment where there is no supplied map, it needs to be able to create this map and localise its pose using it. This combined localisation and mapping process is referred to as SLAM (Simultaneous Localisation and Mapping).

The main mapping algorithms are Occupancy Grid Mapping, Grid-based FastSLAM, Graph-SLAM and RTAB-Map.

The Occupancy Grid Mapping is a 2D algorithm where each grid cell is identified as Unknown/Undiscovered Zone, Free Zone or Occupied. This represents a slice of the 3D world.

The Grid-Based FastSLAM approach combines SLAM (Synchronised Location and Mapping) using a MCL (Monte Carlo Localisation) Algorithm and an Occupancy Grid Mapping. The main advantage of is the MCL particle filter approach but it always assumes there are known landmark positions. Thus it is unable to model an arbitrary environment.

Graph-SLAM uses a graph based approach to represent poses, features from the environment, motion constraints (between two poses) and measurement constraints (ties together a feature and a pose). It solves the full SLAM problem, it covers the entire path and map and not the most recent pose.

This project uses RTAB-Map, which is a Graph-SLAM approach that uses loop closure with Visual Bag-of-Words for optimisation.

The loop closure detection occurs against working memory to constrain the number of images interrogated. Working memory can be transferred and retrieved from long term memory to reduce complexity. The algorithm used for loop closure detection is SURF (Speeded Up Robust Features).

The possible outputs of RTAB-Map are 2D occupancy grid map, 3D octomap or a 3D point cloud.

Robots are of varying dimensions inclusive of height. Whilst mapping a 2d environment may show where fixed walls etc are it does not take into account height. A robot, that is propelled on the floor, may be able to navigate under some obstacles but not others eg a chair vs a large table. Hence the need to understand the environment from a 3D perspective.

However building a 3D map is more costly then a 2D map. This is not only in terms of Compute & Data costs but also in the cost of the sensors required. However, simple sensors such as a single camera may be cheaper but the algorithms required can be more complex.
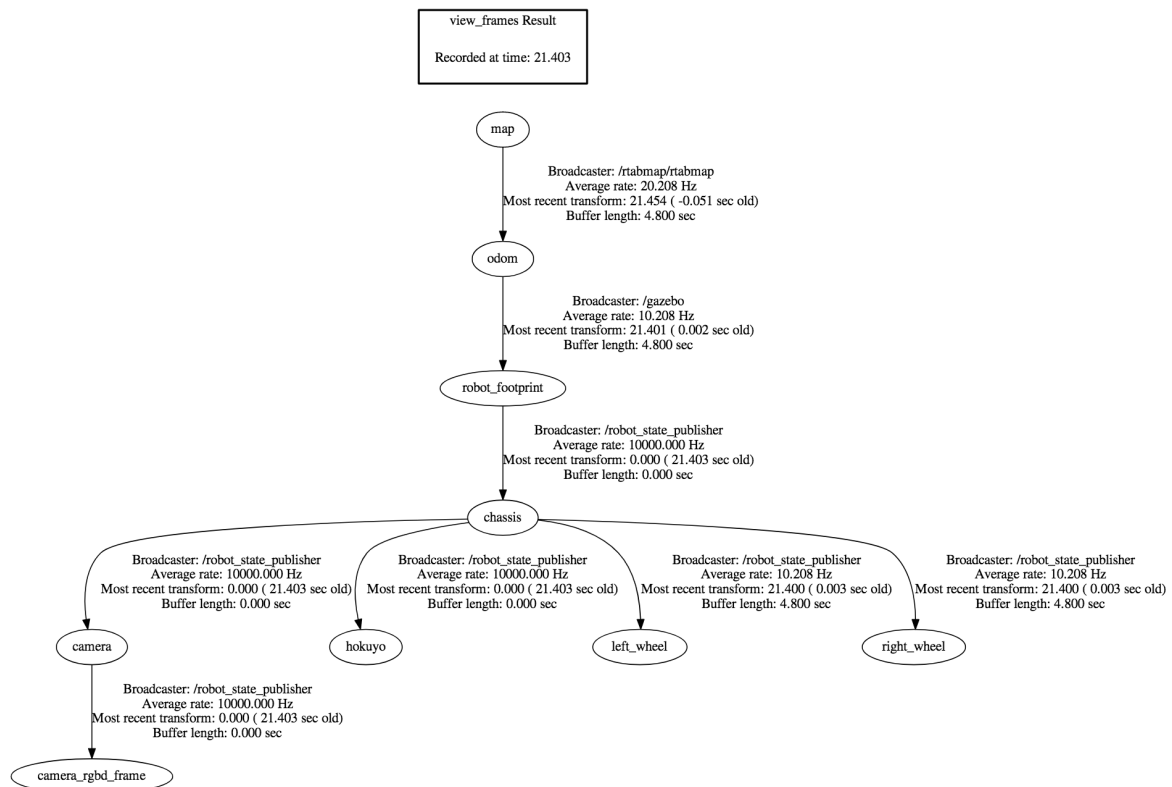
Robot Model Configuration

The robot model used was based on the nick_bot created in the previous project as the student robot model (which had a square base with two actuators for the left and right wheels). The camera was removed and replaced with a kinect leveraging the openni_camera ros package with the gazebo controller Openni Kinect.

No changes were made to the hokuyo laser range finder.

An additional joint was added to rotate the kinect data 180%. It was positioned on the front of the robot so as to not interfere with the laser range finder.

The nick_bot configuration files can be found under the urdf directory.

Visualization of the frames follows

view_frames Result

Recorded at time: 21.403

map

Broadcaster: /rtabmap/rtabmap
Average rate: 20.208 Hz
Most recent transform: 21.454 (-0.051 sec old)
Buffer length: 4.800 sec

odom

Broadcaster: /gazebo
Average rate: 10.208 Hz
Most recent transform: 21.401 (0.002 sec old)
Buffer length: 4.800 sec

robot_footprint

Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (21.403 sec old)
Buffer length: 0.000 sec

chassis

Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (21.403 sec old)
Buffer length: 0.000 sec

Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (21.403 sec old)
Buffer length: 0.000 sec

Broadcaster: /robot_state_publisher
Average rate: 10.208 Hz
Most recent transform: 21.400 (0.003 sec old)
Buffer length: 4.800 sec

Broadcaster: /robot_state_publisher
Average rate: 10.208 Hz
Most recent transform: 21.400 (0.003 sec old)
Buffer length: 4.800 sec

camera

hokuyo

left_wheel

right_wheel

Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (21.403 sec old)
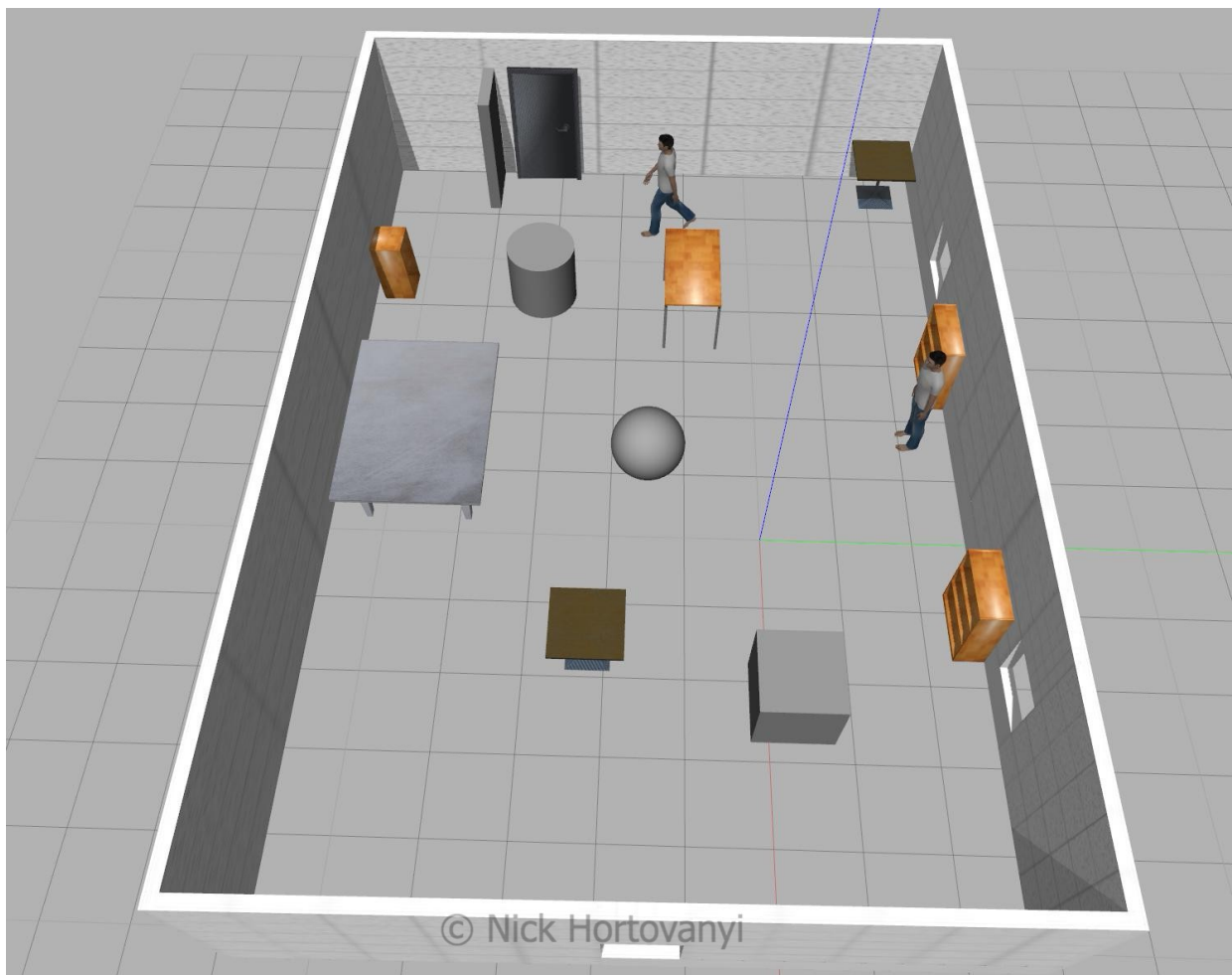Buffer length: 0.000 sec

camera_rgbd_frame

`frames.png`

World Creation

Two worlds were created in gazebo - one supplied as `kitchen_dining.world` and the other student customised `nicks_building.world`

kitchen_dining.world

`nicks_building.world`

Fixtures were selected in `nicks_building.world` to give sufficient points for the SLUR algorithm to detect distinct points. Items were also placed to ensure that the features could not be mistakenly detected twice ie a person was placed next to one of the bookshelves. Sufficient space was left for the robot to navigate.

The following tree diagram depicts the package structure and files that made it up.

```
├── CMakeLists.txt
├── launch
│   ├── config
│   │   └── robot_slam.rviz
│   ├── mapping.launch
```

```
|   ├── robot_description.launch
|   ├── rviz.launch
|   ├── teleop.launch
|   └── world.launch
├── materials
|   └── textures
|       └── kinect.png
├── meshes
|   ├── hokuyo.dae
|   └── kinect.dae
├── output
|   └── rtabmap.db
├── package.xml
├── README.md
├── rtab_run
├── teleop
├── urdf
|   ├── nick_bot.gazebo
|   └── nick_bot.xacro
└── worlds
    ├── kitchen_dining.world
    └── nicks_building.world
```

This tree structure was chosen based on the previous student project that conformed to known gazebo/ros standards with the addition of an `output` directory to store Udacity project writeup requirements.
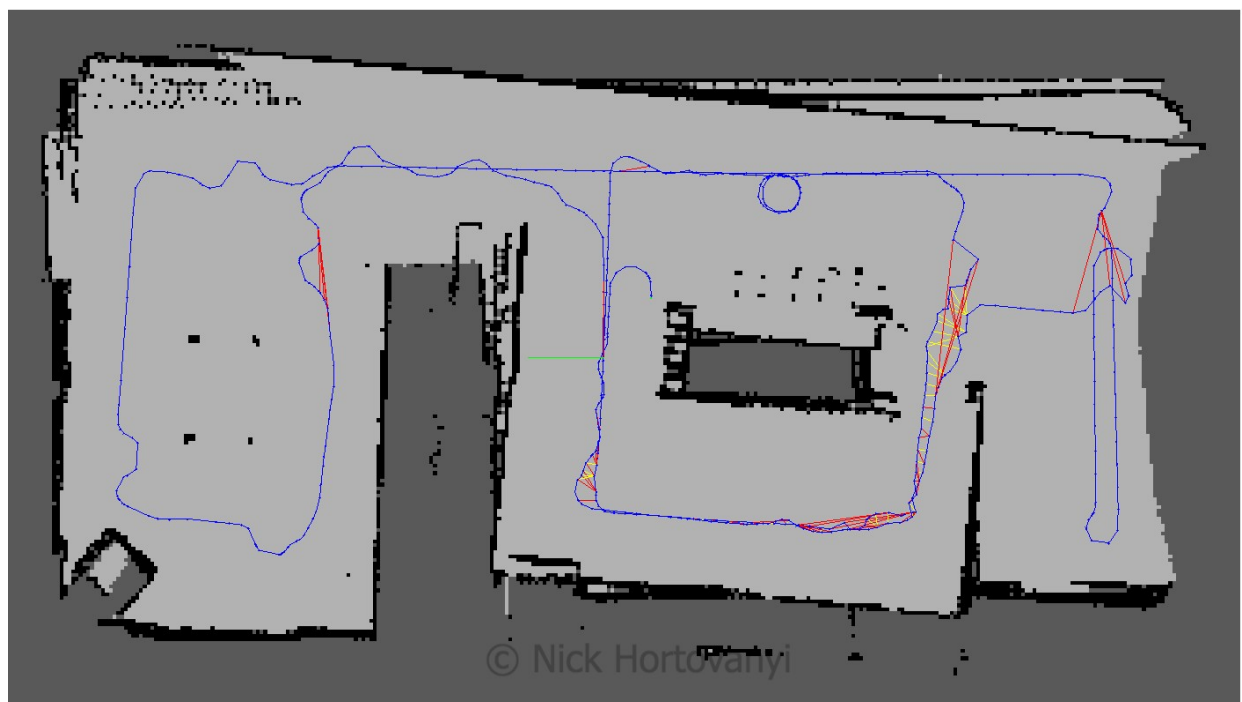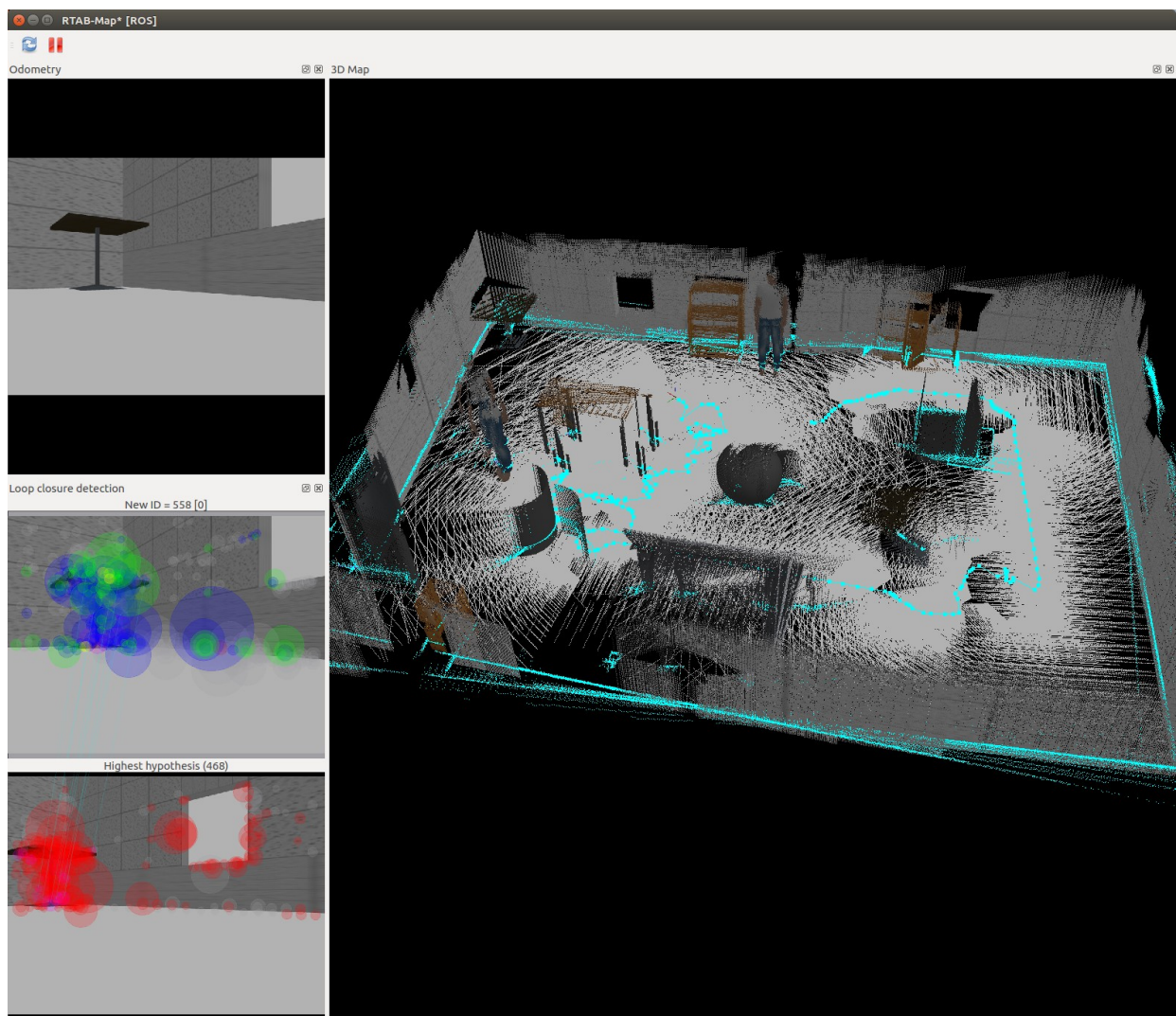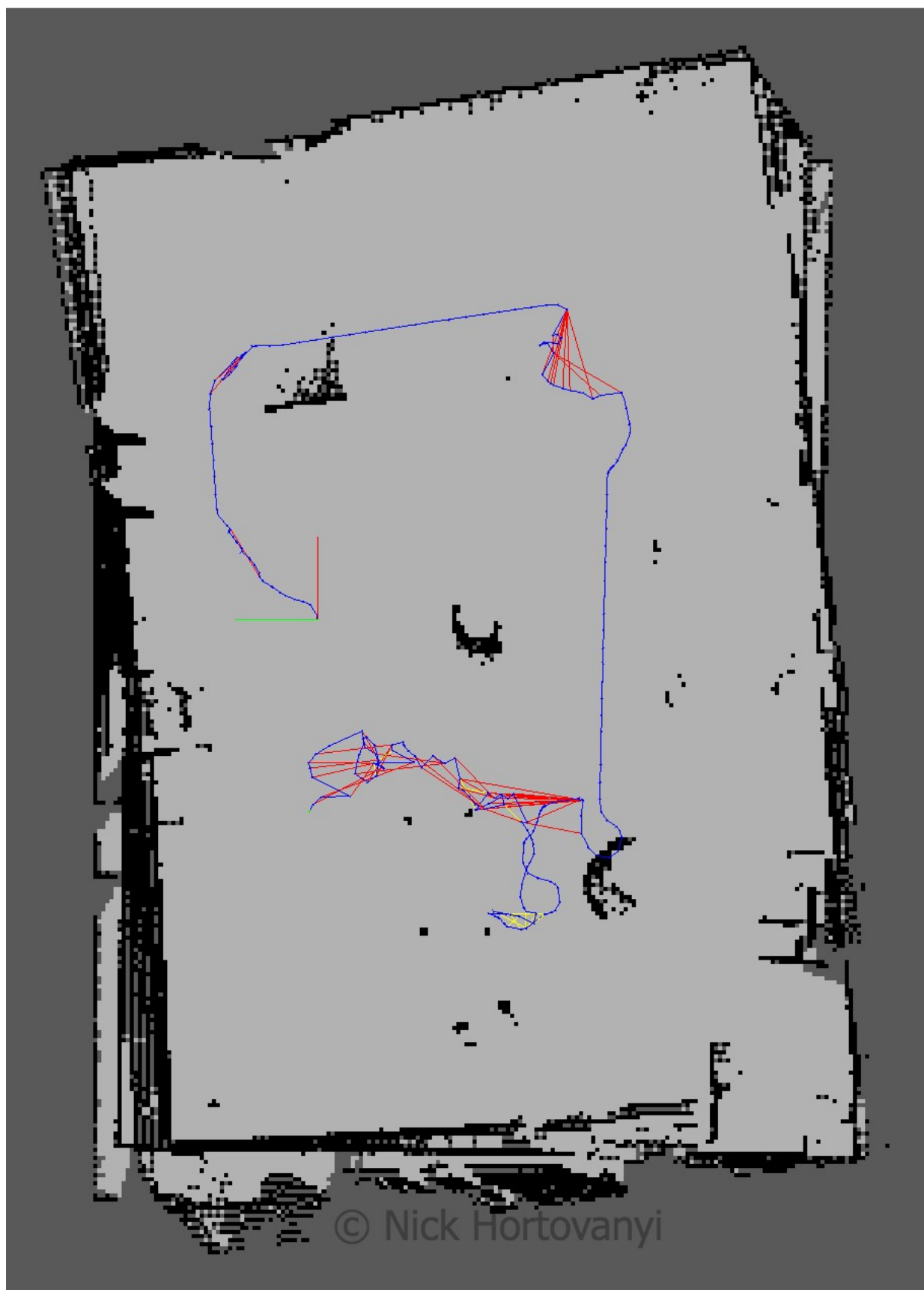
Results

kitchen_dining_3d
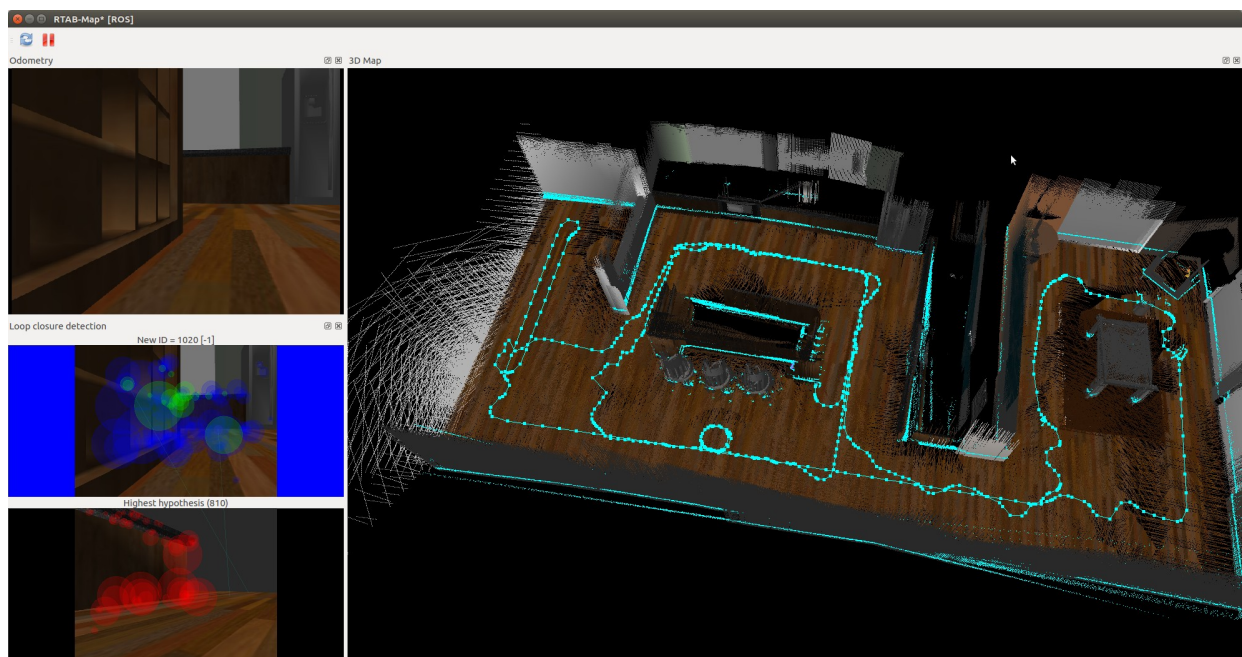
kitchen_dining_2d

nicks_building_3d

nicks_building_2d

Discusion

The robot was teleoped (navigated via the keyboard) around the room. At some points the robot did not move forward. This appeared to be when it started to perform loop closure. `Kp/MaxFeatures` was halved to 200 and `Vis/MinInliers` was reduced from 15 to 10.

However the 3D map quickly started to resemble the physical kitchen dining gazebo model. To improve loop detection rates some, on the spot circles were performed. Of particular note were the features in the main kitchen area. More SURF features were identified there as there was more variation in the surfaces.

The nick building gazebo model wall surfaces were tiled, repeatable pattern with lack of other discerning features sometimes caused the loop closure detection to map to an incorrect previous image. This then distorted the map. Additional features were added to achieve a successful map.

The kitchen_dining model performed significantly better then the student created nicks_building model. This was due to the richer and more complex features of the kitchen_dining model.

Future Work

Mapping is important to help understand the world. There are a plethora of sensors and of interest is the about to arrive solid state lidars. As the price point of these sensors continues to drop it will open up opportunities to create richer and more realistic 3D maps at a cheaper price point.

Being able to map an environment cost effectively to create a replicated virtual world will increasingly be important to allow for the training of deep learning models. We are actively looking to do this and then supplant the trained model back into a robot so it can navigate in the original environment that was mapped.