RoboND SLAM Project Map My World Robot

Noriaki Ibe

Abstract—The third project in term 2 of the Robotics Software Engineer Nanodegree Program requires students to create a 2D occupancy grid and 3D octomap from a provided simulated environment. Furthermore, students will then create own simulated environment to map as well.

Index Terms—Robot, IEEEtran, Udacity

1 Introduction

THE Mapping is one of the main part in Robotics to determine its unknown environment given, especially moving base such as iRobot etc.

The robot must collect observations of features in the world to execute control. The project focuses on Simultaneous Localization and Mapping (SLAM), which is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent's location within it.

SLAM is a chicken-or-egg problem:

- a map is needed for localization and
- a pose estimate is needed for mapping

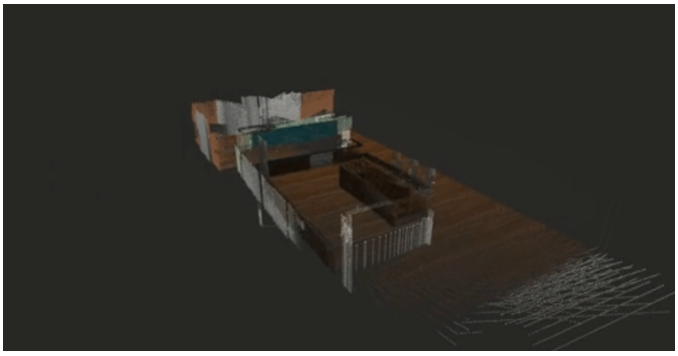

Fig. 1. SLAM example

The lectures of Udacity covers several kinds of SLAM. Then, The project requires students to complete

- Develop own package to interface with the rtabmap_ros package.
- Build upon second project of term2 to make the necessary changes to interface the robot with RTAB-Map. An example of this is the addition of an RGB-D camera.
- All files are in the appropriate places, all links are properly connected, naming is properly setup and topics are correctly mapped, which is including the appropriate launch files to launch the robot and map its surrounding environment.
- When the robot is launched, students shall teleop around the room to generate a proper map of the environment.
- Build own simulated environment and apply the code you used to map the supplied environment on this environment.

2 Background / Formulation

The lecture of Udacity explained two SLAMs such as FastSLAM and GraphSLAM from follows:

- Extended Kalman Filter (EKF) SLAM
- Sparse Extended Information Filter (SEIF)
- Extended Information Filter (EIF)
- FastSLAM
- GraphSLAM

1. SLAM The inputs and outputs to the SLAM problem are Input: Measurements+Controls and Output: Map+Trajectory and it has two key features such as Forms and Nature.

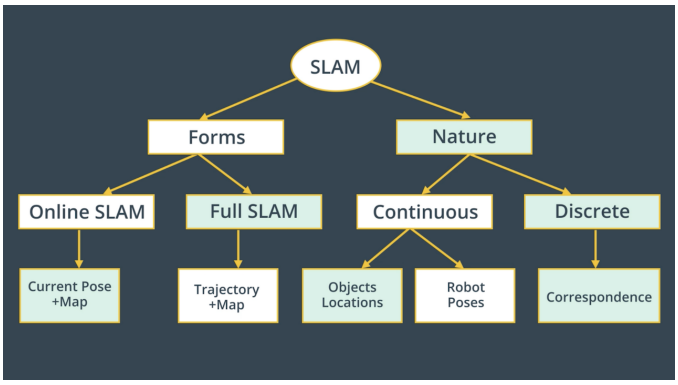

Fig. 2. SLAM Key Features

The SLAM problem has two Forms as Online SLAM and Full SLAM.

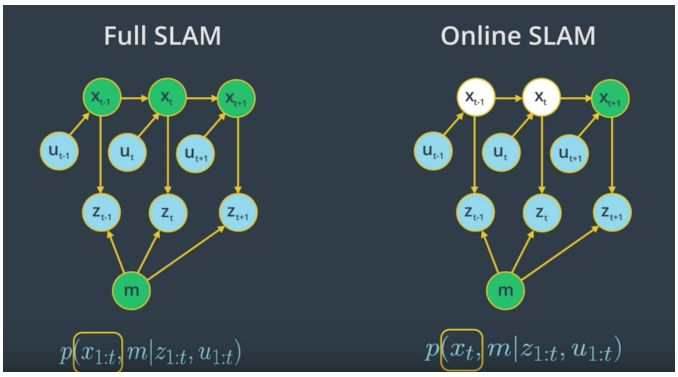

Fig. 3. SLAM time:t, pose:x and measurements:z

As shown the above, Full SLAM estimates all the variables(Map + Trajectory) that occur throughout the robot travel time, instead Online SLAM estimates variables(Map + Pose) that occur at time t only.

The second key feature of the SLAM problem relates to its nature. SLAM problems generally have a continuous and a discrete element.

- Continuous: Robot continuously senses its pose and the location of the objects.
- Discrete: Robot has to identify if a relation exists between any newly detected and previously detected objects.

While performing SLAM, a mobile robot is establishing a relation between newly and previously detected objects. SLAM problem has to do with Nature: Discrete and This discrete relation between objects is known by correspondence. Then, correspondent value c could be added into Graph.

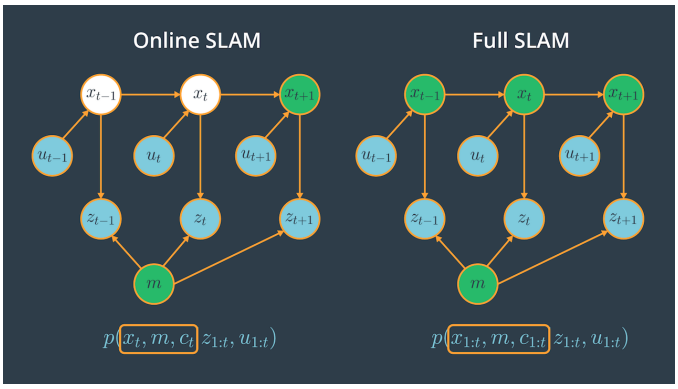

Fig. 4. SLAM Key Features

But the particle filter approach (EKF, MCL etc.) to SLAM in this current form will scale exponentially and is doomed to fail, this is because the map is modeled with many variables resulting in high dimensionality.

2. FastSLAM FastSLAM can be solution under a custom particle filter approach.

The FastSLAM algorithm(MCL + Low-Dimensional EKF) solves the Full SLAM problem with known correspondences.

- Estimating the Trajectory: FastSLAM 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.
- Estimating the Map: FastSLAM uses a low dimensional EKF to solve independent features of the map which are modeled with local Gaussian.

Currently, three different instances of the FastSLAM algorithm exist such as FastSLAM 1.0, FastSLAM 2.0 and Grid-based FastSLAM. The third instance of FastSLAM is really an extension to FastSLAM known as the Grid-based FastSLAM algorithm, which adapts FastSLAM to grid maps. The most difference between Grid-based FastSLAM 1,2 and Grid-based FastSLAM is Grid-based FastSLAM has no related to any landmark instead FastSLAM 1.0, 2.0 use the landmark so that FastSLAM 1,2 don't fit to an arbitrary environment. with the grid mapping algorithm, Grid based FastSLAM can model the environment using grid maps without predefining any landmark position.

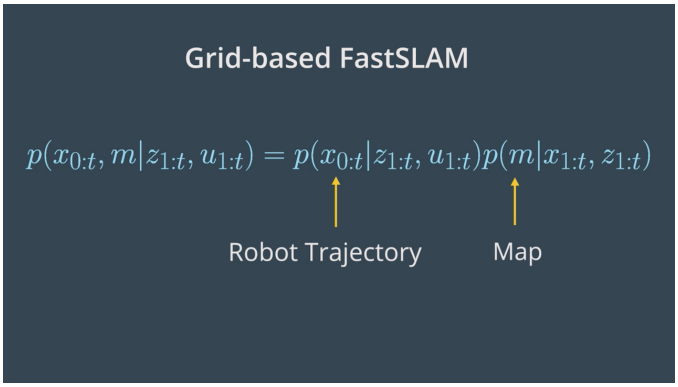

Fig. 5. Grid-based FastSLAM

To extend FastSLAM to Grid-based FastSLAM, it needs three different techniques:

- Sampling Motion- $p(x_t|x_{t-1}^{[k]}, u_t)$: Estimates the current pose given the k-th particle previous pose and the current controls u with MCL .
-
- Map Estimation-$p(m_t|z_t, x_t^{[k]}, m_{t-1}^{[k]})$: Estimates the current map given the current measurements, the current k-th particle pose, and the previous k-th particle map with Occupancy Grid Mapping..
-
- Importance Weight-$p(z_t|x_t^{[k]}, m^{[k]})$: Estimates the current likelihood of the measurement given the current k-th particle pose and the current k-th particle map with MCL.

So the steps of Grid-based FastSLAM shall be taken as follows:

1) Previous belief
2) Sampling motion
3) Importance weight
4) Map estimation
5) Regampling
6) New Belief

ROS has ROS:gmapping package which is based on the Grid-based FastSLAM algorithm to map an environment. And students try SLAMLAb with Udacity Workspace.

3. GraphSLAM

Graph-based Formulation is follows:

- Use a graph to represent the problem
- Every node in the graph corresponds to a pose of the robot during mapping
- Every edge between two nodes corresponds to the spatial constraints between them

And the goal is finding a configuration of the nodes that minimize the error introduced by the constraints J as follows:

$$JGraphSLAM = x_0^T \Omega x_0 + \sum_t (x_t - g(u_t, x_{t-1}))^T R_t^{-1} (x_t - g(u_t, x_{t-1})) + \sum_t (z_t - h(x_t, m_j))^T Q_t^{-1} (z_t - h(x_t, m_j))$$

$$\Omega_0 = \begin{bmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{bmatrix}$$

The first element in the sum is the initial constraint - it sets the first robot pose to equal to the origin of the map. The covariance,Ω_0 represents complete confidence, which are working with multi-dimensional graphs and multi-dimensional constraints, it makes sense to use a more intelligent data structure.

4. RTAB-Map

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D, Stereo and Lidar Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. In case of 3D SLAM With RTAB-Map, the steps are categorized as Front-end and Back-end as follows:

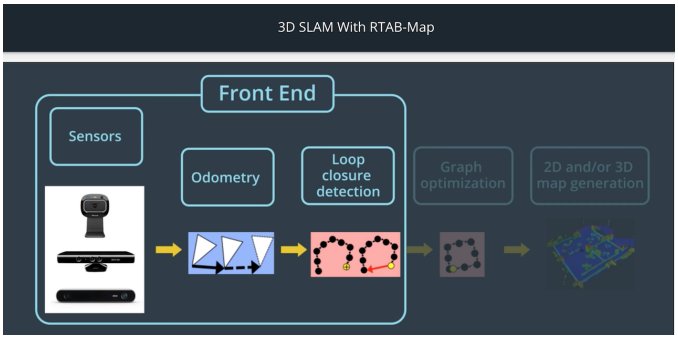

Fig. 6. 3D SLAM With RTAB-Map Front-end

Front-end has functions to use sensor data to obtain odometry and loop closure constrains and loop closure detection is using bag-of-words. The bag-of-words is steps to make inverted index from image database by extracting features as follows:

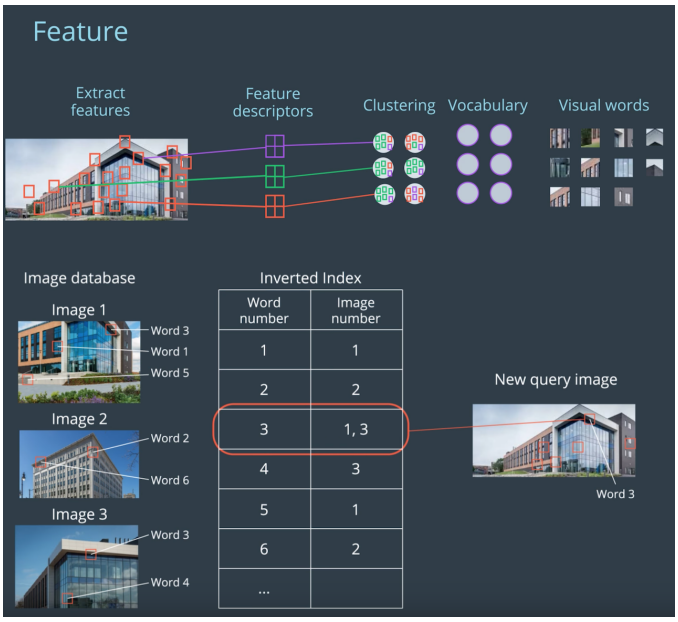

Fig. 7. bag-of-words

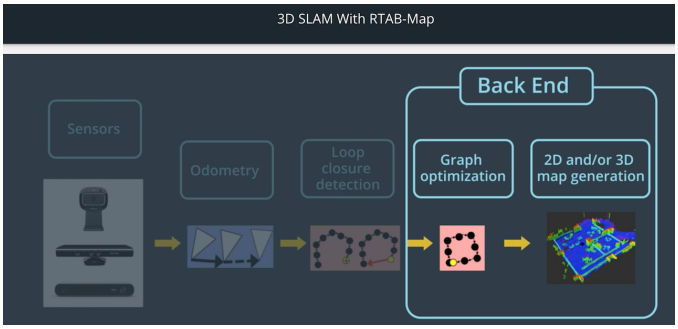

Fig. 8. 3D SLAM With RTAB-Map Back-end

Back-end has functions as Graph optimization and Map-output.

In RTAB-Map, loop closure is one of important rules. When loop closure is disabled, tt is not an accurate representation of the environment. This is caused by the robot not using loop closure to compare new images and locations to ones that are previously viewed, and instead it registers them as new locations. Then, RTAB-Map is able to outputs a 2d Occupancy grid map, 3d occupancy grid map (3d octomap), or a 3D point cloud.

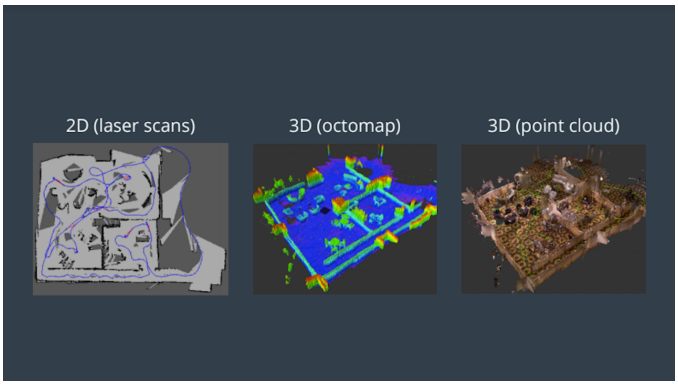

Fig. 9. RTAB-Map output

3 SCENE, ROBOT CONFIGRATION AND TUNING

Robot configuration The project built the robot model based on the last project titled "RoboND-Localization-Project"

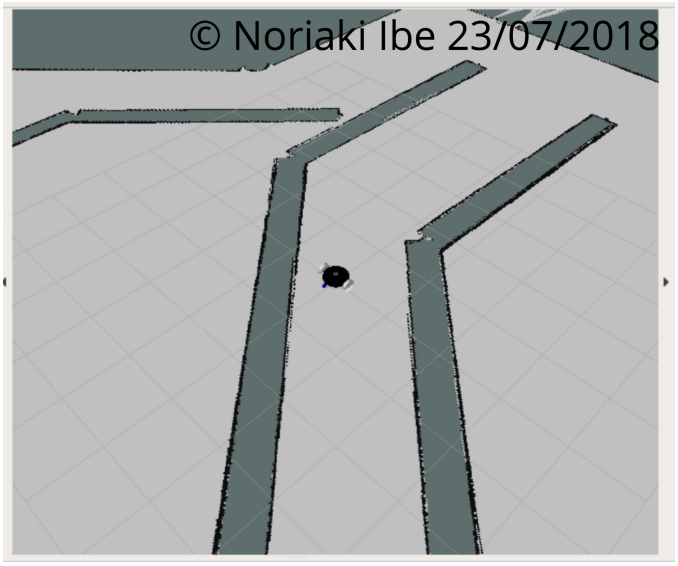

Fig. 10. Own Robot.

Then, the project just add front wheel, back wheel and changed size of side wheel in udacity_bot.xacro. Then, the project follows lectures as follows:

2. Extending your Robot Creation - Sensor Upgrade udacity_bot.gazebo is updated to give that an upgrade from RGB camera to the RGB-D camera.

Then, all frames was coordinated as follows:

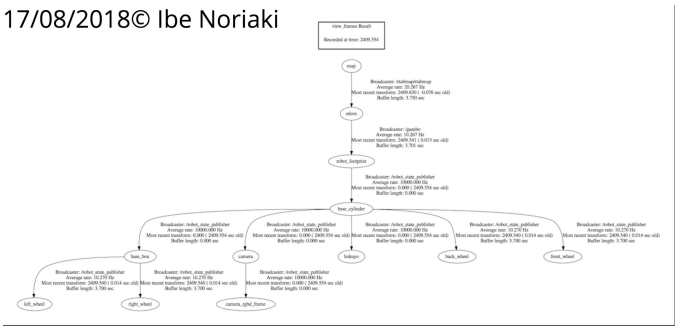

Fig. 11. Frames coordination

3. Extending your Robot Creation - Launch Files The project created all of the necessary launch files based on the last project titled "RoboND-Localization-Project" too.

1) Launch the gazebo world and own robot by world.launch. The last project include launching rviz, but it's deleted for step 4.
2) Launch own teleop node by teleop.launch.
3) Launch your mapping node by mapping.launch. mapping.launch is give in this lecture. Additionally, this project use rtabmapviz, which is an additional node for real time visualization of feature mapping, loop closures, and more. Then, mapping.launch was modified as instructed in this lecture, too.
4) Launch Rviz by rviz.launch. rviz.launch including configration file titled "robot_slam.rviz" are provided.

```
wget https://s3-us-west-1.amazonaws.
    com/udacity-robotics/Term+2+
     Resources/P3+Resources/Student+
     Project+Materials.zip
```

SCENE On the channel"udacity_map_my_world" in the slack "https://udacity-robotics.slack.com/", There is discussion about error to use "kitchen_dining.world" provided. Additionally, @froohoo shares the fixed one on this git "https://github.com/froohoo/RoboND-MapMyWorld" so that this project use it as "kitchen_dining_v2.world".

Fig. 12. Provided Environment: kitchen

Additionally, the project creates own environment to map by Gazebo named myworld_v1.world and myworld_v2.world.

Fig. 13. Created Environments: myworld_v1 and myworld_v2

Tuning After several attempts, there are difficulty to detect environment like spiraling map issue.

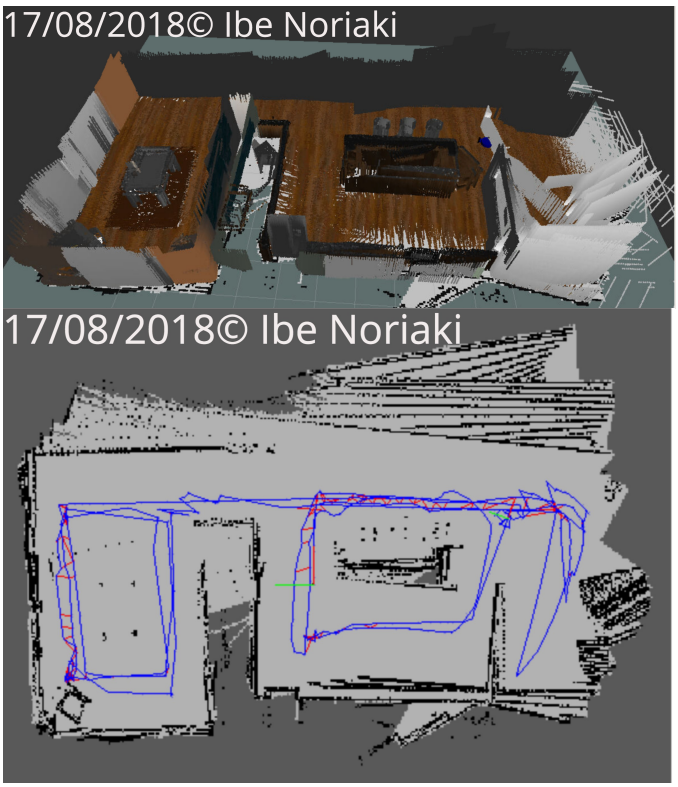

Fig. 14. Created Environments: myworld_v1 and myworld_v2

There is an discussion on slack channel@Udacity and the shared link, then the project set params as follows in mapping.world.

```
<param name="Reg/Strategy" type="string"
    value="2"/> <!-- 0=Visual, 1=ICP, 2=
    Visual+ICP -->
```

Launching This section guides how ROS package to generate a 2D and 3D map of that environment. following steps should be taken to launch and run Ros in Udacity workspace after extend catkin_ws.tar.gz under /home/-workspace/.

```
### 1st Terminal  ( you can set name of
    world in world.launch)
$ cd /home/workspace/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roslaunch slam_project world.launch

### 2nd Terminal
$ cd /home/workspace/catkin_ws
$ source devel/setup.bash
$ chmod +x src/slam_project/teleop
$ roslaunch slam_project teleop.launch

### 3nd Terminal
$ cd /home/workspace/catkin_ws
$ source devel/setup.bash
$ roslaunch slam_project mapping.launch

### 4th Terminal
$ cd /home/workspace/catkin_ws
$ source devel/setup.bash
$ roslaunch slam_project rviz.launch

### You can see rtabmap after run project
$ rtabmap-databaseViewer ~/.ros/rtabmap.db
```

4 RESULTS

4.1 Kitchen

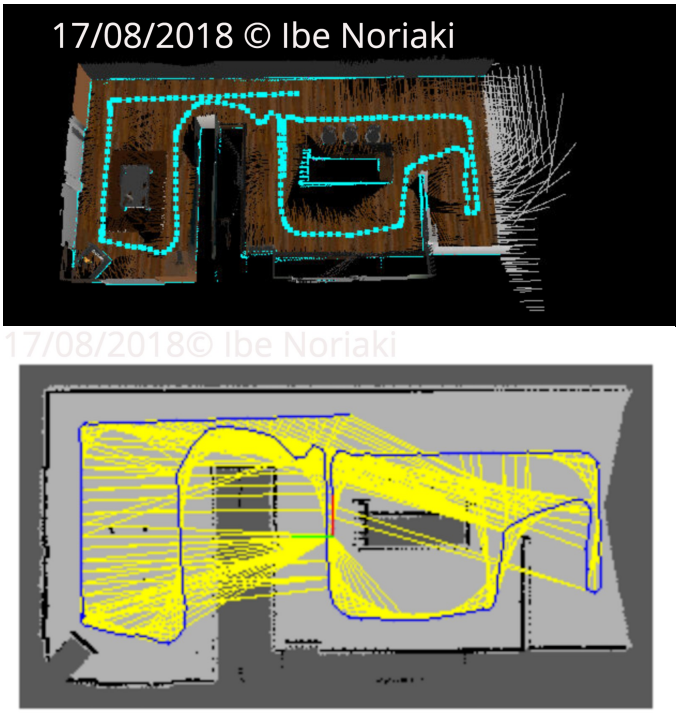

Fig. 15. Kitchen 3D and graph map

4.2 Myworld 1

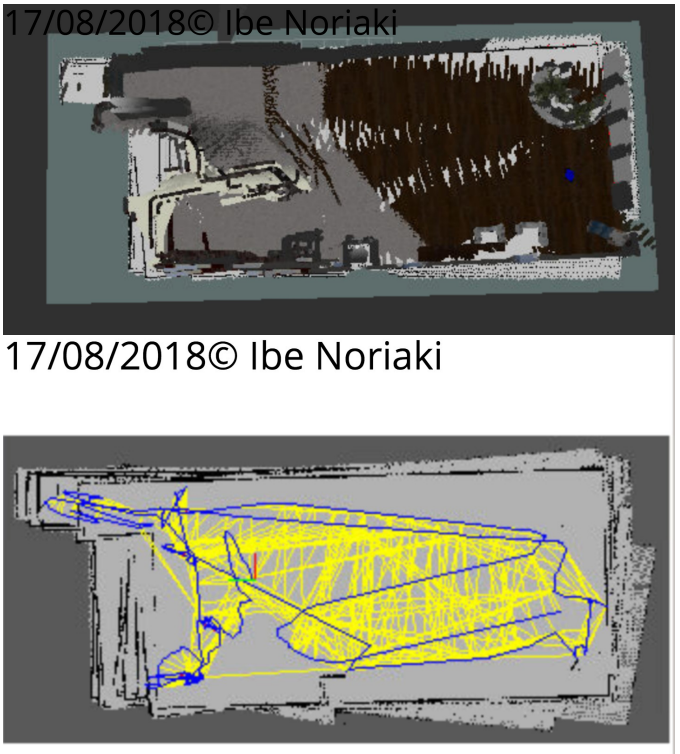

Fig. 16. Kitchen 3D and graph map

4.3 Myworld 2

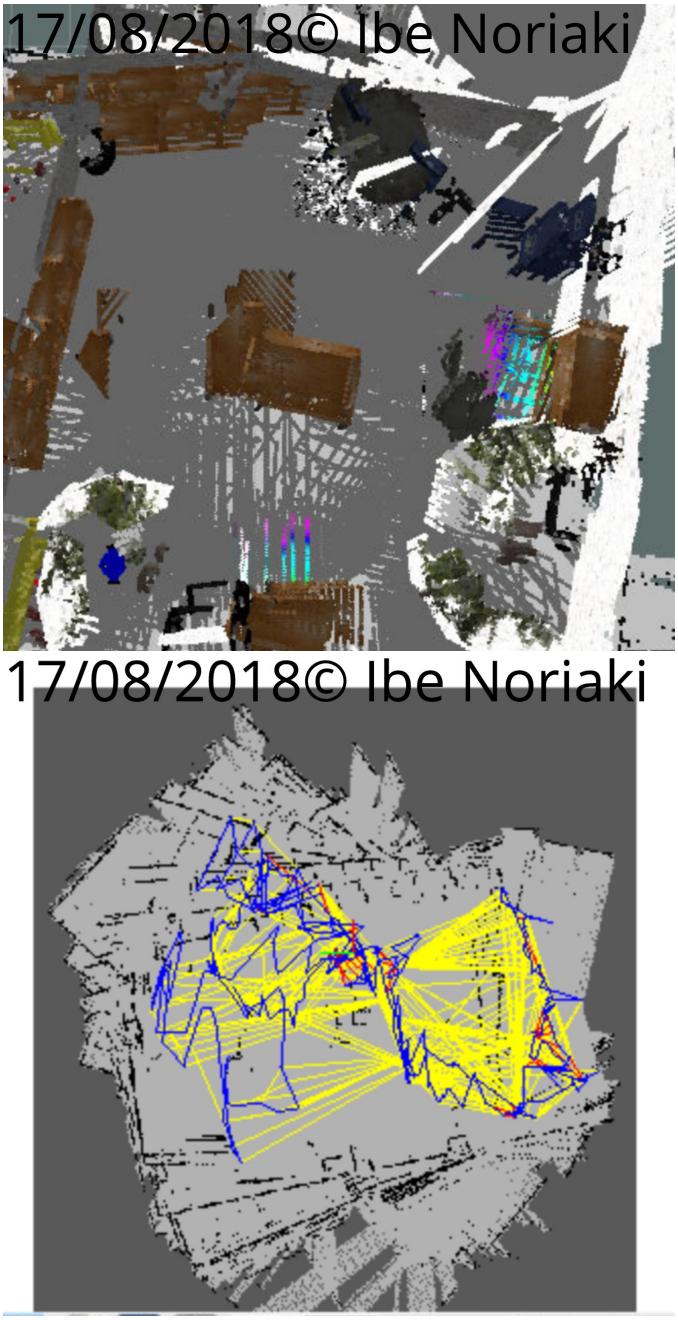

Fig. 17. Kitchen 3D and graph map

On the kitchen and myworld_v1, the model performs well rather than on myworld_v2.

5 Discussion

There was difficulty spiraling map issue, it looks there was function failure of bag-of-words at Front-end to use sensor data to obtain odometry and loop closure constrains and loop closure detection. When the project tried on myworld_v2, there are a lot of overlapped rather than others.

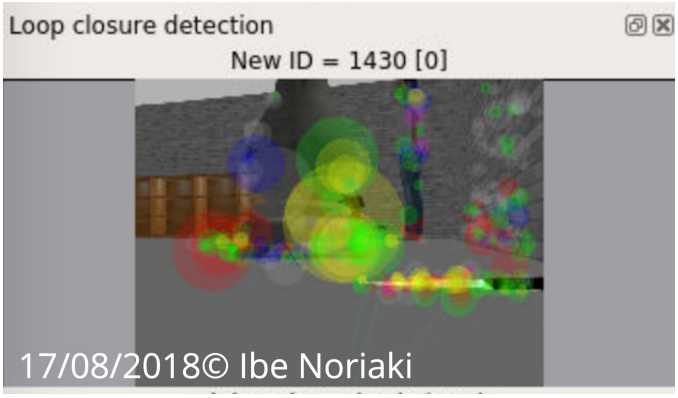

Fig. 18. Overlapping on loop closure constrains

So it's good example to show some limitation of RGB camera especially the less contrast environment.

6 Conclusion / Future work

The project could perform at least as the requirements stated on Background / Formulation. And it shall perform well on the environment provided. Of course, there are a lot of part which should be improved. But As stated on Discussion, there shall be point to improve on input into bag-of-words as front-end. RGB could be converted to several futures to fit to the environment as some student learned already on Self-driving-car Nanodegree. It shall be great exercise to combine the knowledge we learned.

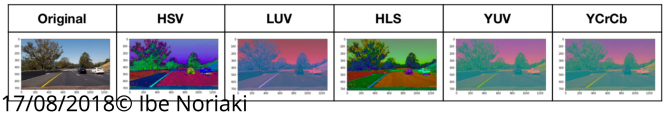

Fig. 19. Kitchen 3D and graph map