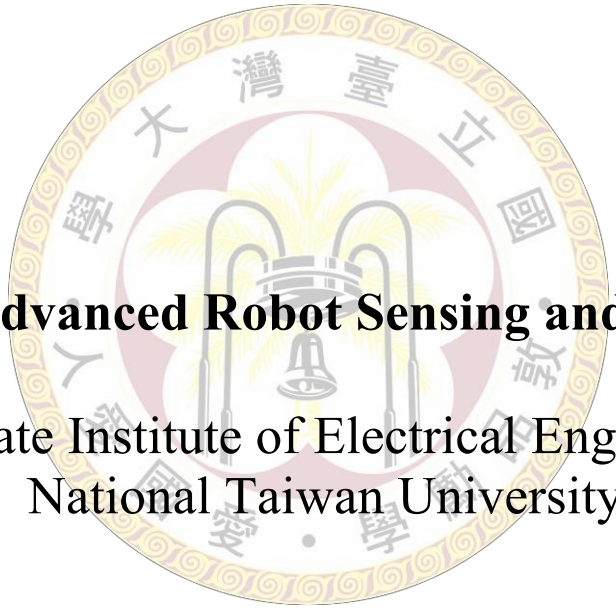
Final Project Report

Informative Map-based Mobile Robot Autonomous Navigation with Nature Language Command



Advanced Robot Sensing and Control

Graduate Institute of Electrical Engineering
National Taiwan University

Advisor

Professor Ren C. Luo

Student

Shang-Lun Lee Chin-Hao Hsu Rollin Gimenez

2019.06.28

Table of Contents

I. Introduction 2

II. Related work 3

III. Problem statement 5

IV. Hardware 6

1. Motor 6

2. Driver 6

3. Laser Range Finder (LRF) 6

4. Operator 7

5. Amazon Echo 7

V. Algorithm 8

VI. Experiment Result 12

References 16

I. Introduction

Recently, the ability to build both the geometric and semantic meaning map is a crucial topic. In tradition, we only focus on the geometric maps that provide us the basic information for the robot to navigate. But for the intelligent robots, they must have the semantic information to interact with environment more smoothly and flexibly.

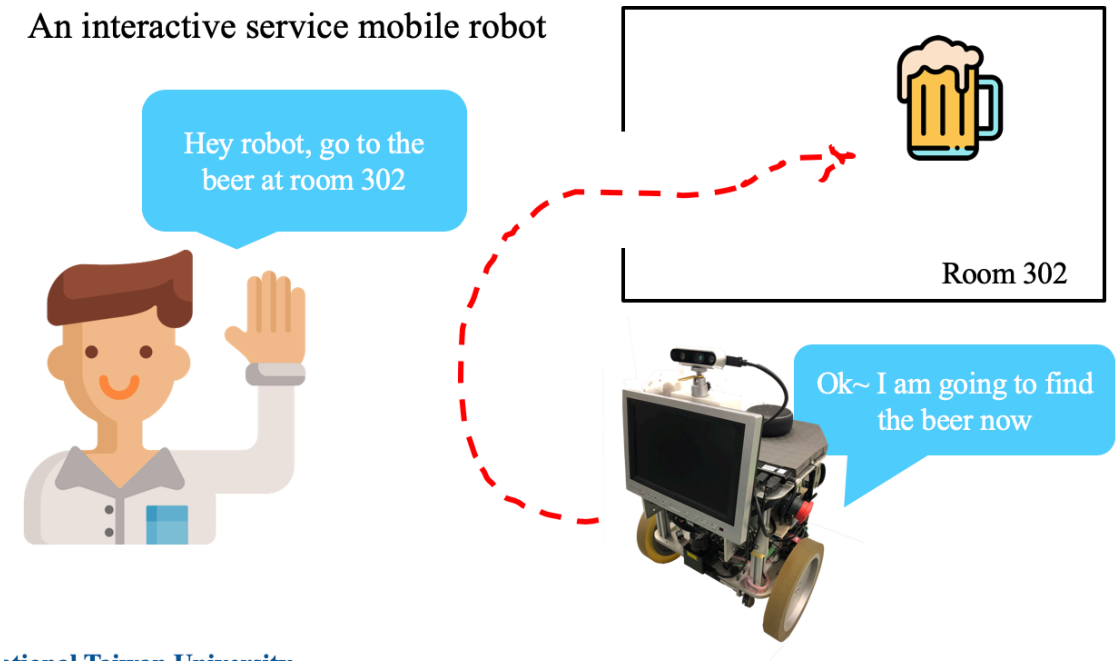
In this paper, we propose a method to build the map with object information in real time. Different from some works that store the whole size of objects in the map, we pay more attention to where the objects are by adding the point which is center of object on the map.

With the development of SLAM technology and object detection, we can use only the visual sensor to complete the production of geometric maps and object detection. Deep-learning-based object detection can even realize multi-object detection and have high accuracy.

On the visual SLAM, the concept of keyframes are often used to help camera for localization and tracking. Then, object recognition can assign class label to the keyframes. Finally, The semantic map will have both the pose of the keyframes in SLAM and each keyframe has the class label.

Compared to previous works which only using vision sensor, we choose to combine laser sensor with camera. We adopt the laser-based SLAM to build maps and use camera to detect objects in the environment. After mapping, object which we are interested in will be added in the map.

After completing mapping, our object's data has two attributes, one is the position of the it on the map coordinate, and the other is the “best viewpoint” of it. By recording the observation point of the object, we can return to the viewpoint of the object while navigating instead of colliding with it.



II. Related work

There are three ways, mapping, localization and path planning, let the robot accomplishes the goal which is given by the user. In the figure2, it shows the overview of their related and we implement our solution at ✯.

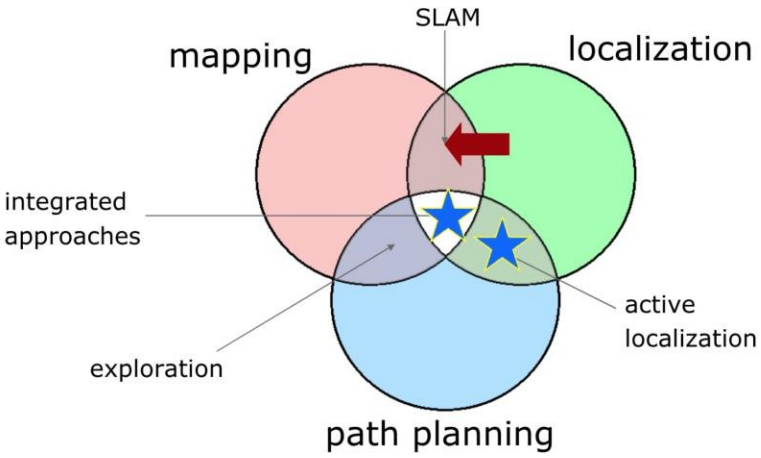


Figure 2. the overview of our method.

In robotic mapping and navigation, simultaneous localization and mapping (SLAM) 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. While this initially appears to be a chicken-and-egg problem there are several algorithms known for solving it, at least approximately, in tractable time for certain environments. Gmapping is one of the solution. It is a package that contains a ROS wrapper for OpenSlam's Gmapping. The gmapping package provides laser-based SLAM, as a ROS node called slam_gmapping. Using slam_gmapping, you can create a 2-D occupancy grid map (like a building floorplan) from laser and pose data collected by a mobile robot.

We use Adaptive Monte Carlo Localization(AMCL) for the localization method. It is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses

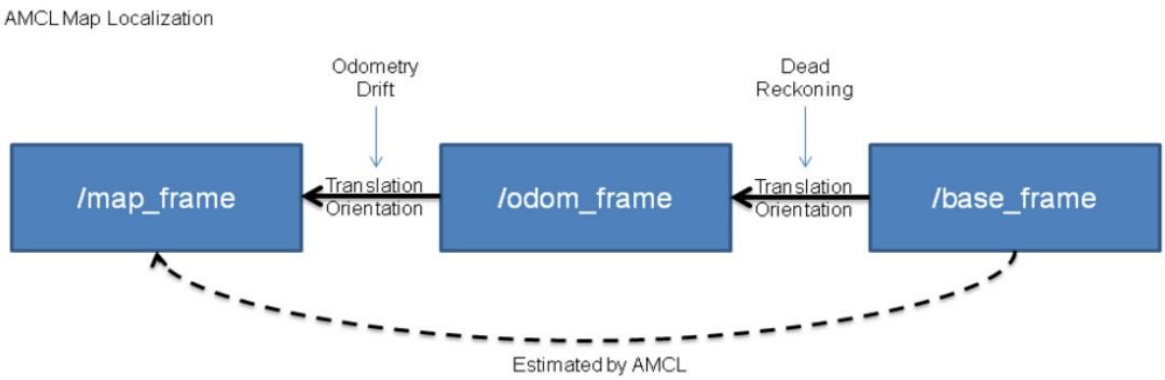


Figure 3. the framework of the AMCL.

a particle filter to track the pose of a robot against a known map.

The Navigation Stack is fairly simple on a conceptual level. It takes in information from odometry and sensor streams and outputs velocity commands to send to a mobile base.

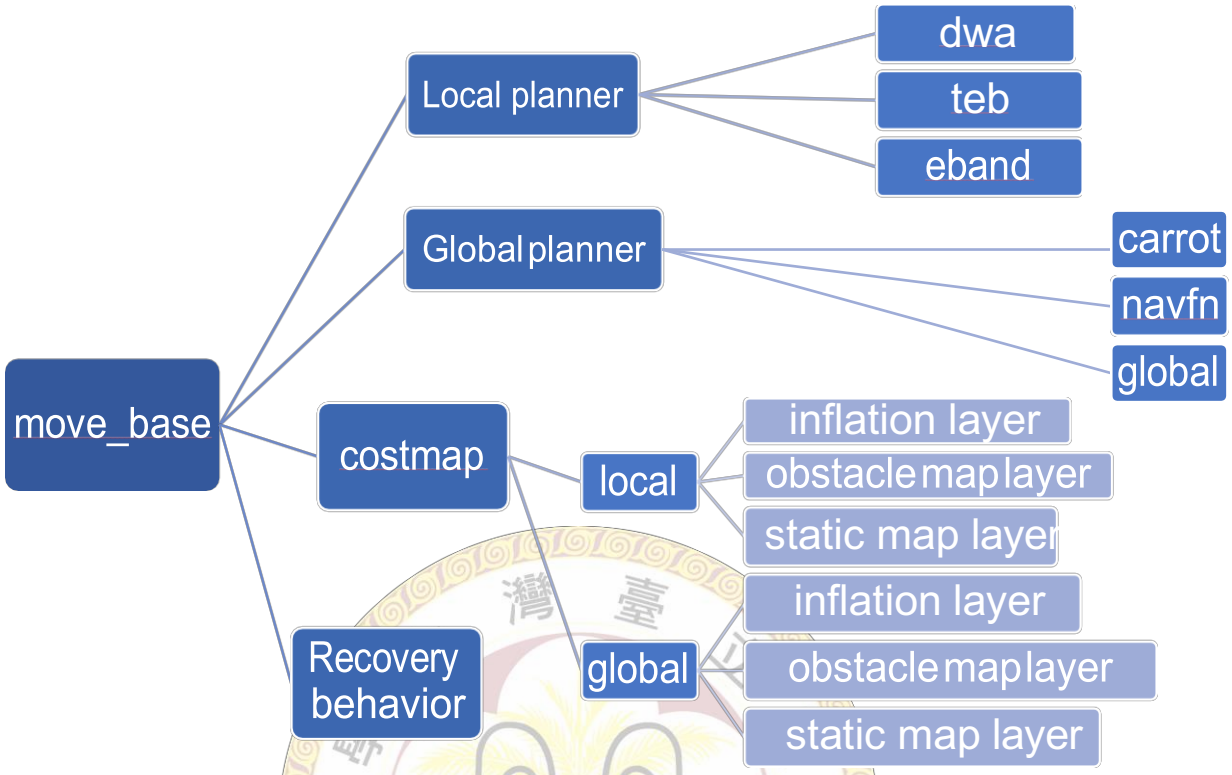


Figure 4. ROS Navigation Stack.

Use of the Navigation Stack on an arbitrary robot, however, is a bit more complicated. As a pre-requisite for navigation stack use, the robot must be running ROS, have a tf transform tree in place, and publish sensor data using the correct ROS Message types. Also, the Navigation Stack needs to be configured for the shape and dynamics of a robot to perform at a high level.

The move_base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task. The move_base node also maintains two costmaps, one for the global planner, and one for a local planner that are used to accomplish navigation tasks.

The dwa_local_planner package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. This value function encodes the costs of traversing through the grid cells. The controller's job is to use this value function to determine dx,dy,dtheta velocities to send to the robot.

The basic idea of the Dynamic Window Approach (DWA) algorithm is as follows:

1. Discretely sample in the robot's control space (dx,dy,dtheta)

2. For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
3. Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
4. Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
5. Rinse and repeat.

The teb_local_planner package implements an online optimal local trajectory planner for navigation and control of mobile robots as a plugin for the ROS navigation package. The initial trajectory generated by a global planner is optimized during runtime w.r.t. minimizing the trajectory execution time (time-optimal objective), separation from obstacles and compliance with kinodynamic constraints such as satisfying maximum velocities and accelerations.

The current implementation complies with the kinematics of non-holonomic robots (differential drive and car-like robots). Support of holonomic robots is included since Kinetic. The optimal trajectory is efficiently obtained by solving a sparse scalarized multi-objective optimization problem. The user can provide weights to the optimization problem in order to specify the behavior in case of conflicting objectives.

III. Problem statement

The hardware is the foundation of all, which contains necessary components needed in this work, such as operator, sensors and motors.

The system of the mobile robot is a self-built body of robot, which is made by steel with designed gears wheels, and holders to install motors, power supply, sensors and controller.

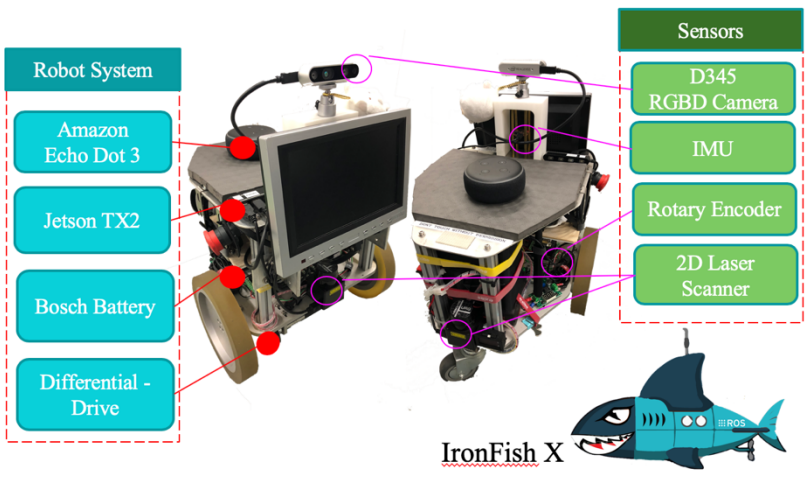
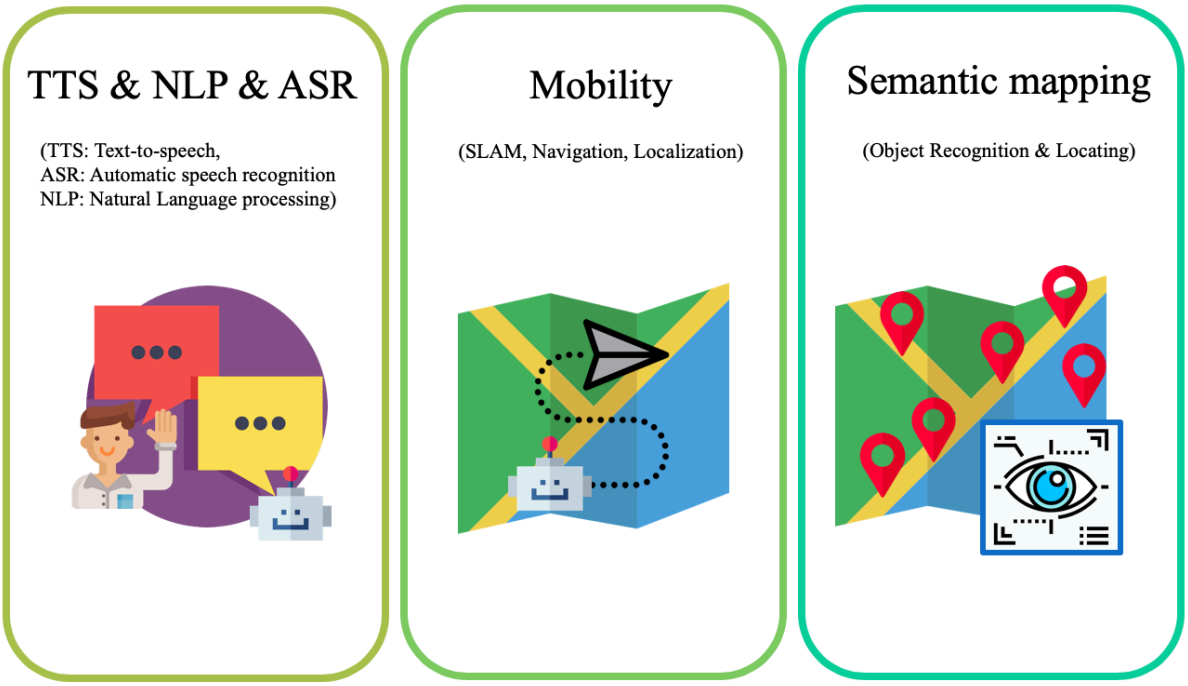


Figure 5. The system of the mobile robot.

We aim to solve 3 main problem:



We would introduce the detail of each part later.

IV. Hardware

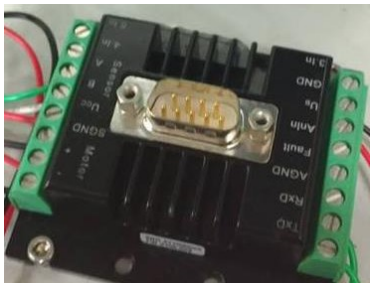
1.Motor

To drive a mobile robot, two *FAULHABER* mounted under the structure. The motors are selected because the high power generating ability and accurate positioning, which can make robot to move across rugged environment while recording odometry precisely.

Figure 6. Motor.

2.Driver

A servo motor needs the driver to control motion. As this reason, the *FAULHABER* chosen with its long operating duration and stability. With odometer installed, the diver can control the motor and record paths simultaneously, together with feedback to the controller, which in significant important to the robot sensing and control tasks makes it the best to use

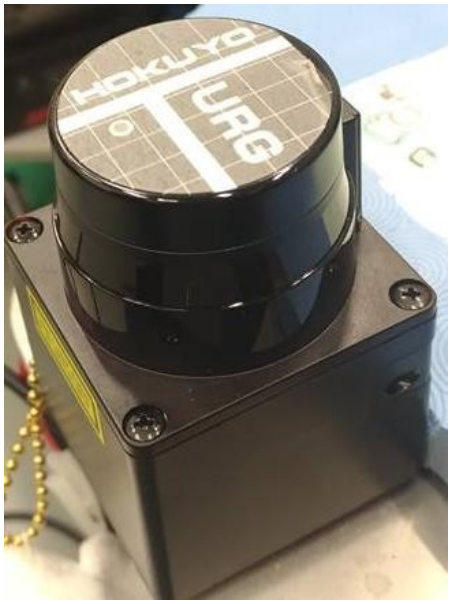


Figure

in this work.

3.Laser Range Finder (LRF)

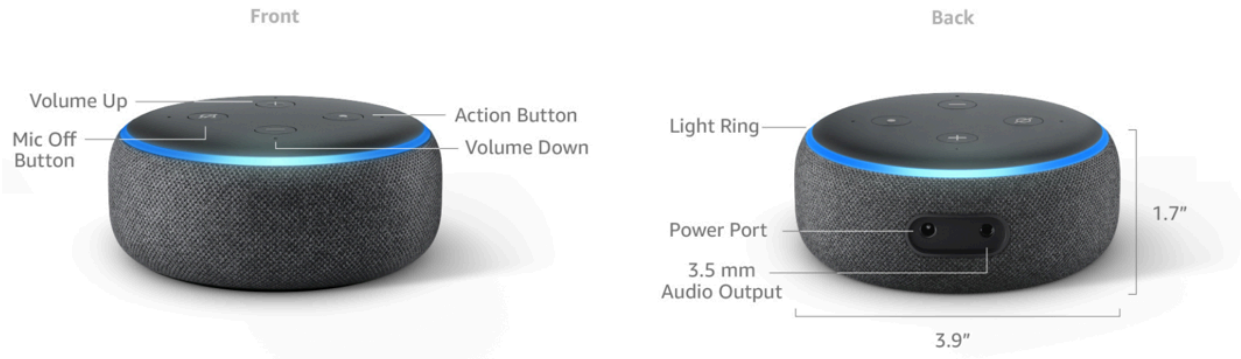
The well-known *Hokuyo* often used in researching and we install two *Hokuyo* on the front and rear side of robot. The LRF can scan an area across270 degreeswith0.25° angular resolution and a measuring depth up to30 meters. This sensor not only can be used in mapping work, it's also capable to help robot safety with collision avoida



4. Operator

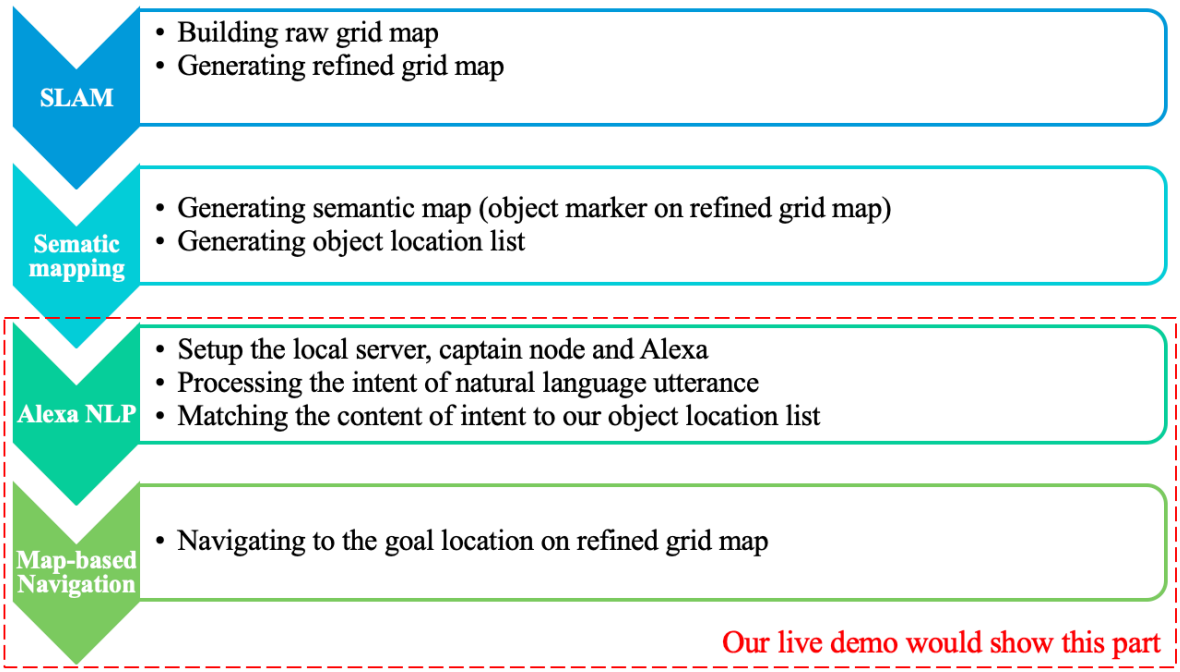
At last, we need the operator to receive, calculate, send data and command the robot and the response, reaction ability is also influenced by computation capability of the operator. So in this case, the robot is mounted with *Nvidia TX2* designed for heavy calculation, image processing and huge data storage.

5. Amazon Echo Dot 3

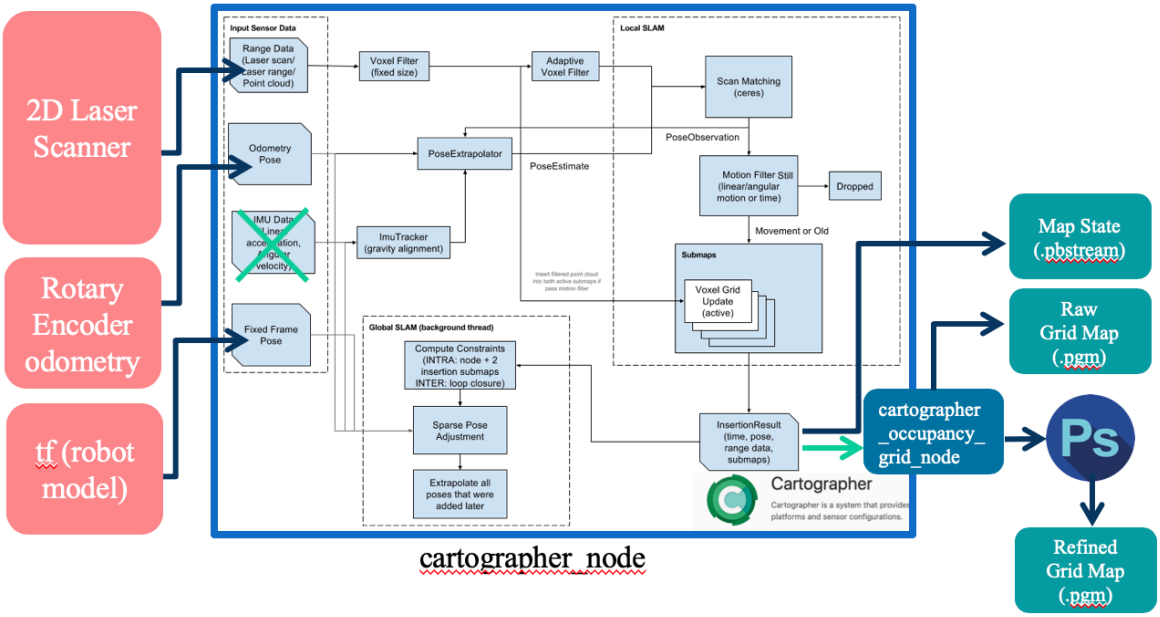


IV. Algorithm

1. System Overview



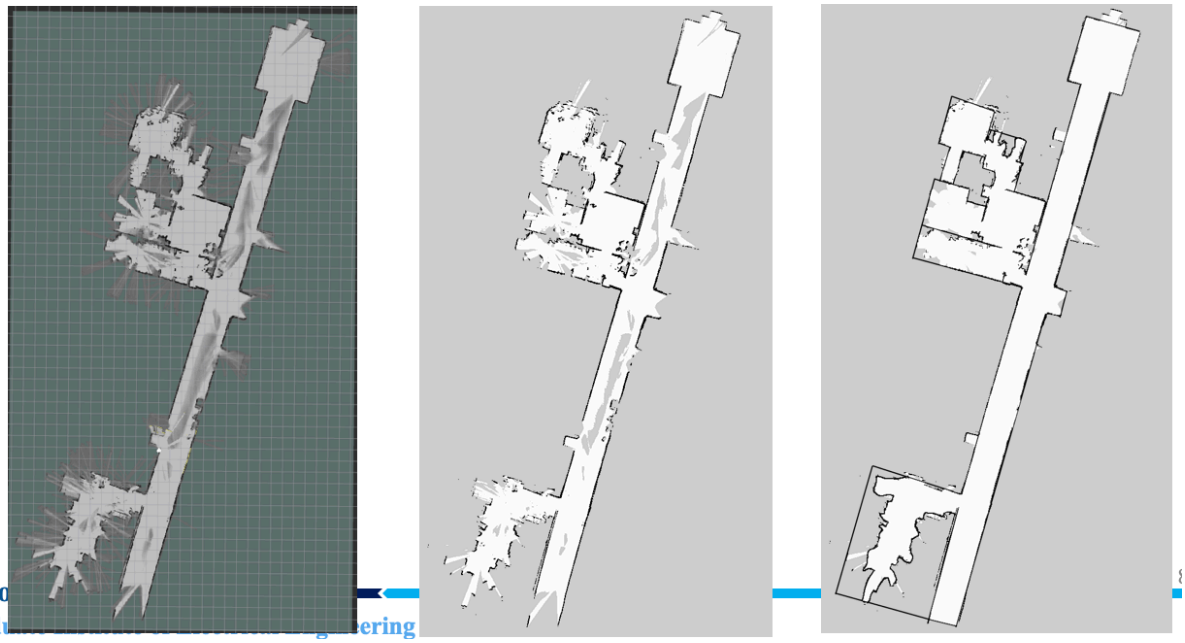
2. Cartographer SLAM



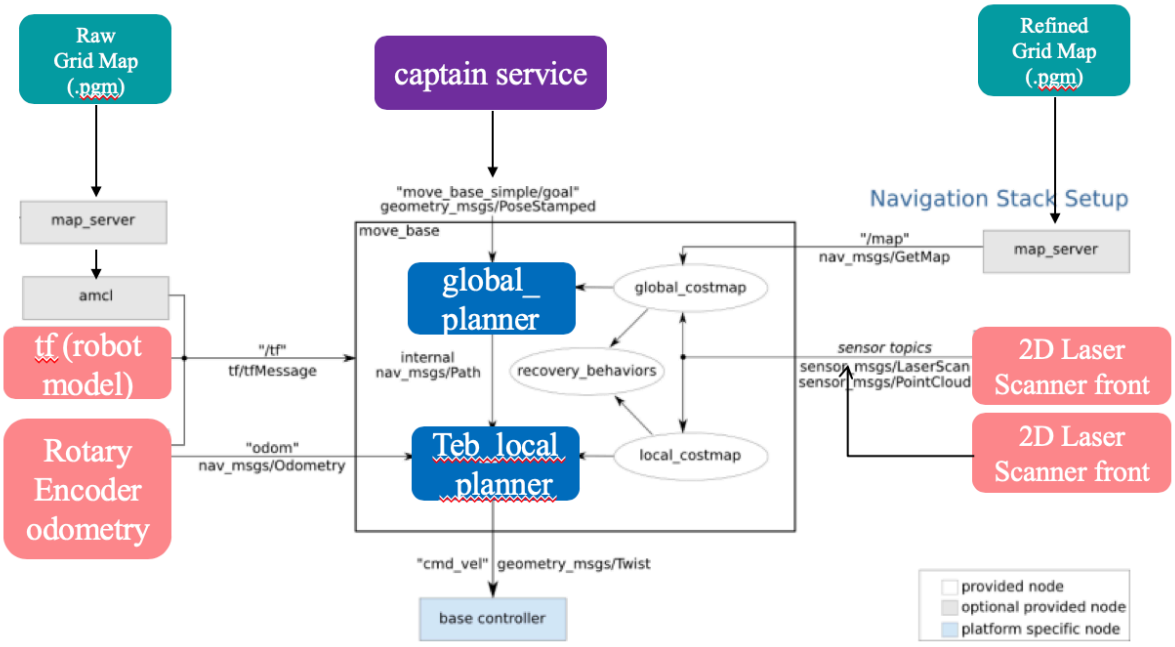
3. Map loading

We use three different map for this work

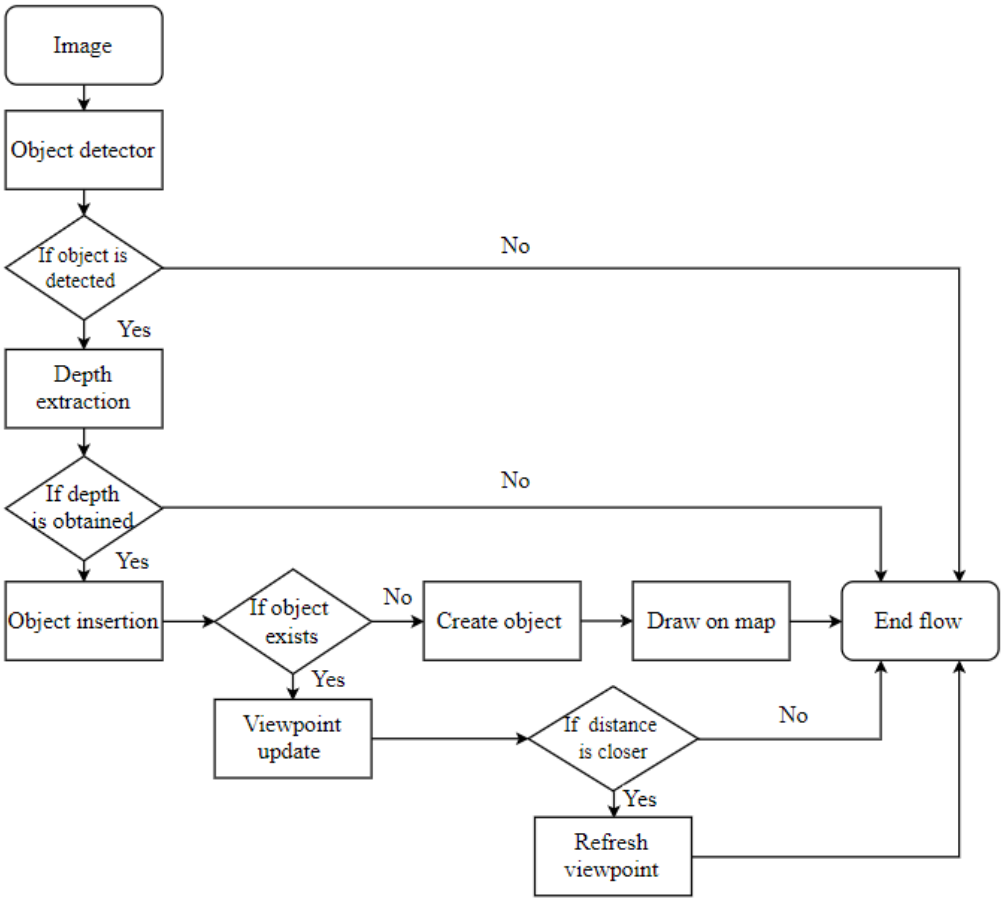
Map State (.pbstream)	Raw Grid Map (.pgm)	Refined Grid Map (.pgm)
Saved by cartographer rosservice Used for cartographer localization	Saved by map_server Used for amcl localization	Modified by Adobe Photoshop Used for navigation



4. Navigation Structure



5. Semantic Mapping Flow Chart



6. Semantic Mapping Detailed Work

A. Object detector

We choose the YOLO V3 as our object detection method. YOLO V3 provides a class label and a confidence score $0 \leq s \leq 1$for every proposal. It has demonstrated highly competitive results on the established computer vision benchmarks.

B. *Depth extraction*

After object detection, we will have the bounding boxes of objects which have x y coordinate. Then, we are able to get the center coordinate of boxes. Having the center coordinate, we can use it to obtain the depth value from point cloud.

C. *Object insertion*

After succeeding in getting depth value, the object pose is able to be calculated. The distance between this object and each object which is already on the map will be measured. If the distance is below threshold, this object will be considered have already existed. Then, move to the viewpoint update step.

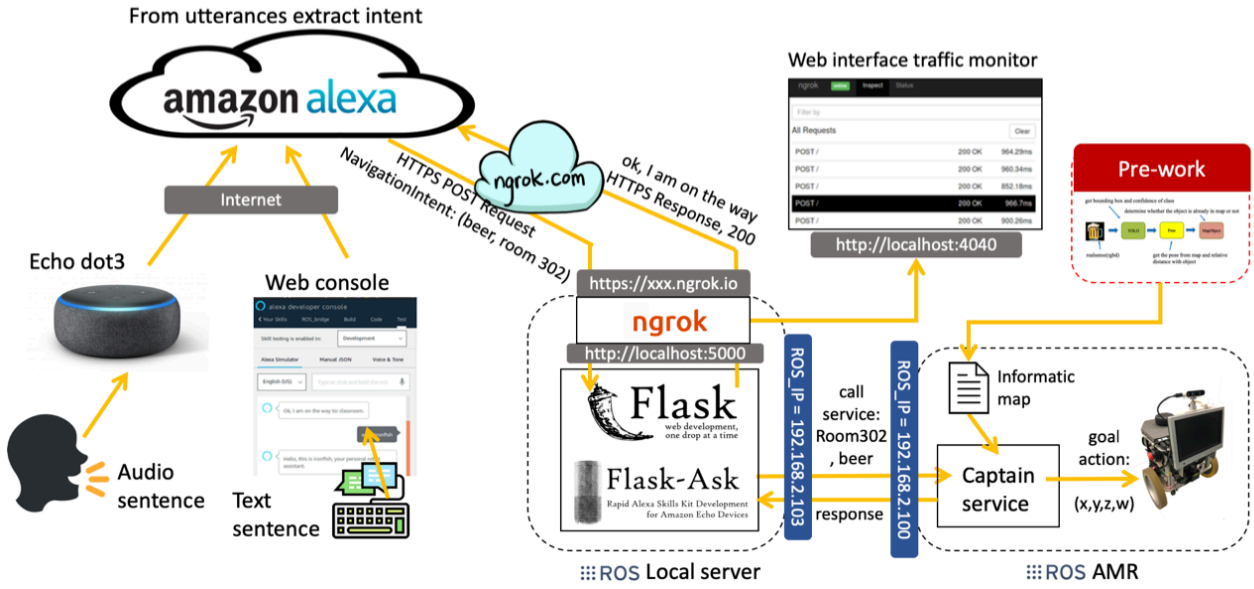
D. *Viewpoint update*

Viewpoint is set according to where the object can be seen properly. During mapping, the same object could be observed by camera several times. Each time the same object is observed, we will update the viewpoint of object by the score of below function

E. *Create object*

The object will be created on this step. The pose of the object, the pose of the robot (viewpoint), and the class label of object. For observability, we will put the marker of object on the map based on where the object pose is.

7. Alexa x ROS



V. Experiment Result

We want our robot successfully to go to anywhere assigned on the map without collision or getting lost. So that we design two experiments to test the navigation abilities of robot.

The first experiment is making robot driving around our lab. We would first start at Shannon's seat and robot is assigned to go to the front of 304 front door. Then it would be assigned to go to the end of corridor as below photos show.

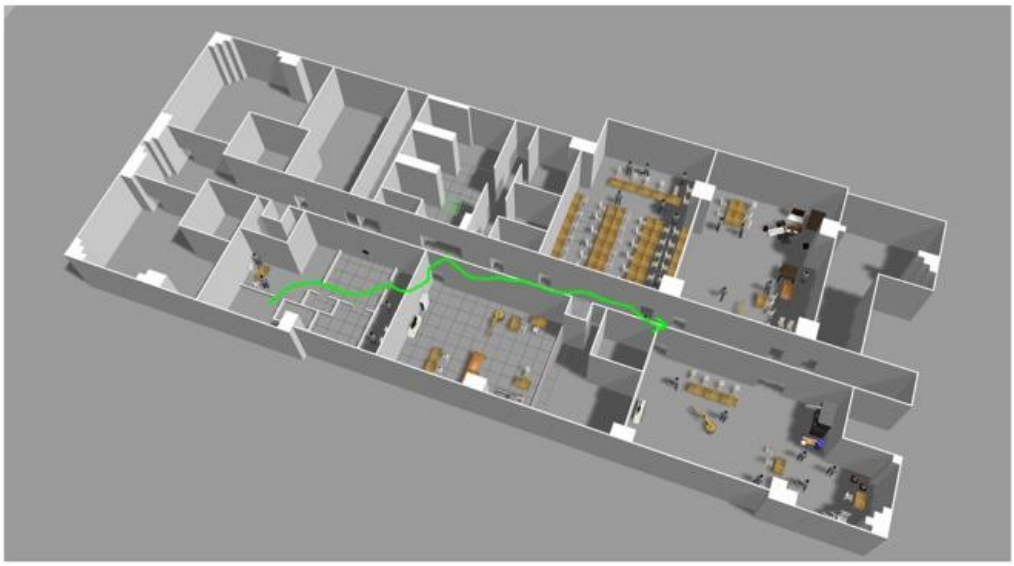


Figure 9. From Shannon's seat to the front of 304 front door.

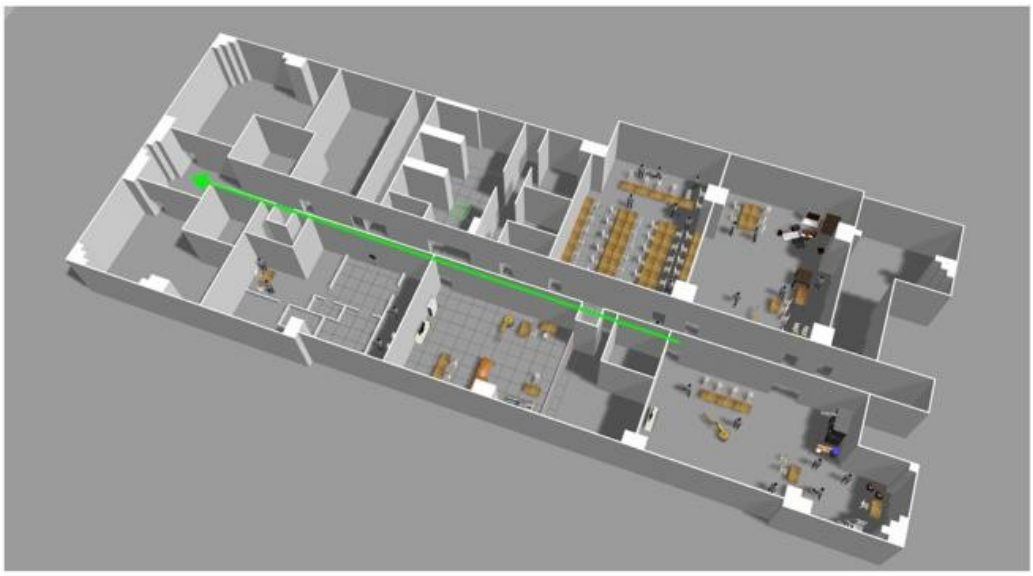


Figure 9. From the front of 304 front door to the end of corridor.

During the experiment, voluntary people would walk around the robot, blocking it unintentionally or intentionally. See how robot and its algorithm work. Finally, we collect the result of the experiments and make a table as below. The first solution we tried is what our lab originally used. The second solution is implementation of [1]. The third solution is the default setting of latest version 0.4 teb_local_planner with dual laser and backward allowed.

Solutions and corresponding navigation abilities	(Original) navfn & dwa default with single laser and forward only	global & dwa fine tune with single laser and forward only	global & teb default with dual laser and backward allowed	Our approach
In static environment without collision	O	O	O	O
In static environment without stuck	X	O	X	O
In dynamic environment without collision	X	O	O	O
In dynamic environment without stuck	X	X	X	O
Human first	X	X	X	X

Can go through narrow way	X	X	X	O

Table 1.

The experiment1's video is available at the attachment: experiment1_video.mp4.

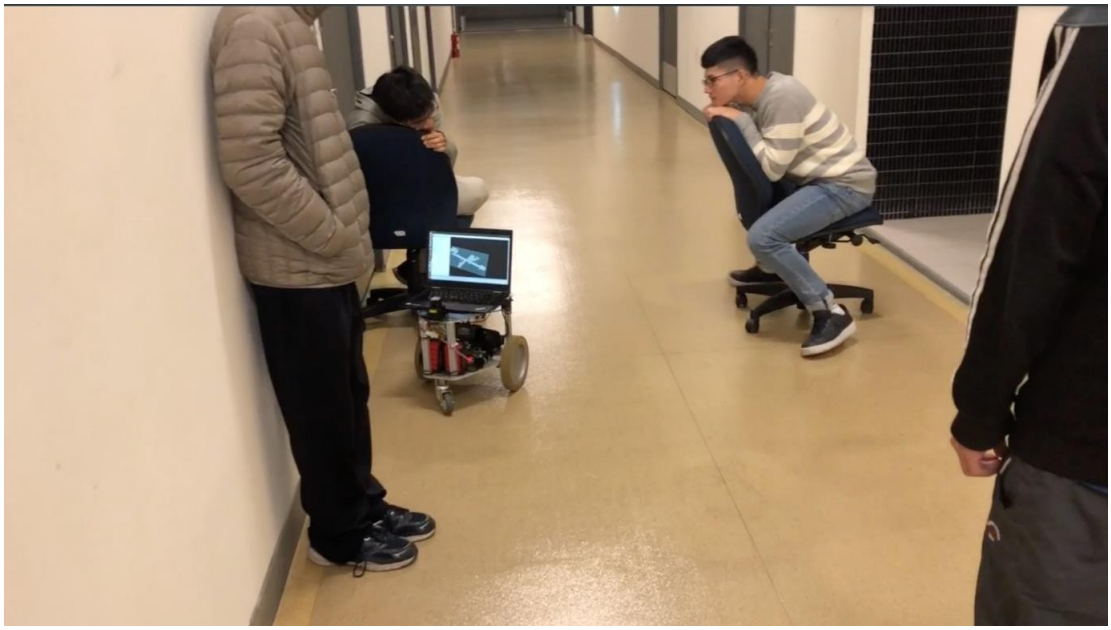


Figure 10. The picture of the experiment1.

However, in the experiment1, we only test the behavior of robot at normal and natural situation, there is no extreme condition applied. So we launch the second experiment which is assigning robot to go through a extremely narrow way. See if it could pass through smoothly without collision or not. The result of this experiment is showed as the table above.

The experiment2's video is available at the attachment: experiment2_video.mp4.

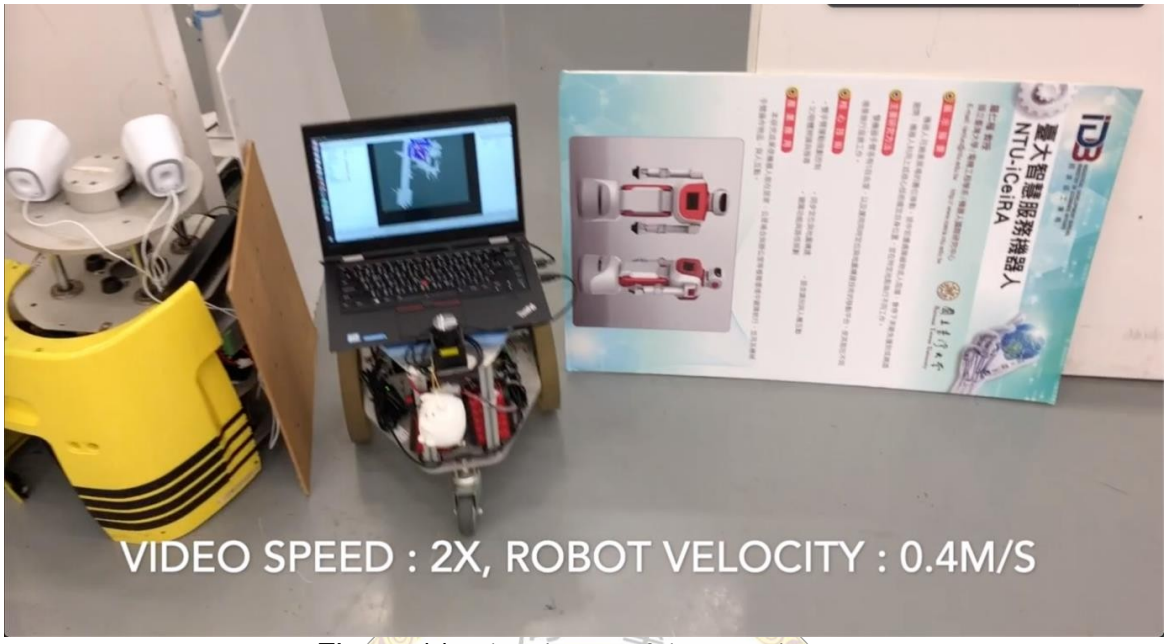


Figure 11. The picture of the experiment2.

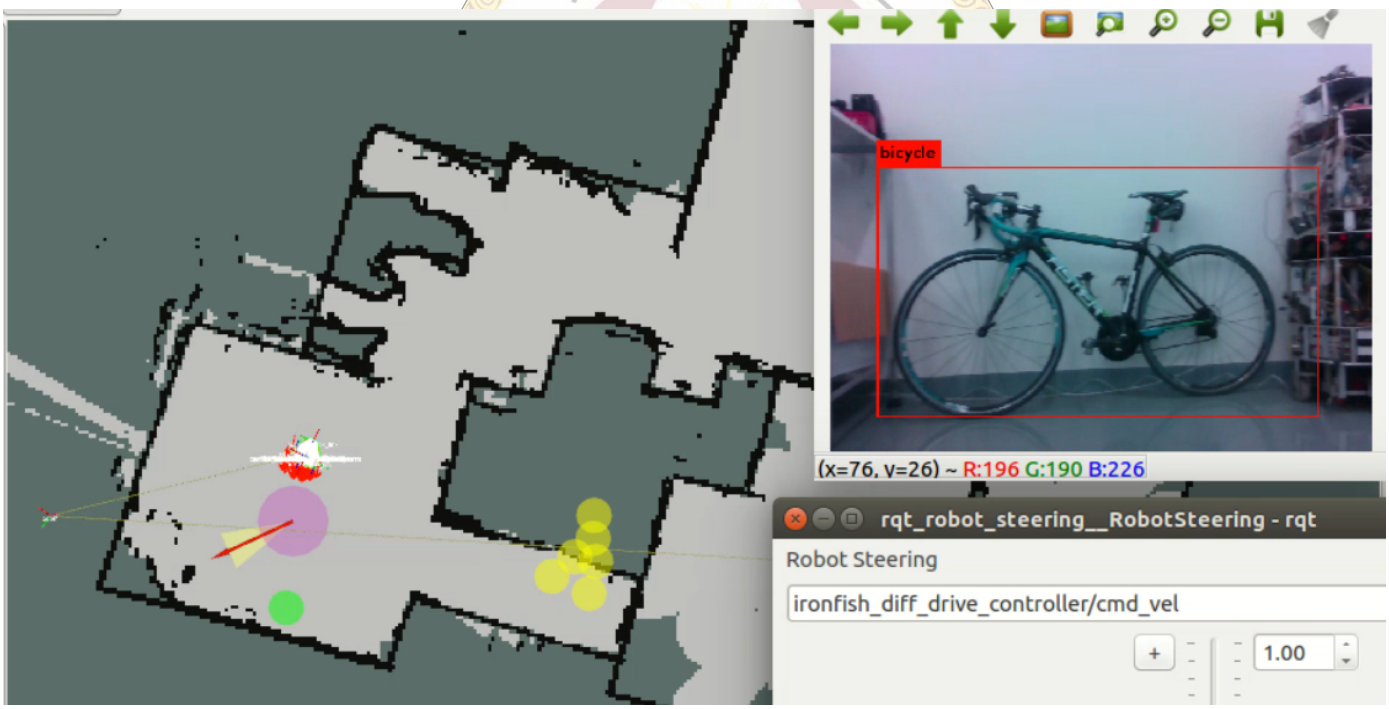


Figure 12. The picture of object "bicycle".

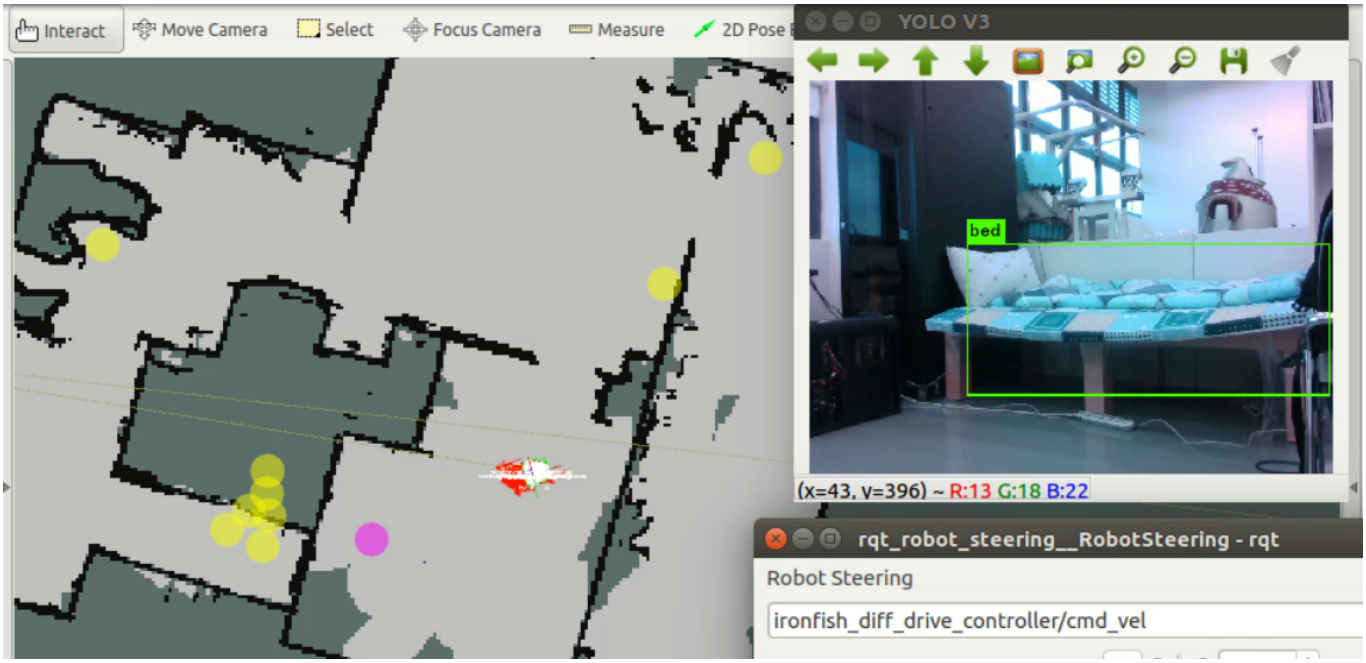


Figure 13. The picture of object “bed”..

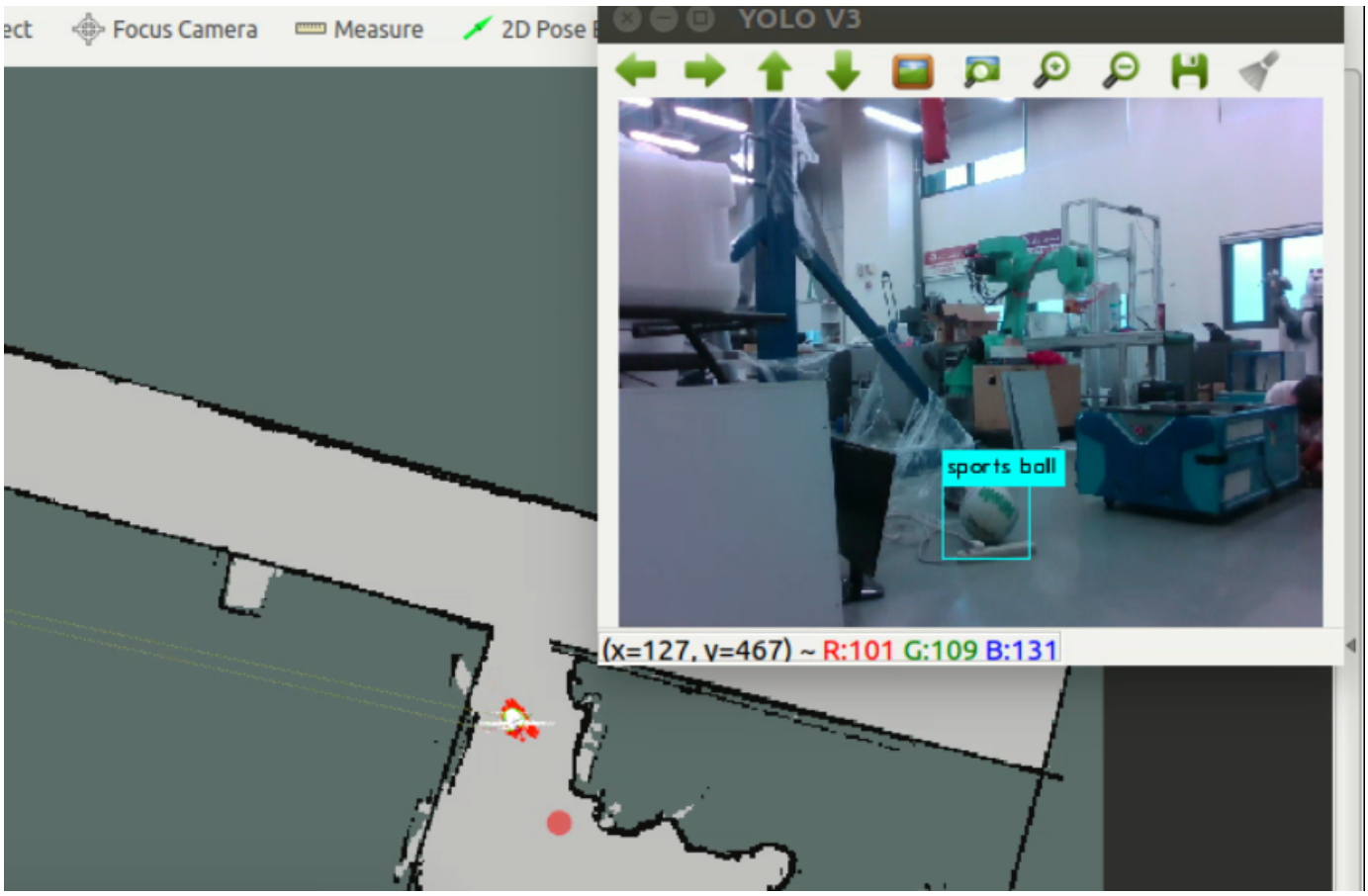


Figure 14. The picture of object “sports ball”..

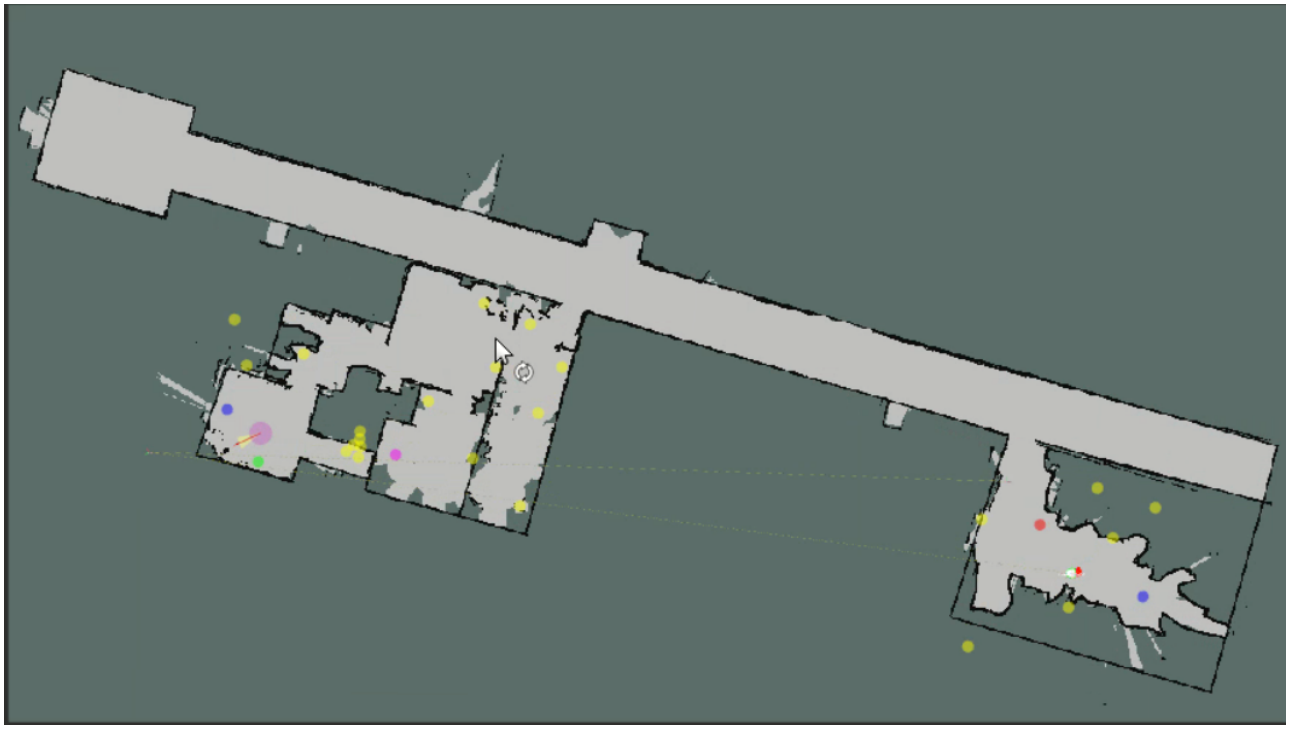


Figure 15. The picture of the whole map.

VI.References

[1] M. Labbé and F. Michaud, “RTAB-Map as an Open-Source Lidar and Visual SLAM Library for Large-Scale and Long-Term Online Operation,” in Journal of Field Robotics, accepted, 2018.

[2] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots, 34(3):189–206.

[3] Grisetti, G., Stachniss, C., and Burgard, W. (2007). Improved techniques for grid mapping with Rao- Blackwellized particle filters. IEEE Transactions on Robotics, 23(1):34–46.

[4] https://openslam-org.github.io/gmapping

[5] http://wiki.ros.org/gmapping

[6] http://wiki.ros.org/ros_control

[7] http://wiki.ros.org/urg_node

[8] Kohlbrecher, S., Rose, C., Koert, D., Manns, P., Kunz, F., Wartusch, B., Daun, K., Stumpf, A., and von Stryk, O. (2016). RoboCup Rescue 2016 team description paper Hector Darmstadt. Technical report, Technische Universitaet Darmstadt.

[9] K. Zheng, *Ros navigation tuning guide*, 201

[10]Niko S¨underhauf, Trung T. Pham, Yasir Latif, Michael Milford, “Meaningful Maps With Object-Oriented Semantic Mapping”

[11] M. Bjelonic "YOLO ROS: Real-Time Object Detection for ROS",