TurtleBot Navigation

11911035 Yuru Li
11810421 Xinyuan Wei

I. Introduction

In this final lab, we are supposed to finish it in the form of a competition. The task for us is to integrate all the functions on TurtleBot3 that have been developed through this whole semester, which are mainly focused on the lane following and the navigation control. The main objective is to earn a score by finishing the task provided as much as we can. This is an open selection problem. Our group chose to complete the lane following more times as much as possible within five minutes and was companied by P2 navigation and target search. The competition environment map is shown below:

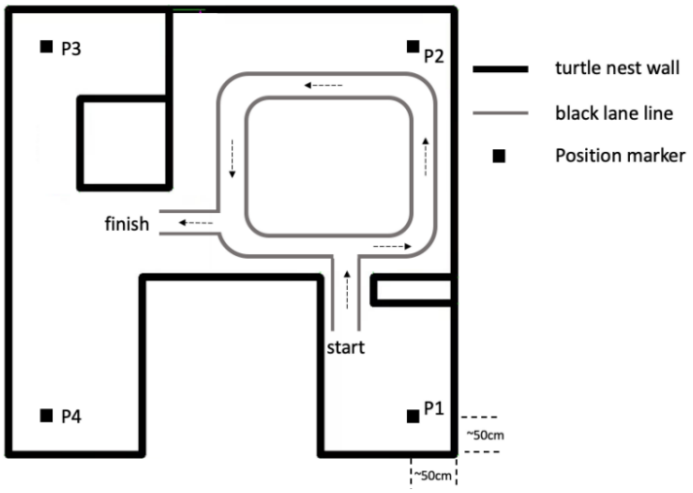


Fig 1. Robot environment map

Map and missions

The above figure shows the model of the environment of competition in reality. Four blocks represent the position that the TurtleBot should arrive in by auto navigation. The track is used for lane following. If we only do the lane following, we should control the TurtleBot enter and come out in the same gate, which is labeled start. And if we combine lane following with the navigation, we should control the TurtleBot to enter the start gate and come out at the finish gate. In addition, there will be put some Aruco Markers with different IDs on the corners at the top of the map for TurtleBot to detect.

II. Experimental Platform

Robot---TurtleBot3

TurtleBot is a low-cost, personal robot kit with open-source software. With TurtleBot, we will be able to build a robot that can drive around the place, see in 3D, and have enough horsepower to create exciting applications. The TurtleBot kit consists of a mobile base, 2D/3D distance sensor, laptop computer or SBC (single-board computer), and the TurtleBot mounting hardware kit. There are 3 versions of the TurtleBot model. Each version has its own advantages and shortages.

The robot provided for this lab is TurtleBot3, which is a ROS standard platform robot. TurtleBot3 is a small, affordable, programmable, ROS-based mobile robot for use in education, research, hobby, and product prototyping. The goal of it is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality, while at the same time offering expandability. TurtleBot3 is evolved with a cost-effective and small-sized SBC that is suitable for robust embedded systems, 360 degrees distance sensors, and 3D printing technology. The core technology of it is SLAM, Navigation and Manipulation.

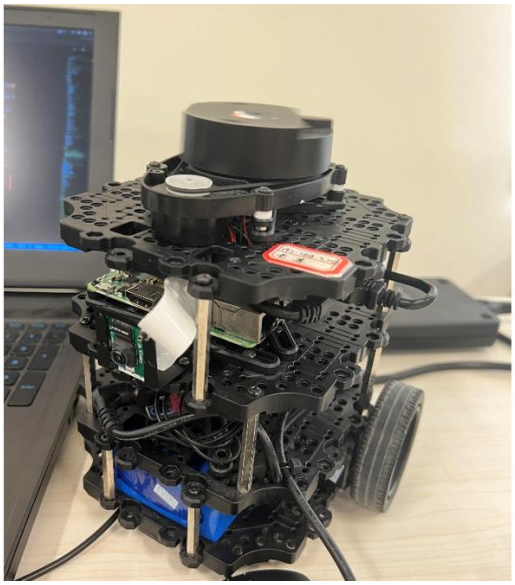


Fig 2. TurtleBot3

Sensors---Lidar & Camera

There are two sensors been used for this lab, which are Laser Rader and the camera. The principle of Lidar is to transmit the detection signal to the target, and then the received signal is reflected back from the target. And then, by comparing and processing the reflected signal with the transmitted signal, the information about the target can be obtained. The Radar here is used for navigation to know where the robot is and the way to go. In addition, the camera here can do real-time monitoring, which can reflect the running state of the TurtleBot3.

Computer

The computer used in this experiment to connect with the TurtleBot is Dell G7 whit 16GB RAM. CPU is Intel Core i7-8750H, GPU is GeForce GTX 1060 with Max-Q Design.

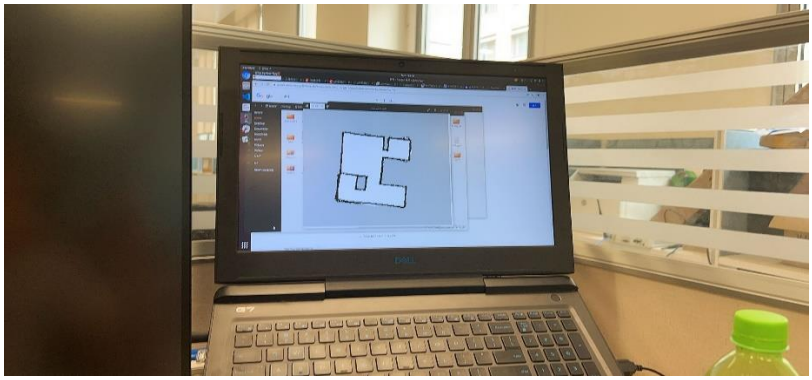


Fig 3. The computer used

ROS

ROS stands for Robot Operating System, and is a highly flexible software architecture for writing robot software programs. It contains an amount of tools, library code, and conventions designed to simplify the process of creating complex, robust robot behavior across robotics platforms. The primary goal of ROS is to provide code reuse support for robotics research and development. ROS is a framework for distributed processes (aka "nodes") that are packaged in packages and feature packs that can be easily shared and distributed. ROS also supports a federated system similar to a code repository, which also enables collaboration and publishing of projects. ROS can be summarized as a four-in-one pattern. The first is ROS communication mechanism, namely point-to-point communication. Second, ROS has many development tools, such as RVIZ, etc. Moreover, ROS also has many functions, which exist in the form of a function package in ROS, and its application functions include visual navigation and motion planning. Finally, ROS is not a single piece of software. It exists in an environment where people, companies, and the world are using it, and it's an active ecosystem. The four-in-one diagram is shown below:

Fig 4. Four-in-one diagram of ROS

III. Approaches

Mapping

Before doing the task itself, we need to do some pre-lab preparations, building a map is necessary and important. Here we use GMapping to realize the goal of creating a map. Based on Lidar, we manage to build a map using a keyboard manipulator by using RVIZ. The robot will be controlled to go over the whole field of the competition environment and use SLAM to generate a map. What should be mentioned is that the accuracy of the map will cause an influence on the navigation. Any disturbance during drawing construction will cause the map to skew or deviate from the original. Hence, when we manipulate the TurtleBot, we need to move it as smoothly as possible to avoid the appearance of bad situations. What's more, the setting of the parameters is also an important component to affect the accuracy of the mapping. We need to estimate and set the value of map size and the map resolution precisely. The establishment of the map needs to be based on the determination of parameters and the environment of the map.

The resulting map we saved is:

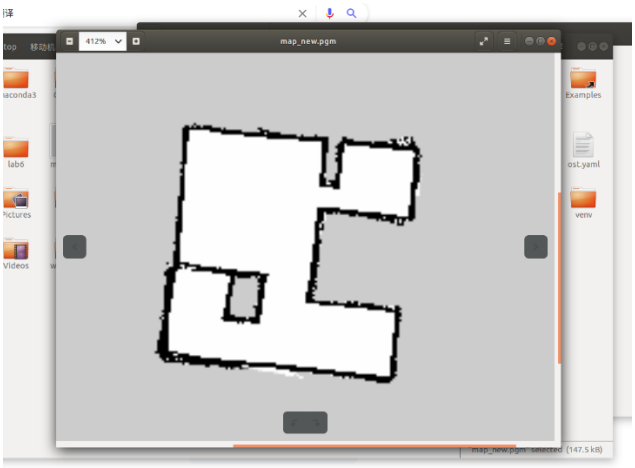


Fig 5. The map we saved

Localization

Since our idea is to start from P1 and do lane following and companied by navigating to P2 and detect the Aruco Marker. We need to locate the position of these two points precisely. The positions can be echoed by `rostopic echo amcl_pose`. The final goal position is

```
[(0.139999901056, 0.12999984622, 0), (0, 0, -0.698, 0.716)],
[(-0.2049999547, -3.91999936104, 0.00), (0.000, 0.000, 0.000, 1.000)],  # point 3
[(3.87499880791, -4.42999982834, 0.00), (0.000, 0.000, 0.99897277669, 0.0453143623188)],  # point 4
[(4.19958301544, -0.264779472351, 0.00), (0.000, 0.000, -0.708046509431, 0.706165802402)],
# point 1 #map_new
[(2.20499968529, -0.860000371933, 0.0), (0.0, 0.0, 0.67721942459, 0.735781116201)],  # begin
[(3.4049996376, -1.020000326633, 0.0), (0.0, 0.0, 0.999646184863, 0.0437337857891)],
# lane_follow far2 fine tune

[(0.224999770522, -0.320000221729, 0.0), (0.0, 0.0, -0.710779897245, 0.703414484975)]  # point2 start
```

From top to bottom are P2, P3, P4, P1, the starting point, the start point of lane following, and the point return to lane following from P2.

Navigation

Here I will introduce the whole procedure for TurtleBot navigating to finish the whole race.

1. Start

In the beginning, start TurtleBot on point 1, then it will automatically navigate to the start point of the lane following. By using the map we built, we continually adjust the position on the map, to make the TurtleBot as close to the actual starting point as possible.

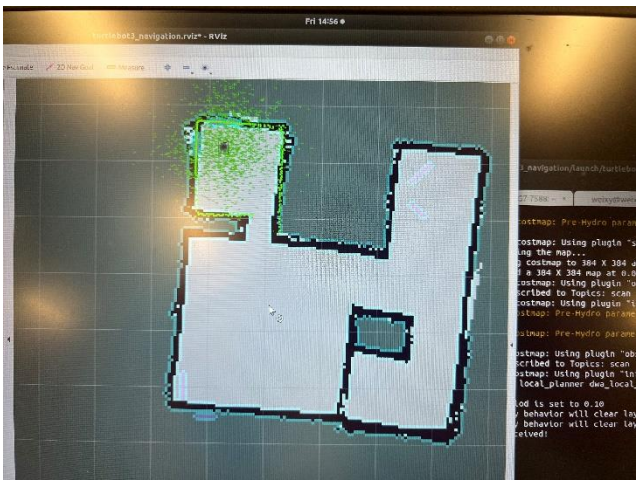
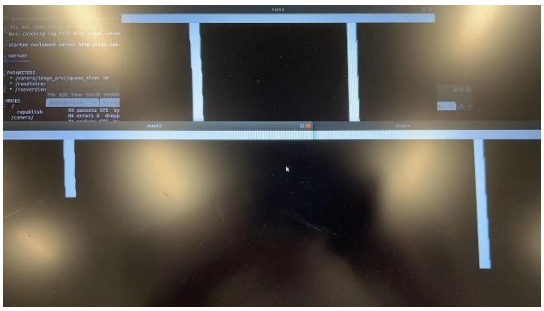


Fig 6. Adjust the position of TurtleBot on the map we built

2. lane following

After TurtleBot reaches the start point, the lane following program begins. The idea we used for lane following is simple. We only use the angle of the lane to control the TurtleBot, its turn logic is very simple, that is, when the upper left part of the video loses the lane, it will think it should make a left turn now, and when the upper right part of the video loses the lane, it will think it should make a right turn now. The schematic diagrams are shown below:



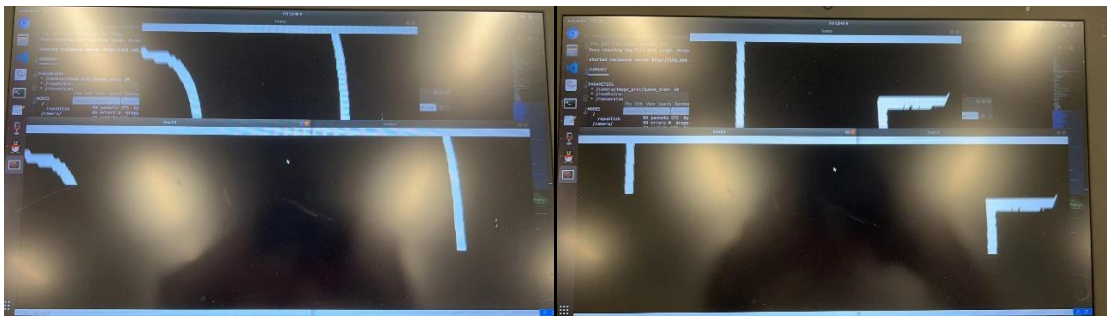


Fig 7&8&9. Normal situation, left lose, right lose (from top to bottom, left to right)

3. Navigate to P2

When TurtleBot reaches the second left turn, it will automatically navigate to the point 2 and then it will back to the lane, heading to the lane direction.

4. Return to original point

After TurtleBot reaches the ending of the lane, it will automatically navigate to point 1 and

repeat the operations again until the five minutes time out.

Command used for starting up the program

 # On TurtleBot

a. Enable video capture

```
roslaunch turtlebot3_autorace_traffic_light_camera turtlebot3_autorace_camera_pi.launch
```

b. Open TurtleBot remote control

```
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

 # On Remote PC

a. Set program environment variables

```
export AUTO_IN_CALIB=action
export GAZEBO_MODE=false
```

b. Receive the video returned by turtlebot on the PC side

```
roslaunch turtlebot3_autorace_traffic_light_camera
turtlebot3_autorace_intrinsic_camera_calibration.launch
```

c. Set camera iso to 120

```
rosrun rqt_reconfigure rqt_reconfigure
```

d. Set program environment variable

```
export TURTLEBOT3_MODEL=burger
```

e. Start up RVIZ SLAM windows

```
roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map_new.yaml
```

f. Start main program

```
rosrun lane_turtlebot3v2 lane_following.py
```

The rqt graph for whole procedure

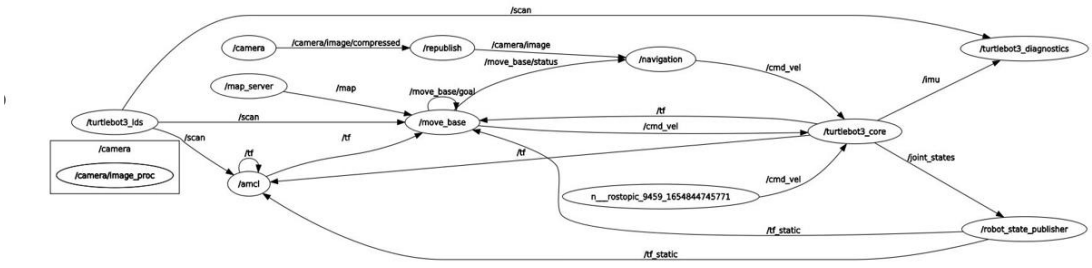


Fig 10. Rqt graph for whole procedure

IV. Results

The whole competition finished in the same environment and each group has two chances. The following gives the results of each round of our group.

Round 1

In round 1 we finish two point five round of lane following and navigate to P1 and P2 for four times in total, and we detect Aruco Marker with correct ID for two times. But our TurtleBot

overpowered the inside lane on the turn. And due to our setting of program, we need to correct the start point of navigation, in this case, the score will be cut down. The mistake of the first round lies in the wrong judgment when starting out that it turning right in advance, which leads to a large degree of deviation in the running process of the car, so it needs to be adjusted manually. However, moving the TurtleBot for a long distance will lead to the complete deviation of the navigation map, so it cannot be retrieved eventually.

Round 2

In round 2, the TurtleBot worked well at first, the lane following and navigate to specific position was done well. However, although the map was not biased, other problems appeared. After it finished two rounds of lane following and went toward the outgate, the navigation had a problem that the TurtleBot was disconnected with the remote PC due to a connection problem, resulting in the car cannot navigate and wandered in the exit.

Resulting figure

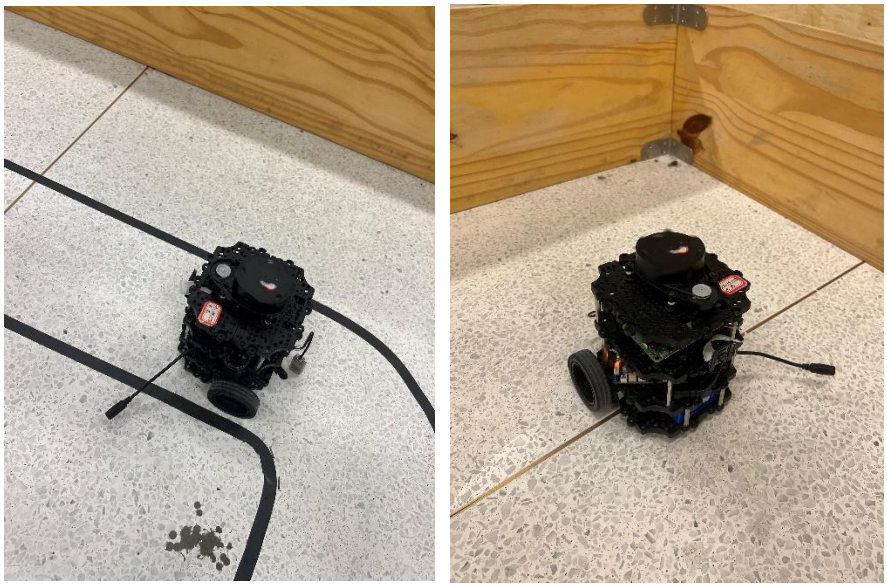


Fig . lane following and navigate to the specific point

The left figure shows the process of lane following while the right one shows the TurtleBot navigate to the required point correctly and stop for a while.

V. Conclusion

In this competition, we realize building map by Gmapping and the combination of lane following and navigation. The TurtleBot can run successfully and finish the task we required, which are localization, lane following and the Aruco Marker detection. However, we still did something not well enough and there still exist some concepts for us to optimize. For example, in the part of the lane following, we adopt the single lane search and take the right lane as a reference. Although this method is feasible, its accuracy and robustness are lower than that of the double lane. In addition, when the TurtleBot ready to enter the lane following track, it is set to run with a fixed speed for a period of time, and then turn right to start the lane following. This method has a high demand for the precise location of the robot to enter the start gate. That is, a small deviation will cause problem

to the whole process of lane following. And this is why our robot fails in round one. We will continue to optimize our program and increase its robustness if we have the opportunity in the future.

VI. Contribution

Xinyuan Wei (50%): main code programming for visual and navigation, robot testing and adjustment
Yuru Li (50%): report writing, code testing and modification, build the map and robot adjustment

VII. Acknowledgement

The code for this whole project is in the attachment.

Thanks for the instruction in detail from Professor Zhan this semester and the help from all TAs and other classmates.