# A 2D-Mapping Car with Face obstacle Recognition and Marker Functions

**Author:** Liwei Che **Instructor:** Xiang Chen, Jie Tang

## Abstract

A 2D-mapping car is based on RPLIDAR and NVIDIA Jetson Board nano using ROS framework and python script to drive. The architecture of the vehicle is divided into three layers according to the functions they are responsible for. The top one is the LIDAR module to detect the surrounding environment feeding back with distance and angle information. The middle layer is the control core of the car, whose functions includes communication with computer and other two layers, process the image data and LIDAR data primarily, give the control signal to the bottom layer. The bottom layer is the control board of the wheels. The vehicle aims at treasure-hunting and environment exploration.

**Keyword:** ROS、Python、OpenCV、Mapping、SLAM

## 1. Introduction

Same situation happens when people suddenly get into a strange and complex building or other environment that is hard to find a way out of it or find a specific object which located at the nearby. It is also impossible for human to remember the complex geographic environment information and draw a map in mind immediately. Considering those facts, it is obvious to solve these problems through techniques. Camera and lidar are both important sensors for robots to perceive their surroundings, which are similar as the eye for human.

Compared with the human beings, robots in recording environmental information are significantly more advantageous. Equipped with lidar and camera, a robot can be a perfect executor of environment detection mission and object hunting mission.

Based on above consideration, this project focus on the construction of a mobile vehicle which shoulder the responsibility of detection and hunting.

## 2. Approach

### 2.1 Overview

The overall framework of the vehicle is shown in figure2. The main architecture is illustrated in following part and figure1 A serial executing results of the car will be shown in the result part. In the following sections, the algorithms will be explained.

### 2.2 Architecture:

The car is divided into three layers:

From top to bottom

1.Lidar to detect surrounding environment.

2.Control unit

3.Dynamical system

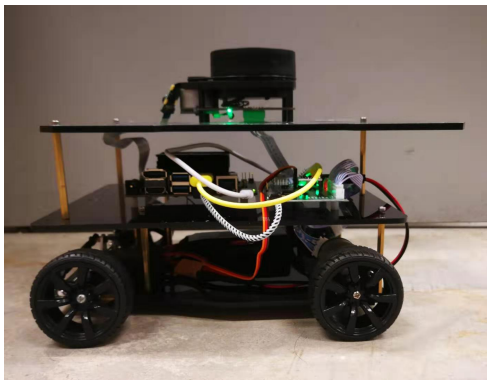

FIGURE1-VEHICLE STRUCTURE

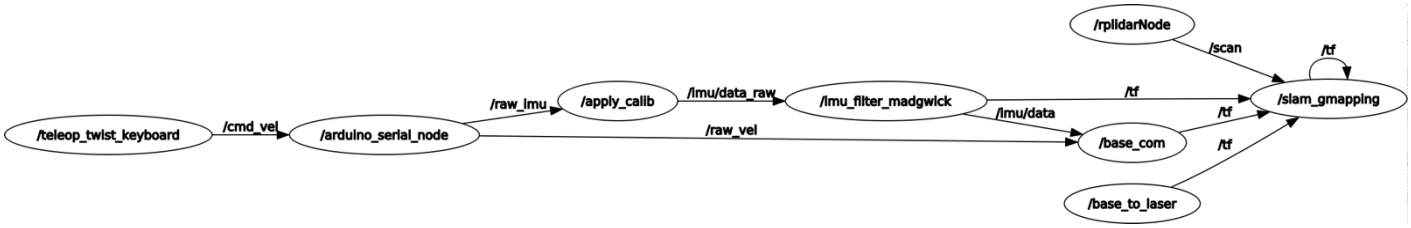


FIGURE2- ROS FRAMEWORK

## 2.3 ROS Framework

There are totally 8 nodes forming the ROS framework of the vehicle.

**Teleop-twist-keyboard:**

This is the generic keyboard control module for ROS. This node will publish a topic called "cmd_vel", which contains the information for vehicle motion.

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit
```

FIGURE3-INTERACTIVE PANEL OF TELETOP-TWIST-KEYBOARD

**Arduino-serial** and **apply-calib:**

Arduino-serial is the module contains the serial functions of Arduino, which in there can transfer the cmd_vel information into raw IMU(Inertial Measurement Unit) data. It subscribes topic "cmd_vel" and publishes "raw_imu" with "raw_vel"

Apply-calib will do further process on those raw IMU data. It subscribes "raw_imu", and pulishes "imu/data_raw".

**IMU-filter-madgwick:**

This module focus on filter which fuses angular velocities, accelerations, and (optionally) magnetic readings from a generic IMU device into an orientation.

Its code based on Sebastian Madgwick's work.

**Base_com:**

This module aims at managing the motion system of the vehicle. It provide "tf" (The relative position of the control unit in the whole system)information for slam_gmapping module and do the calibration and correction work to avoiding unnecessary error caused by excessively sensitivity to minute vibrations.

**rplidarNode** and **base_to_laser:**

These two nodes are the core of lidar module. The former one can provide scan information(distance and angle) and the latter one tells the relative postion of the lidar(in the central axis with height of 18cm).

**Slam_gmapping:**

This node can receive the "tf" topic from other topic to help adjust the spatial coordinate parameters of the map. It will also subscribe topic "scan" to use lidar information construct the surroundings map.

## 2.4 Face obstacle recognition

The object recognition function is achieved through a smart camera installed on the NVIDIA jetson board nano. It provides 720p HD image capture.

Using python OpenCV API, the camera can be driven and give confidence to recognize what it sees.

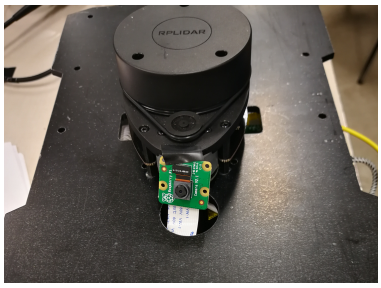


FIGURE4-CAMERA SENSOR AND LIDAR

The inside classifier is based on Cascade classifier provided by OpenCV. Compared with the early CvHaarClassifierCascade, new Cascade can use both Haar-like features and LBP features.

As a method of face detection, LBP appears side by side with Haar, and its single point detection method is the most complex one of the two. Therefore, when the detection points are the same, it takes the longest time to calculate the judgment step without considering the feature calculation time.

Here we choose Haar to reduce the recognition calculation time.

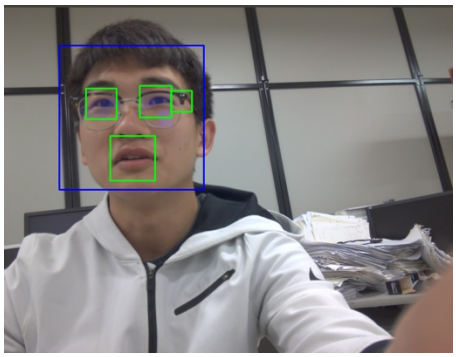

FIGURE5-FACE RECOGNITION RESULT

The detector uses blue rectangular to denote human face and green rectangular for eyes. For saving calculation amount, the recognition of eyes will not based on whole picture captured by camera, instead the blue rectangular area, identified areas of the face, will be trimmed down for eye recognition.

### 2.5 Marker

When people are in a symmetrical building, it is difficult to distinguish their position without reference. It is also very convenient to find specific targets in the treasure hunt and mark them on the map.

The initial goal of this project is to mark the doors in the room on the map. However, due to the lack of training data, the recognition ability has no generalization advantage. Therefore, a distinctive Dr.Robot in the laboratory was selected for recognition and labeling.

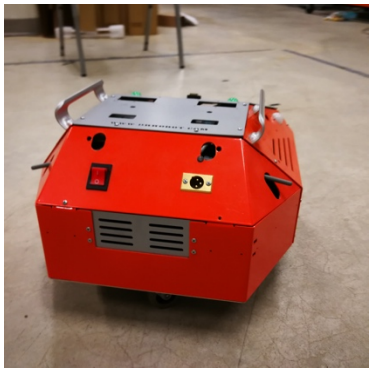

FIGURE6-MARKER TARGET DR.ROBOT MOBILE VEHICLE

Rviz provides a marker function that accepts coordinate information from subscribed topics to mark specific locations on a mapped map. Based on that, the question has been simplified that how to obtain the position of the target.

It is easy to know the current coordinate of the car itself. Thus, once we find the relative position of the target to the car, we can calculate the position of the object.

Using python rplidar package [6], we can store the scan information in numpy array form or txt document. We can do further math process to calculate the desire position.

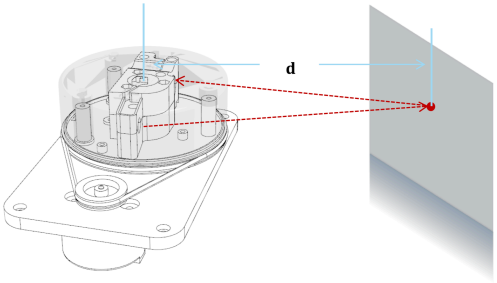


FIGURE7- THE RPLIDAR A1 WORKING SCHEMATIC [7]

| Data Type | Unit | Description |
|---|---|---|
| Distance | mm | Current measured distance value between the rotating core of the RPLIDAR A1 and the sampling point |
| Heading | degree | Current heading angle of the measurement |
| Quality | level | Quality of the measurement |
| Start Flag | (Boolean) | Flag of a new scan |

FIGURE8-SAMPLE OF OUTPUT DATA

Once the car has successfully identified the target, it will stop. The following program will read and record the lidar's feedback data. Since the camera cannot rotate after being fixed, its direction is directly in the forward direction, so the relative Angle between the target and the car is limited at $\mp 20°$.

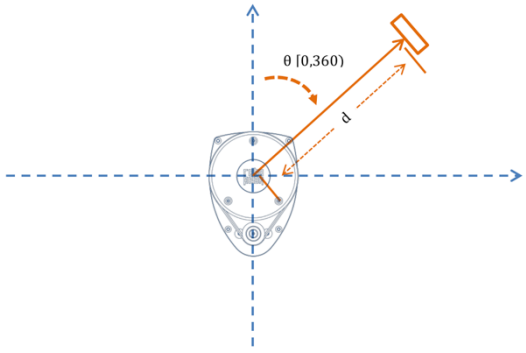


FIGURE9- LIDAR DETECTION SCHEMATIC

At this time, filtering the data whose angle is within the range of $+20°$ to $-20°$, then select the minimum value as the distance between the target and vehicle.

Using simple trigonometry, we can get the Manhattan distance between the target and the car, and then get its absolute coordinates. At this point, the algorithm creates a theme and publishes the coordinates. The marker module of Rivz subscribes to this topic so that objects can be annotated on a map in real time.

## 3. Result

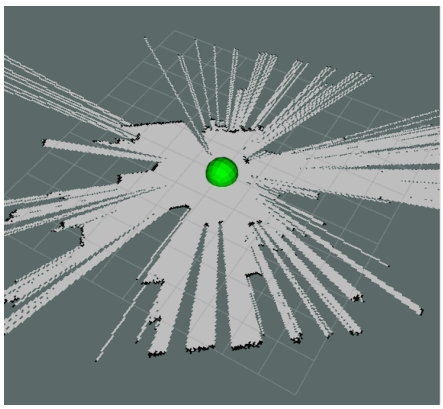

FIGURE10-DETECT EFFECTS IN THE LAB ROOM

## 4. Discussion

This vehicle is a meaningful project, but it also faces many problems in the practical operation. Firstly, due to the limited performance of lidar, the detection effect is not good in the crowded and disorderly environment. Secondly, due to the limited power supply voltage, the car's endurance and stability still need to be enhanced; Third, because there is only one camera, there is a big error in judging the position and relative Angle of objects when carrying out map annotation function.

In the process of further improvement, in addition to the adjustment of the hardware, the interconnection between cars can also be realized. After the car has been built, the effects of interconnecting with other ROS systems have been tested. Multi-vehicle interaction can realize searching for things, escape from secret rooms, provide tools and even search and rescue.

## 5. Conclusion

This project is a preliminary comprehensive application of ROS, python and SLAM, which contributes to the improvement of software and hardware design capability.

## 6. Reference

[1] slam_gmapping-ROS wiki
http://wiki.ros.org/slam_gmapping
[2] cob_people_detection-ROS wiki
http://wiki.ros.org/cob_people_detection
[3]OpenCV: OpenCV Tutorials
https://docs.opencv.org/master/d9/df8/tutorial_root.html
[4] CascadeClassifier-object detection
https://www.jianshu.com/p/53da1fbfd4cd
[5] teleop-twist-keyboard- ROS wiki
http://wiki.ros.org/teleop_twist_keyboard
[6]Python-Rplidar
https://github.com/SkoltechRobotics/rplidar
[7]Rpliadr A1 Support Document
http://bucket.download.slamtec.com/b90ae0a89feba3756bc5aaa0654c296dc76ba3ff/LD108_SLAMTEC_rplidar_datasheet_A1M8_v2.2_en.pdf