Automatic Obstacle Avoidance and Navigation Vehicle

NCTU ECE 07 0310855 陳品維

➢ Introduction:

The purpose of this project is to build a robot which can do self-driving task in an environment with obstacles in it. The robot has IMU, GPS, wheel odometry and LIDAR on it. First I will try to give the robot a pair of latitude and longitude, which is its goal point, and I'll let the robot navigate to the goal point and also avoid the obstacles at the same time.

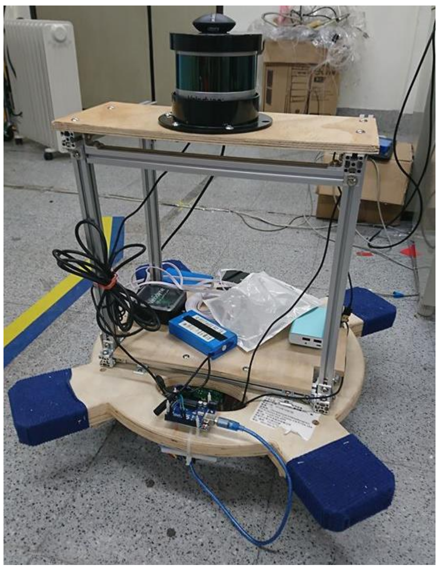
➢ Goals:

- ✓ Localize robot with GPS, IMU, wheel odometry and LIDAR.
- ✓ Use point cloud to cluster and find the obstacle
- ✓ Plan a path to let the vehicle navigate without collision
- ✓ Build a simple global map to describe the obstacle position

➢ Approach:

● Hardware:

I cut the wood and use some Aluminum stick to build the robot. My robot has IMU, GPS, Wheel odometry and LIDAR (Velodyne VLP-16) on it,.

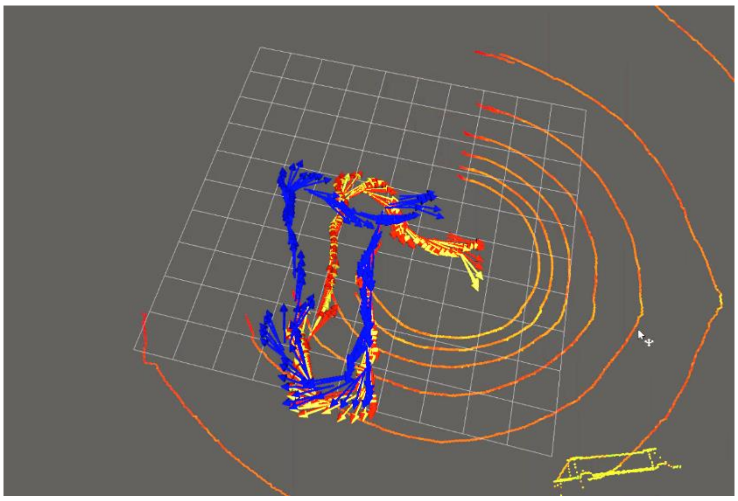


- ## Kinematics

I use simple control method to test the motor system ID, and write the PID controller to let the motor be controlled by PWM signal which publish by ROS geometry_msg/Twist message.

- ## Localization

I use IMU, GPS and wheel odometry to do EKF(Extended Kalman Filter), Also, I add LIDAR measurement to do SLAM (GMapping). And these two way can both get a robot odometry measurement. So I take these two odometry data do EKF fusion again, and then I can get a very good robot localization and mapping.



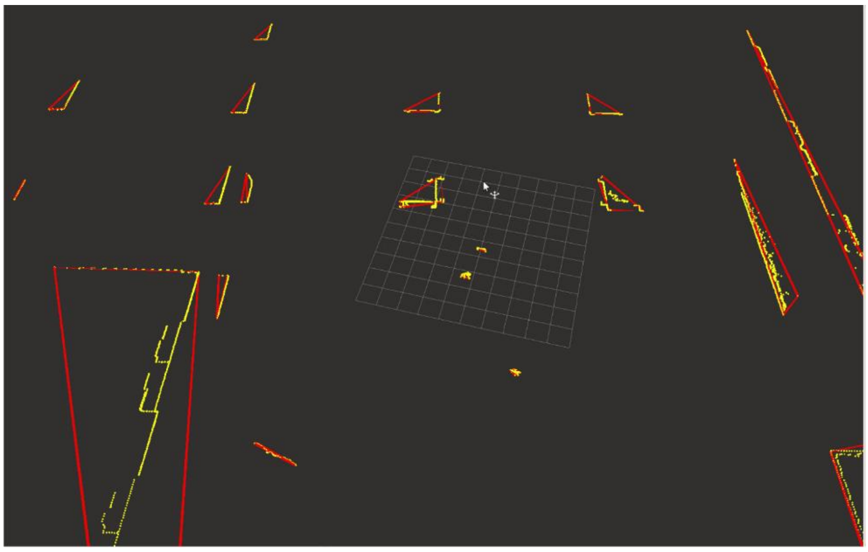
(Blue odometry: from LIDAR SLAM,
Yellow odometry: from IMU,GPS and Wheel odometry)

- ## Point Cloud Processing

First, I do the noise filter, which is to choose a point P and check how many points next to it within radius r. If the number is lower than a threshold, then we can tell it is an outlier noise.

And then I do the plane filter to remove the floor, because the floor will affect our clustering result. The way I do is to use RANSAC to fit a plane which normal project to [0,0,1] is higher than 0.8, so we can infer that the plane is almost horizontal. Moreover, if the number of the plane point is larger than specific threshold, then we can say it is the floor.

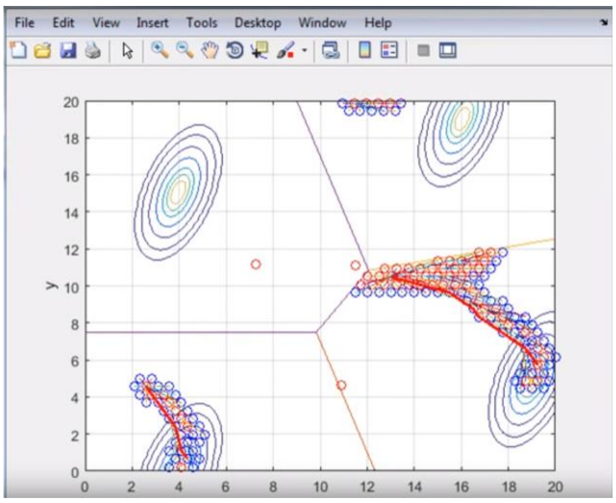
Finally, I am going to find out the obstacles. I use K-means to find the obstacles, and then use a convex hull to present the obstacle by its' vertex. As a result, each obstacle can be present as a list of points.


(the red bounding box is the convex hull of each obstacle)

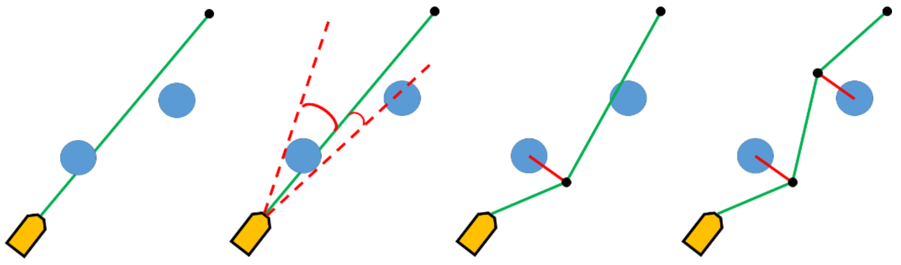
Path Planning

After having the obstacle list and a goal point, then I can do path planning which can lead my robot to the goal point and also avoid the obstacles.

The first method I try is RRT (Rapidly exploring random trees), which will randomly grow a uniform distribution tree until receive the goal point. However, the method takes too long to compute. Then I try RRT with Gaussian distribution, which set the Gaussian mean at the goal point. It can really shorten the computing time, but I still think it take too long to compute.



Eventually, I choose a method refer from the following paper: Real-Time Path Planning for Mobile Robots (2005) [Hui-Zhong Zhuang, Shu-Xin Du, Tie-Jun Wu]. It is a method which can find a minimum angle and a safe distance as a new waypoint.



- ## Path Following

Once I get the path, which is composed with a list of waypoints, then I use pure pursuit algorithm to let my robot follow the path. Finally, it can avoid any obstacle include moving object and arrive the goal point.

  - ➢ Use Gazebo and Rviz to visualize my path following algorithm: https://youtu.be/DdaZkm7_bjI

- ## Demo Video

- ➢ Self-driving at NCTU EE-building indoor: https://youtu.be/ve4Mln4D1xo
- ➢ Self-driving at NCTU EE-building outdoor: https://youtu.be/E1anLPuXjxs