Personal for GSoC 2018

JdeRobot-Academy: robot navigation using Open Motion Planning Library

About Me

Name: Hanqing Xie

Country: China

GitHub: https://github.com/hywel1994

Email: hywel1994cn@gmail.com

541651344@sjtu.edu.cn

Twitter: https://twitter.com/hywel_xhq

Tel: +86 18694500708

Time zone: GMT+8

Project

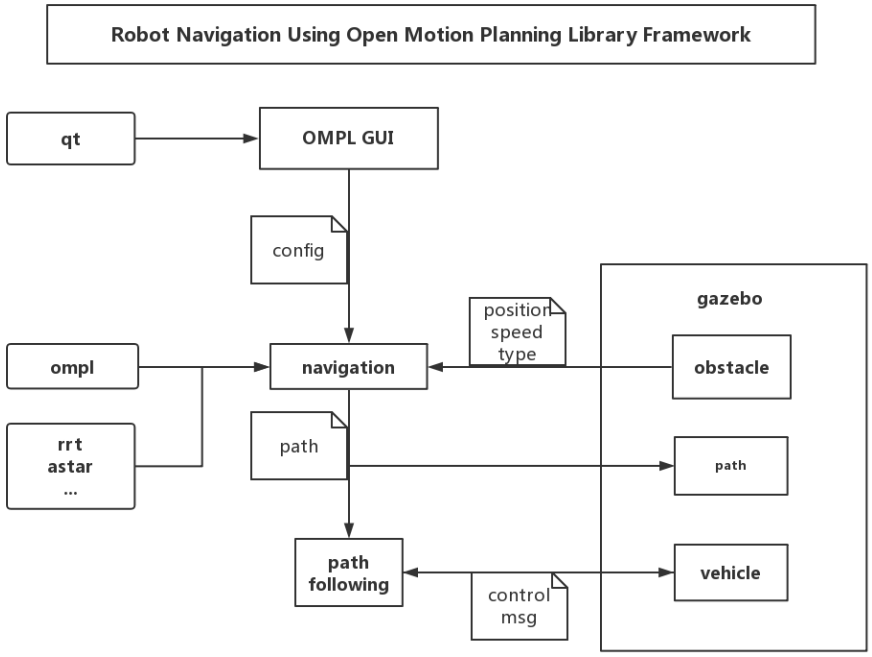
Project Name

JdeRobot-Academy: robot navigation using Open Motion Planning Library

Project Description

The main idea of this project is to introduce the OMPL (Open Motion Planning Library) into JdeRobot-Academy, in a new robot navigation exercise. For this task, the student will develop a new exercise and their solutions using different path planning algorithms of an autonomous wheeled robot or drone which moves along a known scenario in Gazebo.

Overview



Outline

1. OMPL GUI

As the requirement mentioned on the project description page, ompl gui is a user interface in the project. It should manage the navigation's start or stop and set all the parameters of this project such as robot type, start and goal pose's position and rotation, bounding box and some parameters about routing algorithm. And it needs display different parameters when choosing different OMPL planner. These all can be implemented by QT slot and signal mechanism. If the project is in the ROS environment, the tools can publish relevant message or subscribe navigation's service to change its parameter.

2. Navigation

Navigation is the core part of this project. It receives configure from visualization tool and obstacle information from gazebo. After its analysis, call the open motion planning library's api and publish corresponding ROS path message. In this node, navigation node needs deal with many message and many api. If the robot model is devoted to route, this node should update the new planning path based on new robot pose. An in-depth understanding of the OMPL is crucial for implementing an appropriate and effective navigation. It is better that the node has interface for self-path planning algorithm.

3. Obstacle

Although it just needs a known scenario in Gazebo on the project explanation, I think it is more reasonable and convenience that we can change obstacle position freely. I want to design a gazebo plugin for obstacle to publish self-information or subscribe navigation's service to change obstacle information. So, we can modify Gazebo world file and modify obstacle pose in Gazebo to modify the Gazebo scenario easily. Mainly, it is easy to implement and design gazebo scenario without modify other node.

Another plugin needed to implement is collision detection. In gazebo, there is a mature collision function, but I cannot know if it is truly happened collision in gazebo. So I can calculate the distance between robot and obstacle and judge collision by the distance. Then publish the result if collision is occurred.

4. Robot models and their path following functions

When navigation node publish the Path message, a robot model is needed to complete this motion. Conveniently, I will use robot models in the JdeRobot such turtlebot robot and adrone from parrot. Because they are familiar to us and their functions are more outstanding and easy to use. We can also select the better robot models after gazebo simulation experiments depending on the results of navigation. Of course, autonomous wheeled robot and drone are all required to test 3d or 2d path.

5. Gazebo simulation in ROS

Integrate all the above and implement the integral ROS gazebo simulation framework. And we can provide more function on this basis. Through comparing the running time, collision or different between the navigation path and robot model running path, evaluate the difference OMPL algorithm's advantages and disadvantages.

6. Gazebo simulation in ICE

I'm not familiar with ICE communication middleware, so I need several days to learn ICE. It should modify all the ROS interface and transplant the framework into ICE environment. Finally, implement the integral ICE gazebo simulation framework.

7. Summary

After the coding part is completed, I will write a summary to describe our achievement and record video to introduce how to use this project and show gazebo simulation result.

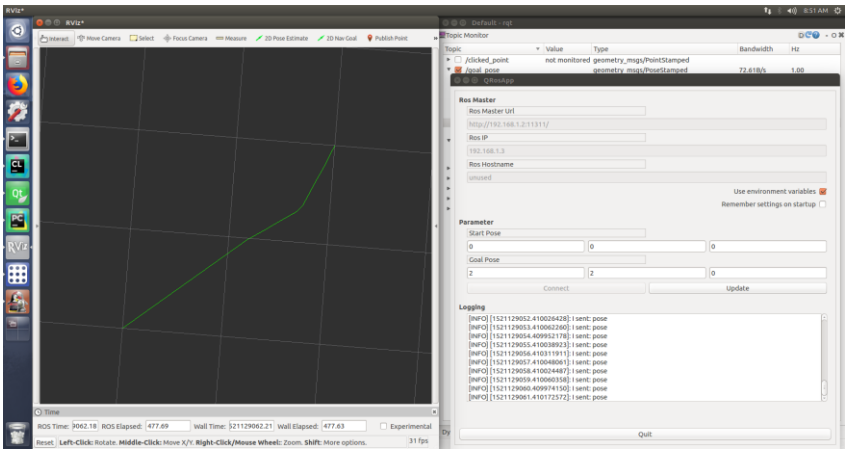
Timeline

If I am accepted to GSoC in this project, I can start work at this from Apr 9th to Aug 12th. I believe it will take about over 12 weeks for me to complete the project, expect for two weeks to have exams in this term. Maybe I will be very busy during that period, but I will be extremely excited for this experience. And before the student projects will be announced in April, I can start early to get familiar with JdeRobot.

Week	Date	Plan
1	Apr 9th - 15th	get familiar with JdeRobot-Academy
2	Apr 16th - 22nd	get familiar with JdeRobot turtlebot robot and adrone from parrot, Implement their path following node
3	Apr 23rd - 29th	find a method to display the ROS path message in gazebo
4	Apr 30th - May 6th	learn ompl's api and analyze ompl code sample
5	May 7th - 13th	develop navigation node with ompl in ROS
6	May 14th - 20th	develop visualization tool with qt in ROS
7	May 21st - 27th	Implement basic simulation framework
8	May 28th - Jun 3rd	Implement obstacle gazebo plugin and gazebo world
9	Jun 4th - 10th	Implement navigation tool with dynamic obstacle
10	Jun 11th - 17th	prepare for exams in this term
11	Jun 18th - 24th	have exams in this term
12	Jun 25th - Jul 1st	Implement the integral ROS gazebo simulation framework
13	Jul 2nd - 8th	test and debug
14	Jul 9th - 15th	get familiar with ICE communication middleware
15	Jul 16th - 22nd	modify the interface and transplant the framework into ICE environment
16	Jul 23rd - 29th	Implement the integral ICE gazebo simulation framework
17	Jul 30th - Aug 5th	test and debug
18	Aug 6th - 12th	sort out data, organize a readme and write a summary throughout this project

Demo

GitHub: https://github.com/hywel1994/gsoc-2018-jderobot-application/tree/master/demo
When I want to apply for this project, I implemented a little demo. It includes OMPL gui node and path planning node. It can set start pose and goal pose in the gui, then calculate and publish the OMPL path planning result and the path message can display in rivz.



Extra Information

Studies

1. Education

2013-2017: Shanghai Jiao Tong University, Ocean and Civil Engineering (Bachelor, major)
2013-2017: Shanghai Jiao Tong University, Industrial Design (Bachelor, minor)
2017-present: Shanghai Jiao Tong University, Ocean and Civil Engineering (master)

2. Reason for participation

Google Summer of Code is a global program that provide us a great chance to make contribution to an open source with the guidance from the best mentors in world. When I knew the information about GSoC, I made a decision to try my best to participate in. I extremely believe it can help me improve my code skill and robot algorithm. If I have the chance to participate in this project, I will try my best to complete this project. And it will be my amazing experience and priceless fortune in my college life.

My research is autonomous navigation for unmanned sailboat. My research focus on sailboat path planning algorithm (Astar, RRT, machine learning) and object detection in sea. As you know, my major is not directly related to computer major, this project is a chance for me to learn more path

planning algorithm from OMPL and prove my programming ability. And this work can be a reference for my unmanned sailboat research. I would like to keep maintaining this project and make more contribution for community after GSoC.

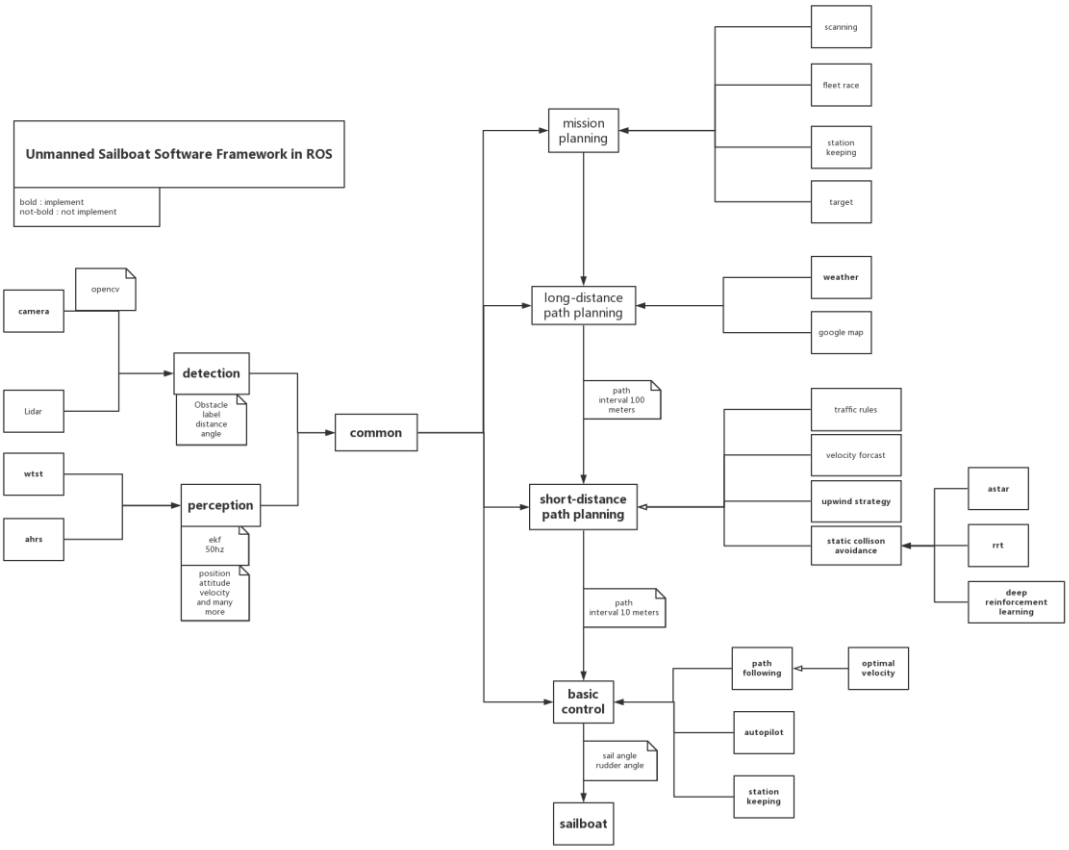
I am looking forward to participate this project with JdeRobot.

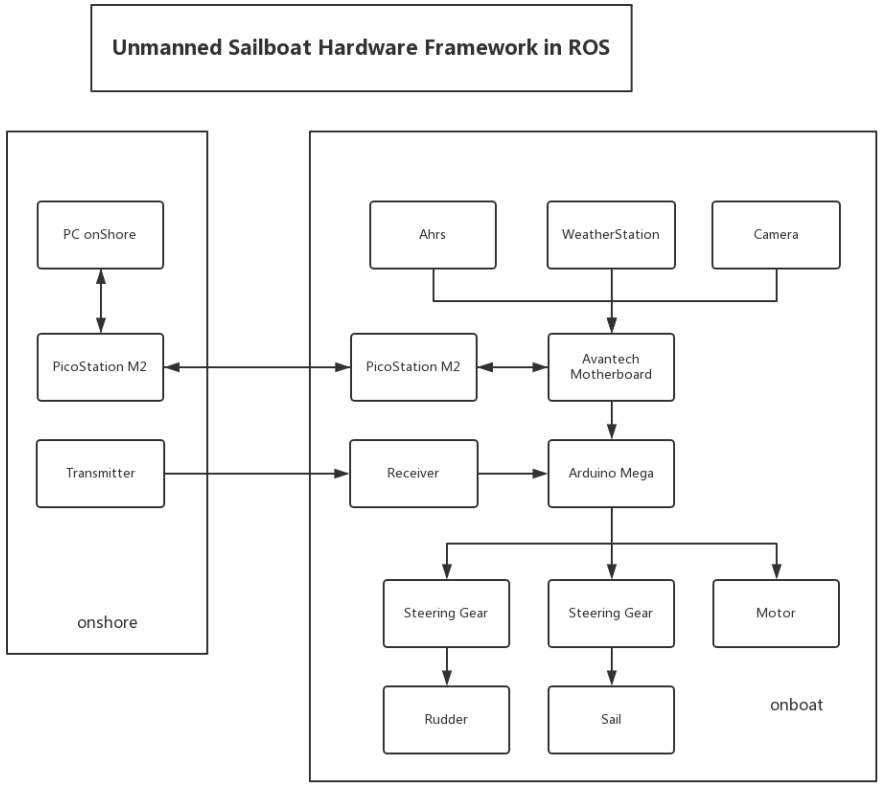
Programming Background

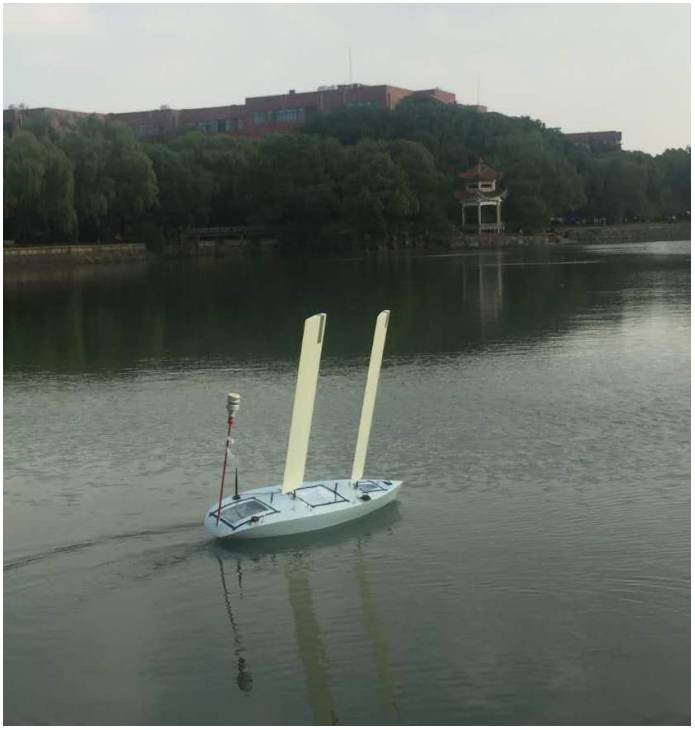
1. Autonomous navigation for unmanned sailboat

GitHub: https://github.com/hywel1994/Sailboat-Ros

In my research, I implement a project about unmanned sailboat in ROS reference to the other unmanned vehicle. Last year, I and my teammates took this system and sailboat to participate the WRSC &IRSC 2017 in Norway. This system is composed of common, driver, basic control, path planning, mission planning, perception, detection and simulation module by c++ and python. Path planning module uses astar and deep reinforcement learning algorithm, perception modules uses ekf algorithm, detection uses opencv library. It is not perfect now and I will continue completing it as my research.



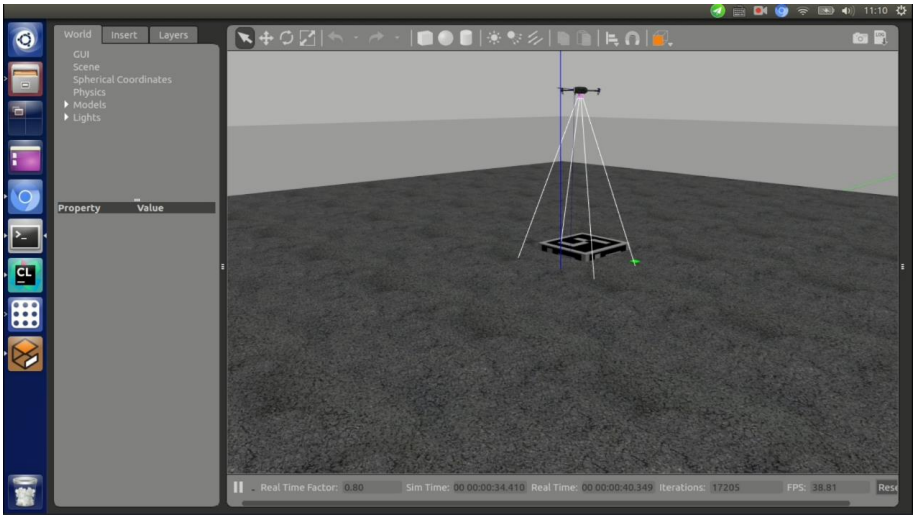

Unmanned Sailboat Hardware Framework in ROS
PC onShore
PicoStation M2
Transmitter
onshore
Ahrs
WeatherStation
Camera
PicoStation M2
Avantech Motherboard
Receiver
Arduino Mega
Steering Gear
Steering Gear
Motor
Rudder
Sail
onboat



2. PX4 precise landing by visual method

GitHub: https://github.com/hywel1994/Aruco_Ros

This programming aims to make drone automatically landed on the aruco. It can improve the landing precise than only using GPS. Then I combine two methods RTK GPS and vision tool to get more landing precise. The code is based on an open source UAV platform px4 and openCV. PX4 is running on a pixhawk, and other code is running in ROS environment on a companion computer (odroid xu4).



3. Automatic inspection of wind turbine blade based on computer vision

I joined in this programming when I was an intern in a drone company a few months ago. This programming project is the biggest project that I have participated in so far. My work was to implement the ROS interface to DJI drone and android app by c++, and transplant the object detection code by python into ROS environment.