EECE 5550 Mobile Robotics - Fall 2019

Project Proposal

Team 10: Lost in Garage

# The project goals (Zimou Gao)

**Scenario:** A small parking lot with some parking cars.

**Intelligent agent:** A Turtlebot2.

**Goal description:**

The Turtlebot does SLAM to understand the environment. During the mapping process, it can detect cars in the parking lot than identify them with specific features, such as license plate or color of the car. After finishing SLAM and identification process, the Turtlebot returns to its original point.

When we send a command with a specific car, Turtlebot will do path planning and find its way to the area of that car. At the same time, it should have an ability of obstacle avoidance to prevent itself from hitting the person walking around.

# An abstract of the project you want to do (Xupeng Zhu, JiaHao Wu)

Now a days, electric car becomes prevalent for commuting. However, one shortcut is that electric car needs to be charged, which takes time. Charging could be daily for ordinary commuter. We propose to build a mobile charging robot that can navigate in the garage, mapping the garage as well cars with license plate, and plug in the charger to a specific car given license plate.

# Detail of steps that need to be taken to achieve the goals and brief literature overview

## 1.Way point navigation --vector + PID (Jiahao Wu)

Way point navigation is a robotic algorithm for motion planning in the case of known robotic positions. When we already know the real-time position of the machine and the attitude, we can use this fuzzy control to control the robot movement.

The main idea of fuzzy control is to translate the information and knowledge that is used by human pilots into fuzzy rules, which can be used by fuzzy controllers. It has strong robustness, good adaptability and fault tolerance. But the weakness is low steady state accuracy due to lack of integral link. It is implemented in PID position controller to tune the parameters of PID gains (𝐾𝑝,𝐾𝑖 ,𝐾𝑑) with the inputs are error (ℯ) and error of rate (ℯ̇). There inputs are obtained from evaluating the feedback of GPS sensor and the reference position given by the user. The system has three output variables which are alpha (𝛼), beta (𝛽) and gamma (𝛾). In first stage of fuzzification, the inputs (ℯ, ℯ̇) are divided into three subsets of fuzzy with trapezoidal membership function between the range from -1 until 1. For the outputs, trapezoidal membership function with two subsets of fuzzy is also used in alpha (𝛼) and beta (𝛽) between the range from 0 until 1 while triangle membership function with four subsets of fuzzy used in gamma (𝛾) between the range from 1 until 6. The Fuzzy Inference System (FIS)has 9 rules. The method of Mamdani with Min-Max was chosen for conducting the combination of fuzzy.

## 2.Mapping --gmapping (Xupeng Zhu)

The grid map will be generated by gmapping algorithm which uses the depth information obtained by Kinect in the turtlebot2. The map will be saved as a 2D occupation grid.

The mapping problem of mobile robot is usually called Simultaneous Localization And Mapping (SLAM). Particle filter use randomly generated particles to represent the location of the robot, then use likelihood to drop particles with low probabilities and keep the one with high probabilities.

One problem is that particles filter will generate too much particles that consume unnecessary computation resources. Rao-Blackwellized particle filter can effectively sampling the particle too solve this problem. GMapping is a variation of Rao-Blackwellized particle filter that suits for grid maps.

## 3.Obstacle avoidance --opti-point hard, kinect package (Jiahao Wu)

When the robot is driving, it will encounter obstacles of different characteristics. In order to accomplish our goal, the robot needs to avoid obstacles autonomously and continue to remember before the path planning.

After building a model of the environment and generating a pre-computed path according to all the methods described in this and the preceding chapter, the robotstarts following the path while it is capturing data from the Kinect sensor. Thisdata is continuously converted to a point cloud. If the path is being intersected by any circle with a center of point cloud and the radius of the robot, and the distance between the current pose and the intersection point is less than some threshold,then robot stops, adds the current point cloud to the world and performs all the steps to build a new model of the environment.

The previous path is retained and if during re-planning the obstacle which blocked the path is removed (for example a dynamic obstacle) then the robot will continue the original path. In other words, the dynamic changes in the environment which do not have any influence on the pre-computed path either they cannot be perceived by the Kinect field of view or they don’t interfere with the path.

## 4.Path planning --A\*, D\*, RRT (Xupeng Zhu)

The present location of the robot will be the starting point, and the goal point i.e., the location of the car, will be given by III.5 vision. The path planning can be conducted by applying the following algorithms in the 2D occupation map obtained by part III.2.

A\* is a graph traverse algorithm that is complete and optimal. A\* utilizes the heuristic function h(n) plus the cost function: f(n) = g(n) + h(n) as priority to traverse the graph. However, A\* has high demand on storage.

D\* is Dynamic A\* in short. Initially D\* algorithm assumes the robot workspace is empty and plan the initial path. Then D\* will include obstacles in the map and replan the path iteratively. Ultimately, D\* can find a path to the goal, if it exists. D\* is more efficient than apply A\* iteratively to deal with dynamic map.

Rapidly-exploring Random Tree(RRT) is a path search algorithm used in search space (in contrast to A\* and D\* in a graph). It is suitable for high dimension space path search. Literally, RRT grow a tree rooted at the start point by randomly sampling point and connect the point to a branch if it is obstacle free.

## 5.Vision (Zimou Gao)

1. **Task description:**

Detect the cars and identify them with Kinect camera. The input of the object detection is an image, and the output is the position and category of the object in the image.

1. **Solution:**
2. **Use Deep Learning for Object Detection.**

**We choose SSD(single shot multibox detector) for this task:** SSD has two components: a backbone model and SSD head. Backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that is able to extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lowerresolution.

**Training Dataset:** PASCAL VOC; ImageNet; CIFAR10;

1. **Use find\_object\_2d package and Depth information to localize the car.**

**find\_object\_2d package:** Simple Qt interface to try OpenCV implementations of SIFT, SURF, FAST, BRIEF and other feature detectors and descriptors. Using a webcam, objects can be detected and published on a ROS topic with ID and position (pixels in the image). This package is a ROS integration of the [Find-Object](http://introlab.github.io/find-object) application.

1. **Comparison of different Algorithm.**

**Two stage detection framework:** Contains the region proposal, first obtains the ROI, then identifies and returns the bounding box to the ROI, represented by the RCNN series method.

**One stage detection framework:** Without the region proposal, the entire image is grided, and each grid is identified and returned, represented by the YOLO series method.

|  |  |  |
| --- | --- | --- |
|  | One-stage | Two-stage |
| Framework | YOLO, SSD, RetinaNet | Fast R-CNN, Faster R-CNN |
| Accuracy | Lower | Higher |
| Efficiency | Faster | Slower |
| Status | Active | Active |

IV. Schedule

Week1: finish turtlebot2 tutorial, run the physical turtlebot2.

Week2: set up Gazebo simulation environment for turtlebot2.

Week3: try to run packages for III.1--III.5.

Week4: fine tune the packages and algorithms, deploy the turtlebot2 in garage to execute the assigned mission.

Week5: adjust the turtlebot2 and record video, start preparing PPT.

Week6: finish PPT, prepare for the presentation.

# V. References:

Mapping:

<http://wiki.ros.org/turtlebot_navigation/Tutorials/indigo/Build%20a%20map%20with%20SLAM>

<https://openslam-org.github.io/gmapping.html>

<https://en.wikipedia.org/wiki/Particle_filter>

Path Planning:

[https://en.wikipedia.org/wiki/A\*\_search\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm)

[https://en.wikipedia.org/wiki/D\*](https://en.wikipedia.org/wiki/D*)

<https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree>

<http://www.ccs.neu.edu/home/rplatt/cs5335_spring2019/slides/prm_rrt.pdf>

**Visual detection:**

<https://github.com/s-gv/libalpr>

<https://github.com/sirius-ai/LPRNet_Pytorch>

<https://github.com/qfgaohao/pytorch-ssd>

<http://wiki.ros.org/find_object_2d>