Instruction for Assignment 3 for Term Project

Hector SLAM(Simultaneous localization and mapping)

Introduction

In project2, we implemented RRT to find a path from initial point to goal point without a collision with obstacles. We used PID controller to control the steering angle of the front wheels of a RC car. In simulation, we can know position of the RC car from gazebo, but in reality, it is necessary to use other methods like GPS to locate the RC car. Dead reckoning is the process of calculating one's current position considering its state, for example linear velocity and angular velocity. However, this method has a disadvantage that the calculated position gradually deviates the actual position as the time passes.

By the way, SLAM enables us to construct or update a map of an unknown environment while simultaneously keeping track of a robot's location within it. Furthermore, since Hector SLAM can be performed without knowing the kinematics of the agent, it is very suitable method for our situation. In project3, you will draw a map and localize the RC car with Hector SLAM. And you will use RVIZ which is a 3D visualizer for displaying sensor data and state information from ROS to display a map and RC car.

Basics of SLAM[1)]

The SLAM problem is building a map while simultaneously determining the robot's position relative to this map when the robot roams an unknown environment. Let us denote time by t, and the robot location by x_t. x_t usually consists of x, y coordinates and heading direction of the robot. From time t=1 to T(terminal time), the path of the robot can be represented by X_T.

$$X_T = \{x_0, x_1, x_2, \dots, x_T\}$$

And u_t is the control vector like steering angle and speed which is applied at t-1 time to drive the vehicle to a state x_t at time t. Then the sequence of control vector

$$U_T = \{u_1, u_2, \dots, u_T\}$$

characterizes the relative motion of the robot. However, since the calculating the next state with the control vector is like dead reckoning, the path X_T inevitably diverge from the truth.

Finally, the robot senses objects in the environment. For example, our RC car detects objects with a laser sensor. Let m denote the true map of the environment. The environment may be comprised of landmarks, objects, surfaces, etc., and m describes their locations. Z_{it} is The observation taken from the vehicle of the location of the i-th landmark at time t. To simply the problem, let's assume that only one landmark is on the map. Therefore, the set of the landmark observations is Z_T

$$Z_T = \{z_1, z_2, \dots, z_T\}$$

Figure 1 illustrates the variables of SLAM problem and their relationship.

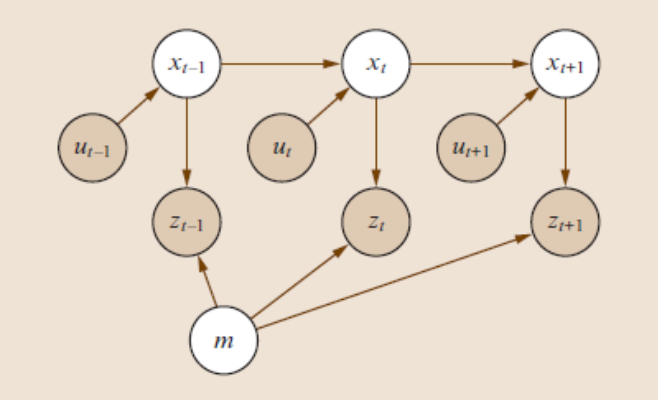

Figure 1 Graphical model of the SLAM problem

With the variables, the SLAM problem can be represented as finding the world map m and the sequence of the robot states X_t from the control and measurement data, U_t and Z_t. In probabilistic form, the SLAM problem requires that the probability distribution

$$\mathrm{P}(x_t, m|\mathrm{Z}_t, U_t, x_0)$$

be computed for all times T. And once the location of the robot and map are defined, observations are independent to the location of the robot and map. It is called the observation model and described as

$$\mathrm{P}(z_t|x_t, m)$$

And next state of the robot is only dependent on the state of the robot at the previous time and the control vector. It is called the motion model and described as

$$\mathrm{P}(x_t|x_{t-1}, u_t)$$

Now, the SLAM algorithm is implemented in a standard two-step recursive (sequential) prediction (time-update) correction (measurement-update) form:

Time-update

$$\mathrm{P}(x_t, \mathrm{m}|\mathrm{Z}_{t-1}, U_t, x_0) = \int \mathrm{P}(x_t|x_{t-1}, u_t)\, \mathrm{P}(x_{t-1}, \mathrm{m}|\mathrm{Z}_{t-1}, U_{t-1}, x_0) dx_{t-1}$$

Measurement Update

$$\mathrm{P}(x_t, \mathrm{m}|\mathrm{Z}_t, U_t, x_0) = \frac{\mathrm{P}(z_t|x_t, m)\mathrm{P}(x_t, \mathrm{m}|\mathrm{Z}_{t-1}, U_t, x_0)}{P(z_t|\mathrm{Z}_{t-1}, U_t)}$$

Make Hector SLAM package

1. Extract project3.zip file to the src folder in your catkin workspace.

2. Change CMakeLists.txt of project2 to new CMakeLists.txt we uploaded.

3. Open a launch file and change the last line of the launch file like the screenshot below.

File : ~/catkin_ws/src/racecar-simulator-master/racecar_control/launch/racecar_control.launch

```
  <!-- servo node -->
  <node pkg="racecar_control" type="servo_commands.py" name="servo_commands"
output="screen">
    <remap from="/racecar/ackermann_cmd_mux/output" to="/vesc/low_level/
ackermann_cmd_mux/output"/>
  </node>

  <!-- Allow for Gazebo to broadcast odom -->
  <!--node pkg="racecar_gazebo" name="gazebo_odometry_node"
type="gazebo_odometry.py"/-->

</launch>
```

4. Fill TODO parts of main.cpp code in project3/src folder(Copy and paste your main.cpp code of project2.)

5. ~/catkin_ws$ catkin_make

If you have any errors like {PACAKGE_NAME} is not installed, you can install the package with the command, sudo apt-get install ros-kinectic-{PACKAGE_NAME}.

6. ~$ roslaunch project3 project3.launch

It will open gazebo simulator and RVIZ.

7. ~$ rosrun project3 project3

It will make a map on RVIZ

8. ~$ rostopic pub syscommand std_msgs/String "savegeotiff"

Go to ~/catkin_ws/src/hector_slam/hector_geotiff/maps.

You can find your own map like hector_slam_map_##:##:##.tif

Submission Format

Compress your project folder including all your project files and upload it with your map on the eTL. The name of the compressed file should be **"IS_Project_03_[TeamName].tar.gz"**.

Reference

1) Durrant-Whyte, Hugh, and Tim Bailey. "Simultaneous localization and mapping: part I." IEEE robotics & automation magazine 13.2 (2006): 99-110.