Introduction to Unmanned Ground Vehicles

Instructor: Micho Radovnikovich
Email: mtradovn@oakland.edu
ECE 495/595 – Winter 2017

Homework #6 – Configuring the ROS Nav Stack

Due: Wednesday, April 12th

The purpose of this assignment is to become more familiar with the navigation stack built into ROS, as well as to obtain more experience creating launch and YAML parameter files. The goal is to use the Gazebo simulation of the Mantis robot developed in lecture and integrate it with **move_base** so that Mantis can navigate Maze World like Roundbot can.

Provided Files

The **homework6** package will be initialized with the following files, which will be pushed into the central homework repository when Homework #6 is assigned:

- Bare-bones **CMakeLists.txt** and **package.xml**.
- Launch file that will eventually run the entire system and the test script (**homework6/launch/homework6.launch**). You will have to modify this.
- Test script (**homework6/test/test_homework6.cpp**).

Step 1 – Create move_base YAML files

Copy the YAML files from the **maze_world_navigation** package into a folder called **yaml** in the **homework6** package. All the parameter values should work, but you will have to adjust TF frame and topic names in the YAML files to match the Mantis system.

Step 2 – Create Drive Control Node

The **move_base** node will output a **geometry_msgs/Twist** command for Mantis to follow, but the Mantis simulation only responds to two separate wheel speed command topics. Therefore, a drive control node needs to be written.

> Mantis is a differential drive vehicle similar to Roundbot. However, the kinematic parameters are slightly different, and the wheel speed command topics have different names. Copy your differential drive kinematics node from Homework #3 and add it as a new node in the **homework6** package.

> Check Mantis' URDF file to determine the wheel radius (r_w) and wheel separation (W) parameter values and update your kinematics code with these values.

> Identify the names of the wheel speed command topics and make sure you modify the drive control node to publish to these topics.

Step 2 – Modify homework6.launch

By default, **homework6.launch** runs Gazebo, loads Maze World, and runs the test script. Some hints of what needs to be added:

> Spawn the Mantis model in Gazebo and run the simulated wheel controllers and LIDAR sensor.

> Run **move_base** and load the configuration parameters from the YAML files created in Step 1.

> Run the new drive control node created in Step 2. Use **remap** tags as necessary to be sure the drive control node subscribes to the command topic coming from **move_base**.

> "Solve" the localization problem by publishing a zero-translation, zero-rotation static transform from **map** to **world** frames.

> Start Rviz and load a configuration file that shows useful things like the global and local costmaps, the global plan, and a model of Mantis.

Step 3 – Test and Debug the System

Launch **homework6.launch**. The test script will publish a goal to **move_base**, and Mantis should start exploring Maze World by following the global plan. Mantis will eventually reach its goal, at which point the test script will report a success.

If things don't behave as expected, there can be a lot of potential causes. Use the various tools at your disposal like **rqt_graph** and TF **view_frames** to debug the system, paying attention to common errors like TF frame tree disconnections and topic name mismatches. Also pay attention to any warnings or errors appearing on the terminal while the system is running; these are usually helpful to pinpointing issues as well.

What to Submit

Submit Homework #6 by pushing the following to your repository:

> Modified **CMakeLists.txt**.

> Modified **homework6.launch**.

> C++ source files for drive control node.

> YAML parameter files for **move_base**.

> Rviz configuration file to display what's going on in Mantis' brain.