CS 237B: Principles of Robot Autonomy II
Section 1: Revisiting AA274A

Our goals for this week's section (and next):

1. Prepare a stable ROS environment on your own machine
2. Create project repositories for your new group
3. Refamiliarize yourself with the AA274 Turtlebot stack
4. Reproduce the final demo baseline from AA274A (autonomous food delivery)

By the end of next week, you should have a working demo of the autonomous food delivery task. The TAs will check off your group as soon as you can show this demo.

1 Environment Setup

Starting AA274B/CS237B, we will only support VM and Ubuntu Native environments. If you don't have the environment, let's go ahead and set it up again: Link to the guide

Problem 1: Once done, please include a screenshot of your work environment running the following launch file:

```
roslaunch asl_turtlebot section4_demo.launch
```

2 Create a new Github repository

As with AA274a, you should use Github to collaborate on your final project with your group. Create a *PRIVATE* fork of the `asl_turtlebot` repository on Github and add the members of your group:

https://github.com/stanfordasl/asl_turtlebot

As we update the `asl_turtlebot` repository throughout the quarter, you should make sure pull these changes into your own forks.

3 Test the AA274 Turtlebot stack

Now, grab your assigned Turtlebot and laptop and set up your new repository fork (only on your laptop):

```
cd ~/catkin_ws/src
rm -r asl_turtlebot
git clone <your repo url> asl_turtlebot
cd ~/catkin_ws
catkin_make
```

If you remember from last quarter, ROS master runs on the Turtlebot with the core drivers, but RViz and `navigator.py` (which encompasses the state machine, controller, and planner), run from your laptop as ROS nodes.

Let's run the master launch script on the turtlebot:

```
ssh aa274@<robot name>.local
roslaunch asl_turtlebot turtlebot3_bringup_jetson_pi.launch
```

One of the nodes that the `turtlebot3_bringup_jetson_pi` launch file starts is `gmapping`, which uses the LIDAR readings to perform SLAM (simultaneous localization and mapping), giving us an *occupancy grid* map of the environment around the robot, as well as an estimate of the robots position within this map. You can visualize this map in RViz on your laptop.

To connect to ROS master running on the turtlebot, you have to set the `$ROS_MASTER_URI` and `$ROS_HOSTNAME` environment variables on your laptop. We provide the `rostb3` script to make setting these environment variables easier. First, open

```
~/catkin_ws/src/asl_turtlebot/rostb3.sh
```

and replace `<turtlebot_name>` with with the name of your turtlebot. Now, in the terminal, run

```
rostb3
```

and make sure the updated `$ROS_MASTER_URI`, `$ROS_IP` and `$ROS_HOSTNAME` settings are just as you intended. You need to rerun this command every time you open a new terminal window. To test that your connection is working, you can try running the following commands:

```
rostopic list
rostopic echo tf
```

Now, we should be ready to launch rviz:

```
rviz
```

Problem 2: Add `velodyne_points_throttle`, `camera_relay/image` and `map` visualization in RViz. Include a screenshot the visualization in your submission.

In a new terminal window, teleop the turtlebot to test its motors (after placing the turtlebot on the ground!)

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

4 Autonomous food delivery

If it's been a while since you took the class and need a refresher on the final project setup from AA274A, you can see the old project description here:

http://asl.stanford.edu/aa274/project.html

To summarize, the demo is comprised of two stages. In the first stage, your task to first navigate the environment via teleop or Rviz nav goals to build the environment map, as well as record the locations of food items detected in the camera (you provide the food you want to detect). In the second stage, we will

give you a list of food to deliver, and the turtlebot should autonomously pick up each food item (by revisiting their recorded location) and come back to the home position.

If you are working with your old group from AA274A, you can simply reuse your project code from last quarter. If you are with new people, figure out how to combine your code so you have a nice baseline to work from for the rest of the quarter. You have two weeks to finish building this stack.

The commands for running the demo depends on your projects, but at minimum, you should run the `navigator.py` script, which listens for Rviz nav goals, runs the A* planner, and executes the trajectory/pose controllers.

```
rosrun asl_turtlebot navigator.py
```

You probably created a launch script to simultaneously launch `navigator.py` and Rviz, which may look something like:

```
roslaunch asl_turtlebot my_nav.launch
```

Now test out your demo, and once everything is operational again, deliver some food to your hungry TAs :)

Problem 3: Demo AA274A Final Project with a TA. TA will check you off on Gradescope.