AA 274: Principles of Robot Autonomy
Final Project
Demo Day: Wednesday Dec 08, 8:30 - 11:30 am

The final project demo will be held over zoom on Wednesday December 08, from 8:30 - 11:30 am. Each group will be given 12-minute slots to present and demo their project (we recommend presenting and demoing simultaneously to maximize time efficiency). We will have 3 groups doing their demo on Genbu at the same time, in separate breakout rooms, to get through the final project evaluations quicker. A Google sheet with time slot sign-ups is here. Your entire group should be present for your project demo; contact the teaching staff immediately if you have a conflict.

You should use `asl_turtlebot` as a starting point, but you are free to make any changes or restructure to better suit your project.

Baseline

The baseline project consists of completing the mission, command center visualization, and a quick pitch presentation for your project. The baseline implementation is worth 70% of total credit, and is not sufficient for full credit. Your group will need to come up with extensions to make up for the remaining 30%. See the **Extensions** section for more details.

Mission:

You are responsible for building your own map for the mission. We have provided a starter launch file, `project_sim.launch`, that brings up `project_city.world`, however you are free to build your own world with comparable complexity (please check in with the TAs if you are unsure whether your custom world has comparable complexity). You will will also need to add objects to your map either through the Gazebo gui or by editing the `.world` file (see `signs.world` as an example). There are a number of readily available objects in the Gazebo model database, and you are free to search for other object models.

Exploration: Use human-operated waypoint following to navigate through the world, and build a map. Your robot must avoid colliding into obstacles.

1. Use SLAM node gmapping (no need for EKF you wrote) for robot pose and occupancy grid. Navigate using rviz clicks or hard-coded waypoints. This will use A* for path planning, controllers for tracking, pose controller, etc. that already exist in navigator.py.

   Your robot should discover all walls, explore the map fully, and return to the starting point.

2. Detect and record object locations through TF frames and broadcasting on a topic. While the robot is exploring, run the CNN detector by setting the parameter `use_tf = true`. When an object is detected, its type and location should be published on a topic, and recorded so your robot can “rescue” it later. Since you are free to build your own map, we suggest using objects that the CNN can reliably detect.

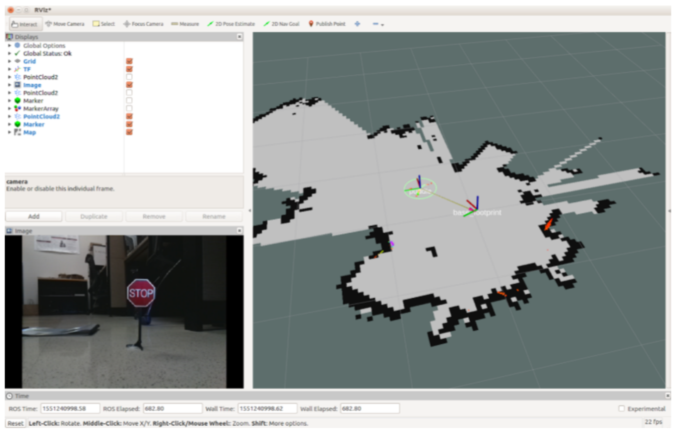

Figure 1: Example baseline command center.

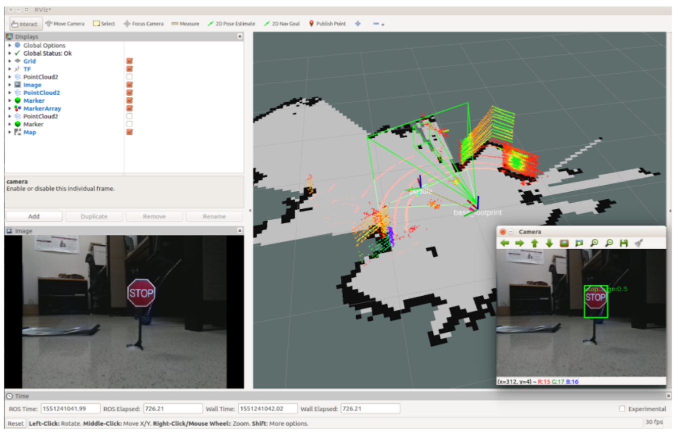

Figure 2: Example command center with extensions.

> **Your robot should detect at least three different objects. After the exploration phase, you should provide the TA with a list of objects that your robot has detected that are fair game for the resue phase of the mission.**

Rescue: The TAs will input a list of ∼3 objects (from your detected objects) that your robot will need to rescue. **Robot must autonomously navigate back to the objects, in any order. Stop at an object location for 3-5 s to "pick it up". Robots should also return to the starting point after the mission is completed.**

Command Center Visualizations:

Using RViz, visualize the map, the Turtlebot, and desired poses with markers. See Figures 1 and 2 for examples.

Pitch Presentation:

The demo of your robot should also include a three minute presentation:

- Describe your overall software stack.

- Explain your design decisions.
- Highlight what makes your robot unique!
- Include extra slides to tell us about details of your robot.

Extensions

The majority of extensions will count towards 10% of the final project grade, but the TAs reserve the right to give extra credit (15%) for exceptional extensions, and partial credit (5%) for minor extensions. To get started, some extensions that have been done in the past are:

- Implementation of graph search algorithms in C++ (15%).
- Using smach to visualize state machines (15%).
- Using TSP to find the fastest rescue route (15%).
- Fully autonomous exploration using frontier exploration (15%).
- Detecting and stopping at stop signs, people, etc (10%).
- Detecting speed limit signs and dynamically reconfiguring the control limits (10%).
- Using lidar point scans for collision avoidance (10%).
- Using RRT* instead of A* for routing (10%).
- Publishing markers at the location of located objects (10%).
- Map dilation (10%).
- Visualizing field of view of the camera on RViz (10%).
- Detecting and following a moving cat/dog (10%).
- Publishing a "meow"/"woof" when your robot detects a cat/dog (5%).

You are also free to come up with your own extension! Talk to the TAs about your ideas. You can tweak the setup, add objects, etc. to help you demo your extensions. We will also give extra credit if you come up with a fun theme!

Summary

Due Date: Dec 08, 8:30 - 11:30 am

Deliverables:

1. Implementing the baseline mission (40%).
2. Command center visualization in RViz (20%).
3. Three minute pitch presentation (10%).
4. Extensions to the baseline implementation (30%).