Section 7: Frontier Exploration

Overview

The goals for this section:

1. Run Frontier Exploration on the TurtleBot.
2. Use the image classifier from Section 6 to perform Frontier Exploration until a stop sign is found.
3. This section uses the heading controller, AStar planner, image classifier, and frontier exploration code that you have been developing throughout the quarter. It is a final test of your autonomy stack! Congratulations on making it this far 😃

Setup

Task 1.1 — Cleanup. Remove directory `~/autonomy_ws` if it exists. This may be leftover from a previous section.

Task 1.2 — Directories. Create the base directory structure for your group's ROS workspace, `~/autonomy_ws/src`.

Task 1.3 — Clone and Branch. Clone your group's GitHub repo to the `src/` directory, and create a new branch called `s7_inperson`.

Task 1.4 — Copy homework code. We will be adding the code you wrote in HW4 to your autonomy stack. Copy your frontier exploration node file to the folder `~/autonomy_ws/src/<group_repo>/autonomy_repo/scripts` and the launch file to the folder `~/autonomy_ws/src/<group_repo>/autonomy_repo/launch` on the AA274a laptop.

Task 1.5 — Build. Update the autonomy repo `CMakeLists.txt` file, and make sure your node is executable. Navigate back to the root of your workspace, `~/autonomy_ws/`, and build your ROS workspace with `colcon build`.

Frontier Exploration (Simulation)

Task 2.1 — Launch simulator. Firstly, we will test out your HW4 code before you can test it out on the actual robot. Start your TurtleBot simulator by running:

```
Shell
ros2 launch asl_tb3_sim root.launch.py
```

Task 2.2 - Launch frontier exploration. In a new terminal, source your workspace, and then launch your frontier exploration:.

```
Shell
source ~/autonomy_ws/install/setup.bash ros2 launch autonomy_repo
<your_frontier_exploration_launch_file>
```

- **Checkpoint — Call a CA to show that your TurtleBot in Gazebo performs Frontier Exploration.**

Frontier Exploration (Hardware)

Task 3.1 - End your simulator process, get a CA to give you a robot based on your **`$ROS_DOMAIN_ID`**, set up the robot using the same steps covered in past sections, and demonstrate that your controller works on the real robot as well.

```
None
ssh aa274@<robot_name>.local

# Enter the password given by the CA
# Start the ROS environment
micromamba activate ros_env

# Bring up the Turtlebot
```

```
ros2 launch asl_tb3_driver bringup.launch.py
```

Task 3.2 - Launch your frontier exploration launch file with `use_sim_time` set to `false`.

```
None
ros2 launch autonomy_repo <your_frontier_exploration_launch_file>
use_sim_time:=false
```

- **Checkpoint — Call a CA to show that your TurtleBot in hardware performs Frontier Exploration.**

Frontier Exploration + Object Detection

We will now integrate frontier exploration with object detection. We will update a copy of the frontier exploration script and launch file for this purpose.

Task 4.1 - In your frontier exploration node, create a subscriber that listens to the `/detector_bool` topic. When a stop sign is detected, the TurtleBot should stop for `5` seconds before resuming exploration. You may find ideas from Section 6 helpful for implementing this functionality.

One issue is that if the robot resumes spinning while the stop sign is still in view, it may immediately stop again. Though there are many ways to fix this, one possible solution is to introduce a short delay during which stop sign detections are ignored after the robot starts moving. Think about how you can use timing logic in the `/detector_bool` callback to implement this behavior.

Simulation Test

Before moving to the real robot, test your updated frontier exploration node in simulation:

Launch your standard simulation setup (e.g., TurtleBot + map + frontier exploration), using the launch files from earlier tasks.

With the system running, manually simulate stop-sign detections by publishing to /detector_bool from the command line:

Set the detector to true (stop sign detected) once:

```
None
ros2 topic pub /detector_bool std_msgs/msg/Bool "{data: true}"
```

Set the detector to false (no stop sign detected) once:

```
None
ros2 topic pub /detector_bool std_msgs/msg/Bool "{data: false}"
```

You should observe the robot stop for 5 seconds when true is published, then resume exploration. Verify that your “ignore detections for a short period” logic prevents it from immediately stopping again as soon as it starts moving.

- **Checkpoint — Call a CA to show that your TurtleBot in simulation performs Frontier Exploration until it views a stop sign.**

Task 4.2 - To launch the detector, you can launch the driver bringup with camera set to true.

```
None
ros2 launch asl_tb3_driver bringup.launch.py camera:=true
```

Task 4.3 - Having run `bringup.launch.py` on the robot, run the launch file you created in Task 4.2:

```
None
ros2 launch autonomy_repo <launch_file> use_sim_time:=false
```

- **Checkpoint — Call a CA to show that your TurtleBot in hardware performs Frontier Exploration until it views a stop sign.**

Clean Up

Clean up the workstation.

1. Push your code (`add`, `commit`, and `push`) to your `s7_inperson` branch!
2. Create a PR either from the GitHub web page or by running `gh pr create` in your terminal.
3. Verify that you've pushed correctly by checking if the code you wrote is in the repository on GitHub.com
4. Delete the `~/autonomy_ws/` .
5. Log out of GitHub on the workstation.
6. Close all terminals and windows!
7. Shutdown laptop

- **Checkpoint — Call a CA over to sign you out!**