CS-8803 Special Topic: Machine Learning for Robotics

First Intermediary Project: Mapping and Finding Objects

Support: cedric.pradalier@georgiatech-metz.fr
Office hours: 10:00 to 18:00.

Setup

This project use the 3D scene:

```
/cs-share/pradalier/scenes/rosControlKinect3d.ttt
```

Because this project is significantly more complex than the previous ones, more tools are provided. In particular, a task management system has already been prepared for the bubblebot, implementing a Goto(x,y) functionality. Additionally, a road-map of feasible routes is provided as a graph (Python iGraph library). In this road-map, every node is a position in the scene (2D), and an edge between two nodes means that there is a straigh line safe path between these two positions.

Setup you environment by copying the following directories into your catkin workspace and running catkin_make (from `/cs-share/pradalier/cs8804_students_ws/src`):

- `vrep_ros_teleop`
- `floor_graph`
- `floor_nav`
- `ros_task_manager`

Start the simulation in V-Rep and launch the launch_vrep.launch file from the floor_nav package. This launch file might need to be modified to point at the right place for the teleop package. Assuming the launch files starts correctly, nothing should happen.

Run rviz and make sure to add the TF tree and subscribe to the only available MarkerArray topic.

Then run the missions/test_graph.py script from the floor_nav package. The robot should aim for Node 2 and start traversing the scene road graph using a hand-crafted Hamilton path.

If you reach this point, your setup is correct and you can start working on the project objectives.

Project Objective 1

The first objective of this project is to use our floor-detection software to autonomously map the traversable area of the scene. A place is considered traversable if it is flat with a low enough angle with respect to gravity.

The outcome of this objective should be a 2D matrix representation of the world, where each cell (representing a small area in the world) is filled with a label in the set {traversable, not-traversable, unknown/unobserved}. This matrix should be built as an image from the OpenCV library (we've used OpenCV objects in the Hough Transform package) and published on an ros topic that can later be visualized with image_view:

```
rosrun image_view image_view image:=/image/topic/name
```

Because this project does not deal with localisation, this scene is providing a perfect localisation at all time within the TF framework (transformation from /bubbleRob to /world).

Also to display an opencv image, you can use the cv::imshow function. To publish it over ros, you will need to use the cv_bridge package. Both solutions are relatively easy to implement but require to check the documentation.

Project Objective 2

The second objective of this project is to detect cylinders out of the various objects present in the scene and report their locations in Rviz using a MarkerArray message (similar to the one published by the floor detection packages so far). Ideally, the detection should merge the cylinder parameters when they are observed multiple times.

Project Objective 3

In the third objective of this project, we will bring your expertise in probabilistic reasoning to bear on the terrain characterization problem. Revisit your floor mapper software, and model the traversability/non-traversability as a binary probabilistic variable that can be estimated by a Bayesian filter, with a confidence increasing with the number of times a given cell is observed.

Please pay attention to first writing the correct formulas, before coding a solution (a similar behaviour can be obtained by just adding +/-1 to the cell when it is observed with a given status).

Some of you noticed that long-range update could sometimes be imprecise. This can be modeled in your probabilistic model by changing the confidence of long-range observations.

If you want to read further or are interested in a more efficient representations, try to read about log-ratios and occupancy grids.

Implementation Recommendations

You should probably create two/three new packages (floor_plane_mapping, cylinder_detector) in which the development of these objectives will be implemented.

You can either modify one of the floor_plane packages to publish the plane parameters on a ROS topic (most reusable), or copy your floor_plane extraction software to the mapping package and modify it (slightly easier but will harder to maintain and reuse in later projects).

For the cylinder detection, the same applies.

Use TF to project the point clouds in the desired frame (you might need to read the TF tutorials).

If you just use the tasks and navigation scripts as provided, there will probably be holes in you map because the robot will not have looked everywhere. Adding new behaviours in the task framework is relatively easy and could be used to add for instance a scanning behaviour where the robot is rotating on the spot over 360 degrees to scan its surrounding. Check the documentation on Piazza.

If you wish, you can use the Point Cloud Library to implement some of the tasks instead of using your own packages. The overall time saving is not obvious.

Deliverables

Submit the deliverables by email to cedric.pradalier@georgiatech-metz.fr (don't forget to mention you group name). The deliverables list is as follows:

- A video of your system in action (you can use VLC to record your desktop as a video stream).
- A demonstration on Thursday October 12th, during the normal class hours. This demonstration should intend to “sell” the class the performance of your system. It should last approximately 10min and demonstrate your system autonomously mapping (traversable terrain and cylinders) a test environment. The demonstration may use small slide show if you need it better sell your system performance.