Sensors and Sensing
Lab 4: Simple Occupancy Mapping

Todor Stoyanov

October 12, 2018

1 Objectives and Lab Materials

The objective of this lab is to bring together some of the concepts about sensors and sensor data processing we have discussed during the course. In particular, you will implement an occupancy grid mapping node, use the LRF noise models you obtained in the previous lab, and integrate measurements at different locations, provided by the odometry module you developed in lab 2.

For this lab you are given:

- MakeBlock kit assembly, including a MegaPi board and two DC motors. (we refer to the MegaPi board as **arduino** from now on).
- Raspberry Pi with Ubuntu and the Arduino IDE pre-installed. (we refer to the Raspberry Pi as **pi** from now on).
- A YDLidar F4 laser range finder (LRF).
- A desktop PC with ROS kinetic installed (we refer to this computer as **desktop** from now on).

The following tasks will guide you through setting up your mapping system. In your report, describe the steps you have taken to solve each task, as well as the results you obtained and the methods you used to verify that the system functions properly. Explain what experiments you performed, how and why. Provide your C++ source codes as an attachment and demonstrate the final system to the lab assistant. If you think it is necessary, you may take pictures of the system in action, and/or include screenshots.

1.1 Preliminaries

Before starting with this lab please verify you have performed these preliminary steps:

- Make sure your robot is configured, the laser sensor is mounted, the odometry module is loaded onto the arduino, and that you have either ethernet or wireless connectivity to your robots.
- On the pi: make sure you still have the github code for lab3 into your workspace.
  `git clone https://github.com/tstoyanov/sensors_lab3_2018.git`

2 Task 1: Occupancy Mapping Module (15 points)

- Using the code from last lab as an example, create a new catkin package `my_occ_mapper`.
- Create a new node in your package, and create a subscriber to laser range finder messages.
- Provide as parameters to your node the map resolution, the map frame name, and the $x - y$ size of the map.
- Implement a method to allocate and de-allocate a byte or short int array for storing the occupancy map.
- Implement a method to update a single cell in the array with a new observation (observed empty or observed occupied).
- Implement a method to trace a single ray of a laser scanner and update cells along the ray. Use the noise model you obtained in the previous lab to decide which cells along the ray should be updated as occupied.
- Use the ray-tracing method to implement a map update procedure. For the first part of the lab, you can assume the laser is located in a fixed position relative to the map. You will, however, need to relax this assumption for task two so it might be a good idea to already at this stage design the function to allow for specifying the pose of the laser scanner.
- Link the map update procedure to the callback of the laser subscriber, so as to use new laser messages for updating your map.
- Implement a method to convert your occupancy map to a `nav_msgs/OccupancyGrid` message.
- Implement a method to publish the current map to a specified topic.
- Link the map publishing method to a timer so that your node publishes updated maps at a frequency of 0.5Hz.
- Test your implementation while keeping the robot stationary.

3 Task 2: Integrating Odometry for Pose Sensing (10 points)

- Modify your occupancy mapping module and add a subscriber for `geometry_msgs/PoseStamped` messages. Store the most recent pose of the robot obtained from odometry, as well as the timestamp of the most recent message.
- In the map update callback, check if the laser message received and the last odometry message are sufficiently close in time (e.g., less than half a second). If that is not the case, ignore the laser message.
- Use the odometry message to transform the latest laser scan to the pose specified by odometry. Make sure that the odometry pose is in the same frame id as the map. Do not forget to add the offset between the robot base and the laser frame into the transofrmation.
- Using the modified node try moving your robot to create a map of the environment. Move the robot slowly and only using a linear velocity initially.
- After this try rotating the robot in place. How does the map look?
- Drive along a figure 8 or a square and generate a map of the environment. What are the dominant sources of error?