Coding Problem Set 2 - Occupancy Grid Mapping

Joshua Mangelson

September 13, 2021

Purpose: The purpose of this lab is to give you hands on experience implementing and the ability to play around with occupancy grid mapping.

1 Setup and Initial Instructions

1.1 Pulling the Code

You will be able to access the code for each of the labs in this course via Git. For this lab specifically, you will be able to access it via the following link: `https://bitbucket.org/byu_mobile_robotic_systems/lab2-occupancy-grid-mapping`

1.2 Setting Up Your Environment

The programming language for this lab is Python. We assume you will be using Python3. The code has the following dependencies:

1. matplotlib - (this is a fundamental python graphing library)
2. numpy - (this is a fundamental python matrix library)
3. tqdm - (used to visualize progress)
4. pytest - (unit test runner)

On ubuntu, you should be able to install what you need via the following commands, run from the cloned repository:

```
sudo apt-get install python3
sudo apt-get install python3-pip
pip3 install -r requirements.txt
```

Alternatively, if you are using conda, activate the environment that you setup in Coding Problem Set 1 and then run the following from the cloned repository:

```
pip install -r requirements.txt
```

and throughout the rest of these instructions, if you are using conda, replace 'python3' in commands with 'python'

1.3 Code Overview

The code needed for this lab is found inside the **src/occupancy_grid_map.py** file. This file contains 2 classes and one function:

- **OccupancyGrid** - this class defines the implementation of an occupancy grid.
- **OccupancyGridMap** - this class uses the OccupancyGrid class to define the implemenation of an occupancy grid map.
- **main** - this function is the function that is run if you type the following command in a terminal:

```
python3 src/occupancy_grid_map.py
```

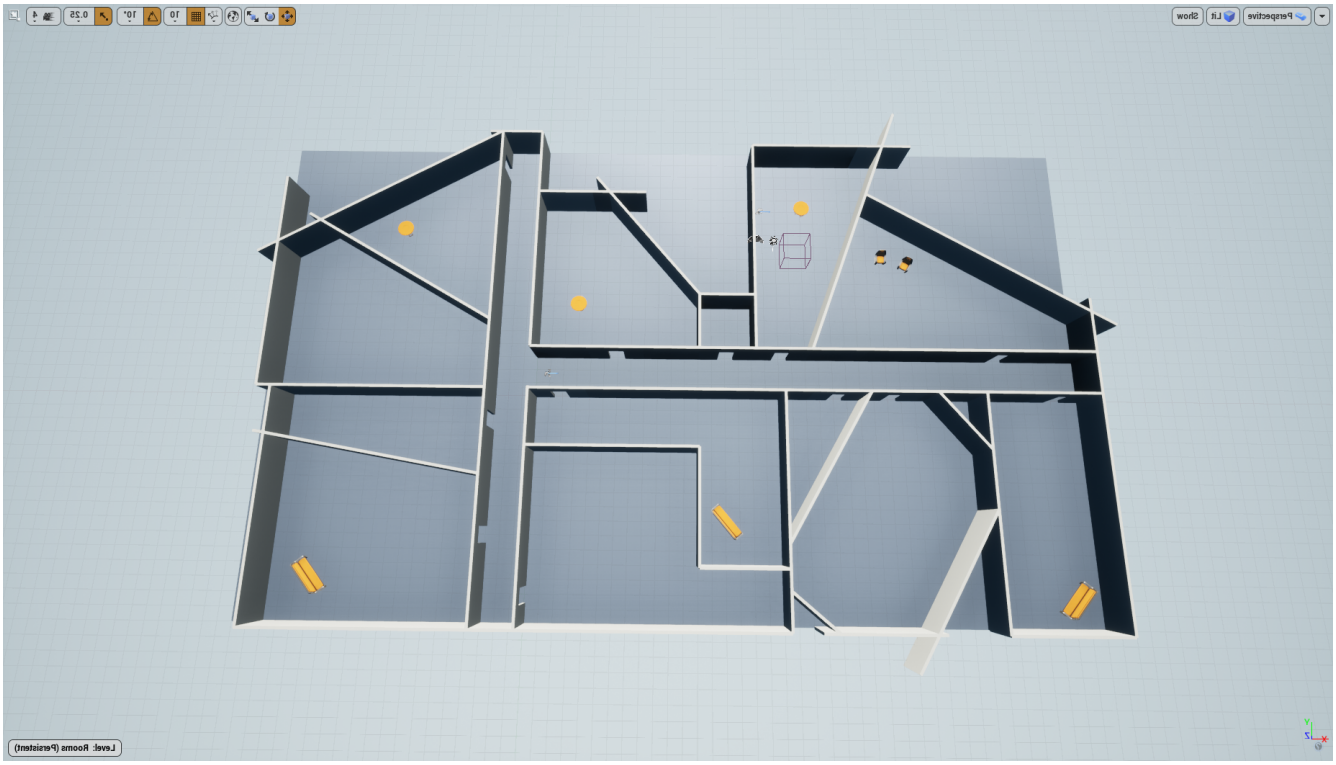

Figure 1: An overhead view of the building you will map.

2 Lab Overview

For this lab, you will implement the occupancy grid mapping algorithm and use it to build a map of a building from a stream of simulated data.

Try running the following command from the repository root directory:

```
python3 src/occupancy_grid_map.py
```

This command calls the **main** function which loops through all of the data stored in **rooms-dataset.p** and plots it to the screen. **Your goal in this lab is to process this stream of data to build a map of the building shown in Figure 1**.

To do this, you will need to finish implementing the occupancy grid mapping algorithm we discussed in class.

For more info on running the file, ie a list of the available options, run:

```
python3 src/occupancy_grid_map.py --help
```

2.1 Data Stream

To generate the data used in this lab, we simulated a robot with 20 laser range finders spread out at various angles. Each of these laser range finders is directed in a slightly different direction on the front of the robot. We then navigated the robot through a simulated building, entering each room one-by-one.

At each time step we recorded the current "ground-truth" pose of the robot as well as the range returned by each of the 20 laser range finders when the robot was at the recorded pose.

The data stored in **data/rooms-dataset.npz** includes the following:

- **angles** - a list of 20 angles (in degrees) representing the angles of the corresponding laser-range finders
- **x_t** - a list of lists of the form

$$[[x_0, y_0, \theta_0], [x_1, y_1, \theta_1], \cdots, [x_n, y_n, \theta_n]]$$

  where each internal list specifies the xy-position and θ-orientation of the robot.

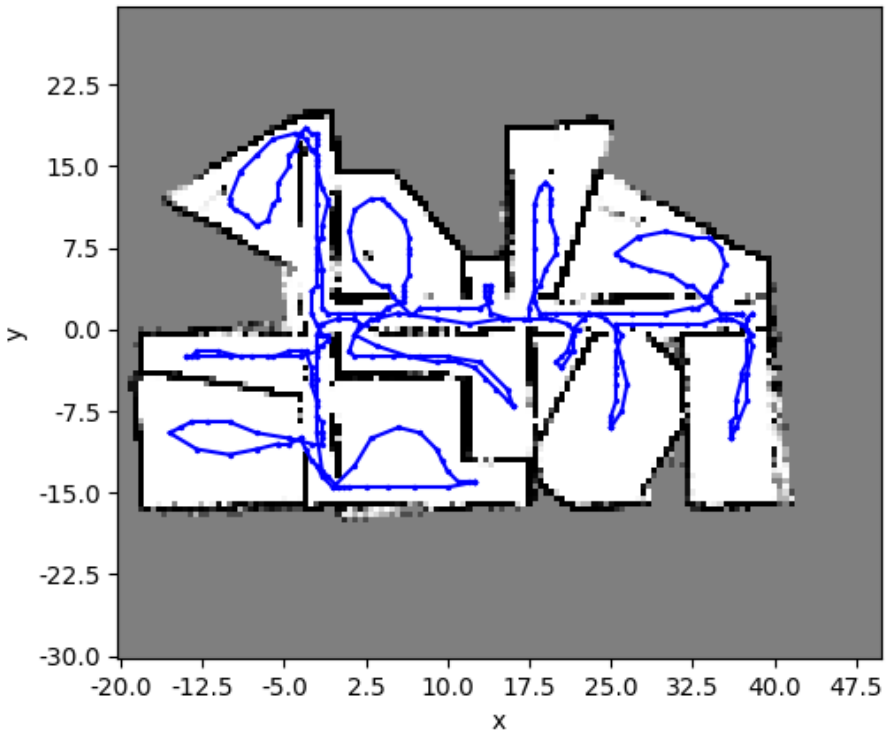

Figure 2: Example final occupancy grid map.

- **z_t** - a list of lists of the form

$$[[z_{0,0}, z_{0,1}, \cdots, z_{0,m}, \cdots, z_{0,19}], [z_{1,0}, z_{1,1}, \cdots, z_{1,m}, \cdots, z_{1,19}], \cdots, [z_{T,0}, z_{T,1}, \cdots, z_{T,m}, \cdots, z_{T,19}]]$$

  where each internal list specifies the set of 20 range measurements $\{z_{t,m}\}_{m=0}^{19}$ returned from the laser-range finders at time t.

2.2 Implementation

The classes in **src/occupancy_grid_map.py** outline the code that you will need to complete. I have completed several functions for you. Take a look through the code to see what they are.

You will need to complete the implementation of the following functions:

- **update_cell_with_meas_logodds**
- **laser_range_inverse_sensor_model**
- **find_cells_to_update_for_ray**
- **integrate_laser_range_ray**
- **main**

TODO: Implement the above functions to implement occupancy grid mapping. In the end, you should have a map of the above room that looks like the map in Figure 2.

TODO: Include a picture of your final map in your lab writeup.

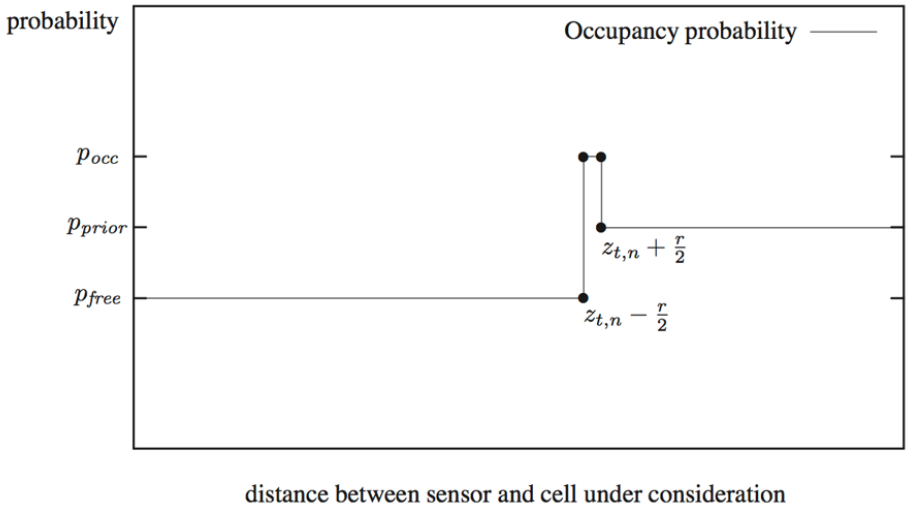

Figure 3: Laser Range Finder Sensor Model

2.3 Laser Range Finder Inverse Sensor Model

To implement the laser range inverse sensor model, implement a function that outputs the logodds update that should be added to a given cell. This should be the inverse of the function in Figure 3, with the output in logodds form instead of probabilities.

Select values for p_{free}, p_{occ}, and p_{prior} that seem likely to you.

TODO: After getting things working, play around with adjusting these values and running the live plotting. Include pictures in your lab writeup.

QUESTION: Why do these differences in the map occur?

2.4 A Note on Plotting

The **main** function steps through the full data stream and records data to plot every 10 time steps. If the **-p** option is set when the python file is ran, then the function will physically plot to the screen each time this data is recorded. Otherwise, if the **-p** option is not used, then the function will wait until all of the data has been processed before plotting it.

Plotting live is useful for debugging, but is much slower than waiting until the end to plot. You may need to switch between these as you complete the lab.

QUESTION: What about the way we handle occupancy grids contributes to this?

2.5 Testing

Included in the lab are a number of tests to ensure that everything has been implemented properly. To run them, when in the repo directory simply run

```
pytest
```

Note these tests are necessary, but not sufficient conditions, meaning if they fail, you've done something wrong, but if they pass, it's not a 100% guarantee that all your code is functioning properly.

TODO: Include the output of '`pytest -v -rN --tb=no --no-header`' in your writeup.

2.6 Wrap Up

TODO: Create a short 30 second video of you showing your finished map, explaining your understanding of the method, and explaining your answer to one of the questions in the writeup.

TODO: Upload as **separate** attachments to Learning Suite:

- Your video (as a .mp4 file)
- Your writeup (as a pdf file)
- A zipfile of your code (this should only contain a single file inside: **occupancy_grid_map.py**)

.