COMP 417: Assignment 1
Occupancy Grid Mapping

This assignment will focus on getting familiar with Player/Stage robotics environment, and then using it to write mapping programs. Player provides an abstraction layer for different kind of robots and sensors. Stage provides a simulated world and simulated sensors, using which a program written for player can be run as a simulation. See the player stage website[1] for more information, documentation, tutorials, and installation instructions.

The file `randomwalk.cc`, provided with this assignment demonstrates the use of some of the functions which might be useful in doing this assignment. Simply run `make` to compile the code. Running the program involves two steps. First, run player with the configuration file for one of the worlds, which have been provided.

```
$ player triangle.cfg
```

This will start the simulation environemnt. Now, open another terminal window and run your program:

```
$ make
$ ./randomwalk
```

Q1. Simple Mapping (30pts)

Write a program called `simpleMap`, which builds the map in the following way: Start with an empty map represented by an array of integers, each representing a $0.05m \times 0.5m$ cell on the map. Each cell C_i holds a binary value 1 or 0, indicating the occupancy of that cell. Mark a cell as occupied if falls in the middle of a sonar cone's arc. Generate the path taken by the robot using a simple random walk. You are encouraged to use the code from randomwalk.cc. Run the program for the duration you see fit for each map.

Q2. Occupancy Grids with Bayesian Updates(55pts)

In this section, we will explore the occupancy grid algorithm, which improves upon the above algorithm by taking into account a sensor model.

[1] http://playerstage.sourceforge.net

We would like to model the occupancy probability $\mathbb{P}\{C_i = \text{occ}\} = \mathbb{P}\{C_i\}$, and the probability that the cell is not empty $\mathbb{P}\{C_i = \text{empty}\} = 1 - \mathbb{P}\{C_i\}$, for each cell C_i in the world. Normally this is implemented as log odds ratio.

Let $\mathbf{z}^t = \{z_1, \cdots, z_t\}$ be all the sonar measurements, and $\mathbf{x}^t = \{x_1, \cdots, x_t\}$ be the robot pose till time t. Our goal is to then compute the log odds ratio at time t:

$$l_{t,i} = \log \frac{\mathbb{P}\{C_i|\mathbf{z}^t, \mathbf{x}^t\}}{1 - \mathbb{P}\{C_i|\mathbf{z}^t, \mathbf{x}^t\}}.. \tag{1}$$

Using Bayes filtering, we can recursively compute log odds ratio from the estimates in the previous step using the following expression:

$$l_{t+1,i} = l_{t,i} + \log \frac{\mathbb{P}\{C_i|z_t, x_t\}}{1 - \mathbb{P}\{C_i|z_t, x_t\}} - l_0. \tag{2}$$

Here l_0 is a constant and represents the prior occupancy as a log odds ratio.

$$l_0 = \log \frac{\mathbb{P}\{C_i\}}{1 - \mathbb{P}\{C_i\}} \tag{3}$$

Without any information, we can assume that a cell is equally likely to be occupied or be empty, and set $l_0 = 0$.

The function $f_{ism}(C_i, z_t, x_t) = \mathbb{P}\{C_i|z_t, x_t\} = \mathbb{P}\{C_i = \text{occ}|z_t, x_t\}$ is sometimes referred to as the *inverse sensor model*. It returns the probability that a cell is occupied, given a robot pose x_t, and sensor reading z_t. We would like this function to assign cells within the sensor cone a low occupancy probability, while the cells on the arc of the cone should be assigned a high probability. A simple way to model this function is to use some constant values $p_{\text{occ}} > 0.5$ and $p_{\text{free}} < 0.5$ for these two cases. All cells outside the cone should return 0.5, corresponding to unknown, or equally likely probability.

Tasks:

1. What is the range of value which l_i can take? (5pts)
2. Given l_i, how can you compute $\mathbb{P}\{C_i\}$? (5pts)
3. Derive the update equation 2 starting from the equation 1. (10pts)
4. Implement a program called `occMap`, which uses Bayes filtering to compute log odds ratio l_i for each cell. The program should take an argument indicating how long (in seconds) to run the program in simulation time. The path of the robot should be generated using a random walk. Use cells of size $0.05m \times 0.05m$. Use a simple piecewise function for the inverse sensor model. Choose appropriate values for p_{occ} and p_{free}. Show the log odds ratio for the worlds provided. (30pts)
5. Given log odds ratio l_i for each cell, how can we come up with a binary map of the world, where every cell is either occupied or empty? Generate and show the binary maps for the worlds provided. (5pts).

Q3: Mapping with Real Sonar Data (15pts)

The file `slothSonarData.dat` contains sonar data from a real sonar sensors equipped robot. Each line corresponds to a sonar reading, and the columns are (in order): robotX(in cm), robotY(in cm), robotθ(in degrees), sonarRange(in cm), sonarθ(in degrees).

Tasks:

1. Modify the `simpleMap` and `occMap` programs to work with this data, and show the resulting maps. (5pts)
2. Discuss the differences between player stage simulation data, and the real data. What artifacts do you see in the real data, which are not visible in the simulated data. (5pts)
3. Compare and contrast the maps generated by `occMap` and `simpleMap` programs. (5pts)

Note/Tips:

- There are 16 sonars mounted on the simulated player-stage robot. The first reading corresponds to the sonar on the left side of the robot. Subsequent readings are in clockwise direction.
- Probably the simplest way to output the result is by writing maps to an image file in PGM file format. See *http://netpbm.sourceforge.net/doc/pgm.html* for specifications. PGM files can be read by programs like gimp and be converted to other image formats. However, feel free to use any other method you may seem fit to present the maps.
- Submit uncompiled code, and a PDF file containing all the answers and generated maps proving that your code works.
- Run the program for a time you see suitable for the worlds.