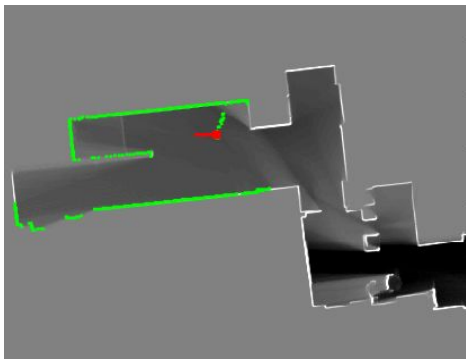
Programming Assignment Week 3

Occupancy Grid Mapping

1. Introduction

In this assignment you will be implementing the Occupancy Grid Mapping algorithm for a 2D floor map. You will incorporate range sensor readings and known poses at each of the measurement times in order to build a map.The figure below depicts an example of range measurements (green dots) from a mobile robot (red dot with an indicated heading) and an intermediate occupancy grid map.

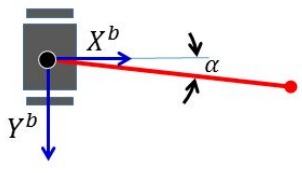


Occupancy Grid Mapping Example

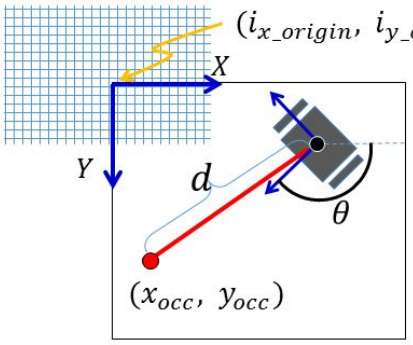
2. Prerequisites

Understanding Coordinate Transformations [Lecture 3-2-3]

1. [**Map coordinate frame *vs*. body coordinate frame**]
   The sensor readings are measured in the robot's body coordinate frame. In order to orient them into the world frame, one must rotate the lidar readings using the yaw estimate and translate the reading with the position estimate.



Definition of Body Frame



Robot on Map Frame

Distance measurement: d (in α)
Known state: (x, y, θ)

'Real' location

$$\begin{bmatrix} x_{occ} \\ y_{occ} \end{bmatrix} = \begin{bmatrix} d\cos(\theta+\alpha) \\ -d\sin(\theta+\alpha) \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

Grid index

$$\begin{bmatrix} i_{x,occ} \\ i_{y,occ} \end{bmatrix} = ceil\left(resol * \begin{bmatrix} x_{occ} \\ y_{occ} \end{bmatrix}\right) + \begin{bmatrix} i_{x\_origin} \\ i_{y\_origin} \end{bmatrix}$$

2. [**Map as array *vs*. map coordinate frame**]
   The final map format should be a 2D array, which has the index (1,1) for the top-left-corner. We will keep the orientations of the map frame the same as the array map, with the positive x-axis towards the right and the positive y-axis pointed downward. However, the origin (0,0) of the metric map is required as the initial pose of the robot, which you need to locate on a corresponding point of the map array. Note that the map as an array can be accessed only via integer indices.

   *Note: In order to access a point on the map array, you will use the tuple (y,x), which corresponds to the MATLAB convention to handle an image data.

Important Parameters

1. **Map resolution** [Lecture 3-2-3]
   The resolution determines how finely your map will be discretized. Specifically, in our assignment, we will define this parameter as the number of cells to subdivide 1 meter. For example, if you take 10 for this parameter, then each grid cell will represent a 0.1x0.1 (m^2) area.

   *Note: this is the inverse of *r* which appears on the slide pp.6-11 of Lecture 3-2-3.

2. **Map size and origin** [Lecture 3-2-3]
   Remember that the extent we can build a map is finite, but it should be large enough to cover all our range measurements. Although the map size needs to be flexibly adjustable in the ideal case, we will keep it at a fixed size in this assignment for simplicity. In addition, you will need to specify the origin cell for the starting point for mapping.
   In the assignment, you will not need to worry about having an invalid range measurement that goes far out of the map. A reasonably big map size and a proper origin are given as parameters for both the practice data and the test data, and you can focus on log-odds updates.

3. Log-odd update parameters [Lecture 3-2-2]
   log_odd_occ: This parameter is for updating the occupied cell measurements.
   log_odd_free: This parameter is for updating the free empty cell measurements.
   log_odd_max: The maximum value for log_odd of your map.
   log_odd_min: The minimum value for log_odd of your map.

   *Note: Having max/min values of log_odd prevents your map from becoming too certain about the occupancy of a given cell.

3. Instructions

Algorithm Implementation

1. You will complete a function that take the parameters, sensor data, and robot pose data as input, and returns a 2D occupancy grid map. The signature of the function is given below.

```
function M = occGridMapping(ranges, angles, pose, params )
```

2. Implement your function to take the parameters as input and build a map accordingly. Please DO NOT hardcode parameters inside the function.
3. In order to apply the body-to-map coordinate transformation, you may use the input argument *pose*: *pose*(1:2) are the x,y location on the map and *pose*(3) is the heading angle of the robot.
4. Your function should return a 2D array map of the right size specified by *params*.
5. *practice.mat* is the data provided for developing your algorithm. You may take some part of the whole data, which has about 4000 instances of lidar scans.
6. A test script *example_lidar.m* is provided to help visualize the range measurements.
7. A test script *example_test.m* is provided to help set the parameters and visualize your result.

Evaluation and Submission

To submit your result to our server, you need to run the script runeval in your MATLAB command window. Please specify the path where the test data file is located. A script will then evaluate your occGridMapping and generate two output files, SubmissionMap1.mat and SubmissionMap2.mat, to be uploaded to the Coursera web UI. You may submit your result multiple times, and we will count only the highest score towards your grade. The score reflects how many cells are built correctly in terms of occupancy.