ESE 650 - Learning in Robotics
PROJECT 4
Simultaneous Localisation and Mapping
Anushree Singh

INTRODUCTION:

Creating a 2D map and also simultaneously localizing the robot pose was the aim of this project. The data set provided to us consisted of Wheel Encoder data for odometry, IMU data, and Hokuyo LIDAR data and also kinnect data for coloring the ground plane. The algorithm that I followed to do SLAM is explained below.

OCCUPANCY GRID MAPPING:

OGM is used to maintain the map of the world. I used a grid resolution of 10 cm and initialized the map to have all ones. Each grid space is either free or occupied depending upon what the robot observes at everytime stamp. Depending on the robot's pose and the LIDAR range values the value at the respective grid is given. I am incrementing the grid space by one, everytime the LIDAR scan hits an obstacle. I have neglected any value greater than 5 meters as this might be erroneous or a ceiling while on a ramp and also ignored values less than 50 cm as these might correspond to floor while coming down the ramp.

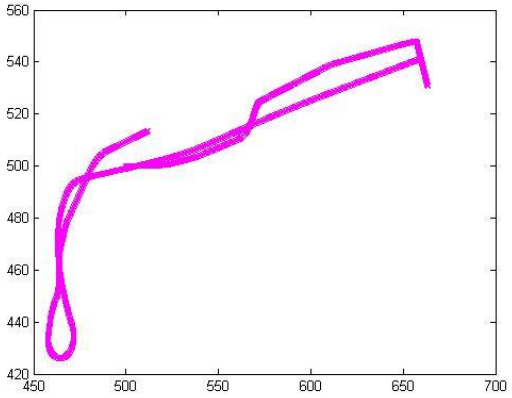
MOTION MODEL:

Our robot is a differential drive robot, with four wheels. I calculated the forward motion of the front axle and the back axle and calculated the approximate centre of the robots as the average of the two. These calculations were done using wheel encoders (resolution: 180) and IMU's gyro readings using the forward kinematics of differential drive robots. The IMU and encoder timestamps were synchronised. Also removed bias from the IMU data. And corrected the readings by taking care of the sensitivity.

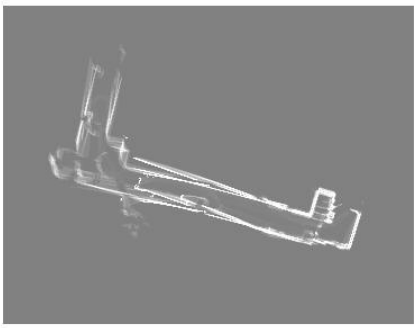
Front Wheel motion
```
Fm = (deltaR + deltaL)/2;
Fm = Fm*2*pi*R/res;
```

Rear wheel motion
```
RFm = (RdeltaR + RdeltaL)/2;
RFm = RFm*2*pi*R/res;
```

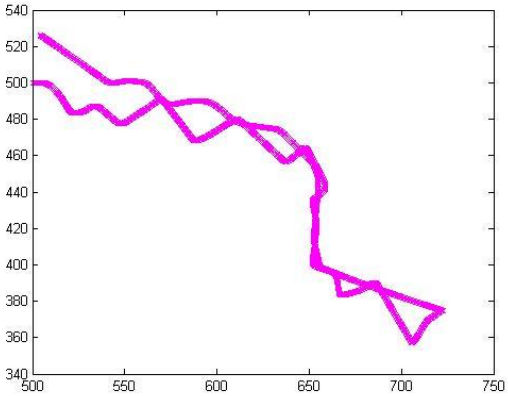
Delta is the encoder count, res is the resolution, R is the wheel radius.
```
xR_next = xR_current + RFm*cos(theta);
yR_next = yR_current + RFm*sin(theta);
```



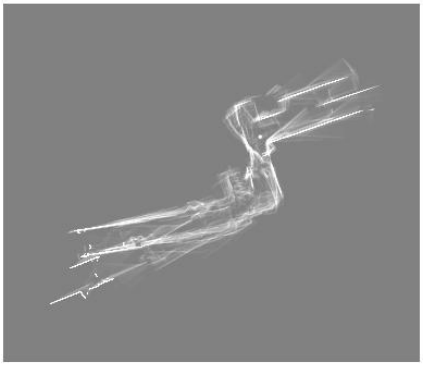
Robot Path for Test set 1



Dead Reckoning for test Set 1



Robot Path for Test set 2



Dead Reckoning for test set 2

SLAM Particle filter:

To combine the pose localization with the maps being generated I implemented a particle filter with 200 particles. I aslo tried with 50 and 100 particles however my optimal. As the IMU and encoder data has errors I incorporated a distribution (normal distribution) in accordance with the number of particles chosen for the particle filter.

The noise models for gyro and encoder are as follows:

```
encoderNoiseX =
 deltax.*unifrnd(-2,2,[150,1]);

encoderNoiseY =
deltay.*unifrnd(-2,2,[150,1]);

gyroNoise =
deltaTheta.*unifrnd(-3.5,3.5,[150,1]);
```

The jitter to be in the range of -2 times encoder change to 2 times encoder change, and -3 times gyro change and 3 times gyro change was optimally chosen after a lot of tweaking.

The initial position was first moved/ updated using the motion model. Then generated 200 jittered particles using the noise model mentioned above. For each of these particles a pseudo map was generated in order to weight these particles effectively. The particels that created LIDAR hits more approximate to the previous hits were given more weight and vice versa. From these weights the particle with the maximum weight was used to create a new map which would then be used in the next iteration for comparision. For Weighting I also implemented the approach to calculate the map correlation and then shift the data point (jitter) by necessary amount, but it made the code very slow and hence I did not include it in my final implementation of SLAM. For

Transformation of the LIDAR points to the map grid iused the following equations.

thetaE = theta_robot + angles_LIDAR;

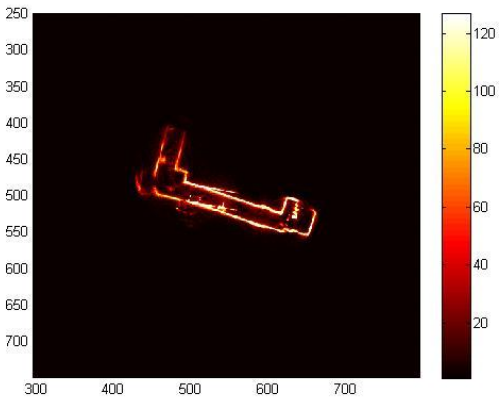
xgrid = round(xRobot + range_LIDAR.*cos(thetaE)*10);

ygrid = round(yRobot + range_LIDAR.*cos(thetaE)*10);

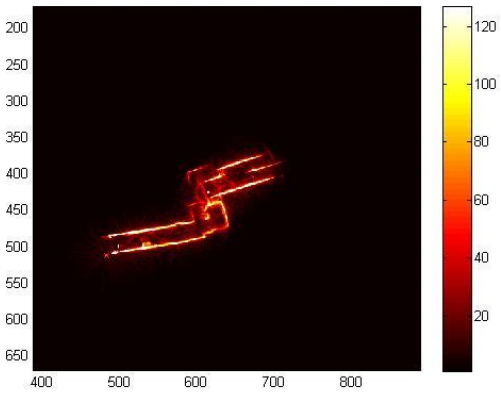
In order to maintain diversity in the particles I resampled the particles at every time step. The motion model was applied on these 200 particles in the next iteration. And this recurssive procedure was done for the number of time stamps.

RESULTS:

Some of the results of my implementation of SLAM are as follows.



Map for test set 1



Map for Test set 2

FUTURE WORK:

Ground plane colouring using the kinnect data in the next step. Also implementing the weighting technique for the particle filter in a more sophisticated way.