Sam Willenson (shw58)
ECE 5242

Project 3 Writeup (SLAM)

Introduction

In this assignment, we focus on mapping and localization of a robot in an indoor environment. The information we are given is data from wheel encoders and range sensors. Using this we integrate the wheel odometry information with a 2D LIDAR sensor, to build a 2D occupancy grid map of the walls and obstacles of our given environment.

Problem Formulation

How do you create a map of an environment given odometry and LIDAR data?

Technical Approach

We are given 2 scripts as starter code, test_load_data, and load_data which load our LIDAR data in the form of arrays consisting of t, pose, res, rpy, scan for each time step as well as IMU and encoder data.

For the odometry only part, using load_data, I start to build my slam algorithm by first getting my IMU data to give me acceleration and gyroscopic values in the x, y, and z directions, as well as the time steps. Next I use get_encoder from load_data to get my values for the speed of each of the 4 wheels on the robot. I average both the right wheels and both left wheels separately to essentially model this as a 2 wheeled robot, simplifying subsequent equations. I also average the (averaged) left and right values to get an estimate for my forward distance per time step. The theta values are found by taking the difference between the Left and Right speeds, and dividing this by the width of the robot wheel base. Because of wheel slippage, the width actually needs to be set to a larger value than the true width of the robot. This value was found through trial and error to be a width = 75 cm.

$$\Delta x = \frac{e_L + e_R}{2} \cos\theta$$

$$\Delta y = \frac{e_L + e_R}{2} \sin\theta$$

Calculate angular odometry

$$\Delta\theta = \frac{e_R - e_L}{w}$$

Next I iterate through the encoder data to get my dx and dy values for distance traveled using my Forward value, as well as the cosine and sine of my theta value for each time step, and start building a sum of x and y distance travelled to place my robot on the map. Finally I plot the LIDAR data at every 100 time steps to visualize the data accordingly.

To start implementing my SLAM algorithm, I wrote many helper functions. One to compute weights, resample particles based on these weights, create a grid map, update this occupancy grid, and update my particles as to where they are. Using these functions, I create an occupancy grid map, initialize my position arrays, and start iterating over my encoder data. Within this new SLAM code, my loop will contain many robots (or particles). I update these particles and add jitter to them to account for not 100% accurate measurements from the real world data. When my weights start to be dominated by too little of these particles, I will resample them u.

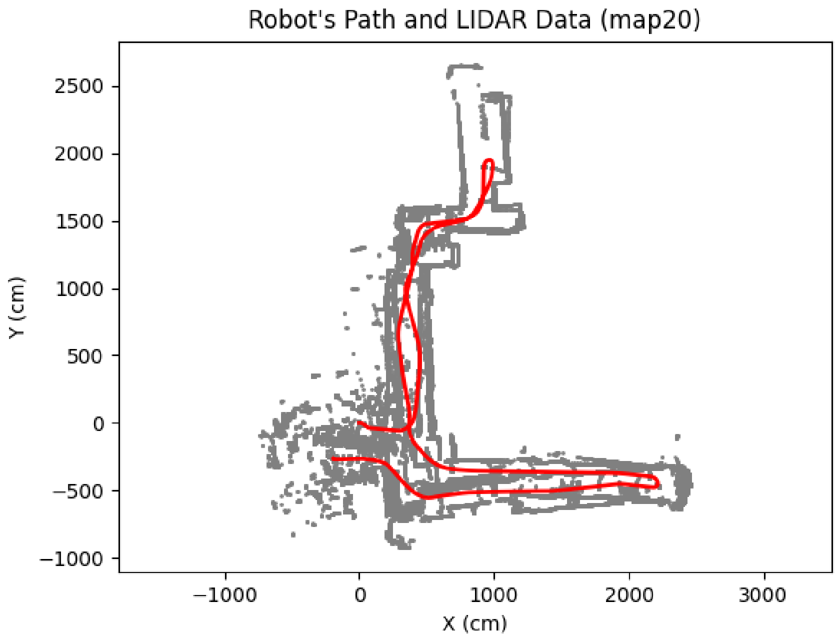
Here are a few equations I used to implement this SLAM algorithm.

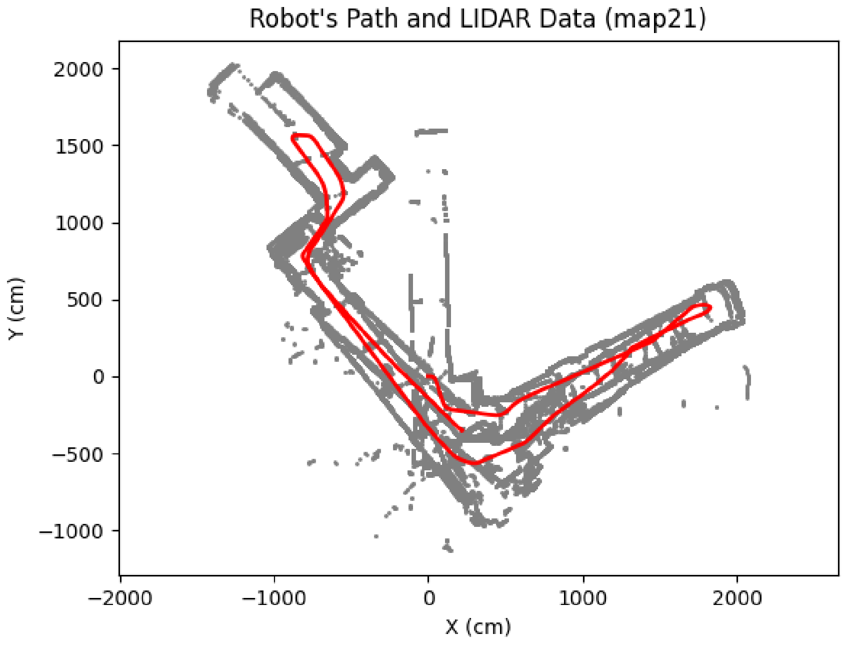
$$n_{effective} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2} \qquad w_i' = w_i \cdot corr(p_i)$$

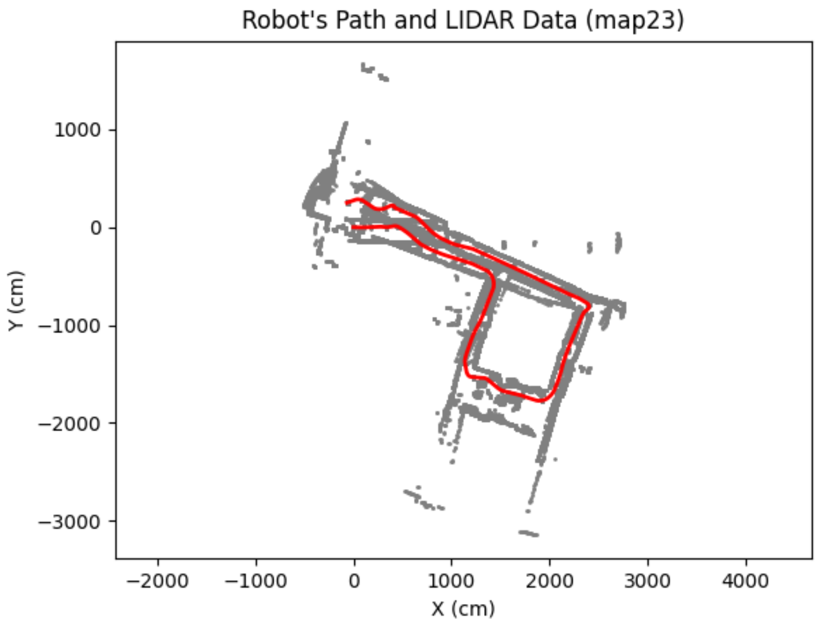
My correlation function was based on Pwall = exp(z) / (1 + exp(z)), where z is my log odds ratio from -10 to +10 that increases each grid for each detected hit and decreases for each detected pass through. The function increases by 0.9 for hits and decreases by 0.1 for misses.

Results on Training Data

Below is an image of my results for the mapped out region of my environment found from my encoder data for each training map.

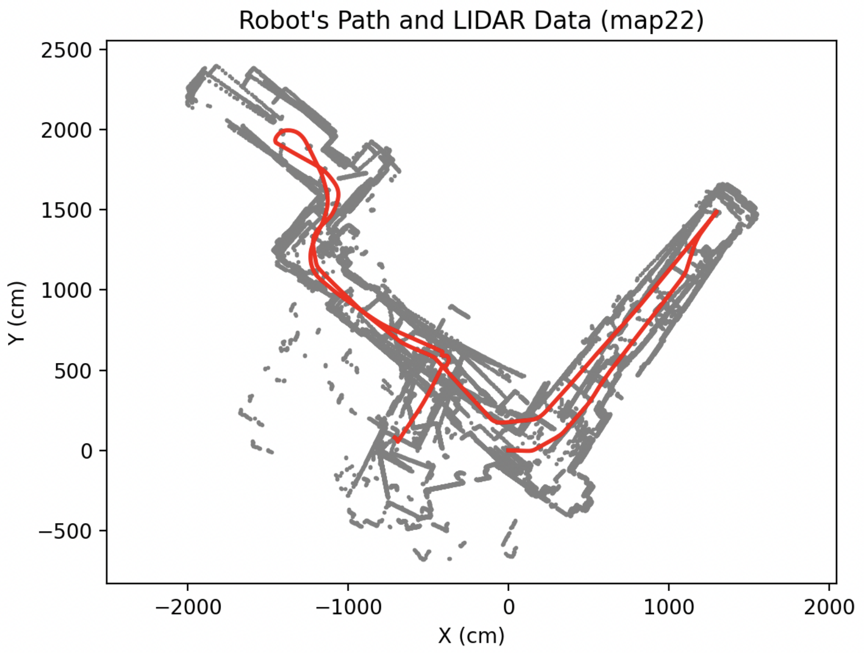

Robot's Path and LIDAR Data (map20)
Y (cm)
X (cm)
2500
2000
1500
1000
500
0
−500
−1000
−1000
0
1000
2000
3000


Robot's Path and LIDAR Data (map21)
Y (cm)
X (cm)
2000
1500
1000
500
0
−500
−1000
−2000
−1000
0
1000
2000

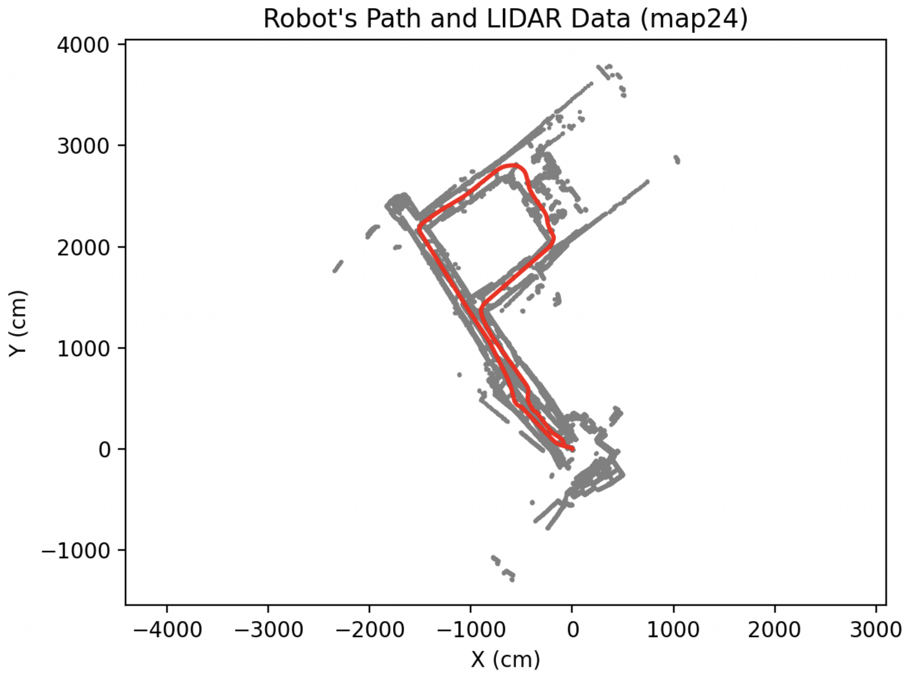

Robot's Path and LIDAR Data (map23)
Y (cm)
X (cm)
1000
0
−1000
−2000
−3000
−2000
−1000
0
1000
2000
3000
4000

Results on Test Data

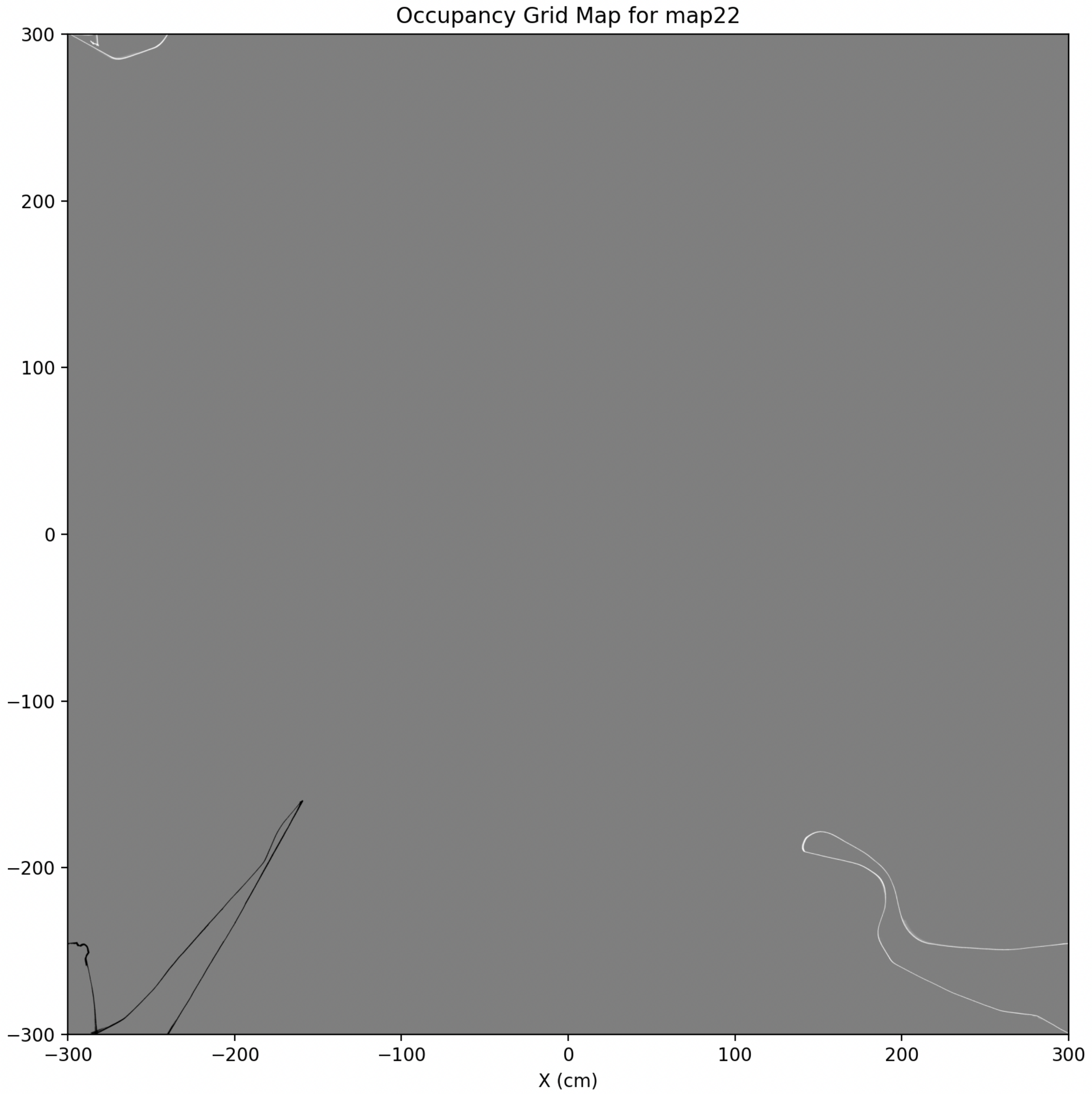
Odometry Only:

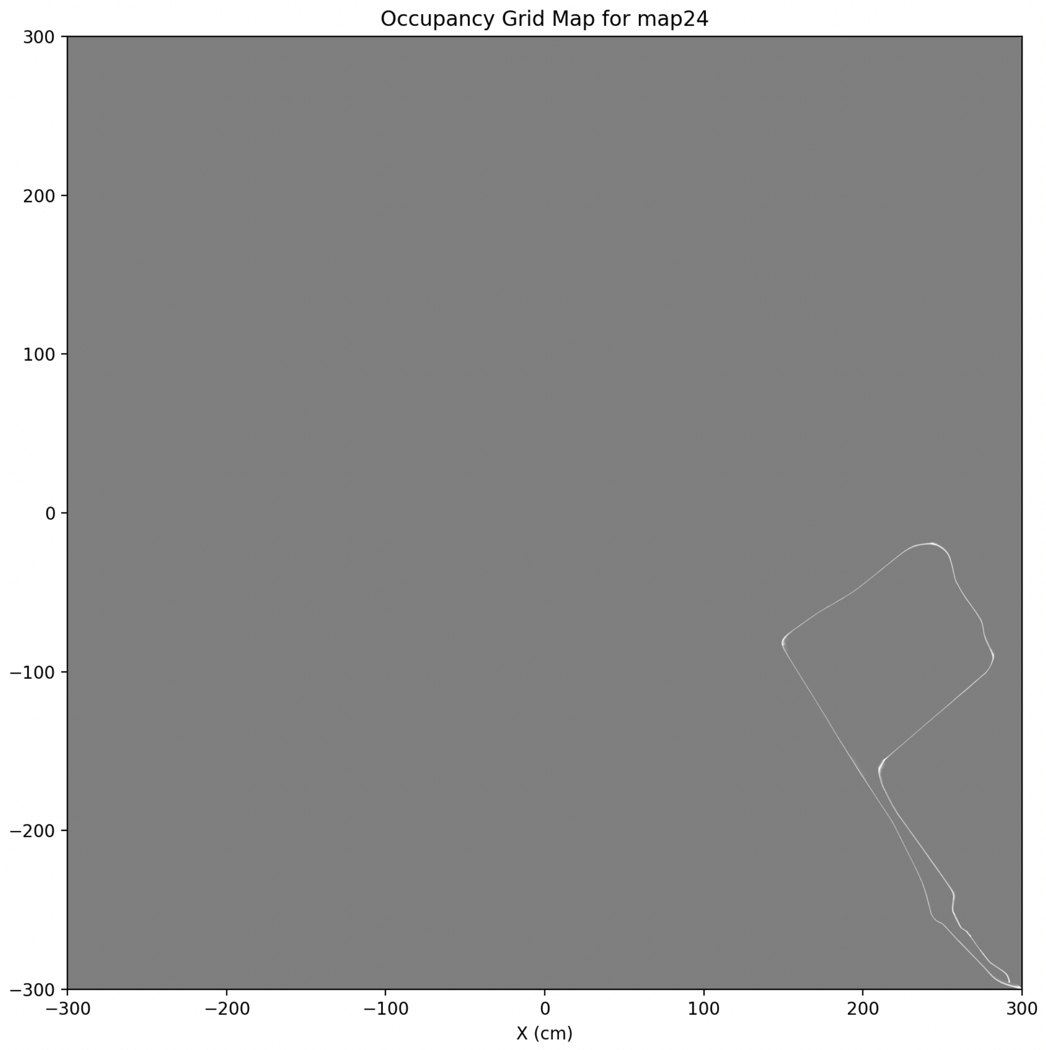


The 22 map dataset was the only set I had to tune my wheel base for. I used wheelbase = 69cm for this map and for all other datasets used wheelbase = 75.



SLAM:


Occupancy Grid Map for map22
300
200
100
0
−100
−200
−300
−300
−200
−100
0
100
200
300
X (cm)



Discussion

Overall I can not deem this venture a success. While my odometry only code does provide successful results, my SLAM code is outputting clearly bad results. My map for the datanumber 22 is producing something that does not look nothing like maps and I clearly still have some debugging to do. My map for datanumber 24 looks like it is just the robot path from my odometry part, so I am confused with where my code goes wrong. My SLAM maps should be more accurate than my pure odometry maps but instead they provide nothing of substance. I believe my error lies somewhere in how I am accessing the Cython MapUtils_fclad.pyx code. I also believe I have an error in my weights calculations and resampling but it is hard to debug when my map provides nothing useful.

Sam Willenson (shw58)
ECE 5242

Project 3 Writeup (Odometry Only)

Introduction

In this assignment, we focus on mapping and localization of a robot in an indoor environment. The information we are given is data from wheel encoders and range sensors. Using this we integrate the wheel odometry information with a 2D LIDAR sensor, to build a 2D occupancy grid map of the walls and obstacles of our given environment.

Problem Formulation

How do you create a map of an environment given odometry and LIDAR data?

Technical Approach

We are given 2 scripts as starter code, test_load_data, and load_data which load our LIDAR data in the form of arrays consisting of t, pose, res, rpy, scan for each time step as well as IMU and encoder data.

For the odometry only part, using load_data, I start to build my slam algorithm by first getting my IMU data to give me acceleration and gyroscopic values in the x, y, and z directions, as well as the time steps. Next I use get_encoder from load_data to get my values for the speed of each of the 4 wheels on the robot. I average both the right wheels and both left wheels separately to essentially model this as a 2 wheeled robot, simplifying subsequent equations. I also average the (averaged) left and right values to get an estimate for my forward distance per time step. The theta values are found by taking the difference between the Left and Right speeds, and dividing this by the width of the robot wheel base. Because of wheel slippage, the width actually needs to be set to a larger value than the true width of the robot. This value was found through trial and error to be a width = 75 cm.

$$\Delta x = \frac{e_L + e_R}{2}\cos\theta$$

$$\Delta y = \frac{e_L + e_R}{2}\sin\theta$$

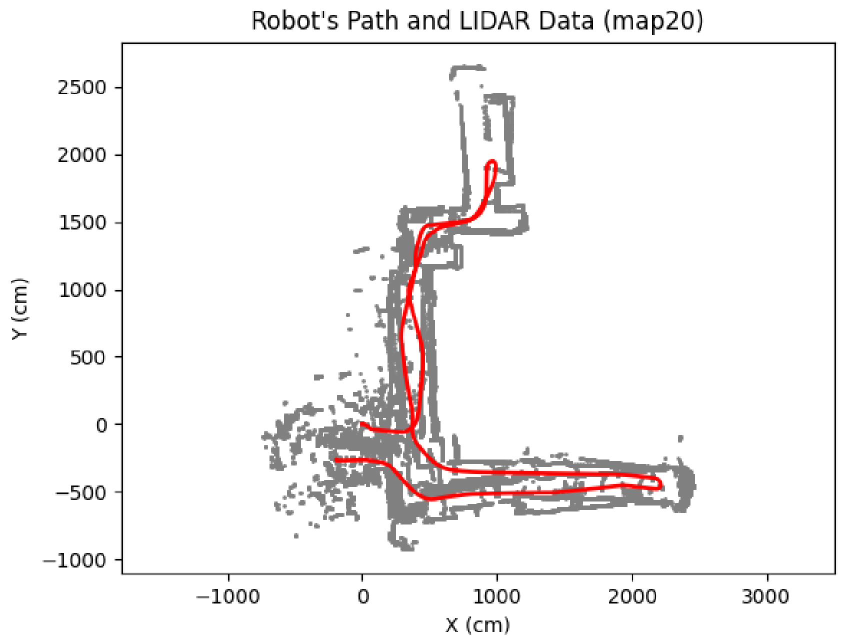
Calculate angular odometry

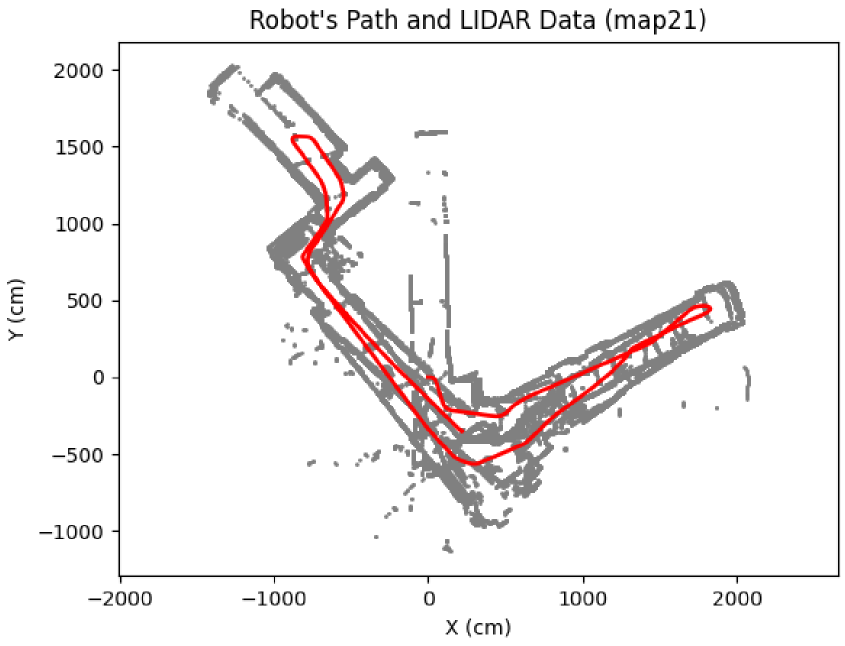
$$\Delta\theta = \frac{e_R - e_L}{w}$$

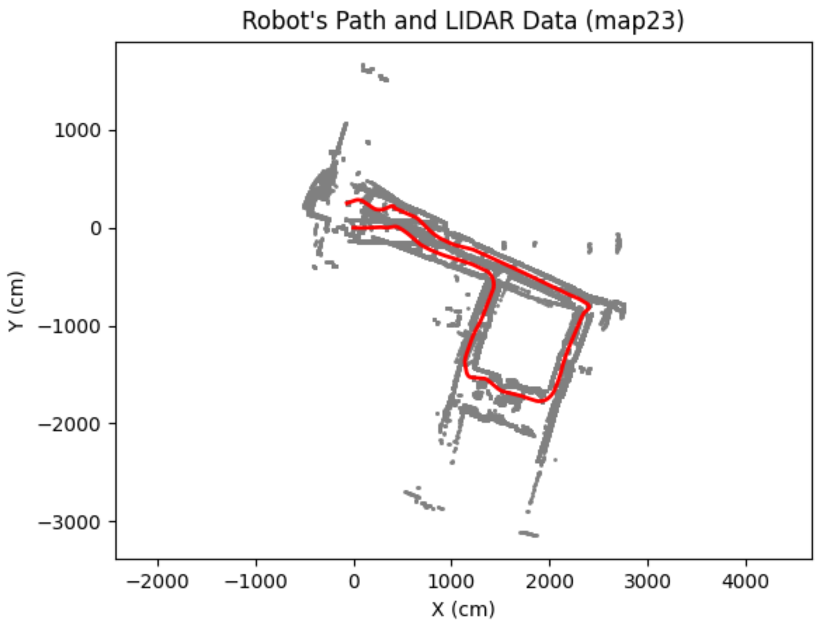
Next I iterate through the encoder data to get my dx and dy values for distance traveled using my Forward value, as well as the cosine and sine of my theta value for each time step, and start building a sum of x and y distance travelled to place my robot on the map. Finally I plot the LIDAR data at every 100 time steps to visualize the data accordingly.

Results on Training Data

Below is an image of my results for the mapped out region of my environment found from my encoder data for each training map.







Discussion

Overall I can deem this venture a success. The robot path matches the LIDAR data to a reasonable degree of accuracy, and the LIDAR data forms shapes that resemble walls to a reasonable degree, both of these things lead me to believe my code is giving me correct results for the robot path and map. There is a little bit of error with some outlier points from the LIDAR data visualization, but since this is odometry only that is within my expected error. The full SLAM code should improve upon these maps and clean up my outliers.