ECE5242 - INTELLIGENT AUTONOMOUS SYSTEMS

PROJECT 3 - SLAM

APRIL 10, 2020

EVA ESTEBAN
CORNELL TECH

Abstract

The aim of this report is to provide an in-depth analysis of the different algorithms developed to allow a mobile robot to perform Simultaneous Localization and Mapping (SLAM) in an unknown environment. This document describes the approaches implemented, the results obtained and the performance of the algorithms.

Table of Contents

1 Introduction

In the field of robotics, Simultaneous Localization and Mapping (SLAM) is the problem of constructing and constantly updating a map of an environment that is unknown to the robot while at the same time keeping track of the robot's location in the map [1].

In this project, algorithms were developed in order to implement the SLAM technique in an indoor environment. For this, odometry, inertial and range measurements that had been previously collected using a mobile robot were used. The sensors the robot used to collect these measurements include wheel encoders, a Light Detection and Ranging sensor (LIDAR), and an Inertial Measurement Unit (IMU).

The project can be divided into two main parts. The first part consisted of tracing the path of the robot using pure wheel odometry measurements, and creating a 2D map of the environment by using the LIDAR range readings. The outcome of this part of the project is a 2D map in which the motion of the robot can be visualized. The second part consisted of implementing the complete SLAM technique by simultaneously localizing the robot and constructing the surrounding environment.

The different sections of the project were tested by using both train-test splits of the training data and real test data. The techniques implemented together with the results obtained can be found in sections 2 to 5 of this document.

2 Problem Formulation

The problem tackled in this project is the application of the SLAM technique by a robot within an unknown environment. Given a robot that is put in an initially unknown environment, this consists of the robot estimating its pose while at the same time constructing a map of the environment. The data available to complete this challenge consists of measurements from wheel encoders, a LIDAR, and an IMU.

3 Technical Approach

3.1 First Part of the Project

The first step consisted of reading the encoder data and estimating the path of the robot using dead reckoning. Dead reckoning in robotics is a navigation process that consists of estimating a robot's current position based on the previous position and advancing that position based on estimates over time [2]. The encoder data measures the ticks of each wheel. There are four encoders, one for each wheel: Front Right (FR), Front Left (FL), Rear Right (RR), Rear Left (RL). Next, the distance travelled by a wheel for each tick was calculated. The diameter of each wheel is 254 millimeters. Therefore, the radius of each wheel is 127 millimeters. The longitude of each wheel is $2 * \pi * r = 2 * \pi * 127 = 797.964534012$ millimeters. Each resolution of the wheel corresponds to 360 counts. Therefore, each count the distance travelled is $\frac{797.964534012}{360} = 2.21656815003$ millimeters. According to the datasheet, a forward motion will add a positive value to the counter, therefore a backwards motion will substract it. Regarding the width of the robot, it was initially set to a value of approximately 376 millimeters. However, in order to account for skid steering, higher widths were tested and a final width of 733 millimeters was selected as the robot width giving the most accurate path.

Using these values, the angle θ measured counter-clockwise from the horizontal axis, as well as the X and Y position of the robot at each stage were calculated by applying the mathematical relations: $\theta = \frac{d_R - d_L}{width}$, $X = \frac{d_L + d_R}{2} * cos(\theta)$ and $Y = \frac{d_L + d_R}{2} * sin(\theta)$, where d_L is the distance travelled by the left wheel and d_R is the distance travelled by the right wheel. This is suitable because the position change between measurements is small and allows for circular motion approximation. The full mathematical calculations can be found on the **epe26_project3.ipynb** file. The function that performs this functionality is **get_path()**.

The next step in the project consisted of estimating the 2D map of the environment using LIDAR measurements. The first step consisted of matching the timestamps for the encoder and LIDAR data. For each encoder timestamp, the closest LIDAR timestamp and its corresponding measurements were chosen. The code that performs this functionality can be found in the functions **find_closest()** and **combine_timestamps()**.

The next step consisted of creating a 30x30 meters probabilistic map with a resolution of 0.05, known as a 2D occupancy grid. The meters were converted to cells. Each cell contains a value which represents the log odds of the probability of the cell being occupied vs. the probability of the cell being free. Once the cells had been initialized to zero which indicates equal probabilty of being occupated or free, the next step consisted of iterating through each entry of the encoder and LIDAR data. At every timestamp, the log odds of the cells being hit by a LIDAR ray was increased by 0.5, and the log odds of the cells in the trajectory of the LIDAR ray that were missed are decreased by a value of 0.01. An upper bound limit of 10 and a lower bound limit of -10 were imposed in order to prevent the probabilities from reaching ∞ or $-\infty$. Once the map had been updated, it was converted to grayscale and displayed and saved as an image. The trajectory of the robot in the map was also displayed in the image in blue. The mathematical expressions for transforming the data into the appropriate coordinate systems and performing the tasks described in this section of the report can be found in the block of code corresponding to creating a map and in the function **update_map()**.

3.2 Second Part of the Project

The second part of this project consisted of increasing the accuracy of the algorithm developed in the first part by implementing SLAM using a particle filter. The particle filter approach uses a number of samples to represent the posterior distribution of the localization and mapping processes given a set of noisy observations [3].

The number of particles N = 50 was found to be the most efficient in terms of the balance between performance and running time after testing several values within the interval 20-80. The first step consisted of initializing N particles located at (0,0) with an angle theta of 0. For each timestamp of the combined timestamp array explained in the sections above, the next odometry position was calculated for each particle. For the first timestamp, all the particles had the same trajectory, hence the occupancy grid was updated according to the lidar data for one of the particles. For each of the subsequent timestamps, once the odometry had been calculated, a small amount of random Gaussian noise of mean 0 and variance 0.0085 was added to the particle location in the map. The variance value was chosen by testing different values and choosing the one with the best outcome. This was followed by a calculation of the lidar hits for each particle in the map, which was done analogous to the procedure explained in the first part of this document.

Once the lidar hits had been calculated, the correlation of each of the N particles with the existing map from the previous iterations was calculated. The correlation is the sum of the log odds probability of each cell the lidar hits in the occupancy grid. The correlations were capped at 0 to avoid negative values, and were normalized, hence giving a single weight for each particle that indicates how well it fits the map. Once each particle had been assigned a weight, the number of effective particles was calculated as $n = \frac{(\sum_{i=1}^{N} w_i)^2}{\sum_{i=1}^{N} {w_i}^2}$. If the number of effective particles dropped below a threshold of N/2, resampling was applied. The resampling procedure applied in this project consisted of systematic resampling. The resampling algorithm together with the code to apply the particle filter can be found in the function **particle_filter()**. The results for both this approach and the approach from the first part of the project can be found in the next section of this report.

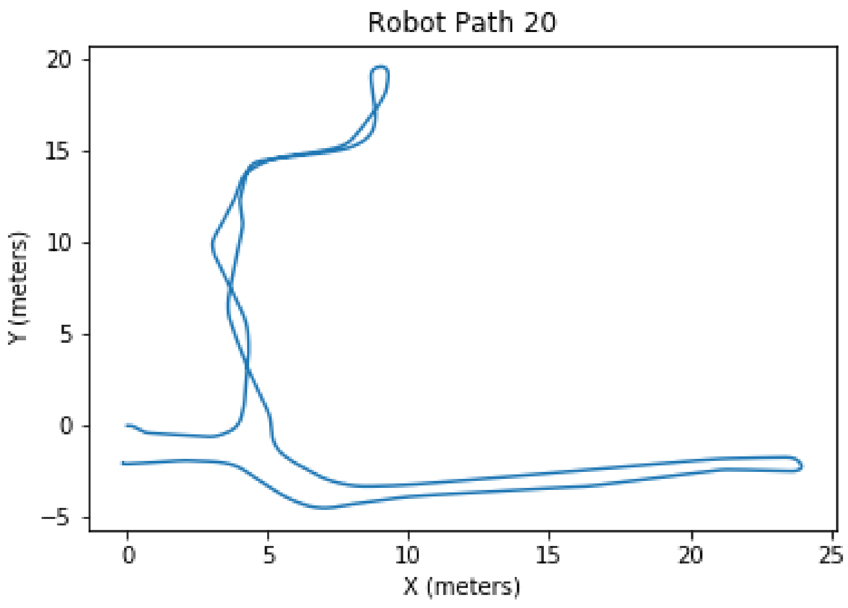

Figure 1: Dead Reckoning Robot Path for Dataset 20.

4 RESULTS

4.1 TRAINING DATA

The odometry path obtained using dead reckoning and a robot width of 733 millimeters on the datasets 20 and 23 for the first part of the project can be found on Figures 1 and 2. Moreover, the resulting 2D occupancy grid maps together with the robot motion in that map are shown on Figures 3 and 4.

Regarding the second part of this project, the resulting 2D occupancy grid and robot path obtained using a robot width of 733 millimeters and a particle filter with N = 50 on the training datasets 20 and 23 can be found on Figures 5 and 6, respectively. Videos showing the construction of the map using a particle filter and the robot path within the map at every timestamp can be found on the links below. Moreover, a fast version of the videos can be found inside the .zip folder of the project.

- Map 20: `https://www.youtube.com/watch?v=2V1MZIzOMyA`
- Map 23: `https://www.youtube.com/watch?v=Q4tAHkstUb8`

4.2 TESTING DATA

Regarding the testing datasets, the results for the 2D occupancy grid and robot path for datasets 21, 22, and 24 can be found on Figures 7, 8, and 9, respectively. Videos showing the construction of the map using a particle filter and the robot path within the map at every timestamp can be found on the links below. A fast version of the videos can also be found inside the .zip folder of the project.

- Map 22: `https://www.youtube.com/watch?v=_882s67lkwU`
- Map 24: `https://www.youtube.com/watch?v=_zaTAsv2jvY`
- Map 21: Video not available publicly due to YouTube length constraints.

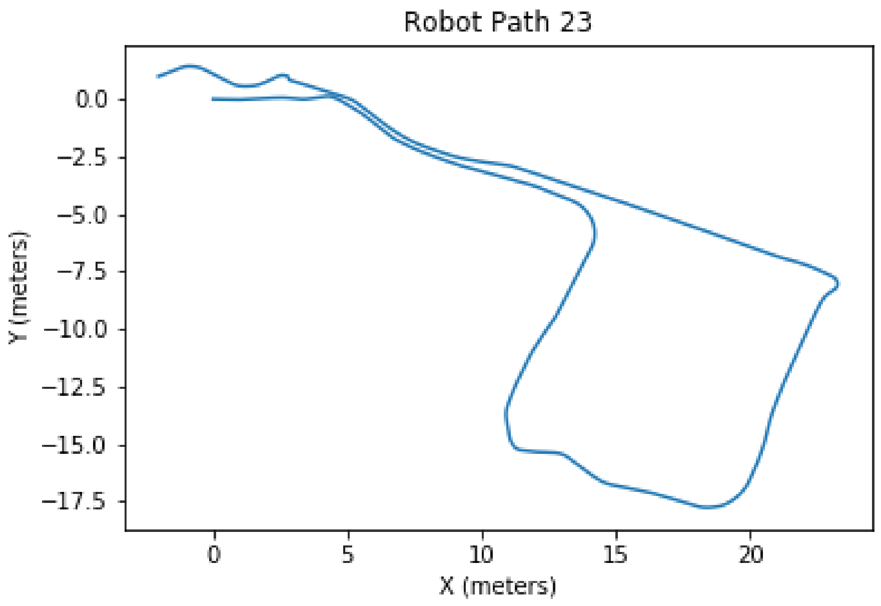

Figure 2: Dead Reckoning Robot Path for Dataset 23.

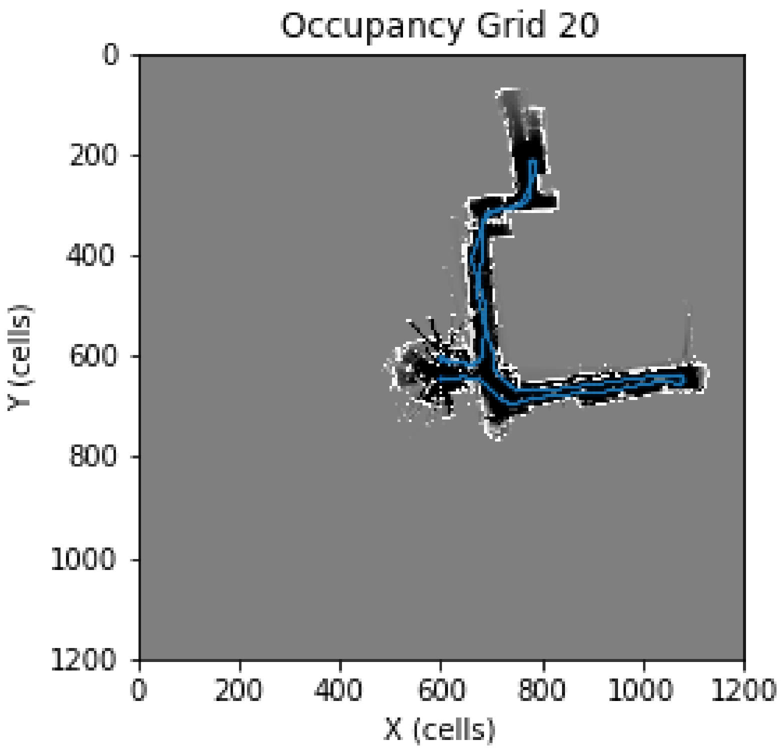

Figure 3: Occupancy Grid Map for Dataset 20.

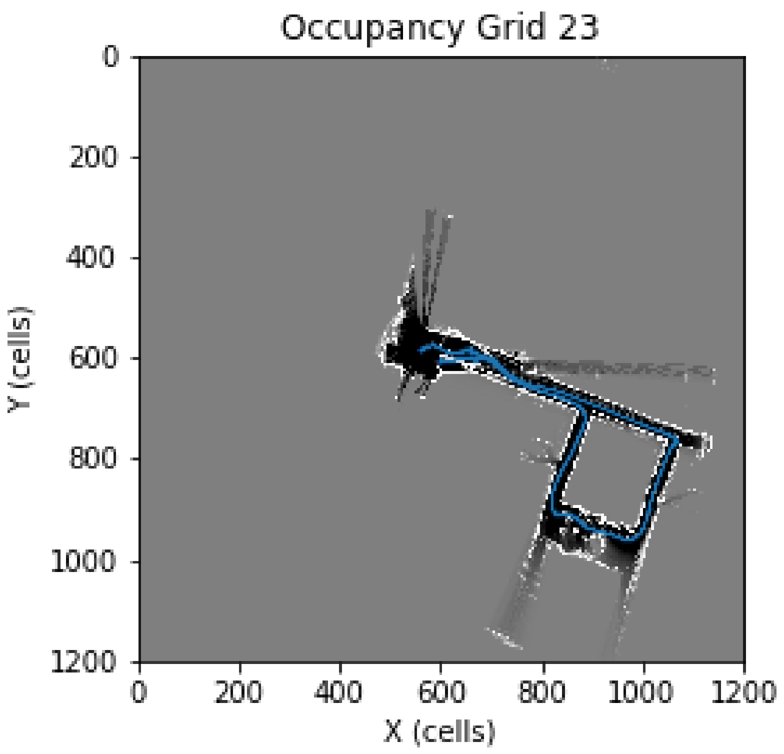

Figure 4: 2D Occupancy Grid for Dataset 23.

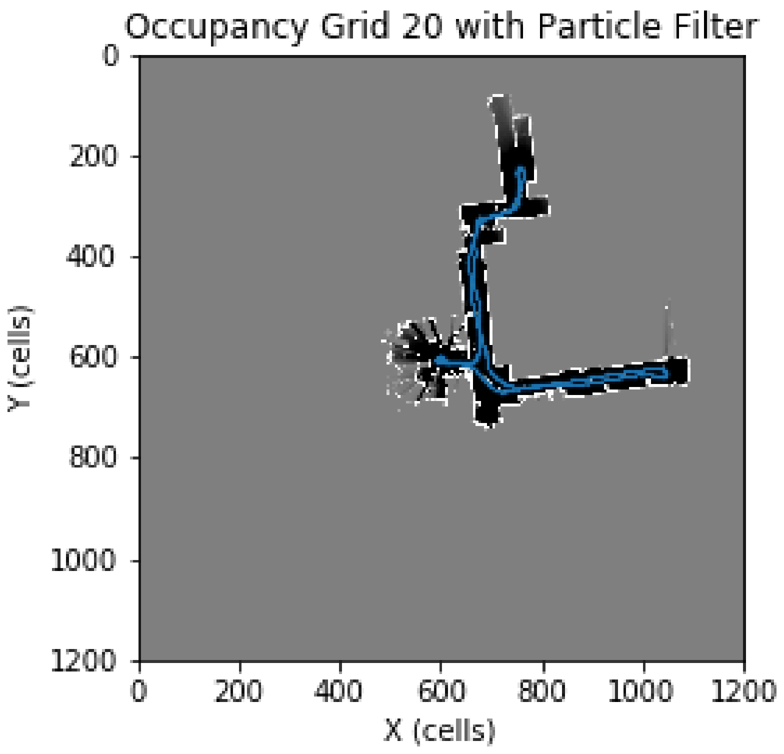

Figure 5: Occupancy Grid Map for Dataset 20 using Particle Filter.

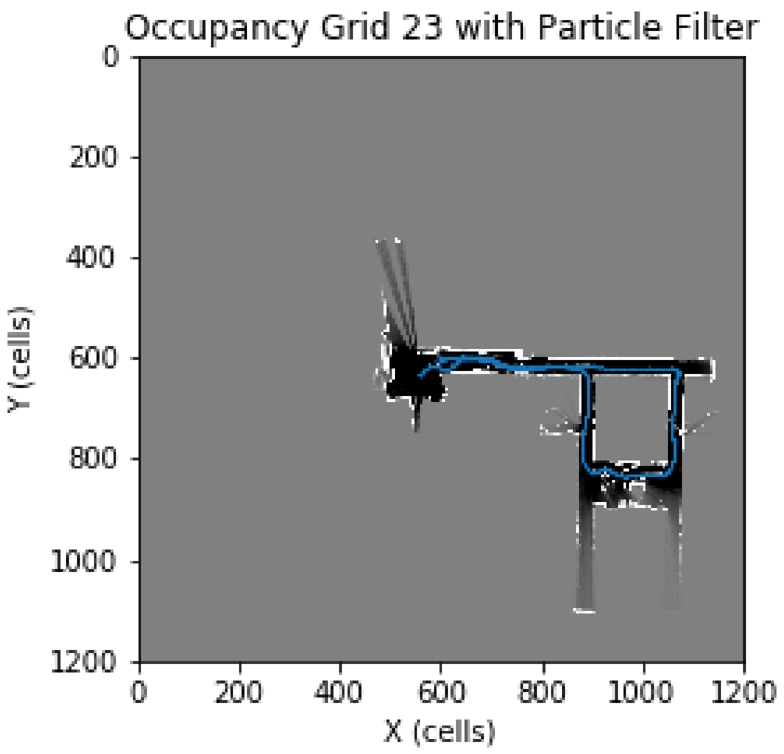

Figure 6: 2D Occupancy Grid for Dataset 23 using Particle Filter.

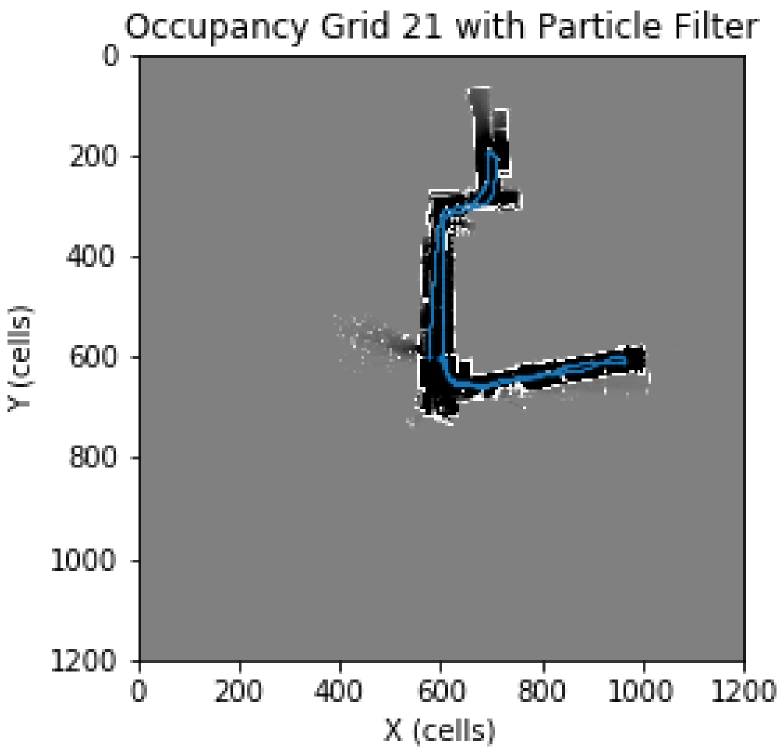

Figure 7: 2D Occupancy Grid for Dataset 21 using Particle Filter.

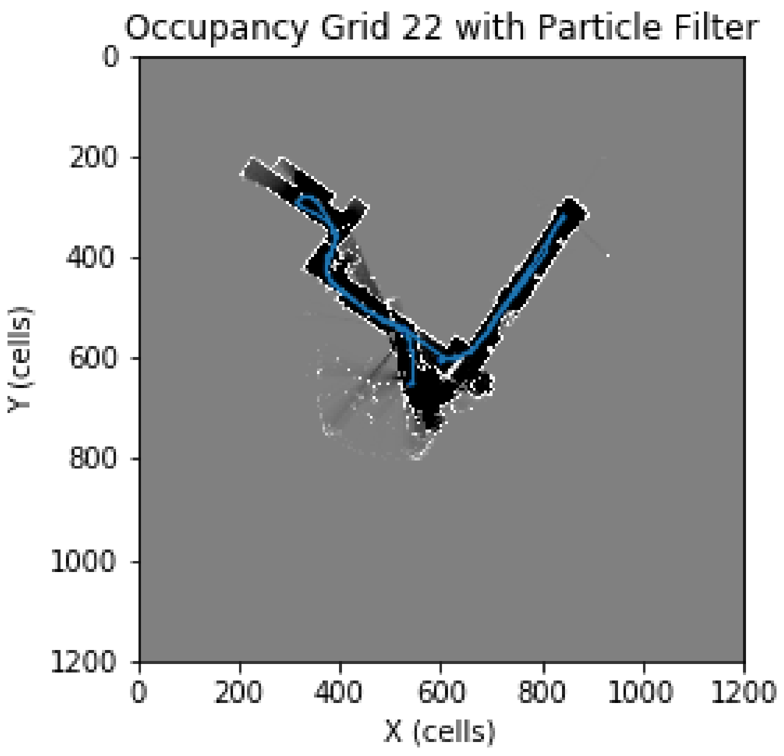

Figure 8: 2D Occupancy Grid for Dataset 22 using Particle Filter.

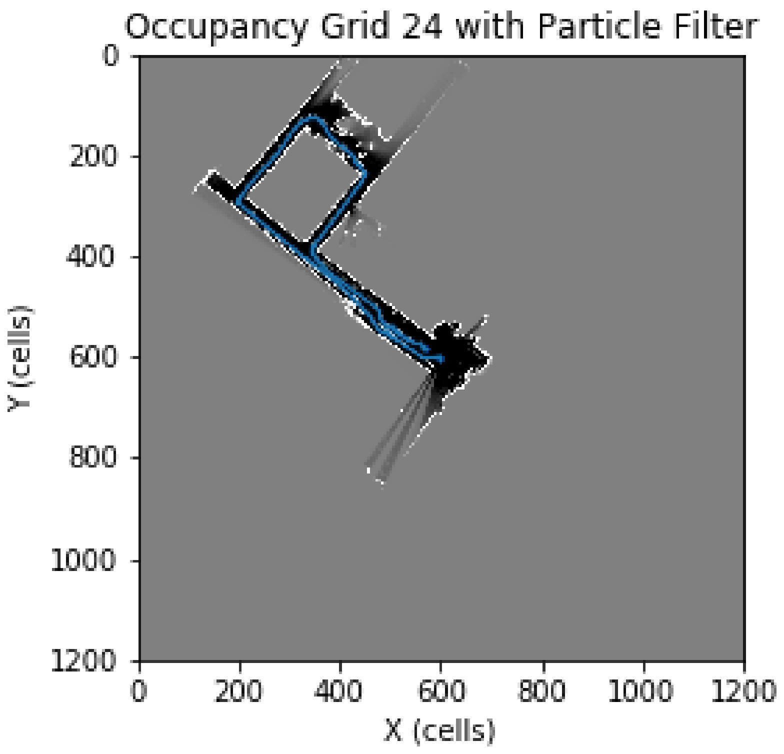

Figure 9: 2D Occupancy Grid for Dataset 24 using Particle Filter.

5 DISCUSSION

Regarding the first part of the project, the deadlines were successfully met and the occupancy grid map closely resembles the robot path calculated and the solution provided for both sets of training data. Nevertheless, some inaccuracies can be found in the 2D occupancy grids for the datasets 20 and 23.

Regarding the second part of the project, the results when using a particle filter approach show a decrease in the number of inaccuracies, which leads to the walls being more straight and giving a more accurate representation of the environment. Additionally, a lot of 'ghost' hallways disappeared when using the particle filter implementation. These improvements came at the expense of a longer running time. All in all, it can be concluded that the SLAM algorithm with a particle filter performed better than the algorithm that uses only one particle, and that the maps provide a clear depiction of the enviroment and the robot's position within it to the user.

6 FUTURE WORK

Future work includes fine-tuning the width of the robot further by using additional training data to improve the right side of map 24 and the bottom part of map 22. A greater number of particles could be implemented and tested for better accuracy and to remove some of the 'ghost' hallways in map 22. In addition, a threshold for only updating the map when the changes are significant could be implemented to efficiently reduce the running time. Finally, IMU data can be used to remove the parts of the map that correspond to LIDAR hits on the ceiling of the building.