Simultaneous Localization and Mapping (SLAM) with Particle Filter and Occupancy Grid Map in 2D Plane

Daniel Pak

***Keywords*— Simultaneous Localization and Mapping (SLAM), particle filter, occupancy grid map, lidar scans, odometry, IMU, map correlation**

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) has been an important topic in robotics in recent years. For mobile autonomous robots, localization allows for accurate movement from one point to another, leading to more reliable completion of tasks, and mapping is important for exploring new environments. In this project, we explored one method by which these tasks can be performed simultaneously.

II. PROBLEM FORMULATION

The main challenge was to accurately create a 2D occupancy grid map and estimate the robot's pose within that map. The x-y coordinate and yaw could be estimated using the built-in odometry and IMU measurements respectively, but both had considerable amount of noise and drift. As a result of imperfect pose estimations of the robot, mapping with dead reckoning results in abysmal occupancy grid maps. Clearly, a more careful analysis was needed to make sense of the noisy data. In this project, we used particle filter and log odds accumulation method to solve this problem.

III. TECHNICAL APPROACH

A. Setup Overview

At each time point, a lidar scan provides the distances of the closest objects to the sensor itself in a 2D plane (Fig. 1). The measurements are at angles -135 to 135 degrees at 0.25 degree intervals. Lidar scans were used to shape the outer boundaries of the maps, and the contour created by the boundaries was considered empty space.

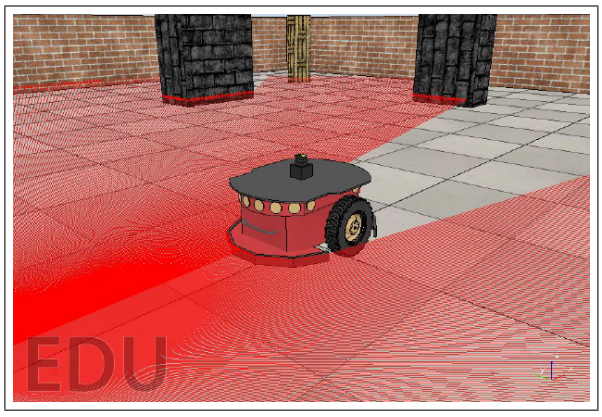

Fig. 1. Example of a lidar measurement

The lidar was placed on a humanoid robot's head (Fig. 2). The humanoid could turn its neck in yaw directions and move its head in pitch directions. The IMU and odometry sensors were in the body frame, some distance away from the center of mass.

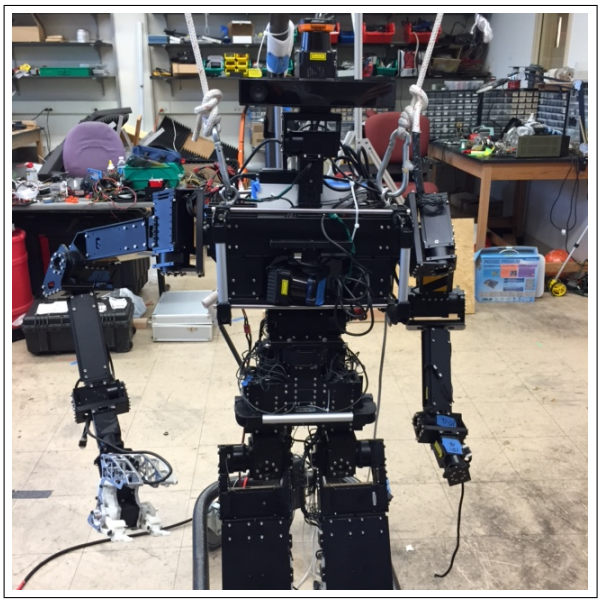

Fig. 2. Humanoid robot used for data collection

B. Transformation of frames

Due to the robot's freedom to move its head in yaw and pitch directions, the lidar scans had to be converted from head frame to body frame. I simply used a 2D rotation matrix with neck angle to rotate the lidar scans to match with the body frame. Then, in order to remove any potential floor scans, I used the head angle and calculated a height value for the scans, and removed any that had a threshold of less than 0.1 meters from the ground. I also converted all remaining lidar scans to be the normal distance from the robot's head, since the surroundings should be recorded at normal distance in the occupancy grid map.

Another necessary transformation was with the odometry data. Since the data were given in an inaccurate global frame, which was rotated with yaw values provided together with the odometry information, I rotated the odometry values to the local frame by applying a 2D rotation matrix. Afterwards, the values were converted to the correct global frame using the yaw values from the IMU.

C. Procedure Overview

1) Define pose $p := (x, y, \theta) \in SE(2)$. Initialize 100 particles to $p = (0, 0, 0)$ and log weights $w = \log(\frac{1}{N})$ where $N =$ number of particles
2) Update the map with the best particle (i.e. one with the highest weight) using the transformed lidar scans and adding log odds ($\log(9)$ for hit points and $\log(\frac{1}{9})/4$ for empty spaces). The empty spaces were given less weight so that scans far away from the sensor, which were relatively less accurate, do not completely erase parts of the map.
3) If $\frac{1}{\sum \exp(w)^2} < N_{eff} = 1.5$, resample using the stratified resampling method
4) Apply the motion model with every particle. This entails applying the transformed odometry and adding a different amount of random noise to each value of pose. The amount of noise for each drastically impacted the quality of SLAM, and had to be tuned for each environment.
5) Calculate the map correlation between the existing map and the particle's lidar scans. This is simply summing the number of occupied cells that coincide with the positions of the lidar scans. Update the log weights by adding the map correlation, and normalize such that $\sum \exp(w) = 1$.
6) Repeat from step 2.

IV. RESULTS

The results for one of the training sets are shown in Fig. 3 and 4. The rest of the figures are attached as Appendiex for better visual arrangement of figures. To provide comparison between the naive dead reckoning method and the carefully tuned particle filter + map correlation method, I have provided a figure for both cases for each dataset (Fig. 3 and 4 for training set 0). Needless to say, the proposed algorithm performs much better at both localization and mapping.

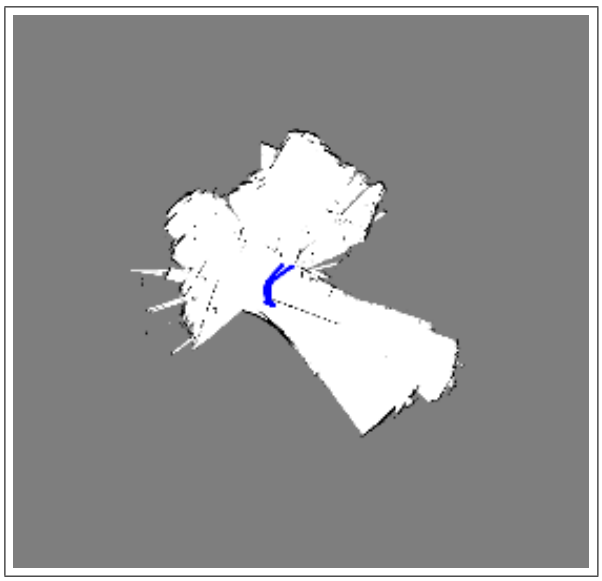

Fig. 3. Training set 0 - mapped with dead reckoning

Fig. 4. Training set 0 - final SLAM algorithm

Additionally, I have included an example of a SLAM map with less optimal parameters from preliminary results given by code submitted on Thursday to show the impact of parameter tuning in the performance of the algorithm (Fig. 5).

Fig. 5. Training set 0 - Original code (parameters not optimized)

Another noteworthy visualization implemented in this project (that was not included in the original code submission) was plotting the current position and orientation of the robot in real time, on the map being drawn. Along with these pieces of information and the current positioning of the lidar scans, it was much easier to determine which parameters needed to be changed to increase performance (Fig. 6).

V. DISCUSSION

As previously mentioned, there were many variables at play that drastically impacted the performance of the al-

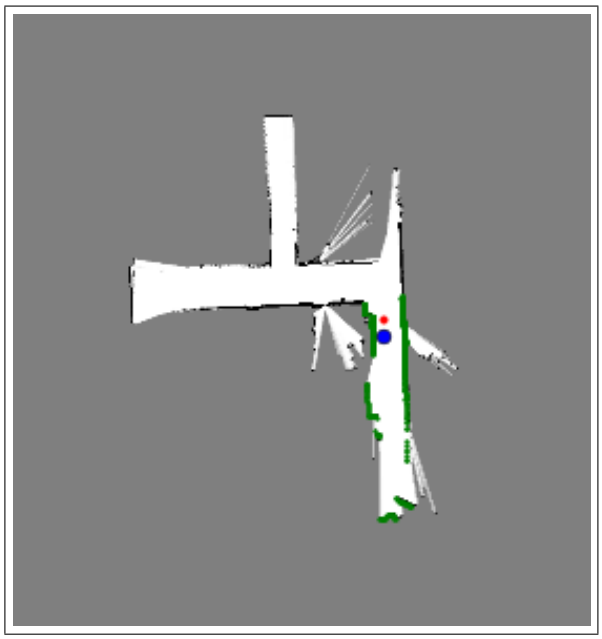

Fig. 6. Testing set map in progress, plotted with current position of robot (red), orientation of robot (facing in the direction of blue), and lidar scans (green).

gorithm. This was a big challenge in the earlier parts of the project, when there were many uncertainties and no conviction of the correctness of the algorithm.

The first and most important variable was the noise parameters. All noise values were drawn from a Gaussian distribution, with a mean of 0 and varying standard deviations. The standard deviation for noise in x-y coordinates generally ranged from 0.001 to 0.01 (meters) for time increments of 1 (with larger jumps in time increments, I multiplied the noise standard deviation by a factor of increments), and that in yaw generally ranged from 0.05 to 0.1 (radians). Even the slightest change in these numbers caused the map to look drastically different, which in turn affected localization as well. One thing I tried was adjusting the level of noise based on the rate of change in yaw estimations, since if seemed like most errors were accumulated at turning points of the robot. This method was not used to produce the final results, but it did help the maps with sub-optimal noise parameters at various turning points of the robot, which was a promising result.

Another variable was the log odds accumulation limit. This variable was most important when I was implementing a version of the code where the log odds map was used instead of the binary map to calculate the map correlation. In that case, since the robot was stationary at the beginning of data acquisition, the log odds accumulation caused the initial points to have too much weight in map correlation compared to new points, which seemed to adversely affect the update in particle pose. The log odds limit of 100 was set to mitigate this effect.

When the log odds map was used instead of the binary map, intuitively it made more sense because it would also take into account the parts of the map that were recorded as empty spaces. However, I observed that this caused some issues when turning corners, especially when the number of scans was limited to a smaller area of the map and the scans themselves were noisy, because the particle chose to line itself up with the unexplored parts of the map to avoid summing the negative weights from the empty cells. Especially when cells have accumulated a lot of negative log odds from previous measurements, this type of avoiding the lidar scans to match with the unoccupied cells occurred frequently, which caused problems with pose estimations.

Reduced log odds for empty spaces was another important modification to the original algorithm. Since there are many instances where a long hallway was mapped by lidar scans very far from the robot, there were many instances where the contour created by those scans completely covered the existing boundaries and made them unoccupied cells. Since the map correlation was the summation of occupied cells, this was a major problem in computing the update step. Therefore, I used reduced log odds to record the empty cells, with the rationale that lidar scans far from the robot are most likely less accurate than those close by.

Another important modification for the binary map was changing the criterion for marking empty or occupied. Initially, I used the provided equation to convert log odds to probability and thresholded that value as greater than or less than 0.5 to mark occupied or free respectively, but I learned that the map looked much better when the threshold is more strict. Since the probability is already 0.1 with one $\log(1/9)$ accumulation and 0.9 with one $\log(9)$ accumulation, the original measure was very extreme in assigning occupied or empty in the map, which caused problems for map correlation. Instead, I used thresholding values on the log odds values themselves such as occupied when greater than 5 and empty when less than -3, so that it is more straight forward in terms of parameter adjustment (it is much harder to adjust the parameter when log odds is converted to probability because it includes an exponential function).

One of the most noticeable challenges in this project was the turning points for the robot. Since the robot is turning from one hallway to another, a lot of the unexplored cells are being recorded as being empty or occupied for the first time in the algorithm, but the yaw drift and lidar scan noise are both greatest at these points, leading to unstable initial mapping of the new hallway. This was a big personal takeaway from this project, along with the fact that real-time visualization helps tremendously with parameter tuning.

VI. REFERENCES

All other information was acquired from Unviersity of Pennsylvania's ESE650 lectures, discussions on Piazza, official websites of Python and the respective libraries.