ESE 650 Project 4 report: Simultaneous localization and mapping (SLAM) using particle filters

Dhruva Kumar

1 Introduction

This project aims to simultaneously localize and map a robot in an unknown indoor environment using odometry data, IMU, and a lidar scanner. A particle filter based approach is taken to achieve the objective. Particle filters effectively model a probability distribution (dynamic model) as a set of discrete states. Each particle has an associated weight which describes its confidence in its current estimate according to the observation. The THOR robot is used for this project.

2 SLAM

- A log odds grid map is used to map the bot and update its values (occupied cells and free cells) based on the lidar scan hits. A 1500 x 1500 map is used with a grid resolution of 5cm per cell.
- The head pitch, neck yaw and torso yaw are taken into consideration.
- Initially, 100 particles are taken and are initialized with position (0,0) and the orientation is randomized according to a normal distribution with a variance of 20°. The log odds grid map is updated with the lidar scans of the first iteration. The particle weights are initialized to one.
- **Dynamic update**: Subsequently for each iteration, the motion of all the particles are updated according to the odometry data:

$$\begin{bmatrix} x_i^t \\ y_i^t \end{bmatrix} = \begin{bmatrix} x_i^{t-1} \\ y_i^{t-1} \end{bmatrix} + \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} + \begin{bmatrix} \mathcal{N}(0,\sigma_x) \\ \mathcal{N}(0,\sigma_y) \end{bmatrix} \tag{1}$$

$$\theta_i^t = \theta_i^{t-1} + d\phi \tag{2}$$

$$\text{where, } d\phi = \phi_i - \phi_{i-1}$$

$$\alpha = \theta_{i-1} - \phi_{i-1}$$

  (x, y, θ) is the pose for each particle i for the t^{th} iteration. ϕ is the yaw of the bot derived from integrating the gyroscope of the IMU.
  Equation (1) and (2) is the motion model for the particles. The position of each particle is updated by its previous position + the difference in x and y from the odometry data rotated by α. Here, α is the difference between the previous orientation of the bot and previous bot yaw. Gaussian noise with a variance of σ is also added to the position of each particle. The orientation θ of each particle is updated by the difference in the yaw procured from the odometry data. σ_x and σ_y are kept as 0.3. The variance induced in the position of the particles mattered significantly in performance. The higher the value, the better the performance. However, it shouldn't be set too high for all the particles to diverge.

- **Weight update**: The weights of the particles are updated and resampled if required.

- The weights of the particles are set based on the lidar hits according to the pose for each particle. The particle whose map corresponds highly to the previous log odds map is given more weight. In this case, the weights are calculated by summing up over the lidar hits or occupied cells by a factor of 0.02. This worked pretty well. The current weight vector is multipled by its previous weight which is used as a prior here to use the previous information. This is then normalized to sum upto 1.
- Before computing the weights, the lidar scans neglected for invalid ranges. This is done for scan < 0.3, scan > 30 and for pitch correction. The lower limit 0.3 should be as low as possible to account for all the hits. However, it should also account for instances of gantry hits. 0.3 seemed to work well in general. If the head pitch is above a threshold, scans at the sides ($0° \pm$ thresh-angle and $180° \pm$ thresh-angle) are retained, while the rest is removed. Ranges from 35 to 55 °worked well for thresh-angle. The threshold for the head pitch was computed based on:

  $$scan_{center} > \frac{r}{\sin\gamma}$$

  Here, $scan_{center}$ refers to the lidar scan of the forward position where the bot is looking at. r refers to the diagonal of the right angled triangle if the height of the triangle is the height of the bot. γ refers to the head pitch. The above formulation worked well for the train set. However, a hard thresholding of pitch ¿ 0.25 radians worked better and this was used.
- Resampling is done if the following condition is satisfied:

  $$N_{effective} = \frac{(\sum_{i=1}^{N} w_i)^2}{\sum_{i=1}^{N} w_i^2} < threshold$$

  The threshold was kept as 0.2. When resampled, the weight vector is set to ones. This threshold should be less. If it's too large, resampling is done really frequently and at any given time, there will be a few set of particles in the cluster.
- The chosen bot position is the particle with the highest weight.

- **log odds map update**: The log odds map is updated with the lidar scans of the best particle chosen.
  - The occupied cells are incremented by $10 \log r$. $\log r$ was kept as 0.02.
  - The free cells in the map were decremeted by $\log r$.
  - The values of the log map were limited to 3 to prevent certainty. This threshold mattered significantly in the performance of SLAM.

3 Results

The loop closure for the test set doesn't happen perfectly. But it's still pretty decent nonetheless. The performance on the train data is pretty good.

References

[1] Dr. Daniel Lee ESE 650 lecture notes.

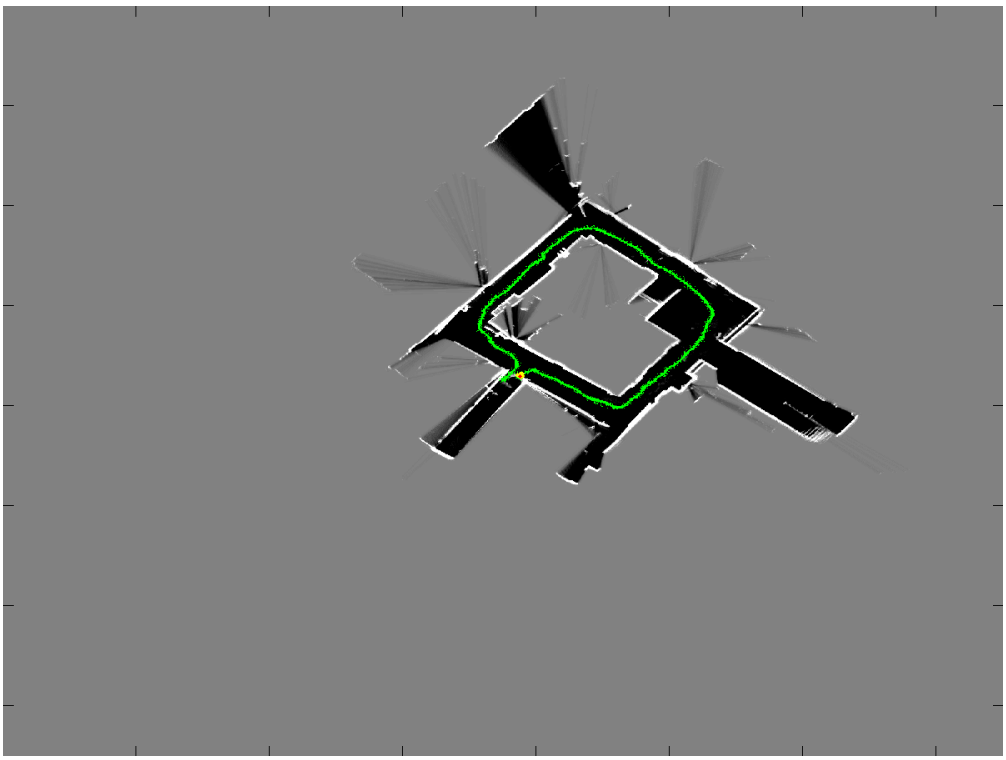

Figure 1: Test set with path

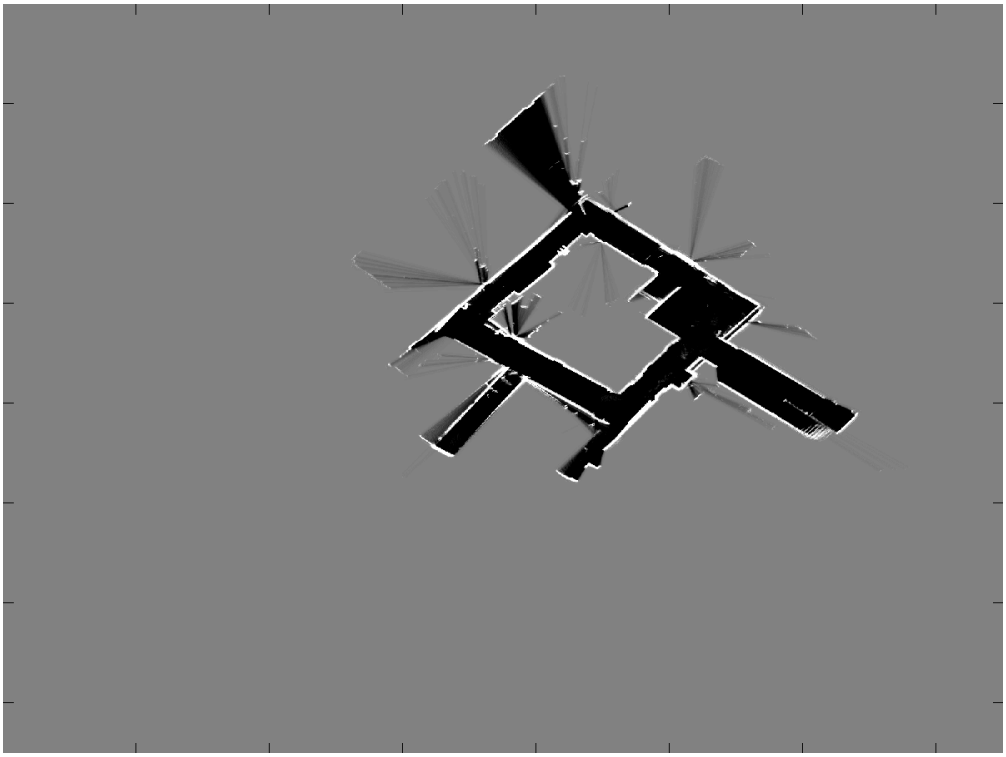

Figure 2: Test set

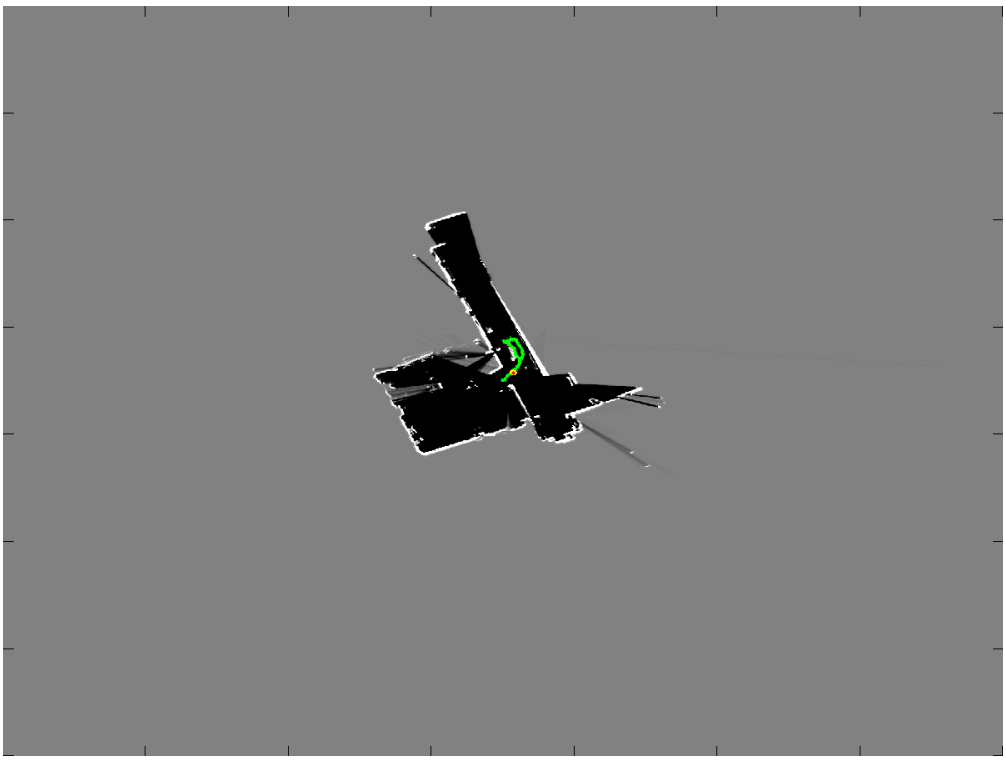

Figure 3: Dataset 0 with path

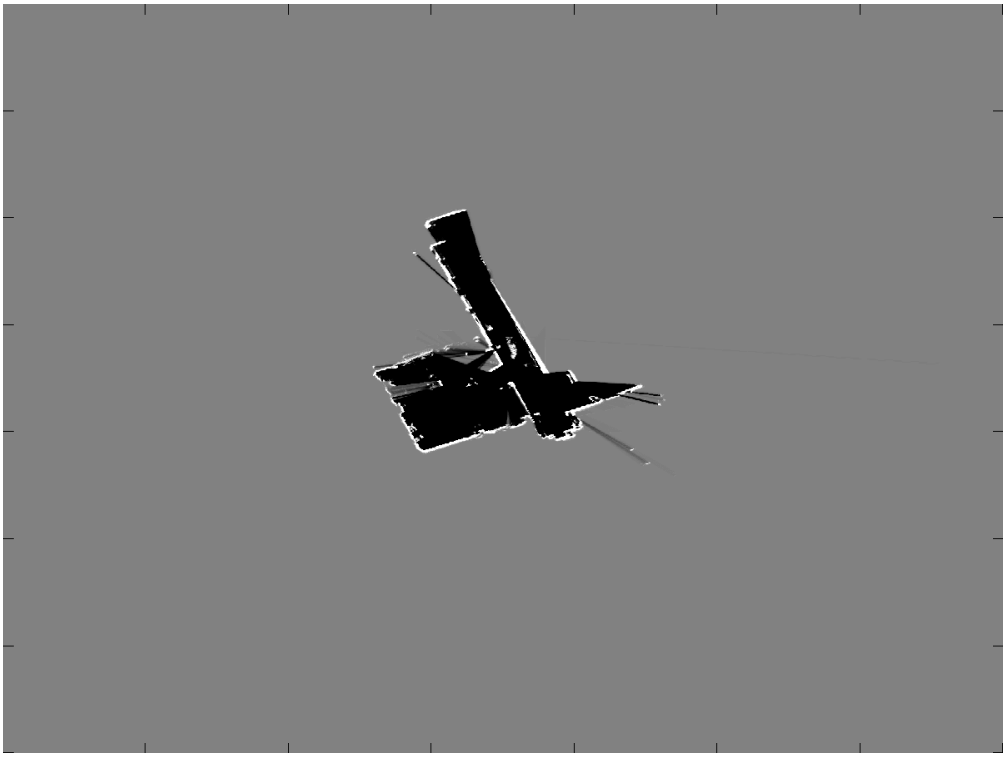

Figure 4: Dataset 0

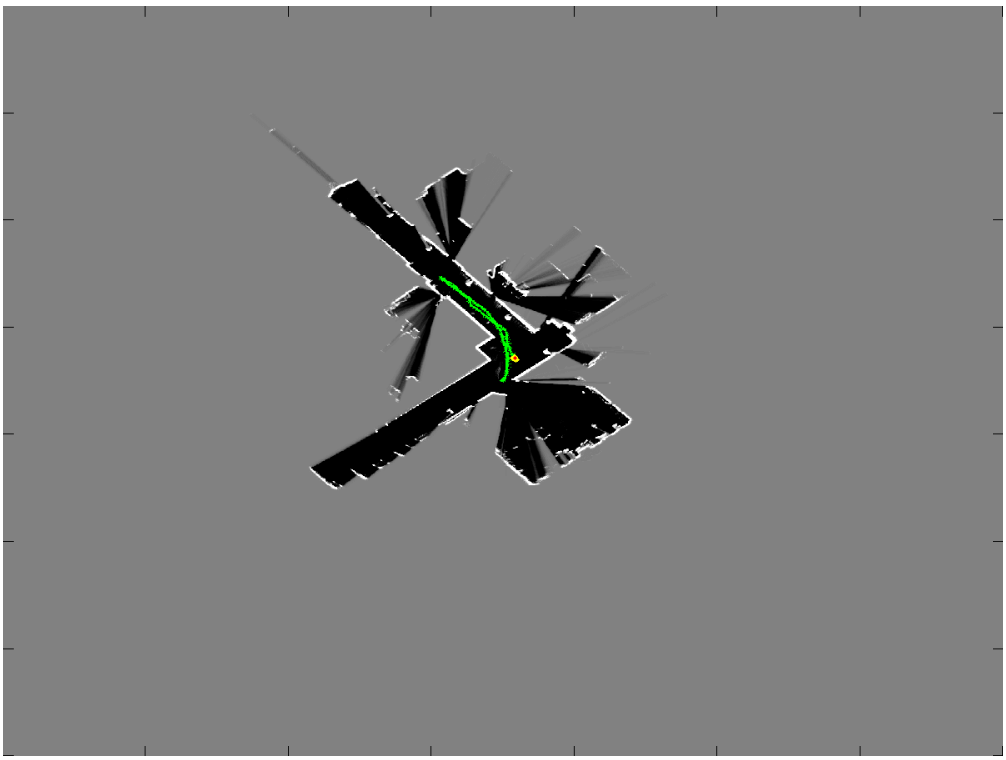

Figure 5: Dataset 2 with path

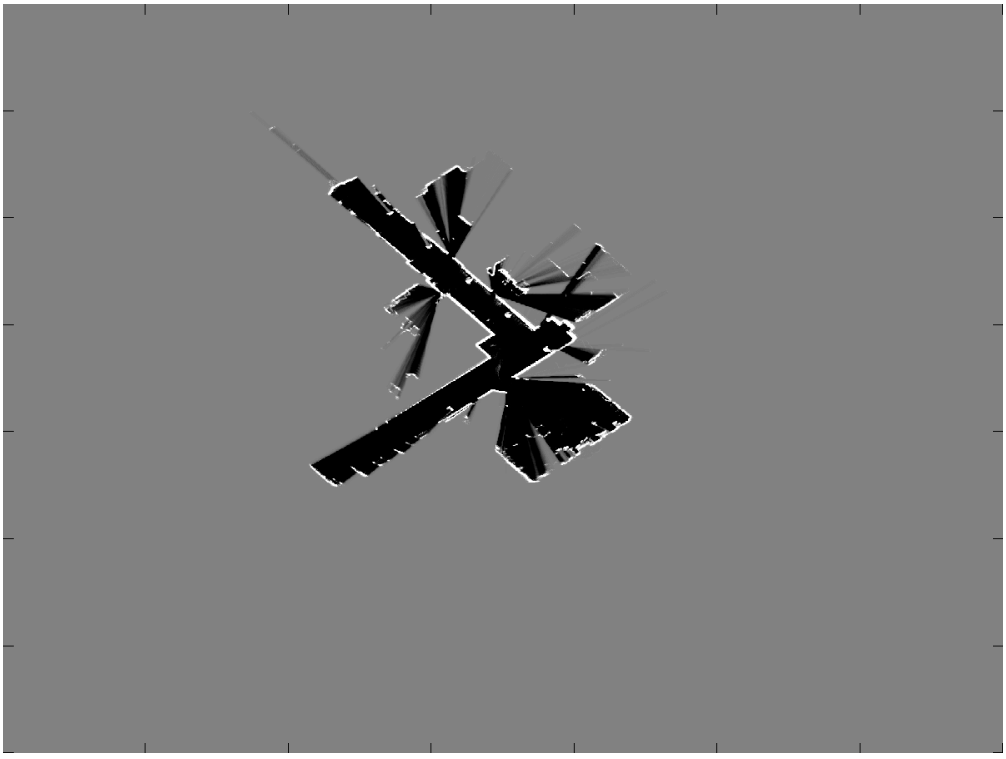

Figure 6: Dataset 2

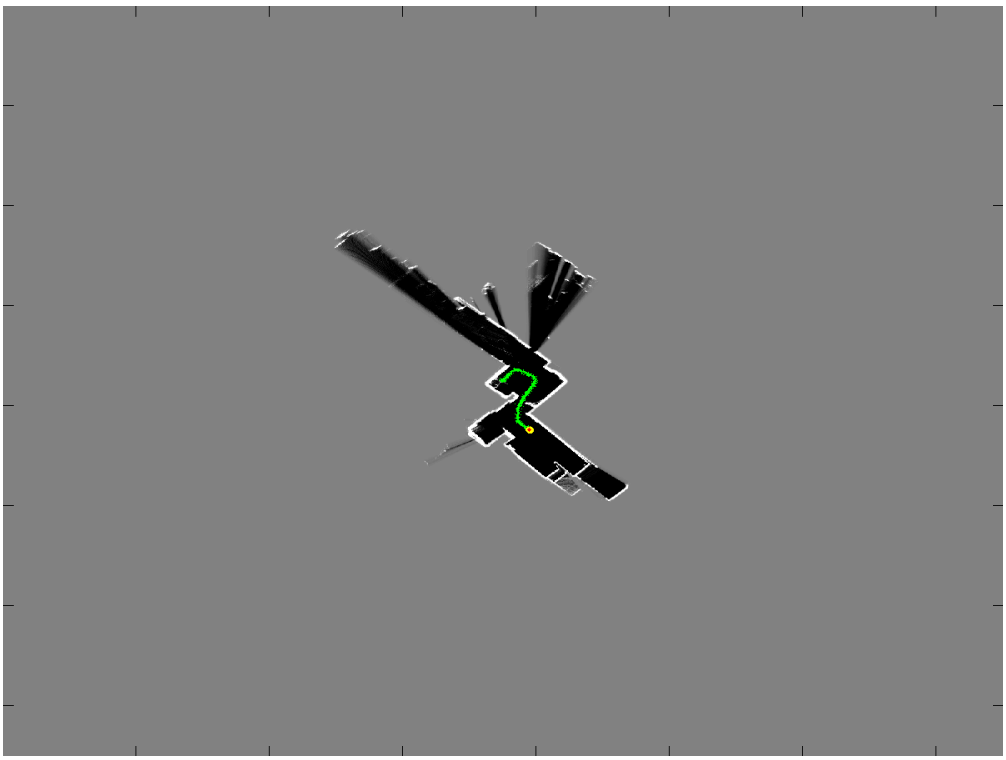

Figure 7: Dataset 3 with path

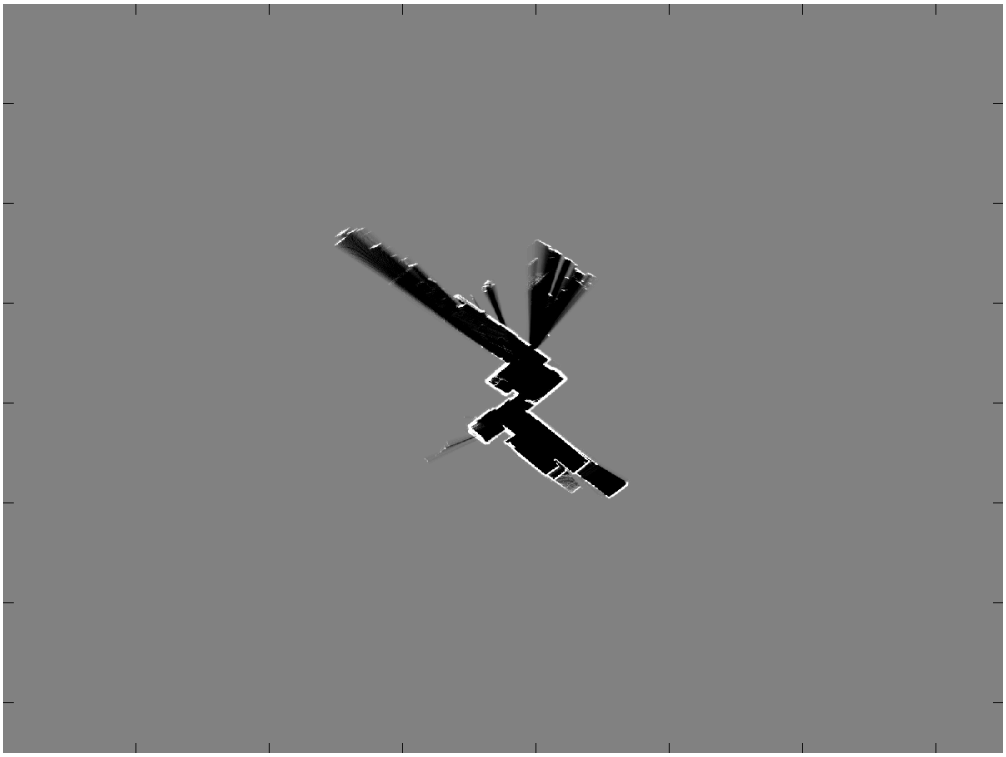

Figure 8: Dataset 3