Project 2 - Particle Filter SLAM and Texture Map

Siddharth Dey
Department - Electrical and Computer Engineering
Major - ISRC
PID - A59015531
email - sidey@ucsd.edu

***Abstract*—In project 2, the first problem is to estimate the robot's trajectory and generate the static map using the IMU and encoder data for odometry measurements and LiDAR scans for range measurements obtained from Hokuyo UTM-30LX. The particle filter algorithm is used to address this problem. The second objective is to generate a texture map from the RGB and disparity values of the Kinect RGBD camera and the trajectory generated in the first problem.**

I. Introduction

The field of robotics has undergone significant advancements in recent years, with a particular focus on the development of Simultaneous Localization and Mapping (SLAM) techniques. One such method is the Particle Filter algorithm, which has been increasingly utilized for 2D robot SLAM applications. This project aims to explore the implementation and performance of Particle Filter-based SLAM in a 2D robotics environment, analyzing the accuracy and efficiency of the algorithm in mapping and localizing robots in real-world scenarios. Through a comprehensive evaluation, this study intends to provide valuable insights into the effectiveness of Particle Filter-based SLAM for 2D robotics applications. This report also investigates the potential application of the algorithm in generating Texture Maps. Specifically, the trajectory obtained in the first part of the study and the RGB images and disparity maps acquired from the Kinect camera is analyzed to create a 2D color map of the floor surface.

II. Problem Formulation

A. Simultaneous Localization and Mapping (SLAM)

A differential-drive robot is moving in a 2d map with obstacles. The robot is equipped with four sensors - Encoders, IMU, Hokuyo LiDAR scanner, and RGBD Kinect camera. The encoders and IMU are to be used to obtain odometry information, including linear and angular velocity. The Hokuyo sensor gives a horizontal LiDAR scan with a 270° field of view and a maximum range of 30 m and provides distances to obstacles in the environment. Each LiDAR scan contains 1081 measured range values. The timing of the five sensors is different and has to be synchronized before use.

The first objective is to estimate the trajectory of the robot $X_{0:T}$, which includes the 2D robot pose (x_t, y_t) and the robot's angular orientation θ_t at each timestamp:

$$trajectory = X_{0:T} = \begin{bmatrix} x_{0:T} \\ y_{0:T} \\ \theta_{0:T} \end{bmatrix} \tag{1}$$

Under the Markov assumption, the Bayes filter can be used for estimating the trajectory. It surmounts to keeping track of two pdfs, the predicted pdf $p_{t+1|t}(x_{t+1})$ and the updated pdf $p_{t+1|t+1}(x_{t+1})$ at each timestamp as shown below:

$$\begin{aligned} p_{t+1|t}(x_{t+1}) &:= p(x_{t+1}|z_{0:t}, u_{0:t}) \\ p_{t+1|t+1}(x_{t+1}) &:= p(x_{t+1}|z_{0:t+1}, u_{0:t}) \end{aligned} \tag{2}$$

where u_t is the control input at time t and z_t is the observation. Starting with a prior pdf $pt|t(x_t)$, the Bayes Filter needs the motion model function f or equivalently probability density function p_f and the observation model function h or equivalently probability density function p_h as defined below:

$$\begin{aligned} x_{t+1} = f(x_t, u_t, w_t) \sim p_f(.|x_t, u_t); w_t = \text{motion noise} \\ z_t = f(x_t, v_t) \sim p_f(.|x_t); v_t = \text{observation noise} \end{aligned} \tag{3}$$

Given the motion model and observation model, the Bayes Filter update becomes:

$$\begin{aligned} p_{t+1|t}(x) &= \int p_f(x|s, u_t) p_{t|t}(s)\, ds \\ p_{t+1|t+1}(x) &= \frac{p_h(z_{t+1}|x) p_{t+1|t}(x)}{\int p_h(z_{t+1}|s) p_{t+1|t}(s)\, ds} \end{aligned} \tag{4}$$

The Bayes Filter needs (4) to be solved consecutively at each time step.

The static map **m** also has to be generated, where the map cells m_i can be defined as independent Bernoulli random variables as shown below:

$$m_i = \begin{cases} +1(Occupied), & \text{with prob } \gamma_{i,t} \\ -1(Free), & \text{with prob } 1 - \gamma_{i,t} \end{cases} \tag{5}$$

where the probability $\gamma_{i,t}$ is defined as the probability of the map cell being occupied given all the sensor measurements and trajectory up to time t as shown below:

$$\gamma_{i,t} = p(m_i = 1|z_{0:t}, x_{0:t}) \tag{6}$$

The mapping objective is to estimate the map cell probability at the final time stamp $\gamma_{i,T}$, which can be then used to generate a binary occupancy map by using a suitable threshold.

B. Texture Map

The second part of the project is to use the RGBD images and the estimated robot trajectory to produce a 2D color map of the floor surface. After obtaining the depth of each RGB pixel using the disparity map from the Kinect camera, the colored points need to be projected from the camera frame to the world frame. After obtaining the world coordinates of the image pixels, the pixels belonging to the floor need to be selected. A second grid map is created with the same resolution as the occupancy grid, with its cells colored with the RGB values according to the projected points.

III. Technical Approach

A. Motion model

The robot follows a Differential-drive Kinematic Model, where the state X is defined as a vector containing the position $p = (x, y) \in R^2$ and the yaw angular orientation $\theta \in (-\pi, \pi]$ in the world frame. The Discrete-time Differential-drive Kinematic Model becomes:

$$X_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(x_t, u_t) := X_t + \tau_t \begin{bmatrix} v_t cos(\theta_t) \\ v_t sin(\theta_t) \\ \omega_t \end{bmatrix} \quad (7)$$

To account for motion noise, a 2-D Gaussian motion noise is added to the control inputs, linear and angular velocities as shown below:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t + \epsilon_t) \\ u_t &= (v_t, \omega_t) \\ \epsilon_t &= N(0, \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}) \end{aligned} \quad (8)$$

B. Sensor Calibration

The robot is equipped with the following sensors:

- Encoders: There are four encoders attached to the four wheels of the robot. Given the four encoder counts [FR, FL, RR, RL] corresponding to the front-right, front-left, rear-right, and rear-left wheels, the linear velocity is calculated as:

$$v_t = (FR + FL + RR + RL)/4 * (0.0022/\tau_t) \quad (9)$$

- IMU: The IMU sensor is used to obtain the yaw rate as the angular velocity in the differential-drive model in order to predict the robot's motion
- Hokuyo: Each LiDAR scan contains 1081 measured range values at angular intervals of $0.25°$. The radial distance and the angular position of the lidar scan give the coordinates of the point in cylindrical coordinates. It is first converted into Cartesian coordinates in the lidar frame. It is then transformed into the robot frame and finally into world coordinates by using the robot's trajectory.
- Kinect: An RGBD camera provides RGB images and disparity images. Using the disparity map and camera intrinsic matrix, the location of the pixels in the optical frame is obtained which is transformed into the regular frame. The angular orientation and position of the depth camera with respect to the robot center are provided, which is used to transform the pixel points in the robot frame. The robot's trajectory is used to finally transform them into the world coordinates.

C. Particle Filter Algorithm

The particle filter uses particles with locations $\mu[k]$ and weights $\alpha[k]$ for k = 1, ..., N to represent the pdfs $p_{t|t}$ and $p_{t+1|t}$:

$$\begin{aligned} p_{t|t}(x_t) &= \sum_{k=1}^{N} \alpha_{t|t}[k]\delta(x_t - \mu_{t|t}[k]) \\ p_{t+1|t}(x_{t+1}) &= \sum_{k=1}^{N} \alpha_{t+1|t}[k]\delta(x_{t+1} - \mu_{t+1|t}[k]) \end{aligned} \quad (10)$$

After substituting these pdfs into the Bayes Filter equations, the prediction step becomes:

$$p_{t+1|t}(x_{t+1}) = \sum_{k=1}^{N} \alpha_{t|t}[k]\delta(x_{t+1} - \mu_{t+1|t}[k]) \quad (11)$$

and the Particle Filter Update Step to rescale the particle weights based on the observation likelihood becomes:

$$\begin{aligned} &p_{t+1|t+1}(x_{t+1}) = \textstyle\sum_{k=1}^{N} [\frac{\alpha_{t+1|t}[k] p_h(z_{t+1}|\mu_{t+1|t}[k])}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] p_h(z_{t+1}|\mu_{t+1|t}[j])}] \times \\ &\delta(x_{t+1} - \mu t + 1|t[k]) \end{aligned} \quad (12)$$

A LiDAR correlation model is defined to estimate the likelihood model $p_h(z|x, m)$ for LiDAR scan z obtained from sensor pose x in occupancy grid m. The LiDAR scan likelihood is set proportional to the correlation between the scan's world-frame projection $y = r(z, x)$ via the robot pose x and the occupancy grid m:

$$p_h(z|x, m) \propto corr(r(z, x), m) \quad (13)$$

where the correlation $corr(r(z, x), m)$ is defined as:

$$corr(y, m) = \sum_i 1(y_i = m_i) \quad (14)$$

The particle weights are then updated using the scan-map correlation:

$$\alpha_{t+1|t+1}[k] \propto corr(y_{t+1}[k], m)\alpha_{t+1|t}[k] \quad (15)$$

Most updated particle weights become close to zero because a finite number of particles is not enough to represent the state pdf. Resampling is performed to avoid particle depletion by adding new particles at locations with high weights and reducing the particles at locations with low weights. It focuses the representation power of the particles to likely regions. Resampling is applied if the effective number of particles N_{eff}:

$$N_{eff} =:= \frac{1}{\sum_{k=1}^{N} (\alpha_{t|t}[k])^2} \quad (16)$$

becomes less than a threshold:

$$N_{eff} \leq \frac{N}{10} \tag{17}$$

To resample, N particles are drawn with replacement with probability $\alpha_{t|t}[j]$ and the particles $\mu_{t|t}$ are added with weights $\frac{1}{N}$ to the new particle set.

D. Mapping

The map update problem is converted into Log-odds occupancy grid map update as follows:

$$\lambda_{i,t} = \lambda_{i,t-1} + \Delta\lambda_{i,t} \tag{18}$$

where the log-odds of the Bernoulli random variable m_i is defined as:

$$\lambda_{i,t} = \lambda(m_i|z_{0:t}, X_{0:t}) = log(\frac{\gamma_{i,t}}{1-\gamma_{i,t}}) \tag{19}$$

Assuming an 80% correct sensor, the inverse measurement model $\Delta\lambda_{i,t}$ becomes:

$$\Delta\lambda_{i,t} = \begin{cases} +log(4), & \text{if } z_t \text{ indicates } m_i \text{ is occupied} \\ -log(4), & \text{if } z_t \text{ indicates } m_i \text{ is free} \end{cases} \tag{20}$$

Once the log-odds occupancy grid map is obtained at the final stamp $\lambda_{i,T}$, it can be converted into a binary map by taking the logistic sigmoid of each cell and putting a threshold.

Given the robot's position and the end point of the LiDAR scan, bresenham2D is used to find the free cells using the ray tracing algorithm in 2D.

E. Texture Map

As mentioned earlier, the Kinect RGBD camera is used to obtain the coordinates of the pixels in the world frame by using the disparity map and the robot's trajectory. To obtain the points belonging to the floor, the z coordinate of the final transformed points should be close to 0. These points are selected and their RBG values are projected onto the occupancy grid.

IV. Results

A. Mapping

The robot is assumed to start with identity pose. To make sure that the transforms are correct before starting to estimate the robot pose, only the first LiDAR scan is considered initially to generate the map. Table I shows the plots generated by the first LiDAR scan. The free cells are colored yellow and the unknown and occupied cells are colored purple in the plots. bresenham2D is used to obtain the occupied cells and free cells that correspond to the LiDAR scan. bresenham2D returns both the free and the occupied cells. The occupied cell is the last element in the array returned by bresenham2D and hence must not considered together with the free cellls during log-odds map update.

B. Prediction

Before using the Particle filter for prediction and update, a simple trajectory is generated by directly using the motion model (dead-reckoning). The resulting trajectory and angular orientation of the robot is shown in Table II and III respectively. Using the dead-reckoning trajectory, the map shown in Table IV is generated from the LiDAR scans. Once the dead-reckoning is verified, the trajectory for N particles are generated using prediction step with Gaussian noise added to the velocities. Initially, a standard deviation of 0.01 and 0.001 was used for noise added to linear and angular velocities but then the trajectories were too close each other. The trajectories shown in V is generated using a standard deviation of 0.1 and 0.01 for linear and angular velocities respectively.

C. Update

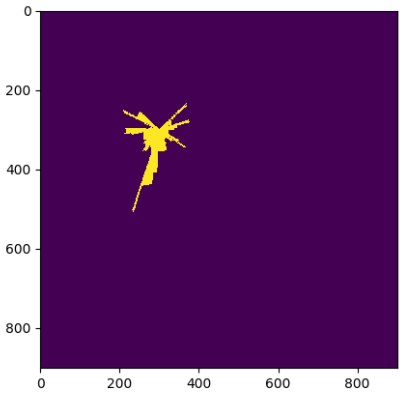
The Particle Filter algorithm with particles N = 100 is used to generate the final occupancy map shown in Table VI. The resolution of the map is 0.05m. While calculating the correlation to estimate the observation likelihood, 9 locations close to the robot's position were used to find the correspondences. The highest value returned by the correlation function was used for the update step. An attempt was made to update the position of the particle to the location with highest correspondence out of the 9 combinations. But it resulted in a distorted map. Hence, this idea was dropped and no update was performed to the particle's position while finding the correlation during the update step.

D. Texture Map

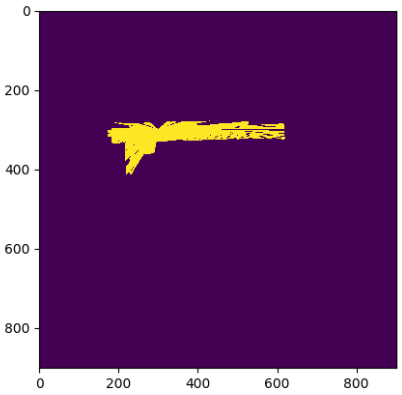
The trajectory generated by the particle filter algorithm is used to generate the texture map as shown in Table VII. The same procedure discussed in the previous section was adotped to generate the texture map of the floor.

E. Video Results

The videos of the occupancy grid map updates and texture map updates with time for both the datasets is also submitted in the videos folder.



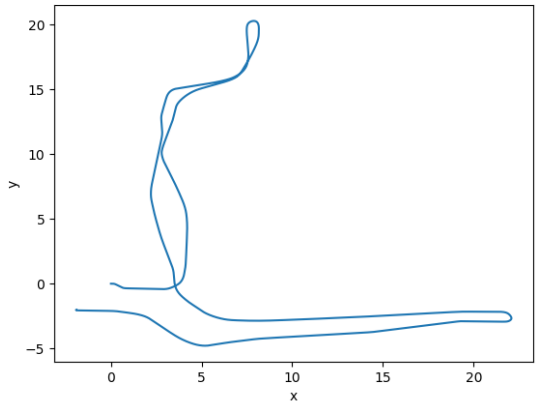
Dataset 20



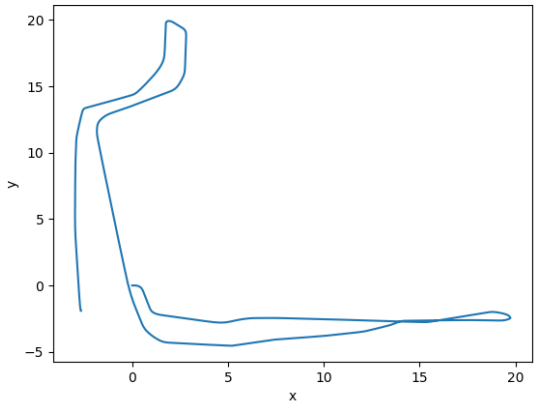
Dataset 21

TABLE I

The map generated using the first LiDAR scan for datasets 20 and 21. The free cells are colored yellow and the unknown and occupied cells are colored purple in the plots.



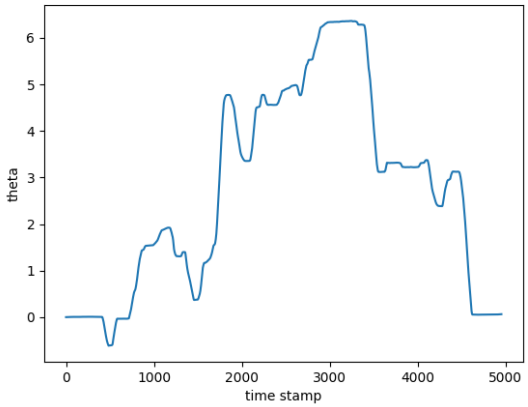
Dataset 20



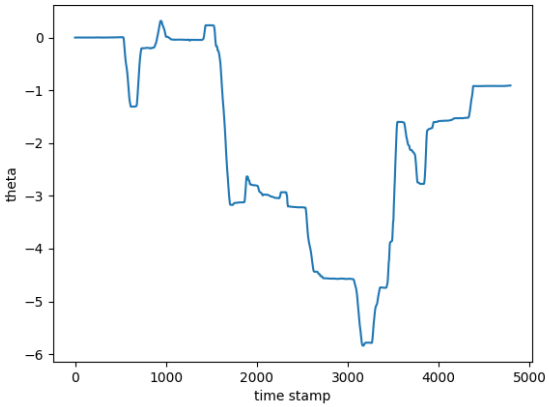
Dataset 21

TABLE II

The trajectory generated by dead-reckoning



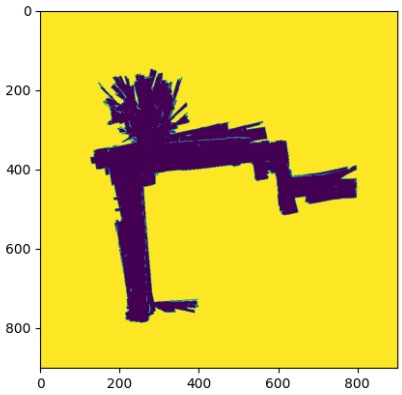
Dataset 20



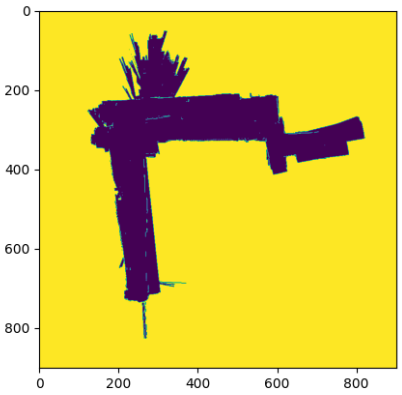
Dataset 21

TABLE III

The variation of angular orientation θ over time, generated by dead-reckoning



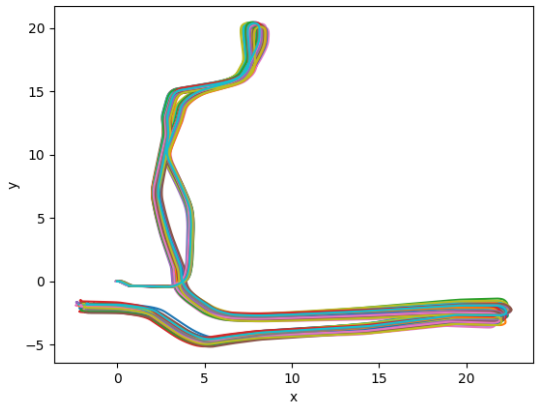
Dataset 20



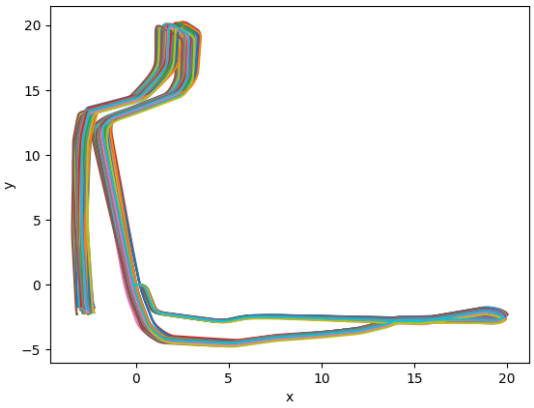
Dataset 21

TABLE IV

Map generated by dead-reckoning as trajectory



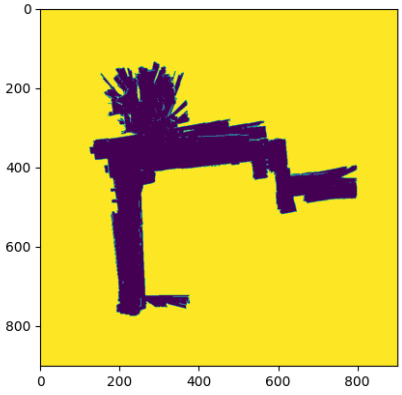
Dataset 20



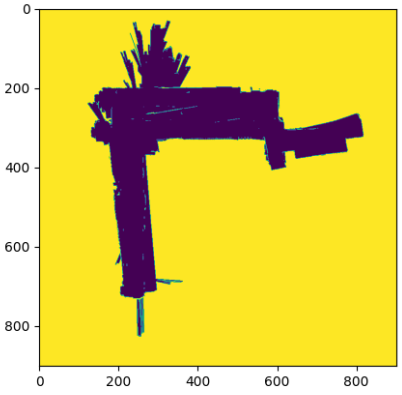
Dataset 21

TABLE V

Trajectory generation by N = 100 particles using prediction step with noise



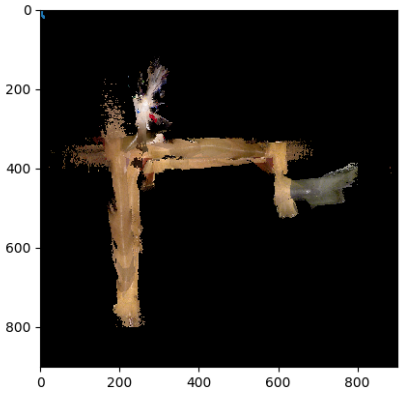
Dataset 20



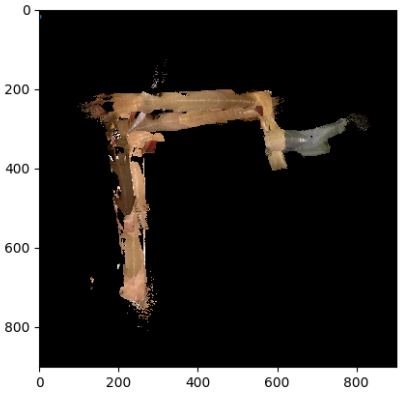
Dataset 21

TABLE VI

Final Occupany grid map generated by the Particle Filter Algorithm



Dataset 20



Dataset 21

TABLE VII

Final Occupany grid map generated by the Particle Filter Algorithm