Particle Filter SLAM

Parthasarathi Kumar
Department of Electrical and Computer Engineering
University of California, San Diego
California, USA
pakumar@ucsd.edu

I. Introduction

Simultaneous Localization and Mapping or SLAM is an important problem in the field of autonomous robots where the agent has to map an unknown environment and at the same time localize itself in the map. This problem has many applications in the area of navigation, path planning, high definition mapping, etc. SLAM is generally done using a combination of sensors like camera, LiDARs, GPS, IMU, etc and involves a probabilistic approach of combining these sensor inputs.

In this paper we approach the SLAM problem on a dataset captured by an autonomous car in an urban environment equipped with a LiDAR(2D), stereo camera, wheel encoders and gyroscope sensors. We fuse these sensor inputs using a particle filter to create a map of the surroundings and simultaneously localize the agent in the created map.

II. Problem Definition

Problem of SLAM involves estimating the robot state x_t at time t and map state m, given the observations $z_{0:t}$ and inputs $u_{0:t-1}$ under Markov assumptions. This can be stated mathematically as computing the joint probability distribution of the agent state and map state as described by (1).

$$p(x_{0:T}, m|z_{0:T}, u_{0:T-1}) \tag{1}$$

To solve this we leverage the motion model and observation model. The motion model describes the state of agent given the previous state and control input as in (2) where f defines the motion model and w_t is the motion noise.

$$x_{t+1} = f(x_t, u_t, w_t) \sim p_f(.|x_t, u_t) \tag{2}$$

$$z_t = h(x_t, m_t, v_t) \sim p_h(.|x_t, m_t) \tag{3}$$

The observation model helps model the surroundings based on the observation z conditioned on the observation model h and observation noise v_t as in (3)

Using (2) and (3) and applying on (1) under the Markov assumption, we can break down the problem into the following equation

$$\begin{aligned} p(x_{0:T}, m|z_{0:T}, u_{0:T-1}) &= p(x_0, m_0) \\ \prod_{t=0}^{T} p_h(z_t|x_t) &\prod_{t=1}^{T} p_f(x_t|x_{t-1}, u_{t-1}) \end{aligned} \tag{4}$$

Applying Bayesian filtering, we obtain the following general equations for the predict (2) and update (3) step respectively.

$$p_{t+1|t}(x) = \int p_f(x|s, u_t) p_{t|t}(s) ds \tag{5}$$

$$p_{t+1|t+1}(x) = \frac{p_h(z_{t+1}|x) p_{t+1|t}(x)}{\int p_f(z_{t+1}|s) p_{t+1|t}(s) ds} \tag{6}$$

III. Technical Approach

The agent state is described by its 2-D position and yaw angle as described below -

$$\begin{aligned} x &= (p, \theta) \\ p &= (x, y) \end{aligned} \tag{7}$$

A. Particle Filter

We extend the formulation in section II and use the special form of Bayesian Filtering called the particle filter to model the probabilities. A particle filter is described as follows -

$$\{\mu^{(k)}, \alpha^{(k)}\}_{k=1}^{N} \tag{8}$$

where each particle is a delta function centered at $\mu^{(k)}$ and has probability $\alpha^{(k)}$ where the probabilities sum up to 1. Each hypothesis can be represneted as -

$$p(x) = \sum_{k=1}^{N} \alpha^k \delta(x - \mu^k) \tag{9}$$

Applying the general Bayesian filter equation, we obtain the following predict steps for the particle filter.

$$p_{t+1|t}(x) = \sum_{k=1}^{N} \alpha^k_{t+1|t} \delta(x - \mu^k_{t+1|t}) \tag{10}$$

As is evident from the equation, this involves movement of the hypothesis center. The update step involves updation of the weightage of the particles based on the following equation-

$$p_{t+1|t+1}(x) = \sum_{k=1}^{N_{t+1|t}} [\frac{\alpha^k_{t+1|t} p_h(z_{t+1}|\mu^k_{t+1|t})}{\sum_{j=1}^{N_{t+1|t}} \alpha^k_{t+1|t} p_h(z_{t+1}|\mu^j_{t+1|t})}] \delta(x - \mu^k_{t+1|t}) \tag{11}$$

B. Motion Model

We use the differential drive kinematic model as the observational model. This model uses the instantaneous angular velocity and linear velocity as control inputs to the state of the agent's state as in (12)

$$x_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(x_t, u_t) = x_t + \tau \begin{bmatrix} v_t cos(\theta_t \\ v_t sin(\theta_t \\ \omega_t \end{bmatrix} \tag{12}$$

To calculate the instantaneous velocity we use the encoder readings and use the following equation to calculate the velocity at a given time interval.

$$\begin{aligned} v &\approx \frac{\pi dz}{4096\tau} \\ v &= \frac{v_L + v_R}{2} \end{aligned} \tag{13}$$

where the wheel diameter is 0.62m as given in the sensor configuration, τ is the change in time and z is the change in encoder reading across the specified time interval. Since we have left and right wheel encodings, we use the mean of the 2 to approximate the velocity of the vehicle.

To calculate the angular velocity we utilize the readings of the gyroscope (FOG) sensor. The FOG sensor gives the change in the the yaw angle which is used to get the instantaneous angular velocity

$$\omega = \frac{\theta}{\tau} \tag{14}$$

We add Gaussian noise of mean 0 and variance 0.1 and 0.001 to the calculated velocity and angular velocity respectively. A different noise value is added for each of the particle in order to ensure the particles do not coagulate.

C. Occupancy Map

To represent the map, we use a 2D grid of a given resolution; 1m for example. A physical point (x, y) is mapped to the 2D grid based on the following formula -

$$\begin{aligned} x_{map} &= ceil(x - x_{min})/res) - 1 \\ y_{map} &= ceil(y - y_{min})/res) - 1 \end{aligned} \tag{15}$$

The occupancy map stores the probability of being occupied. Since its a binary state we use the log odds representation and follow (16) to update the map based on the lidar scan -

$$\lambda_{i,t} = \lambda_{i,t-1} + \Delta\lambda_{i,t} \tag{16}$$

$$\Delta\lambda_{i,t} = \begin{cases} +log(k) & z_t \text{ is occupied} \\ -log(k) & z_t \text{ is empty} \end{cases} \tag{17}$$

The probability can be recovered from the log odds value by applying the sigmoid function.

D. Map Correlation

We use the map correlation as the map observation model for updating the particle weights in the update step.

$$p_h(z_{t=1}|\mu^k_{t+1|t}, m) \propto corr(y^k_{t+1}, m) \tag{18}$$

Given a particle, we transform the lidar scan to the world co-ordinate as per the particle's pose. We then calculate the map correlation value by placing a suitable sized grid around the paricle position to get an accurate correlation value as per the following equation -

$$corr(y, m) = \sum_i \mathbb{1}\{y_i = m_i\} \tag{19}$$

where y is the transformed lidar points.

E. Synchronization and time management

Each given sensor is associated with a timestamp. However, each of the sensor is asynchronous and with different frequencies. The FOG sensor has 10x the frequency of the encoders. The camera, lidar and encoder approximately have a similar frequency.

We use the encoder timestamp as the main timing reference and use it to get the closest reading for each of the other sensor. For efficient access of the sensor reading we use a indexing based approach as described in (18) instead of each time doing a search in the sensor's timestamp array.

$$index_{sensor} = \frac{(t_{current} - t^{sensor}_{min})}{freq_{sensor}} \tag{20}$$

For speeding up the process, we only use every second encoder reading and every 5^{th} lidar reading.

F. Particle Resampling

During the workflow, due to the noisy prediction step, a number of particles' weightage tends to zero where as a few particles have a very high probability. In order to ensure the effective number of particles remains above a threshold value (0.6 times the total number of particles).

G. Texture Mapping

For texture mapping, we use the disparity map from the stereo image pair. Given a pair of stereo image pair, we use the stereo block matching function of opencv to calculate the disparity map. This disparity map is used to create a colored point cloud that is translated to the world coordinate frame as per the following equation -

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_{or} R^T (m - p) \tag{22}$$

The above equations utilize the camera intrinsics as indicated in the sensor configuration.

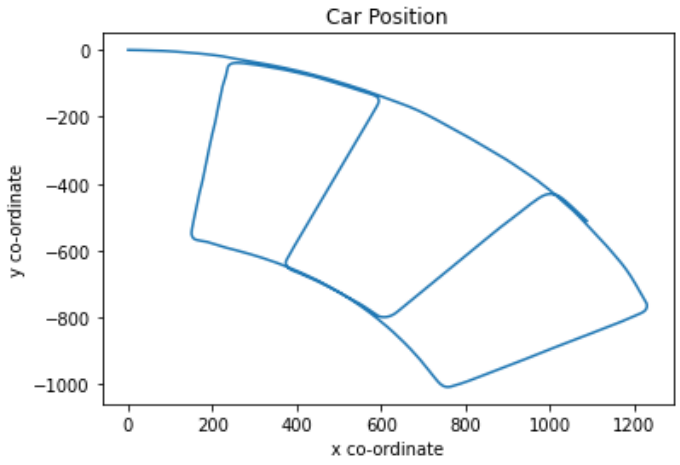

Fig. 1. path traced by a single particle using the differential drive motion model

H. Algorithm

In this section we give a brief overview of the algorithm, covering all the above sections in a chronological order.

The SLAM algorithm begins with the initialization of the particle at the origin with a yaw angle of 0. As stated above the encoder timestamp is used as the time sequence.

1) Predict: In the predict step, the encoder and FOG readings are used to calculate the velocity and angular velocity to apply the differential drive model to predict the next state of the particle.

2) Update: In the update step, we use the lidar measurement to update the particle weights based on map correlation value. This correlation value is calculated for each of the particle where we transform the lidar points to the world points as per the particle pose.

Based on the most confident particle, we update the map using the measured lidar scan. We also resample the particles in case the effective particles fall below the threshold value. At last, the disparity map calculated using the stereo image pair is transformed as per the most confident particle to map the texture to the occupancy map.

IV. Results

A. Single Particle Predict

Figure 1 depicts the path traced by a single particle using the differential drive motion model.

B. Qualitative Experiments

Figure 2 depicts the occupancy map, texture map and path overlayed on the occupancy map for different number of particles.(N=5,50,300)

Figure 3 depicts the occupancy map, texture map and the path traced by the most confident particle on the occupancy map at different instances of time. (N=50)

C. Discussion

Although the plots look similar for different values of N, we can observe the slight improvement as the number increases. To elaborate on this, notice the overlap of roads in case of N = 300 when the vehicle re-traces the path. On the other hand for N - 5, 100 we see significant deviation in the overlapping parts.

The similarity can be attributed to the accurate sensor readings.

D. Parameter Settings

In this experiment we utilize the sensor information provided in the sensor configuration file. The experiments are done across different number of particles.

The lidar points with a range less than 1m and more than 60m are ignored. These points correspond to faraway points and often lead to out of map points.

The map has $[x_{min}, x_{max}, y_{min}, y_{max}]$ set as [-80, 1280, -1080, 80]. The map resolution is set as 1m unless stated otherwise. A value of $log(4)$ is used as the log odd value. For texture mapping we threshold the point cloud at z = 5m which is a reasonable estimate of the ground plane.

References

[1] ECE 276A lecture 8,9,10

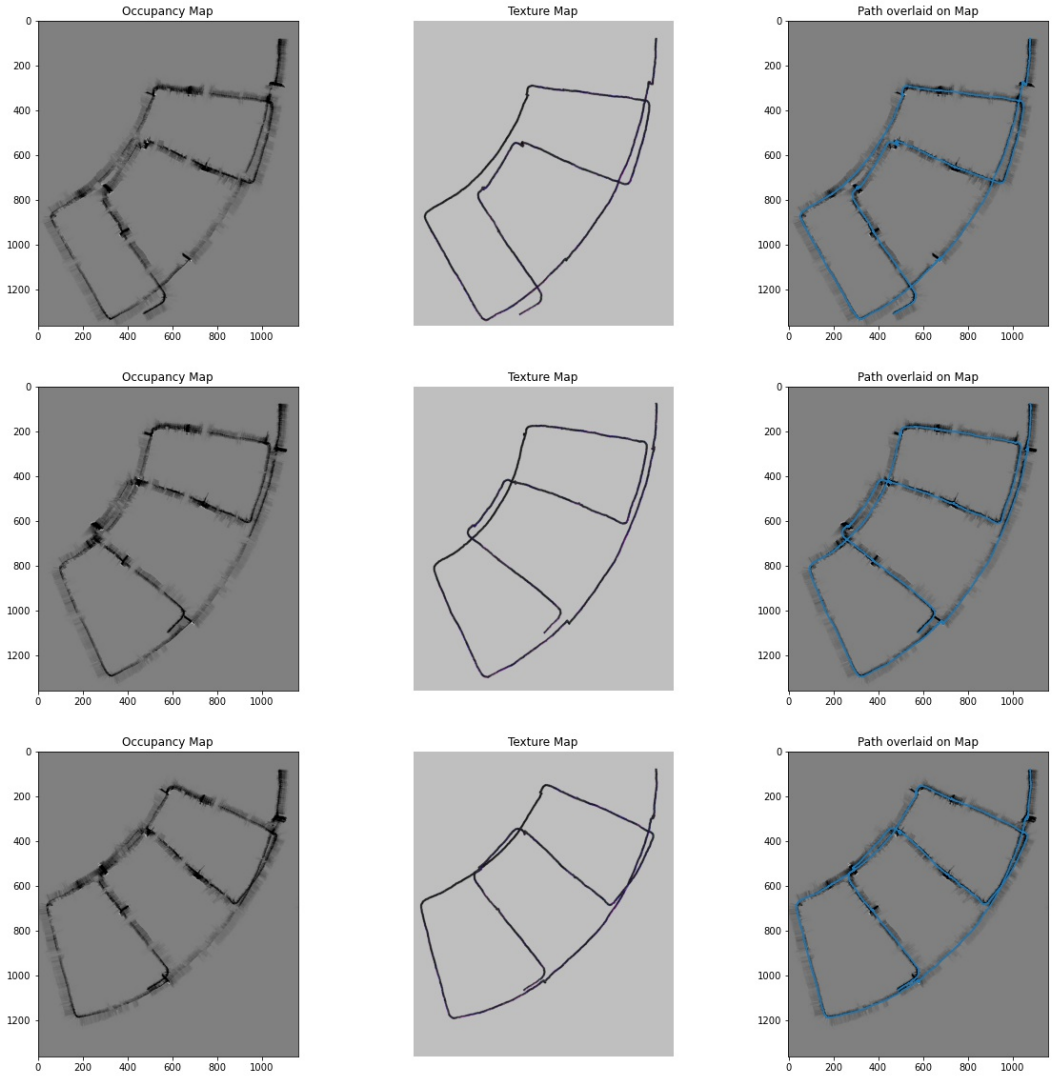

Fig. 2. occupancy map, texture map and path overlayed on the occupancy map for different number of particles. N = 5, 50 , 300 from Left to Right

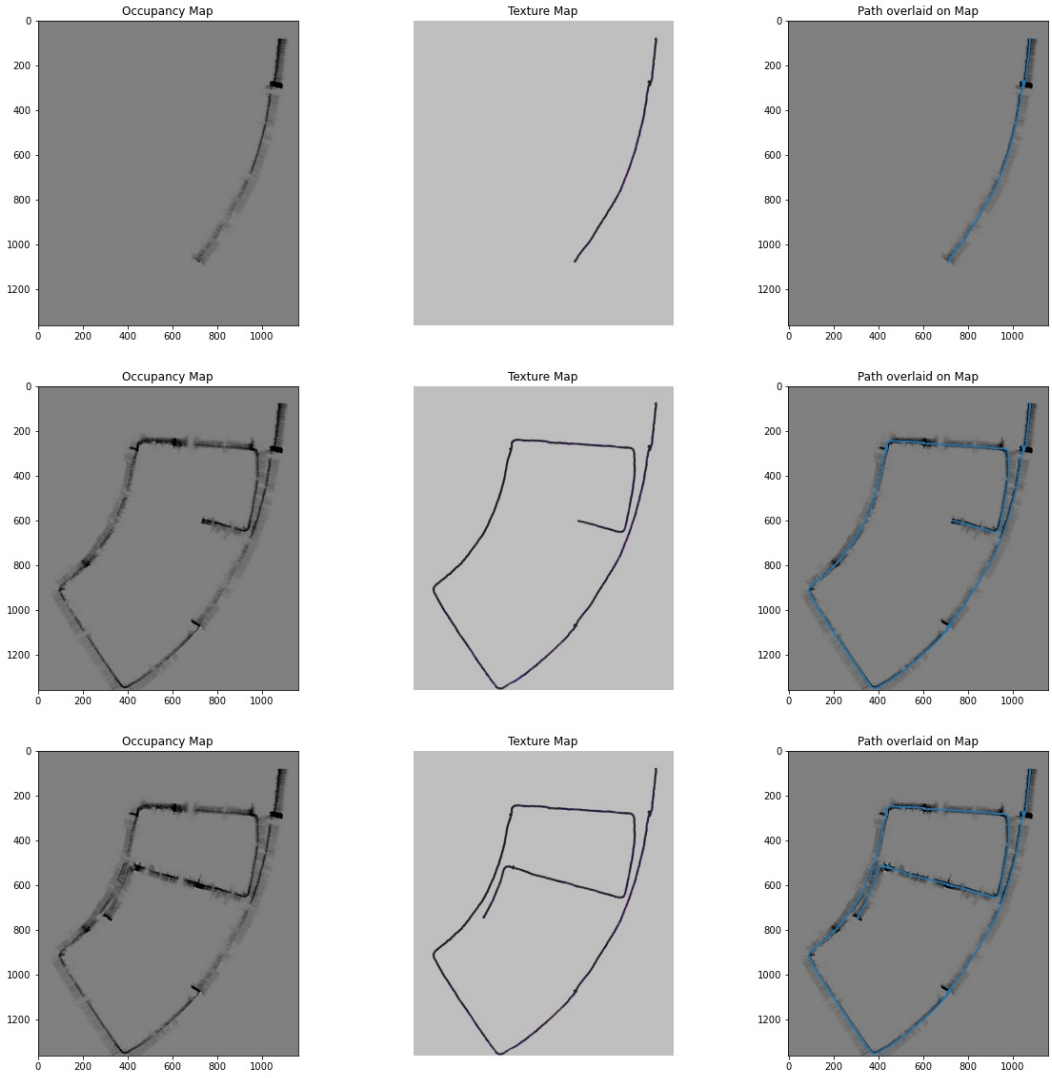

Fig. 3. occupancy map, texture map and the path traced by the most confident particle on the occupancy map at different instances of time. (N=50) (Time increases from top to bottom