ECE 276A PR2: Particle Filter SLAM

Varun Vupparige
Mechanical and Aerospace Engineering
University of California San Diego
La Jolla, USA
vvuppari@ucsd.edu

I. INTRODUCTION

Robotics systems perceive their environments through sensors, and manipulate their environment through things that move. In designing these systems, one has to consider the different kinds of uncertainties such as dynamic and unpredictable environments, noisy sensors, inaccurate actuators and motors, imperfect computational models and many such non deterministic factors. As robots move more towards autonomy space, there would be more emphasis on real time software systems, and thus this makes Simultaneous Localization and Mapping (SLAM) a critical problem in robotics.

Over decades, lot of different SLAM approaches have been researched and have already been in use in many applications such as self-driving cars, unmanned aerial vehicles and underwater surveillance vehicles, Techniques such as Particle Filer, Extended Kalman Filter and neural networks have been researched. In this paper, we describe an implementation of a particle filter for simultaneous localization and mapping (SLAM), and texture mapping by the autonomous car as show in fig1.

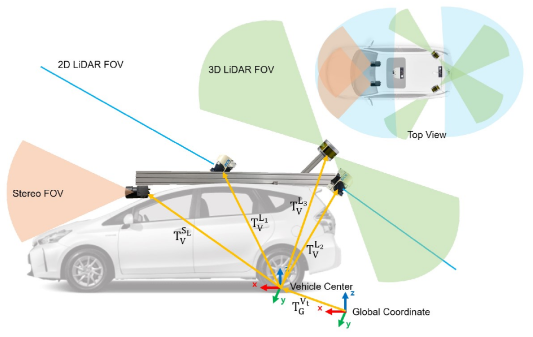


Fig. 1. Autonomous car with lidar 2D, fog and yaw sensors

The car is equipped with front 2-D LiDAR scanner, fiber optic gyro (FOG), and encoders for localization and mapping as well as the stereo cameras for texture mapping. The goal of the project is to use a particle filter with differential-drive motion model and scan-grid correlation observation model for simultaneous localization and occupancy-grid mapping.

II. PROBLEM FORMULATION

In this section, we precisely define the mathematical terms required to describe the problem. Few concepts and knowledge required for defining the problem is also introduced.

A. SLAM problem

The simultaneous localisation and mapping problem can be described as Markov process such that the state x_t is only dependent on previous state x_{t-1}. Given a robots state x_t, and control input u_t, the state at time $t+1$ can be modelled as a probabilistic function given by,

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

where w_t is the sensor/measurement noise. To address the mapping part of the problem, we define an observational model where given robot's pose x_t, observation noise v_t and map of the environment m, we estimate the observations z at time t.

$$z_t = h(x_t, u_t, v_t) = p_f(.|x_t, m) \tag{2}$$

Thus SLAM is the problem of determining the robot state x_t from the control inputs $u_{0:t}$, and determine the environment m from the observations $z_{0:t}$. It can be represented as a probability density function as follows:

$$p(x_t, m|z_{0:t}, u_{0:t}) \tag{3}$$

Using the Markov assumption, this above term can be factorized as shown below. The probability density functions represent in order are prior distribution, observational model, motion model and control policy.

$$\begin{aligned} p(x_{0:T}, m|u_{0:T-1}, z_{0:T}) =& p_0(x_0, m) \prod_{t=0}^{T} p_h(z_t|x_t, m) \\ & \prod_{t=0}^{T} p_h(x_t|x_{t-1}, u_{t-1}) \prod_{t=0}^{T} p_h(u_t|x_t) \end{aligned} \tag{4}$$

B. Occupancy grid mapping

At any time t, we are given lidar scan observations $z_{0:t}$. Using these readings, and our current estimate of the robot's pose (i.e. trajectory) in the world $x_{0:t}$, we generate and maintain a log-odds grid mt that is used to record the robot's current surroundings. Each cell i in the grid mt is modeled as occupied (+1) or free (-1) or unexplored (0) independently of other cells. If we define $gamma_{i;t} := p(m_i = 1|z_{0:t}; x_{0:t})$ and the odd ratio $o_{i,t} := o(m_i|z_{0:t}, x_{0:t})$ is given by:

$$0_{i,t} = \frac{(p(m_i = 1|z_{0:t}, x_{0:t}))}{(p(m_i = -1|z_{0:t}, x_{0:t}))} \tag{5}$$

Here, we can take the log of this expression to update the map log odds. In this way, our update at time t + 1 for any cell of our log-odds grid is simply to add log odds at particular timestamp to the current value of the cell. For the initial cell value, before we have any observations to process, it is fine to initialize the cells to zero. To generate the occupancy grid (a binary image of cells considered either occupied or not occupied), we threshold the log-odds grid based on thresholds chosen via empirical observations.

C. Particle Filter

The probability distribution $p(x)$ of the robot's next location x_{t+1} after the prediction step can be represented as a mixture of delta functions (representing the motion model pf in a general Bayesian filter) weighted by the N particle weights:

$$p_{t+1|t} = \sum_{n=1} (\alpha_{t|t}^{(n)} (\delta((x)); (\mu)_{t+1|t}^{n}) \tag{6}$$

where $(\mu_{t+1|t})$ is the prediction for the nth particle given the motion model. To complete the update step, we simply compute the correlation between each particle's next observation and our current knowledge of the world around us (i.e. compare with the occupancy grid) and weight the particle weights accordingly:

$$p_{t+1|t+1} = \frac{\sum_{n=1} (\alpha_{t+1|t}^{(n)} p_h (z_{t+1} | \mu_{t+1|t}}{\sum_{n=1} (\alpha_{t+1|t} p_h (z_{t+1} | \mu_{t+1|t}} \tag{7}$$

III. Technical Approach

In this section, we will introduce the detailed methodology used to solve the problem explained in the above section. Fig2 gives a bird's eye overview of the solution for the SLAM problem.

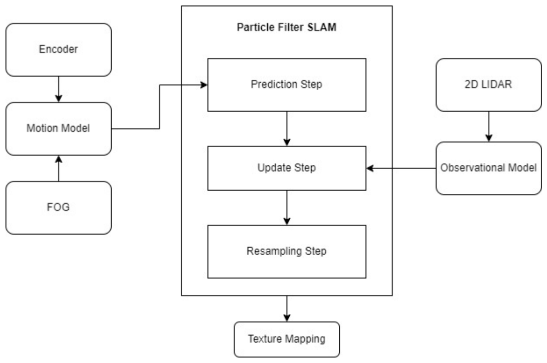


Fig. 2. Bird's eye view of the implemented algorithm

A. Motion and Observational Model

The autonomous car is equipped with different sensors namely a 2D Lidar FOV, linear encoder and fibre opto gyro (FOG) which provide obstacle distance, linear velocity and yaw respectively. Since the timestamps of linear encoder and FOG did not match, we had to synchronise the timestamps. After synchronising the timestamps, linear velocity and angular velocities were calculated by modelling Gaussian noise for each timestamp and particle. Following are the steps involved in particle filter SLAM:

We define the pose of the robot at time t is as a vector $x_t = [x_t; y_t; \theta]$, where x_t and y_t are the coordinates (in meters) of robot in 2D space with respect to a fixed origin, and θ is the heading (in radians) of the robot. Since the odometry contains the difference between position and heading angle at time t and $t-1$, cumulative sum is obtained for the differential drive motion model to predict the motion of particles. Equations of differential drive motion model are as follows:

$$x_{t+1} = x_t + \tau v_t cos(\theta_t) \tag{8}$$

$$y_{t+1} = y_t + \tau v_t sin(\theta_t) \tag{9}$$

For initialising the map, read the lidar data for 1st time stamp. The 2D Lidar sensor has a field view ranging from -5 to 185 degrees with an angular resolution of 0.66 degrees. Data points which too close or too far to the vehicle/car is filtered out. Also, data points corresponding to $z > 1.5m$ are also filtered out since they do not provide any necessary information required for mapping. The coordinates obtained from the 2D Lidar is in polar coordinates and needs to be converted to cartesian coordinates. Furthermore, these coordinates are transformed from sensor frame to vehicle frame using the transformation matrix provided.

$$p_V = T_{VL} p_L \tag{10}$$

The world frame coordinates are converted to cells by initializing a map of size $3000x3000$ metres with a resolution of 1 metre. Log odds are increased at grid cells where LIDAR hit the obstacles and decreased at non occupied cells. The indices of non occupied cells are computed using Bresenham 2D line algorithm.

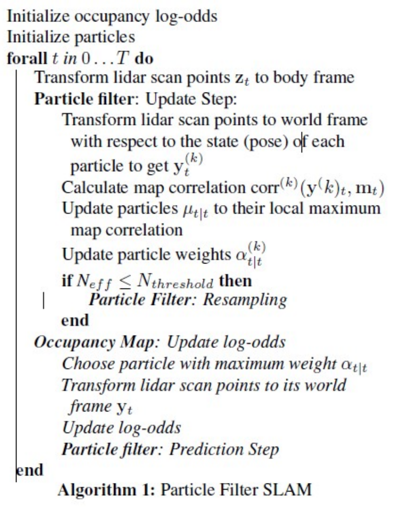
B. Particle Filter

Particle filter methods are a set of Monte Carlo algorithms used to solve filtering problems. It uses a set of particles to represent the posterior distribution of some stochastic process given noisy or partial observations.All particles of the filter are passed through an odometry-based prediction motion model and an observation-correlation based update model. At each time step, the "best" particle is chosen as the closest estimate of the robot's pose. We initialize all N particles to the zero matrix which implies using the robot's initial pose as the origin of the map and the particle weights are all initializes to 1/N .

- Prediction Step: In the Bayes filter, the prediction step is calculated with the prior $p_{t|t}$ over x_t, the control input u_t, and the motion model p_f to compute the predicted density $p_{t+1|t}$ over x_{t+1}. Using the differential drive model, the pose of the robot along with some sensor noise is estimated using the odometry data. In our algorithm, this step is iterated over the range of encoder timestamps.For the noise, we generated independent zero-mean Gaussians for the robot's relative odometry in x, y, and θ (torso yaw).

- Update Step: In the Bayes filter, the update step of the posterior $p_{t+1|t+1}$ over x_{t+1} is calculated with the predicted density $p_{t+1|t}$ over x_{t+1}, the measurement z_{t+1}, and the observation model p_h, to incorporate the measurement information. In this step, correlation between the information obtained from the lidar scans at t+1 to the current occupancy grid is estimated, and each hypothesis a weight proportional to this correlation so we know which ones are most trustworthy. correlation for a given particle and a given scan, we simply computed the sum of cells that agreed between the latest scan and the current binary occupancy grid.
- Resampling step: If we find that most of our particles have weights that make them effectively useless, generate new particles based upon the current distribution of weights of the particles, and give them uniform weights. This is the resampling step of the particle filter. Given a weighted particle set, resampling creates a new particle set with equal weights by adding many particles to the locations that had high weight and few particles to the locations that had low weights. It focuses the representation power of the particles to likely regions, while leaving unlikely regions with only few particles.

The below fig summarises the algorithm of particle filter SLAM.



```
Initialize occupancy log-odds
Initialize particles
**forall** $t$ *in* $0 \dots T$ **do**
    Transform lidar scan points $\mathbf{z}_t$ to body frame
    **Particle filter**: Update Step:
        Transform lidar scan points to world frame
          with respect to the state (pose) of each
          particle to get $\mathbf{y}_t^{(k)}$
        Calculate map correlation corr$^{(k)}(\mathbf{y}^(k)_t, \mathbf{m}_t)$
        Update particles $\mu_{t|t}$ to their local maximum
          map correlation
        Update particle weights $\alpha_{t|t}^{(k)}$
        **if** $N_{eff} \leq N_{threshold}$ **then**
            *Particle Filter: Resampling*
        **end**
    *Occupancy Map: Update log-odds*
        *Choose particle with maximum weight* $\alpha_{t|t}$
        *Transform lidar scan points to its world*
          *frame* $\mathbf{y}_t$
        *Update log-odds*
        *Particle filter: Prediction Step*
end
    **Algorithm 1:** Particle Filter SLAM
```

Fig. 3. Particle Filter: Psuedo Code

C. Texture Mapping

The goal of texture mapping is to form a vector of size $nx3$ where each cell row represents a RGB colour space corresponding to the cell in the occupancy grid map.

IV. Results and Discussion

This section summarises the results of the particle filter slam implemented on localising and mapping an autonomous car. Fig 4 depicts the dead reckoning plot which is the trajectory of the robot according to differential drive model without any motion noise. Fig 5 represents the first unfilitered lidar scan in world coordinates.

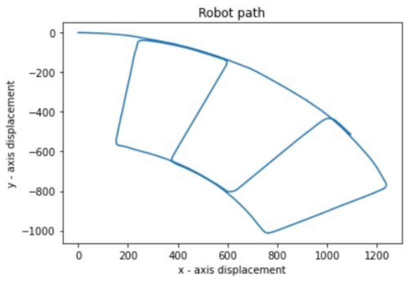


Fig. 4. Dead-reckoning plot of motion model

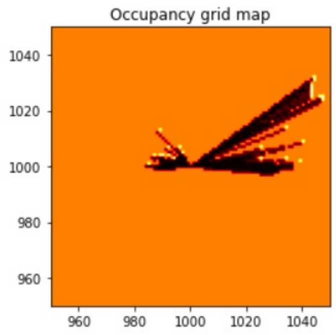


Fig. 5. Lidar scan of 1st timestamp in world coordinates

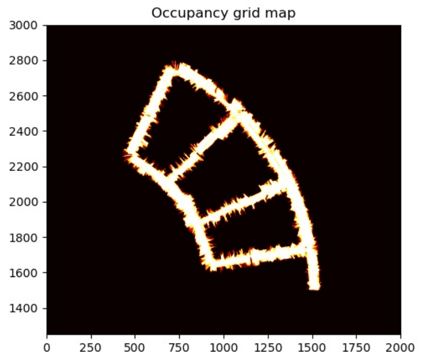


Fig. 6. Occupancy binary map of only lidar scans

Figures below show the results of particle filter SLAM at iterations 20000, 50000 and 95000 respectively with 20 particles and without skipping any sensor data.

Fig 9 and 10 shows the occupancy grid map and binary map for the entire time duration. In fig 10, the blue line represents the trajectory taken by the robot. Due to computational limitations, the algorithm was not implemented for more than 20 particles which would result in a more accurate representation of the map.

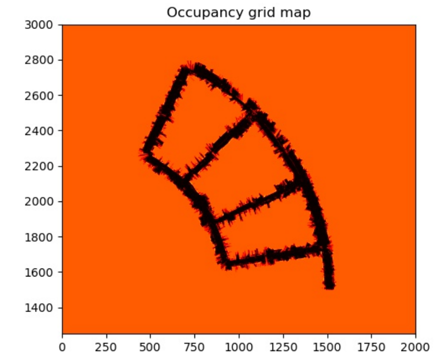


Fig. 7. Occupancy map of only lidar scans

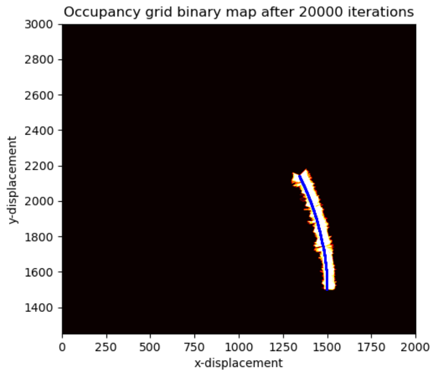


Fig. 8. Occupancy binary grid map at it = 20000

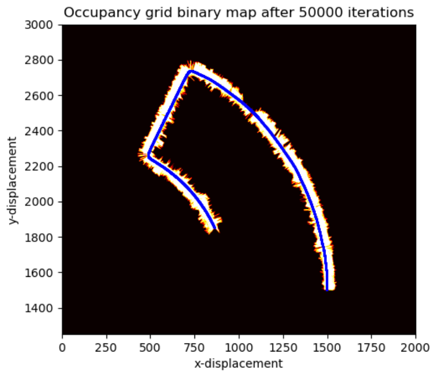


Fig. 9. Occupancy binary grid map at it = 50000

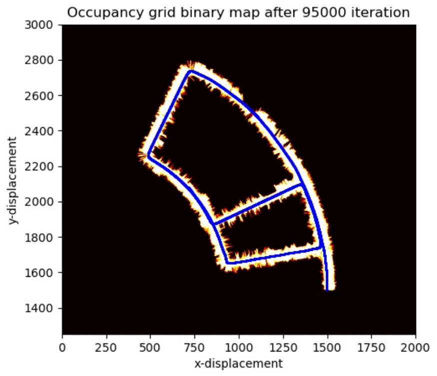


Fig. 10. Occupancy binary grid map at it = 95000

Future scope would include modelling lidar data as a gaussian random variable, filtering the encoder and fog data, implementing a more powerful resampling algorithm.

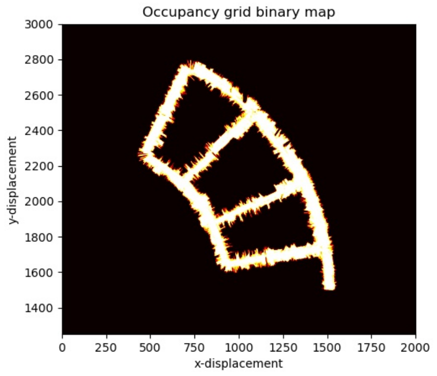


Fig. 11. Occupancy binary grid map

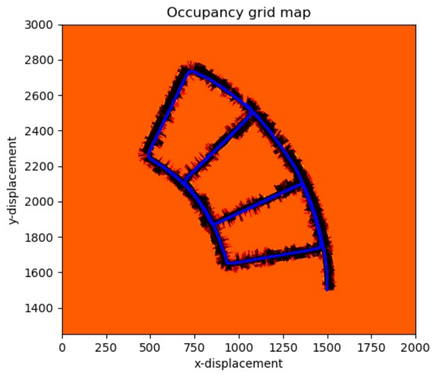


Fig. 12. Occupancy map and robot trajectory