Particle Filter SLAM: An academic project

Orish Jindal
Department of Electrical Computer Engineering
University of California, San Diego
ojindal@ucsd.edu

I. *Abstract*— Autonomous vehicles are equipped with sensors that are used to collect information about the motion and surroundings of the vehicle. With the help of the information collected, our goal is to build a virtual map of the surroundings while simultaneously locating the vehicle in that map. A popular technique which is used for this task is called SLAM: Simultaneous Localization and Mapping. This paper will focus on one of the approaches to solve the SLAM problem for a moving car and the advantages as well as disadvantages of the given implementation.

Index Terms— SLAM, Particle Filter.

II. Importance of SLAM

Development of mobile robots and autonomous vehicles is becoming more and more popular as to replace the humans for the tasks that are monotonous or dangerous. Primary function that these machines need is to navigate in an unknown environment without the help of human. A solution to this problem falls into the family of problems called SLAM. There are various methods in this family such as particle filter, extended Kalman filter, covariance intersection, and GraphSLAM, that are popular for specific type of problems of navigation and creation of a map for any unknown or changing space where a map is either not available or needs to be updated in real-time.

III. Problem Formulation

SLAM is basically a parameter estimation problem for $\mathbf{x}_{0:T}$ (position at time T) and **m (**map of surroundings) given $\mathbf{u}_{0:T-1}$ (control input at time T-1) and observations $\mathbf{z}_{0:T}$ (observations at time T).

There are two parameters that needs to be estimated at each interval: Position of the robot with respect to surroundings and Map of the surroundings in which the robot is moving. They both need the existence of each other to estimate them, so SLAM becomes a chicken - egg problem where it is difficult to keep track of both the parameters simultaneously.

Technically, we need the robot pose to build the map with the help of mapping sensors like LiDAR and again need the map information to estimate the robot pose at any time.

The equation of estimating the position of the robot with highest probability at some time T given the information of all the sensors before that time is as follows:

$$p(\mathbf{x}_{0:T}, \mathbf{m}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_0(\mathbf{x}_0, \mathbf{m})}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}} \prod_{t=0}^{T-1} \underbrace{p(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{control policy}}$$

where the joint pdf is formed using Bayes rule, and Markov assumption properties given the deterministic observation and motion models throughout. For solving this problem, we need to estimate two equations at each time

First: Motion Model

Robot's estimated position for a discrete time interval dataset at each time as a function of its previous position and control input subject to the motion noise is written as follows:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)$$

Second: Observation Model

The observations as a function of robot's position at time t, as a function of its position, map of the environment subject to the measurement noise is as follows:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}_t, \mathbf{v}_t)$$

IV. Technical approach: particle filter SLAM

The goal of the project that is discussed in this paper is to use a particle filter approach with a differential-drive motion model and scan-grid correlation observation model for simultaneous localization and occupancy-grid mapping.

Dataset: We are given datasets of the readings from four sensors namely FOG, Encoder, LiDAR and Stereo Camera. The '.csv' files of first three sensors written above shows the data of each sensor at a given timestamp which are not synchronous with each other. So, the data cleaning was done based on the readings and a master file was prepared after syncing the data at best estimated time intervals that contained all the corresponding readings of three sensors. This makes it easy to formulate the motion model and the observation model to further apply the Bayesian filter technique to solve the SLAM problem. Rest is written stepwise

Step 1 – Trajectory without noise.

Synchronized data of the FOG and Encoder together gives the distance traveled by the vehicle between two timestamps and the direction (as Roll, Pitch and Yaw) in which the vehicle has moved. By ignoring the noise factor of both the sensors, we can obtain the trajectory of the vehicle in ideal conditions by obtaining the x and y coordinates at each timestamp using only the 'Yaw' data. We have ignored the Roll and Pitch data as we already know that the car will neither roll or pitch on the road and the corresponding values that read by the FOG sensor are small disruptions to the ideal motion that happens when car applies breaks and take a sharp turn on a slightly banked road.

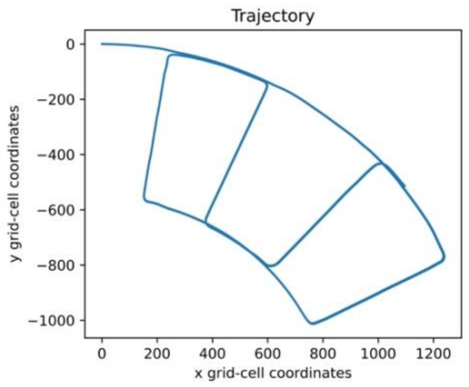
Distance information can be obtained from the encoder readings as follows:

$$\text{meters per tick} = \frac{\pi \times (\text{wheel diameter})}{\text{ticks per revolution}}$$

This is further converted to the velocity by dividing it with the net time take to travel that distance. Also, the FOG will give net change in all the three angles which will give us the angular velocity. A motion model (ignoring noise) is then formulated as follows:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x}_t, \mathbf{u}_t) := \mathbf{x}_t + \tau \begin{bmatrix} v_t \cos(\theta_t) \\ v_t \sin(\theta_t) \\ \omega_t \end{bmatrix}$$

A set of coordinated is obtained via the above written formula which is used to plot the trajectory of the vehicle which is shown below.



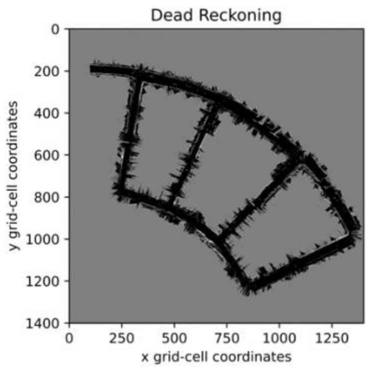
Step 2 – Lidar mapping without considering motion noise.

The LiDAR readings are obtained from the master csv file with synced data. The readings represent the polar coordinates of the rays projected by the sensor that spans from -5 degree to 185 degree with a uniform gap of 0.666 degree between two consecutive rays. A filter was applied on the length of the rays in order to avoid the singularities and edge cases that will constitute to the additional noise. These polar coordinated converted to the cartesian coordinates in the lidar frame which are then converted to the vehicle frame and finally to the world frame. The transformations were done using pose comprising of rotation matrix and position represented as follows:

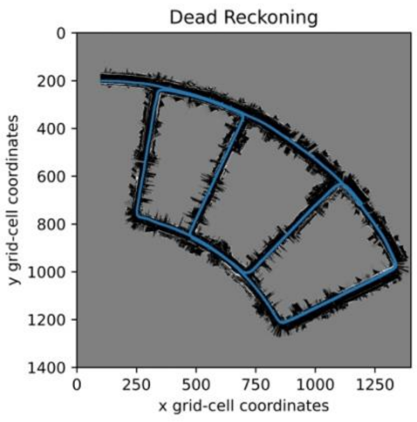
$$\underline{\mathbf{s}}_W = \begin{bmatrix} \mathbf{s}_W \\ 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{p} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{s}_B \\ 1 \end{bmatrix} = T\underline{\mathbf{s}}_B$$

To visualize this, a grid map was initiated the cells of which gets updated with each iteration of robot movement in discrete time using the 'bresenham2D' function. Log odds were used to update the estimate of vacant and occupied cells. In the approach used for writing this paper, twice the log odds were added to the occupied as compared to the log odds deducted from the vacant cells at each iteration. This is done to make our model more precautious about the obstacles that the vacant spaced. Also, clipping is used as an upper bar to avoid the over confidence in our model.

We have the vehicle's pose at each time stamp with which, we can update the grid map with the cartesian coordinates of the endpoints of the rays emitted by the LiDAR sensor with respect to the world frame at each time stamp. By plotting the final map after full iterations, we have a map as follows:



To have a better idea of the estimation, the plotting is done by superimposing the trajectory to the lidar readings. The plot is shown below:



There is an interesting observation that can be noticed from this overlapping of the plots. The map is viewed from the top and the blue line representing the trajectory seems to have always positioned to the right side of the center on the mapping plot. This is because US has a right-side driving rule which means that the vacant area on the right side of the vehicle will always be lesser than the vacant area on the left. This interesting fact softly indicates that our hypothesis is correct so far.

Step 3 – Particle Filter SLAM

In this step, we will apply a more realistic approach to solve the SLAM problem where we consider the possibility of noise in the sensor data while estimating the car pose and map of the surroundings. This will be done using the particle filter technique where we assume the several possibilities for the robot pose, equal to the number of particles with which we initialized the model. Now this step has four parts of its own which runs in loop with each other for each discrete timestamp.

The first part is Prediction.

I this part, we add random gaussian noise to the angular velocity and the linear velocity which constitutes to the noise in the readings of FOG sensor and Encoder respectively. This function will take the state of each particle estimated at the previous timestamp and predict the noise added state according to the control input that is given. Each particle will have equal weight (1/number of particles).

The second part is co-relation.

In this part, we are given the map that has been updated with the best estimated particle state of the previous iteration. The map contains the LiDAR info of each particle till the previous timestamp which will be used to estimate the best particle of current timestamp. A co-relation index is obtained by comparing the map till now and the map suggested by the LiDAR information of each particle at the current timestamp. The weights of the particles are then redistributed according to how much they agreed with our map.

The third part is map updation.

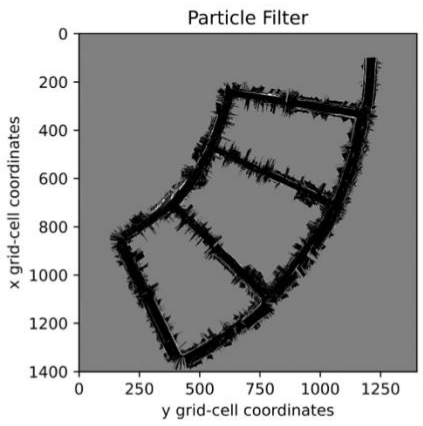
In this part, we select the best particle based on the corresponding weights of each particle that were assigned to the particles in the previous part. Then we update our map believing that the LiDAR rays at current timestamp be emitted from the angle and coordinates of the state of the best particle.

The fourth part is resampling.

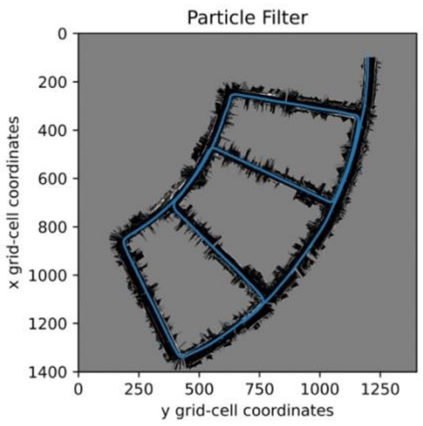
In this part, the weights are redistributed to be equal, and the particles are virtually converged in the direction of the best particle to account for our observation at the current timestamp.

These four steps are repeated in a cyclic order as written until the last timestamp. At the end of all iterations, we will have a map obtained by the best particle of every iteration and with time, each particle will tend to converge to the correct path. This is a particle filter approach to the SLAM problem.

The images of the map then obtained by using 4 particles is shown below. Also, these images are tilted because of the map correlation function that is being given to co-relate the map in order to decide best particle.



To have a better idea of the estimation, the plotting is done by superimposing the trajectory of the best particle to particle filter map. The plot is shown below:



V. Advantages and disadvantages of Particle filter

This method has its own set of advantages and limitations as compared to other methods.

Advantages:

- This is a highly versatile method and works for any observation model and any motion model we can design.
- Particle filters scale very well.
- The algorithm is relatively easy to implement.
- This method works with equal effort even for high dimensional systems as they are independent of the size of the system.

Limitation:

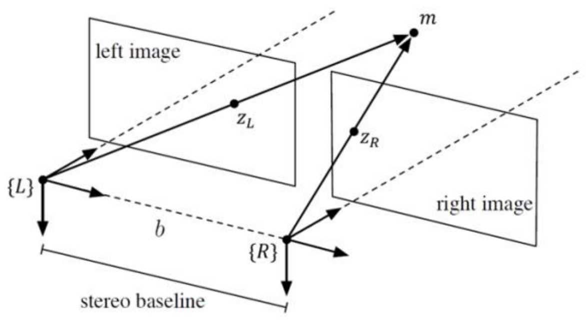
- The assumption of Markov independence is the greatest limitation of this method as we can come across a sensor in which its reading is dependent on each step that has been taken before the current one. Our model will fail to generate good results in this case.

VI. Texture mapping

It is a process of adding color to the map observation sensor. This is done by comparing the physical end points of the LiDAR scan with the actual images of the stereo camera at synchronized timestamps.

In this project, we have to use a stereo camera sensor to perform the texture mapping.

In a stereo camera, there are two perspective cameras rigidly connected to one another with a known transformation. So it gives an advantage of determining the depth of a point from a single stereo observation unlike the normal camera.



The pixel coordinates of a point m in the world frame observed by a stereo camera at position p and orientation R ∈ SO(3) with intrinsic parameters K ∈ R 3×3 are:

$$\underline{\mathbf{z}}_L = K\pi\left({}_oR_rR^\top(\mathbf{m}-\mathbf{p})\right) \qquad \underline{\mathbf{z}}_R = K\pi\left({}_oR_rR^\top(\mathbf{m}-\mathbf{p}) - b\mathbf{e}_1\right)$$

And the final stereo camera model is as follows:

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_ub \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = {}_oR_rR^\top(\mathbf{m}-\mathbf{p})$$

Where, d is the disparity and is noted as

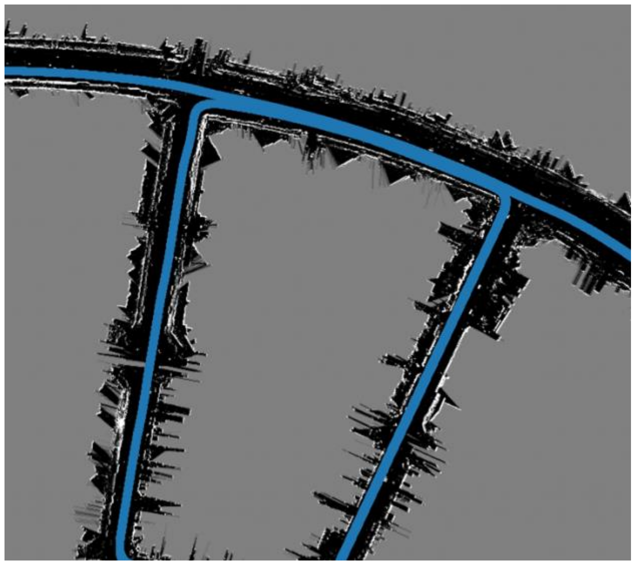
$$\boxed{d = u_L - u_R = \frac{1}{z}fs_ub}$$

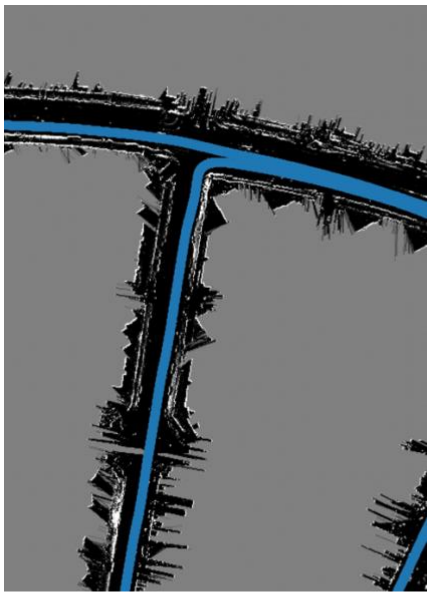
VII. Summary of results

The map plotted without adding noise is almost similar to the map plotted using particle filter slam as it should be. There are very small differences that can only be noticed by zooming the images.

The zoomed comparison images are as follows:

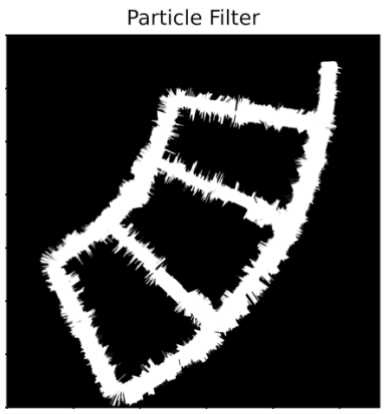
Top image is of particle filter and bottom(on other page) one is without noise.





Like the paths in the below image are less overlapping around the first path split as compared to the upper one.

A binary image of the map just to represent the space that is available for the car to move freely is shown in white whereas the space where the car cannot be present is shown as black. This image is made from the particle filter estimation.


Particle Filter