Particle Filter SLAM

Albert Tan
Electrical and Computer Engineering
University of California, San Diego
San Diego, California
aktan@eng.ucsd.edu

I. Introduction

Autonomous vehicles are equipped with sensors that give information about the vehicle's motion and observations. SLAM (Simultaneous Localization and Mapping) is a technique that uses the data from these sensors to perform mapping and localization, simultaneously. The particle filter, a special case of the more general Bayes filter, is implemented for the task.

II. Problem Formulation

A. SLAM Problem

SLAM is a parameter estimation problem for $\boldsymbol{x}_{0:T}$ and $\mathbf{m}$ given a dataset of the robot inputs $\boldsymbol{u}_{0:T-1}$ and observations $\boldsymbol{z}_{0:T}$. In general, we relate all those parameters as a joint pdf :

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

where the joint pdf is decomposed through conditional probability, Bayes rule, and Markov belief network properties. Depending on the observation model and motion model, different types of bayes filters can be used for SLAM.

B. Localization

Given a map $\mathbf{m}$, a sequence control inputs $\boldsymbol{u}_{0:T-1}$, and a sequence of measurements $\boldsymbol{z}_{0:t}$, infer the robot state trajectory $\boldsymbol{x}_{0:t}$

C. Occupancy Grid Mapping

Given a robot state trajectory $\boldsymbol{x}_{0:t}$ and a sequence of measurements $\boldsymbol{z}_{0:t}$, build a map $\mathbf{m}$ of the environment where the general form of the occupancy grids is:

- Model the map cells m_i as independent Bernoulli random variables

$$m_i = \begin{cases} +1 \text{ (Occupied)} & \text{with prob. } \gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \\ \text{-1 (Free)} & \text{with prob. } 1 - \gamma_{i,t} \end{cases}$$

with the distribution $p(m_i = 1 | z_0 \colon t, x_{0:t})$ being maintained over time

D. Robot Motion Model

To determine how the vehicle moves in the environment, we must use a differential motion model with general form

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x}_t, \mathbf{u}_t)$$

The next state is determined by the input of the previous state into the motion model: $f(x_t, u_t)$. This motion model must be derived using motion data (FOG and encoder).

E. Robot Observation Model

To determine how the vehicle moves in the environment, an observation model $\boldsymbol{z_t}$ must be found below:

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

Due to measurement noise, $\boldsymbol{z_t}$ must be found as a pdf:

$$\mathbf{z}_t \quad \text{has pdf} \quad p_h(\cdot \mid \mathbf{x}_t, \mathbf{m}_t)$$

where the probability of the observation $\boldsymbol{z_t}$ at time t uses the lidar data with the current state $\boldsymbol{x_t}$ and map $\boldsymbol{m_t}$

F. Data Sensors and Dead Reckoning

Since the SLAM problem requires a motion model and observation model, they be evaluated by using the data from sensors above. This introduces the task of processing those sensors to solve the task of dead-reckoning (prediction-only filter). That is, when $\epsilon = 0$ (no noise) and $\# \, of \, particles = 1$.

III. Technical Approach

A. Lidar Processing

The initial mapping is performed using 2-D lidar data provided in lidar.csv

The lidar data as seen in its own frame has coordinates:

$$\bar{\mathbf{m}} = R^\top(\mathbf{m} - \mathbf{p}) = \begin{bmatrix} r\cos\alpha\cos\epsilon \\ r\sin\alpha\cos\epsilon \\ r\sin\epsilon \end{bmatrix}$$

where $\epsilon = 0$ since the lidar reads in 2 dimensions.

To convert from the lidar frame to the vehicle frame, the pose transformation is performed:

$$\begin{bmatrix} S_{world} \\ 1 \end{bmatrix} = {}_{World}T_{Vehicle} * {}_{Vehicle}T_{Lidar} * \begin{bmatrix} S_{lidar} \\ 1 \end{bmatrix}$$

where the poses are given by

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

For ${}_{Vehicle}T_{Lidar}$, the position $\mathbf{p}$ and rotation matrices $\mathbf{R}$ and determined with the lidar2vehicle parameters.

For ${}_{World}T_{Vehicle}$, the rotation matrix is found using a linear transformation:

$$R = R_z(\psi)R_y(\theta)R_x(\phi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

Applying the lidar world frame coordinates to Bresenham produced the first scan below:

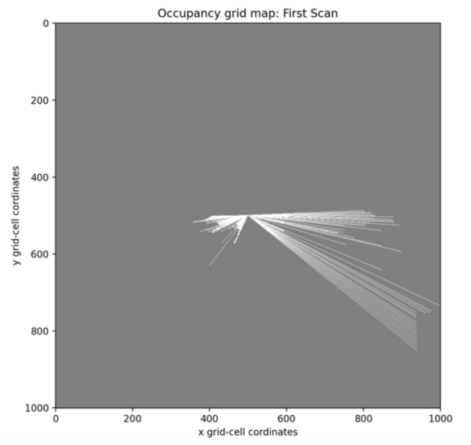

Figure 1. First Lidar Scan in Grid Cells

where grid cells are calculated by

$$\begin{bmatrix} x_{grid} \\ y_{grid} \end{bmatrix} = \begin{bmatrix} floor\ (\frac{x_{world} - a}{resolution}) \\ floor\ (\frac{y_{world} - b}{resolution}) \end{bmatrix}$$

with a and b as region endpoints of the world frame.

B. SLAM Particle Filter

In the project, a Bayes Filter is combined with an occupancy grid mapping to perform simultaneous localization and mapping. The Bayes Filter must maintain the following distribution over time:

$$p(\mathbf{x}_t, \mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1})$$

Rewriting this joint pdf in a similar formulation gives the HMM decomposition:

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

Using a particle filter to maintain the joint distribution, a mixture of particles in the form $\sum \delta(x_t; \mu)$ is used to get the recursive particle prediction and update steps:

- **Prior**: $\mathbf{x}_t \mid \mathbf{z}_{0:t}, \mathbf{u}_{0:t-1} \sim p_{t|t}(\mathbf{x}_t) := \sum_{k=1}^{N} \alpha_{t|t}^{(k)} \delta\left(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)}\right)$
- **Resampling**: If $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha_{t|t}^{(k)}\right)^2} \leq N_{threshold}$, resample the particle set $\left\{\boldsymbol{\mu}_{t|t}^{(k)}, \alpha_{t|t}^{(k)}\right\}$ via stratified or sample importance resampling
- **Prediction**: let $\boldsymbol{\mu}_{t+1|t}^{(k)} \sim p_f\left(\cdot \mid \boldsymbol{\mu}_{t|t}^{(k)}, u_t\right)$ and $\alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)}$ so that:
$$p_{t+1|t}(\mathbf{x}) \approx \sum_{k=1}^{N} \alpha_{t+1|t}^{(k)} \delta\left(\mathbf{x} - \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$
- **Update**: rescale the particle weights based on the observation likelihood:
$$p_{t+1|t+1}(\mathbf{x}) = \sum_{k=1}^{N} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}_{t+1|t}^{(j)}\right)}\right] \delta\left(\mathbf{x} - \boldsymbol{\mu}_{t+1|t}^{(k)}\right)$$

The number of particles n can be selected based on a performance vs computation trade-off. After doing a parameter sweep of n (1 vs 3 vs 50 particles), it was found that more particles does indeed improve SLAM performance, though taking longer to run.

C. Prediction Step with Motion Model and Noise

If a control is given, then the particles will undergo the prediction step. With the weights remaining unchanged, the particle positions will move by $\boldsymbol{\mu}_{t+1|t}^{k} = f(\boldsymbol{\mu}_{t|t}^{k}, \boldsymbol{u}_t + \epsilon_t)$ given noise ϵ_t being added to v_t and w_t . The motion model p_f is given by

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

where $v_t = \frac{\pi dz}{4096 * \tau}$ and ω_t was given directly by FOG. This was then substituted into the prediction equation above. With zero noise and just a single particle, dead reckoning is shown below:

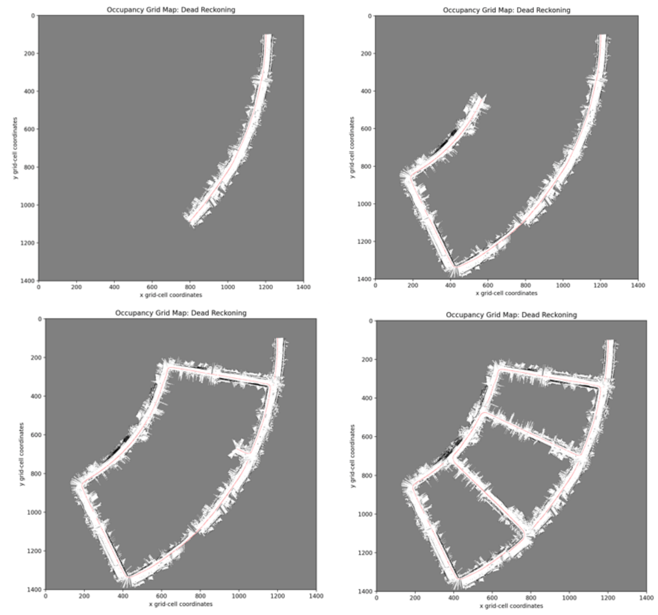

Figure 2. Dead-Reckoning Occupancy Grid over time. Some inaccuracies can be seen in the true trajectory and mapping, which will be corrected with SLAM in the next section

Dead reckoning was implemented as the first step, with zero noise and just a single particle. Because dead-reckoning only involves a prediction-step, it could be improved upon with SLAM.

The next step was to incorporate more particles with noise ϵ_t , which was empirically determined to make sure that there was enough spread between the particles, but not too much noise such that the filter failed to close the loop. An example of excessive noise is shown below:

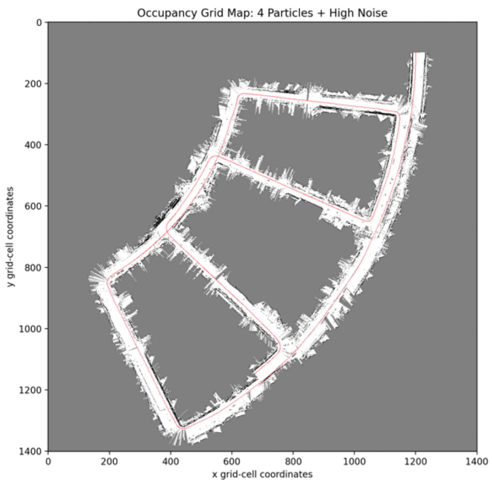

Figure 3. Occupancy Grid with High Variance in Yaw and Velocity, causing the particles to turn prematurely in their trajectory and overshoot the road

The red line trajectory shows that with too much noise, especially in the yaw component from FOG data, the particle will deviate from its proper course and break the feedback loop. The best result was achieved either by scaling the covariance matrix by a factor of 1000x, or having the noise temporally vary with respect to the linear velocity and angular velocity at the particular time. It turned out that empirically scaling the noise such that $\epsilon_{v_t} = \frac{v_t}{200}$ and $\epsilon_{w_t} = \frac{\omega_t}{50}$ so that it was an order of magnitude less than the sensor data itself led to successful results below:

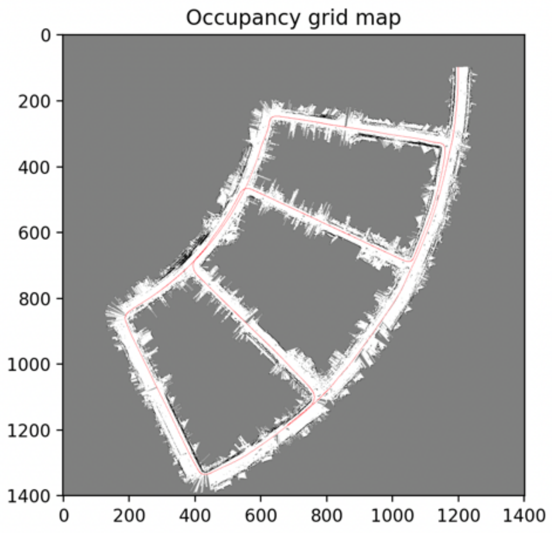

Figure 4. Occupancy Grid with reduced noise, which made the SLAM more stable over time with proper loop closure

D. *Update Step: Laser Correlation and Occupancy Grid*

After the particle has moved according to the prediction step, the weights are then scaled in the update step:

- **Update step**: the particle poses remain unchanged but the weights are scaled by the observation model:

$$\boldsymbol{\mu}^{(k)}_{t+1|t+1} = \boldsymbol{\mu}^{(k)}_{t+1|t} \qquad \alpha^{(k)}_{t+1|t+1} \propto p_h(\mathbf{z}_{t+1} \mid \boldsymbol{\mu}^{(k)}_{t+1|t}, \mathbf{m})\alpha^{(k)}_{t+1|t}$$

The observation model probability was determined by:

$$p_h(\mathbf{z}|\mathbf{x},\mathbf{m}) \propto \exp\left(\text{corr}\left(r(\mathbf{z},\mathbf{x}),\mathbf{m}\right)\right)$$

where $corr(r(z,x),m)$ was computed using the Map Correlation function provided by the PR2Utils.py. Using a 9x9 grid, the maximum correlation was taken for the particle with lidar scans z and used to scale the weights.

Once all the correlations were computed, a softmax was taken over the vector of correlations. The particle with the highest α_k was then selected to project its lidar readings y^k_{t+1} onto the occupancy grid and update the log-odds table, which is modeled in the next section.

One important mechanism to prevent particle depletion is resampling, which localizes the particles as as:

- **Resampling**: If $N_{eff} := \frac{1}{\sum_{k=1}^{N}\left(\alpha^{(k)}_{t|t}\right)^2} \leq N_{threshold}$, resample the particle set $\left\{\boldsymbol{\mu}^{(k)}_{t|t}, \alpha^{(k)}_{t|t}\right\}$ via stratified or sample importance resampling

With SIR resampling, the best result for 50 particles was achieved when $N_{threshold}$ was set to 20% of total particles. This prevented the particles from straying too far away from the correct trajectory by redistributing them at points of high likelihood based on highest particle weights.

E. *Occupancy Grid and Lidar Mapping*

The mapping part of SLAM relies on how the grid is modeled. An occupancy grid is used

- Model the map cells m_i as independent Bernoulli random variables

$$m_i = \begin{cases} +1 \text{ (Occupied)} & \text{with prob. } \gamma_{i,t} := p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \\ \text{-1 (Free)} & \text{with prob. } 1 - \gamma_{i,t} \end{cases}$$

where due to cell independence,:

$$p(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \prod_{i=1}^{n} p(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t})$$

This leads to the log odds occupancy grid mapping:

$$\lambda_{i,t+1} = \lambda_{i,t} \pm \log 4$$

such that anytime a laser hits a free cell, the log odds at that cell increases by $+\log\left(\frac{80\%\ correct}{20\%\ incorrect}\right) = -\log 4$, and increases if the lidar point is an endpoint obstacle

Given log odds $\lambda_{i,t}$, the map $\mathbf{m}$ can be found by

$$m_i = \begin{cases} -1, & \lambda_{i,t} \leq 0 \\ 1, & \lambda_{i,t} > 0 \end{cases}$$

IV. Results

Ideally, the particle filter SLAM should create a more accurate map and trajectory compared to dead-reckoning (1 particle, no noise). This was the case as shown below:

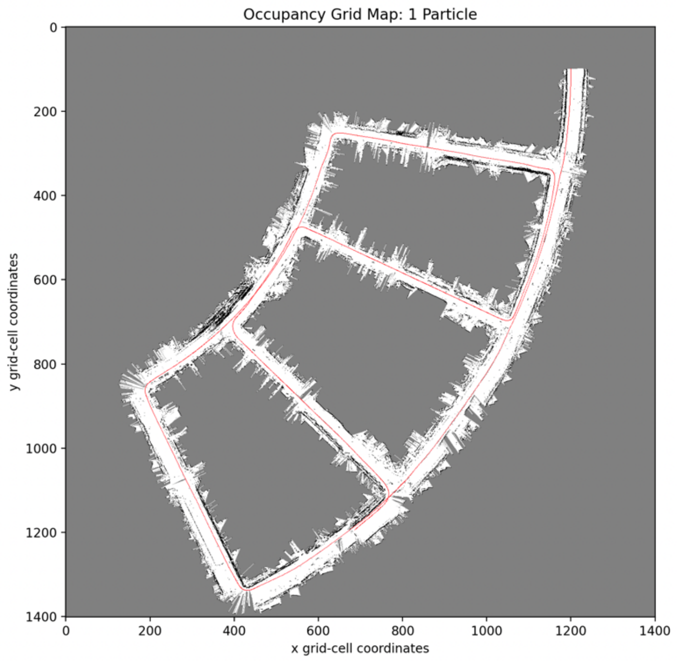

Figure 4. Particle Filter with 1 Particle (Dead Reckoning). Mapping and trajectory is not fully accurate, especially with misalignment at end of route

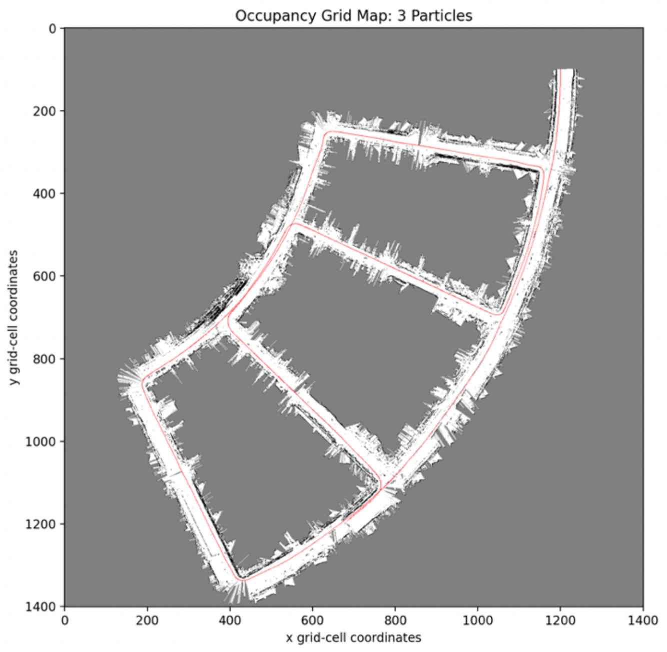

Figure 5. Particle Filter with 3 Particles. Result is slightly improved at the end of the route

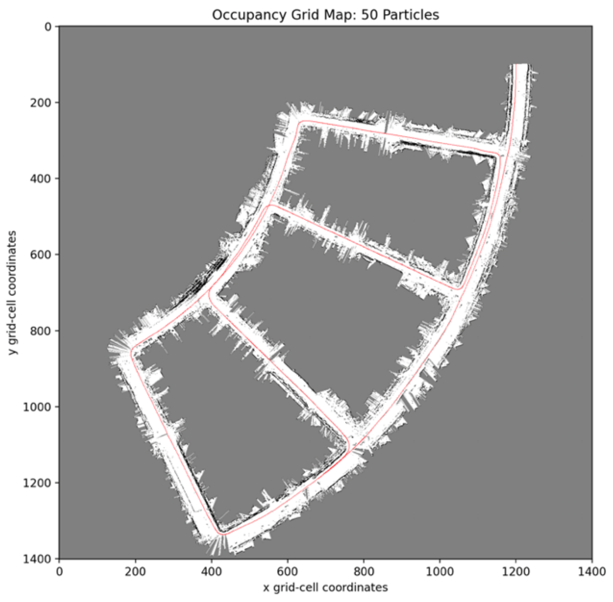

Figure 5. Particle Filter with 50 Particles. Map and trajectory are now fully accurate

The improvements of the particle filter are illustrated at key points of the route:

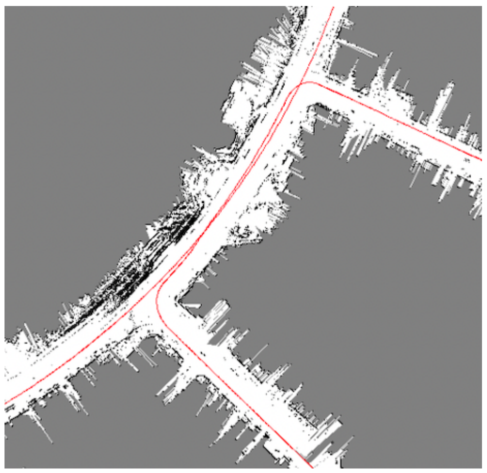

Figure 5. Dead Reckoning result, car fails to stay on right lane

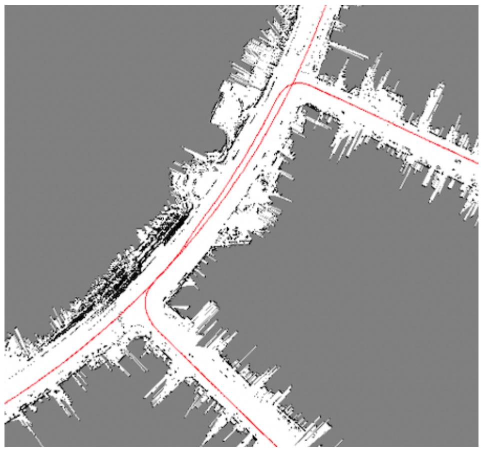

Figure 5. Particle Filter Result: car successfully stays on right lane by using lidar map correlations to localize the 50 particles

Similar improvements for SLAM can be at the very end of the route:

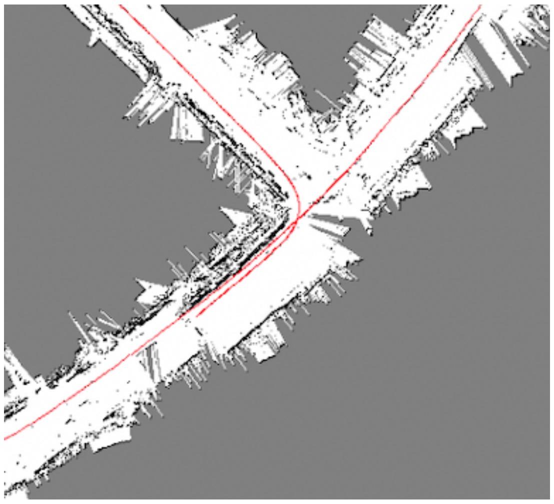

Figure 5. Dead Reckoning result: Trajectory and obstacles are misaligned in the absence of localization correction

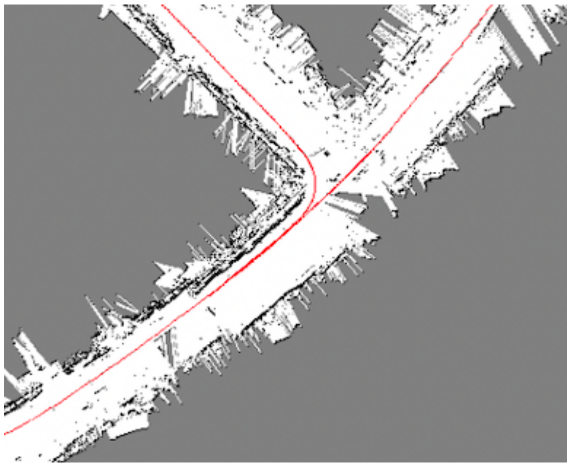

Figure 5. Particle Filter: Trajectory and obstacles are well aligned after lidar corrections

Overall, the particle filter helped to localize the car and make corrections at points where the FOG and Encoder data were inaccurate. This suggests that lidar is a reliable reading that can be used to correct for noisy motion model measurements via SLAM.

While the occupancy grid map discretely defines cells as freed or occupied, the log-odds can be used to display each cell as a smooth continuous gradient of confidence levels. The points of high confidence are colored white (at the free cell, or the roads) and black (at the obstacles, or the edges of the road). The points of low confidence are red (unexplored regions) and orange (regions where lidar data was sparse at the location).

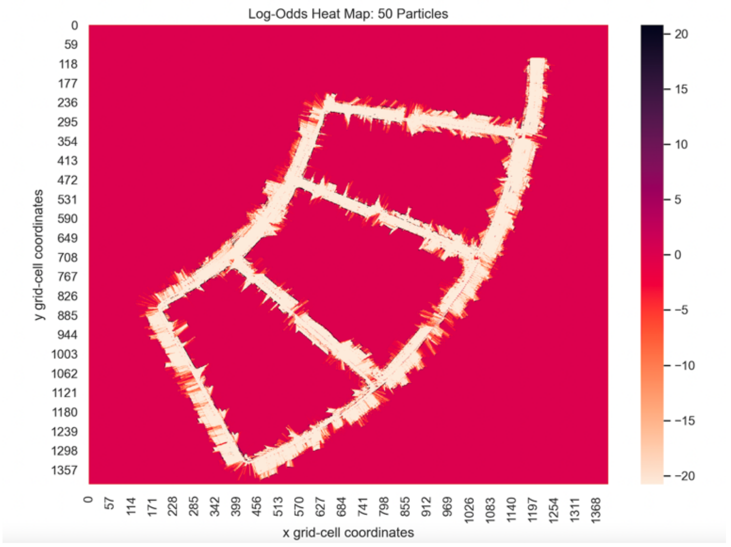

Figure 5. Particle Filter: Trajectory and obstacles are well aligned

Finally, a video of the particle filter SLAM can be seen in the video I uploaded onto youtube:

https://www.youtube.com/watch?v=3HxXUftrKaY&feature=youtu.be&ab_channel=AlbertTan

V. Acknowledgements

This project was done in collaboration with:

- Keshav Rungta
- Ahnaf Ahmed
- ECE 276 Instructors