Bayes Particle Filter SLAM for Autonomous Driving

1st Marcus Schaller
Department of Electrical and Computer Engineering (ECE 276A)
University of California San Diego
La Jolla, U.S.A
mschalle@ucsd.edu

***Abstract*—Simultaneous Localization and Mapping or SLAM plays a crucial role in obtaining a map for autonomous cars. This process allows cars to quickly create a map of their surroundings and correct for errors created by their on-board sensors by localizing themselves on the map. There are a number of techniques used in SLAM to localize a robot and one common technique discussed in this paper is Bayes particle filter. This paper will discuss how the data extracted from a 2D Lidar sensor, encoders, and a fiber optic gyroscope (FOG) is used to localize the car and create a 2D map of the cars environment. It will further discuss how a texture map is created of the vehicle's environment using a stereoscopic camera.**

I. Introduction

SLAM has become a common technique used in many autonomous vehicles. It is crucial for reliably providing an accurate position of where the vehicle is at any given time. While certain sensors like GPS are capable of localizing the vehicle in the world, there are many factors such as the vehicle potentially passing under a bridge that can lead to these sensors becoming inaccurate and unreliable. This is why SLAM is often considered more reliable when compared to these other sensor types. Unlike GPS which locates the vehicle on a map, SLAM creates the map while localizing the vehicle. This inherently creates a sort of chicken vs. egg problem and techniques such as particle filters must be used to create the map and localize the vehicle at the same time [1]. The technique described in this paper is known as a Bayes particle filter. The Bayes filter described in this paper is able to continuously update an autonomous car's location with its most likely position in the world map. This is more effective than relying on a vehicle's motion model to update its position on a map as this often results in errors and contains noise. The Bayes particle filter will further improve this motion model giving an overall more accurate vehicle pose. Finally this project will utilize a stereoscopic camera to create a texture map.

II. Problem Formulation

As stated previously SLAM is a chicken-and-egg problem. This means that often a vehicle that is utilizing SLAM has to generate a map while at the same time localizing itself within that map. This is why SLAM can be broken into two problems that must be formulated. The first of the problems is localization and being able to estimate the robot's trajectory at time *t*. Localization involves estimating a robots state given inputs $\mathbf{u}_{0:T-1}$. These inputs are used in conjunction with the vehicles observations $\mathbf{z}_{0:T}$ to estimate a robots position $\mathbf{x}_{0:T}$ within a map $\mathbf{m}$ which is continuously updated. This mapping is the second problem that must be solved within SLAM. The representation of the SLAM problem can be given as: [4]

$$P\left(x_{0:T}, \mathbf{m}, z_{0:T}, \mathbf{u}_{0:T-1}\right) \tag{1}$$

Due to Markov assumptions the joint probability distribution function can be decomposed as follows:

$$p_0\left(\mathbf{x}_0, \mathbf{m}\right) \prod_{t=0}^{T} p_h\left(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m}\right) \prod_{t=1}^{T} p_f\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}\right) \prod_{t=0}^{T-1} p\left(\mathbf{u}_t \mid \mathbf{x}_t\right) \tag{2}$$

Each of these probabilities represent the following:

$$\text{Prior} = p_0\left(\mathbf{x}_0, \mathbf{m}\right) \tag{3}$$

$$\text{Observation Model} = p_h\left(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m}\right) \tag{4}$$

The observation model is further expanded as:

$$\mathbf{z}_t = h\left(\mathbf{x}_t, \mathbf{m}_t, \mathbf{v}_t\right) \sim p_h\left(\cdot \mid \mathbf{x}_t, \mathbf{m}_t\right) \tag{5}$$

Where *ph* is a probability density function and v_t = observation noise.

$$\text{Motion Model} = p_f\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}\right) \tag{6}$$

The motion model can also be further expanded as:

$$\mathbf{x}_{t+1} = f\left(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t\right) \sim p_f\left(\cdot \mid \mathbf{x}_t, \mathbf{u}_t\right) \tag{7}$$

Where the w_t is represented as motion noise.

$$\text{Control Policy} = p\left(\mathbf{u}_t \mid \mathbf{x}_t\right) \tag{8}$$

Each of these probabilities rely on the on-board sensors as well as an accurate map to be correct. This project focuses on using Bayesian particle filtering and tracks a distribution over state x_t and *m*.

$$p\left(x_t, m \mid z_{0:t}, u_{0:t}\right) \tag{9}$$

A. Provided Data and Sensors

The sensors used in this project include a fiber optic gyroscope as well as encoders on the right and left wheels of the vehicle. The purpose of these sensors are to allow for estimation of the pose of the vehicle in the world frame. The encoder provides how many times the wheels have turned given in the form of encoder ticks ϕ. The gyroscope provides $\Delta\theta$ of the car between each time step. This data will be used to create the motion model for the vehicle. The vehicle also contains a 2D lidar sensor that has a field of view of 190 degrees and an angular accuracy of 0.666 degrees. The lidar sensor also has a maximum range of 80 meters. This sensor will provide observations $\mathbf{z}_{0:T}$ in the SLAM model. Finally, the vehicle is equipped with a stereo camera that has a baseline of approximately 475mm. The stereoscopic camera which contains a projection matrix with which the focal length can be extracted. Using this a texture map can be made.

B. Localization and Occupancy Mapping

Occupancy mapping provides a view of what the car has seen and translates it to a birds eye view of the car. The occupancy map is made up of individual cells of a certain resolution. Each cell has a probability value assigned to it and it can be assumed that each cell value is independent conditioned on the robot trajectory.

$$p\left(\mathbf{m} \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right) = \prod_{i=1}^{n} p\left(m_i \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right) \tag{10}$$

This probability is used to help create a laser correlation model which helps provide more accurate localization. The laser correlation model sets the likelihood of a laser scan z propertional to the correlation between the scan's world frame projection $y = r(z, x)$ via the robot pose x and the occupancy grid m.

$$p_h(\mathbf{z} \mid \mathbf{x}, \mathbf{m}) \propto \exp(\operatorname{corr}(r(\mathbf{z}, \mathbf{x}), \mathbf{m})) \tag{11}$$

C. Texture Map

In addition to creating a map using SLAM this project also involved creating a texture map. Texture maps involve using the stereo camera to calculate the position of a pixel in 3D space and transforming the RGB value over that pixel into the world map. Given each pixel the disparity is represented as the following where u_L and u_R represent the horizontal pixel position in both the right and left pixel frame [2]

$$d = u_L - u_R = \frac{1}{z} f s_u b \tag{12}$$

This leads to solving the following problem which when solved returns the x and y position of the pixel.

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & f_{S_v} & c_v & 0 \\ 0 & 0 & 0 & f_{S_u} b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{13}$$

Once both the pixel map and the texture map problems have been solved the vehicle path can be considered properly mapped.

III. Technical Approach

In order to implement SLAM on the vehicle a number of steps were taken. First rotation calculations were made to translate both the lidar scan and the vehicle path into the world frame. Next, using the vehicles encoders and gyroscope a motion model was created. This motion model was used to implement Bayes particle filter SLAM which corrected the vehicles position based on observations from the lidar readings. Finally a texture map was created of the entire map showing the surface which the car was traveling on.

A. Lidar Scan and Vehicle Orientation

In order to translate the lidar into the world frame homogeneous transformation matrices are used. The lidar scan is a vector in $S \in \mathbb{R}^{286}$ and scans in a circle from -5° to 185° around the lidar. These scans are filtered out if any scan has a distance longer than 75m or shorter than 2m. The lidar sensor scans with polar coordinates of radius *r* and angle θ that must be converted to Cartesian using the following simple equations.

$$\mathrm{x} = \mathrm{r}\cos\theta \tag{14}$$

$$\mathrm{y} = \mathrm{r}\sin\theta \tag{15}$$

These coordinates are translated from the lidar frame to the vehicle frame using the following translation matrix *T*.

$$T_v = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{16}$$

Where t_x, t_y, t_z represent the transition array from the lidar to the gyro x, y, and z, direction respectively. The gyro was determined to coincide with the vehicle frame for simplicity. After this a rotation matrix was made given γ = roll, β = pitch, and α = yaw the rotation matrix *R* is defined as

$$\begin{aligned} R_x(\gamma) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \\ R_y(\beta) &= \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\ R_z(\alpha) &= \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{17}$$

$$R_v = R_z(\alpha) R_y(\beta) R_x(\gamma) \tag{18}$$

As the lidar to vehicle rotation matrix was given, R can be defined as.

$$R_v = \begin{bmatrix} 0.00130201 & 0.796097 & 0.605167 \\ 0.999999 & -0.000419027 & -0.00160026 \\ -0.00102038 & 0.605169 & -0.796097 \end{bmatrix} \tag{19}$$

Combining these matrices a homogeneous transformation matrix is made.

$$\mathbf{H_v} = \begin{bmatrix} \mathbf{R_v} & \mathbf{T_v} \\ \mathbf{0} & 1 \end{bmatrix} \tag{20}$$

By taking the dot product between H and the lidar Cartesian points the position of the lidar scan can be calculated in the vehicle frame.

$$\begin{bmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{bmatrix} = H_v \cdot \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} \tag{21}$$

When the vehicle is not within the world frame or when t>0 the process is repeated using the vehicles position to translate the points into the world frame. Given θ is the angle of the vehicle with respect to the world frame and X_w and Y_w represent the estimated vehicle position within the world frame R_w and T_w are once again calculated as the following. As the world map is 2D, the z component is not needed.

$$T_w = \begin{bmatrix} X_w \\ Y_w \\ 0 \end{bmatrix} \tag{22}$$

$$R_w = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Once again a homogeneous transformation matrix is made.

$$\mathbf{H_w} = \begin{bmatrix} \mathbf{R_w} & \mathbf{T_w} \\ \mathbf{0} & 1 \end{bmatrix} \tag{24}$$

Followed by the dot product to calculate the position of the lidar scan points in the world frame.

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = H_w \cdot \begin{bmatrix} X_v \\ Y_v \\ Z_v \\ 1 \end{bmatrix} \tag{25}$$

This returns the scan in the world frame which can then be plotted on the occupancy map.

B. Motion Model

The motion model is crucial to estimating the position of the vehicle in each state. In order to compute the vehicle pose, encoders and a fiber optic gyro are used to calculate the velocity *V*, the angle of the vehicle θ, and the angular velocity of the vehicle ω at time *t*. The dataset provides Δt which is the change in time and ϕ = which is the encoder tick count. Using this along with wheel diameter D, the meters per tick coefficient can be calculated as:

$$\text{Meters per Tick } \Omega = \frac{(\pi \times D)}{\phi} \tag{26}$$

Using Ω, the displacement of each wheel between time stamps is calculate as:

$$d_R = \Omega \times (\phi_{tR} - \phi_{tR-1}) \tag{27}$$

$$d_L = \Omega \times (\phi_{tL} - \phi_{tL-1}) \tag{28}$$

The instantaneous velocity of each wheel is calculated as:

$$V_R = \frac{d_R}{\Delta t} \tag{29}$$

$$V_R = \frac{d_L}{\Delta t} \tag{30}$$

The average of the right and left wheel velocities will return the linear velocity of the center of the car.

$$\mathrm{V} = \frac{V_L + V_R}{2} \tag{31}$$

The angle and angular velocity is calculated from the gyroscope which gives $\Delta\theta$.

$$\theta = \theta_{t-1} + \Delta\theta \tag{32}$$

$$\omega = \frac{\Delta\theta}{\Delta t} \tag{33}$$

Finally the data is plugged into the discrete-time differential drive kinematic model which returns the position and the control state of the vehicle at time *t+1*. With τ being represented as the time between time steps [2]

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{x}_t + \tau \begin{bmatrix} v_t \operatorname{sinc}\left(\frac{\omega_t \tau}{2}\right) \cos\left(\theta_t + \frac{\omega_t \tau}{2}\right) \\ v_t \operatorname{sinc}\left(\frac{\omega_t \tau}{2}\right) \sin\left(\theta_t + \frac{\omega_t \tau}{2}\right) \\ \omega_t \end{bmatrix} \tag{34}$$

Completion of this step allows the X and Y position of the car to be plotted to anticipate the estimated motion of the car as seen in figure 1.

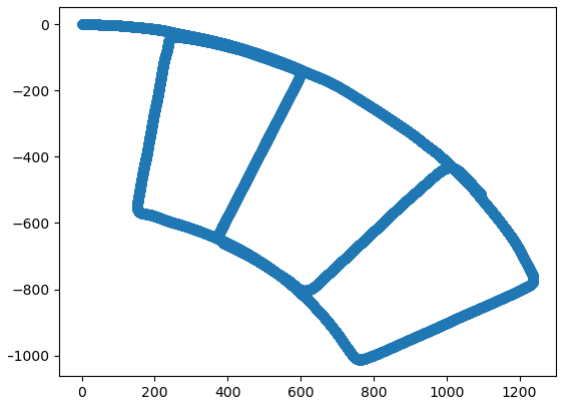

Fig. 1. Dead Reckoning of Vehicle

This model is not always fully accurate as often the sensors on the vehicle have internal noise and drift. The Bayes particle filter is able to add noise to the model and correct the vehicle's position based on the observations provided from the lidar scanner.

C. Bayes Particle Filtering

Bayes filtering is a probabilistic technique used for estimating the state of the vehicle. The Bayes filter keeps track of $p_{t|t}(x_t)$ and $p_{t+1|t}(x_{t+1})$ using a prediction step and an update step to incorporate the measurements [3]. Used in conjunction with particle filters it is possible to calculate the probability $\alpha^{(k)}$ that x is at particle location $\mu^{(k)}$. Using the following notation one can represent the probability mass function of weights over a set of particles.

$$p(\mathbf{x}) := \sum_k \alpha^{(k)} \delta\left(\mathbf{x} - \boldsymbol{\mu}^{(k)}\right) \tag{35}$$

Where δ is the Dirac delta function:

$$\delta(x) := \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases} \qquad \int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{36}$$

This notation will also be used in the Bayes filter to derive the filter. The Bayes filter includes two steps, the prediction step and the update step, both of which follow this notation.

1) Prediction Step: Given a prior density $p_{t|t}$ over x_t and a control input u_t, the motion model p_f is used to compute the predicted density $p_{t+1|t}$ over x_{t+1}

$$p_{t+1|t}(\mathbf{x}) = \int p_f\left(\mathbf{x} \mid \mathbf{s}, \mathbf{u}_t\right) p_{t|t}(\mathbf{s}) d\mathbf{s} \tag{37}$$

Plugging particle representation of $p_t|t$ in the Bayes filter prediction step we get:

$$= \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_f\left(\mathbf{x} \mid \boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t\right) \tag{38}$$

Since $p_{t+1|t}(x)$ is a mixture pdf with components $p_f(\mathbf{x} \mid \boldsymbol{\mu}_{t|t}^{(k)})$ it is possible to approximate it with particles by drawing samples from it to give:

$$p_{t+1|t}(\mathbf{x}) \approx \sum_{k=1}^{N_{t+1|t}} \alpha_{t|t}^{(k)} p_f\left(\mathbf{x} \mid \boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{u}_t\right) \tag{39}$$

In this project the prediction step is calculated by adding noise to x, y, and θ. The noise is represented as Gaussian noise with a mean of 0 and standard deviation equal to $1/10^{th}$ of the instantaneous velocity. This prevents the vehicle position from drifting when it is stopped in traffic. $\mu_{t+1|t}$ is then approximated by extracting random particle samples from within this noise. This can be formulated as.

$$\mu_{t+1|t}^{(k)} = \mu_{t|t}^{(k)} + \mathbf{u}_t + \mathbf{w}_t, \mathbf{w}_t \sim \mathcal{N}(0, \frac{\mathbf{V}(\mathbf{X}, \mathbf{Y}, \theta)}{10}) \tag{40}$$

After the noise is added there is an array of k particles extracted. The particle filter used in this project utilized k = 50 particles. In the implementation described in this paper the particle with the maximum weight is taken.

$$\mathbf{x}_{t+1|t} = \mu_{t+1|t}^{(k)}, k = \operatorname{argmax}\left(\alpha_{t+1|t}^{(k)}\right) \tag{41}$$

In this project $\mathbf{x}_{t+1|t}$ can be considered the best trajectory out of all of the possible trajectory points drawn from with range proposed in equation (40). After this is completed the weights need to be rescaled which is completed in the next step.

2) Update Step: Given a predicted density $p_{t+1|t}(x)$ over x_{t+1} and measurement z_{t+1}, the observation model p_h can be used to obtain the updated density $p_{t+1|t+1}$ over x_{t+1}

$$p_{t+1|t+1}(\mathbf{x}) = \frac{p_h\left(\mathbf{z}_{t+1} \mid \mathbf{x}\right) p_{t+1|t}(\mathbf{x})}{\int p_h\left(\mathbf{z}_{t+1} \mid \mathbf{s}\right) p_{t+1|t}(\mathbf{s}) d\mathbf{s}} \tag{42}$$

Plugging the particle representation into the Bayes filter update step yields:

$$= \sum_{k=1}^{N_{t+1|t}} \left[\frac{\alpha_{t+1|t}^{(k)} p_h\left(\mathbf{z}_{t+1} \mid \mu_{t+1|t}^{(k)}\right)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(j)} p_h\left(\mathbf{z}_{t+1} \mid \mu_{t+1|t}^{(j)}\right)} \right] \delta\left(\mathbf{x}; \mu_{t+1|t}^{(k)}\right) \tag{43}$$

This pdf is a delta mixture therefore no approximation is made. The update step will not change the particle positions but only their weights. To correlate this to the project described in this paper the probability p_h is obtained by computing the local maximum by measuring the correlation of 9x9 pixels around each particle in the world frame. The softmax function is then used on this correlation to calculate p_h. Finally the weights are updated and returned along with the particle state which is determined to have the highest correlation within the world frame.

3) Particle Resampling: Particle depletion occurs when most of the updated particles weights become close to zero because there are not enough particles. To combat this particle resampling must be done to create a new particle set with equal weights. This resampling adds many particles to locations that had high weights and few particles to locations that had low weights. Particle resampling occurs when N_{eff} is less than the threshold. N_{eff} is calculated as:

$$N_{eff} := \frac{1}{\sum_{k=1}^{N} \left(\alpha_{t|t}^{(k)}\right)^2} \tag{44}$$

In this project the threshold was determined to be 20% of the total particles or 10. Stratified and Systematic Resampling was used to resample the weights in this project. This method guarantees that samples with large weights will appear at least once and those with small weights at most once.

D. Occupancy Grid Mapping

This step begins by initializing conditions for the occupancy grid map. In this project since the area of the path was so large it was determined that a the map would be 1400 x 1400 meters with a resolution of 1 meter. The lidar scan in the position was mapped into the occupancy grid by plugging in each relevant point whether that be a point given by the lidar or the robot's position in the world frame. These points can by modeled as x_t and y_t in order to transfer them into the occupancy map the following equations was used.

$$x_p = \operatorname{ceil}\left(\left(x_t - x_{\min}\right) / \textit{resolution}\right) \tag{45}$$

$$y_p = \text{ceil}\left(\left(y_t - y_{\min}\right)/\textit{resolution}\right) \tag{46}$$

The vehicle origin is calculated as well as each of the lidar points. Using the Bresenham 2D ray tracing algorithm a path is traced between the vehicle position to the ending lidar point. The final lidar points of each ray is determined to be occupied by a object or barrier and each of the points traced in between the vehicle and the object are determined to be free. The odds of each point are updated with +log(4) if occupied and -log(4) if free. This way the probability that a cell is occupied can be calculated as:

$$\gamma_{i,t} = p\left(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}\right) = \sigma\left(\lambda_{i,t}\right) = \frac{\exp\left(\lambda_{i,t}\right)}{1 + \exp\left(\lambda_{i,t}\right)} \tag{47}$$

Finally a custom color-map was used to mark out the vehicle path, occupied cells, free cells, and undiscovered cells. Every 100th frame was saved as a .png file to be made into a video which shows the SLAM algorithm at work and is attached to this project.

E. Texture Mapping

Texture mapping involves using the stereo cam to calculate a pixel location in the world frame and overlay a map with the pixel's RGB value. In order to do this the opencv was used to generate a disparity map for each pair of stereo cam pictures. These maps gave the disparity of the pixels between the two figures and using the following equation in conjunction with the focal length fs_u and the baseline b it is possible to calculate the z depth. First however the projection matrix must be used to find focal length. The projection matrix can be represented as the following:

$$P = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{48}$$

This allow one to easily compute the depth z of the pixel from the camera frame.

$$z = \frac{fs_u b}{disparity} \tag{49}$$

Following this u_L and v_L were calculated in the left camera frame. These correspond to the distance from optical center of the left camera frame in the x and y axis respectively to the desired pixel. In order to do this open cv was used to extract the region props of each disparity mask and the coordinates of the pixels were subtracted from the center pixel of the camera frame which in this experiment was (640, 280). Once this information is found the x and y position of the pixel with respect to the camera origin can be found as.

$$x = z\left(\frac{u_L - c_u}{fs_u}\right) \tag{50}$$

$$y = z\left(\frac{v_L - c_v}{fs_v}\right) \tag{51}$$

After this is completed the x,y,z coordinates can be transferred from the camera frame to the world frame using the methods described in part A of this section. However the rotation matrix and the transformation matrix to orientate the camera into the vehicle frame is given as the following.

$$R = \begin{bmatrix} -0.00680499 & -0.0153215 & 0.99985 \\ -0.999977 & 0.000334627 & -0.00680066 \\ -0.000230383 & -0.999883 & -0.0153234 \end{bmatrix} \tag{52}$$

$$T = \begin{bmatrix} 1.64239 \\ 0.247401 \\ 1.58411 \end{bmatrix} \tag{53}$$

IV. Results

The results of this project were very promising. To help describe these results a gif file of showing the SLAM update and the texture map update were included with this report.

A. SLAM results

The parameters used in the SLAM model are as followed:

TABLE I

Log-odds update parameters	Occupied	Free	Unexplored
Values	+log(4)	-log(4)	0

The log-odds maximum and minimum values were 100 and -100 respectively.

TABLE II

Motion model noise	X	Y	θ
Upper bound	$V_x/10$	$V_y/10$	$V_\theta/10$
Lower bound	$V_x/10$	$V_y/10$	$V_\theta/10$

The following are a couple of screenshots of the occupancy map. As it can be seen the red line shows the position in the world frame and the white represents a a occupied zone and black a free zone.

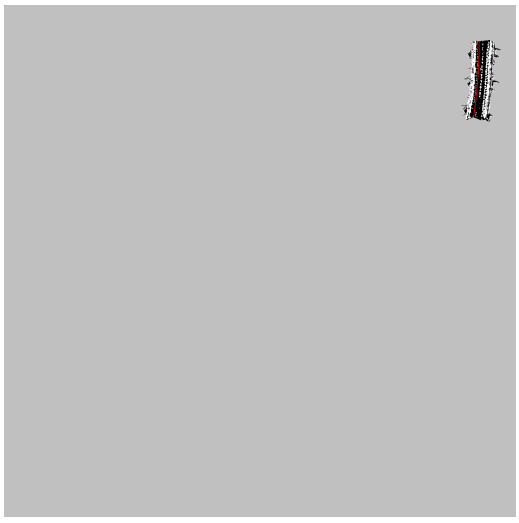

Fig. 2. Early Occupancy Map

Fig. 3. Half Occupancy Map

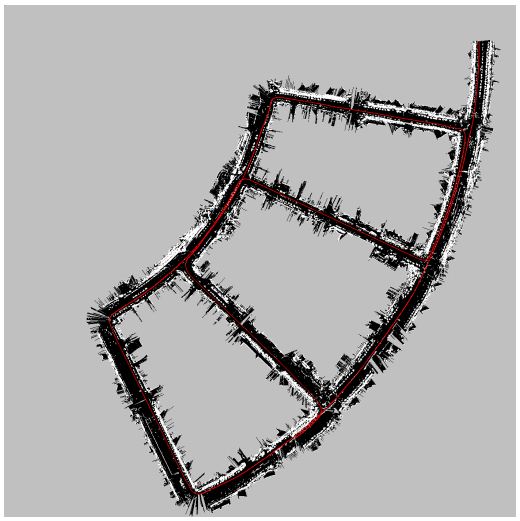

Fig. 4. Full Occupancy Map

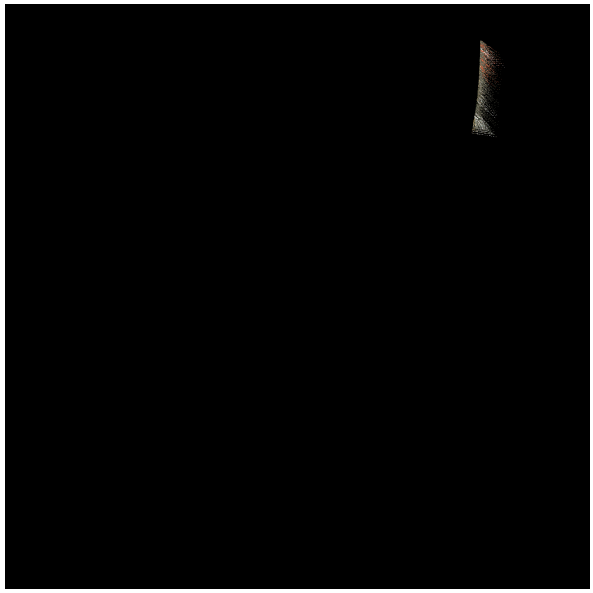

Fig. 5. Early Occupancy Map

Fig. 6. Early Occupancy Map

B. Texture Mapping

Below are screenshots of the texture map at different stages of the vehicle's path. The last few frames of the lidar pictures are out of sync so therefore are not pictures

C. Possible improvements

[h] There are a number of updates that can be made to improve this project. First and foremost the code needs to be further optimized. At 50 particles and using using every 5th lidar scan the algorithm takes about 1 hour and 13 minutes to fully run. This can be accomplished by further vectorizing the mathematical computations as well as possibly implementing multi-threading so the matrix calculations can be done in parallel with the Bayes particle filter algorithm. Further, the resolution of the occupancy map is 1 meter which is quite large. This can be attributed to the fact that the vehicle path covers an area of about 1400 x 1400 meters. Implementing a dynamic map might allow for a smaller resolution without slowing down the algorithm too much. This would especially be useful for the texture map as often more than one pixel fits into one occupancy grid square often resulting in them overwriting each other. This can be seen in the texture map as often the colors in certain areas appear accurate but ultimately are blended with neighboring pixels. In addition, the disparity function from opencv seems to struggle to locate objects resulting in a very sparse texture map.

ACKNOWLEDGMENT

Thank you Professor Atanasov as well as Mo and Arash. I appreciate all the time and effort you spent on putting this

project together as well as all the time you put into answering questions on Piazza.

REFERENCES

[1] D. Agarwal, "How SLAM Works For Self-Driving Cars: A Brief But Detailed Overview," AutoVision News, 05-Jun-2020. [Online]. Available: https://www.autovision-news.com/sensing/how-slam-works/. [Accessed: 21-Feb-2021].

[2] N. Atanasov, "Lecture 7: Motion and Observation Models," in ECE276A: Sensing amp; Estimation in Robotics, 01-Feb-2021.

[3] N. Atanasov, "Lecture 8: Bayesian Filtering," in ECE276A: Sensing amp; Estimation in Robotics, 08-Feb-2021.

[4] N. Atanasov, "Lecture 9: Particle Filter SLAM," in ECE276A: Sensing amp; Estimation in Robotics, 10-Feb-2021.

[5] N. Atanasov, "Lecture 6: Rotations," in ECE276A: Sensing amp; Estimation in Robotics, 27-Jan-2021.