Particle Filter in Simultaneous Localization and Mapping (SLAM) and Texture Mapping

1st Kai Chuen Tan
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
kctan@ucsd.edu, A59011493

I. Introduction

Navigating around an unexplored and unfamiliar environment is not only a challenging task for autonomous vehicles and mobile robots but also a crucial task that enables humans to explore uncharted hazardous territories without the need of risking any human's life. One of the most common hardware that autonomous vehicles and mobile robots used to navigate in outdoor environments is the global positioning system (GPS). The development of GPS navigation technology for autonomous vehicles including unmanned aerial vehicles (UAV), unmanned ground vehicle (UGV), and other automobile robotics has been matured over the years [1]; with the real-time kinematic (RTK) GPS and the global navigation satellite system (GNSS) on an automobile robot, the centimeter-level positioning accuracy can be achieved [2]. In this day and age, most of the UAVs' navigation system uses GPS to navigate safely in the sky; for example, a private drone delivery company in the United States, Zipline, integrated a custom-built GPS navigation system into their UAV system to perform safe flight operation and accurate medical supplies deliveries like blood platelets [3].

Nevertheless, in GPS-denied environments like indoor environments and urban canyons, the navigation tasks difficulty for autonomous vehicles and automobile robots increases. Autonomous vehicles and automobile robots that heavily rely on the GPS as their navigation system will not be able to navigate around safely in GPS-denied environments. If autonomous UAVs and UGVs are able to navigate in a hazardous indoor environment, UGVs and UAVs will be able to perform search-and-rescue missions and safe autonomous flights to save lives. Therefore, in this project, the particle filter in simultaneous localization and mapping (SLAM) with the differential-drive motion and scan-grid correlation observation model is implemented to localize a vehicle and to map its surrounding environment simultaneously using the vehicle sensors including encoders, fiber optic gyro (FOG), 2-D light detection and ranging (LIDAR), and an RGB stereo camera to collect data that helps the vehicle to be aware of its surrounding.

The particle filter in the SLAM algorithm includes the mapping function that plots the trajectories of the vehicle and LIDAR scanned data onto the occupancy-grid map, particle filter SLAM prediction function that predicts the vehicle positions in the future based on the present sensor data measurements from 2-D LIDAR and FOG, particle filter SLAM update step function that uses scan-grid correlation to correct the robot pose, and texture mapping function that transform RGB pixels from the RGB stereo camera onto the occupancy-grid map to color the ground floor. The paper is organized as follows. §II presents the problem formulation. §III describes the technical approach to the particle SLAM and texture mapping. Lastly, §IV presents the occupancy-grid mapping and texture mapping results, and the discussion of the results.

II. Problem Formulation

A. Simultaneous Localization and Mapping (SLAM)

The main objective of the particle filter in simultaneous localization and mapping (SLAM) is to localize a vehicle and build a 2-D occupancy grid map of the environment by using the 2-D LIDAR, FOG, and stereo camera. Therefore, in the SLAM problem, several Markov assumptions can be made given that the sequences of control inputs $\mathbf{u}_{0:T}$ and observations $\mathbf{z}_{0:T}$ are known and observed, respectively, and the sequences of the vehicle state $\mathbf{x}_{0:T}$ and map states $\mathbf{m}_{0:T}$ are unknown and hidden, respectively, where $t \in \mathbb{N}_0$, is the discrete-time steps in nanoseconds, $\forall t = 0, 1, 2, 3, \cdots, T$, where T is the end of the time in nanoseconds. The Markov assumptions are listed in the following [4]:

- The vehicle state, $\mathbf{x}_{t+1}$ only depends on the previous input $\mathbf{u}_t$ and $\mathbf{x}_t$.
- The map state, $\mathbf{m}_{t+1}$ only depends on the previous map state, $\mathbf{m}_t$.
- The map state, $\mathbf{m}_t$, and the vehicle state, $\mathbf{x}_t$ may affect each other's motion.
- The observation $\mathbf{z}_t$ only depends on the vehicle state, $\mathbf{x}_t$, and the map state, $\mathbf{m}_t$.

The motion model of the vehicle can be formulated as follows:

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

where f is a non-linear function of the motion model, p_f is the probability density function of the motion model that describes the vehicle motion to a new state, $\mathbf{x}_{t+1}$, after the control input, $\mathbf{u}_t$ is applied at state $\mathbf{x}_t$, and $\mathbf{w}_t$ is the motion noise.

Besides the motion model of the vehicle, there is an observation model since the 2-D LIDAR sensor is mounted

to the vehicle to map the environment, and the observation model is formulated as follows:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}_t, \mathbf{v}_t) \sim p_h(\cdot|\mathbf{x}_t, \mathbf{m}_t) \tag{2}$$

where h is the function of the observation model, p_h is the probability density function of the observation model that describes the vehicle observation, $\mathbf{z}_t$ depending on $\mathbf{x}_t$, and $\mathbf{m}_t$, and $\mathbf{v}_t$ is the observation noise.

SLAM is a parameter estimation problem to determine the environment of the occupancy-grid map, $\mathbf{m}$ and the vehicle poses, $\mathbf{x}_t$ given a dataset of the vehicle inputs $\mathbf{u}_{0:T-1}$ and observations $\mathbf{z}_{0:T}$. Hence, the SLAM problem can be formulated in a probabilistic form as shown below:

$$p(\mathbf{m}_t, \mathbf{x}_t|\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) \tag{3}$$

The objectives of the SLAM problem also exploit the decomposition of the joint probability density function without considering the control policy term due to the Markov assumptions listed above as shown in the following:

$$p(\mathbf{x}_{0:T}, \mathbf{m}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = p_{0|0}(\mathbf{x}_0, \mathbf{m}) \prod_{t=0}^{T} p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}_t) \prod_{t=1}^{T} p_f(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{4}$$

In this project, the maximum likelihood estimation (MLE) approach is applied to determine the optimal environment of the occupancy-grid map, $\mathbf{m}$ and the vehicle poses, $\mathbf{x}_{0:T}$ as shown in the problem formulation below:

$$\textbf{MLE}: \max_{\mathbf{x}_{0:T}, \mathbf{m}} \log p(\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}|\mathbf{x}_{0:T}, \mathbf{m}) \tag{5}$$

$$\textbf{MLE}: \max_{\mathbf{x}_{0:T}, \mathbf{m}} \sum_{t=0}^{T} \log p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) + \sum_{t=1}^{T} \log p_f(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \tag{6}$$

1) Occupancy-Grid Mapping

Given the vehicle state trajectory, $\mathbf{x}_{0:T}$, the vehicle motion model in Equation 1, and the observation model in Equation 2, an occupancy-grid map $\mathbf{m}$ of the environment can be built. Occupancy-grid mapping represents a vehicle workspace as a 2-D discrete grid that addresses the problem of generating a consistent map of the environment, $\mathbf{m}$ from the noisy and uncertain sensor measurement data or observation, $\mathbf{z}$, with the assumption that the vehicle pose, $\mathbf{x}$ is known [5]. The environment of the occupancy-grid map is divided into a regular grid with n cells, and the cells represent pixels in 2-D; the occupancy-grid represents a vector $m \in \mathbb{R}^n$, and the occupancy-grid map cells $\mathbf{m}_i$, where $\forall i = 0, 1, 2, \cdots, n$, can be formulated as a binary variable as shown below:

$$\mathbf{m}_i = \begin{cases} 1 & \text{if Occupied} \\ -1 & \text{if Free} \end{cases} \tag{7}$$

The occupancy-grid mapping objective is to determine the posterior probability over time as follows:

$$p(\mathbf{m}|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \tag{8}$$

With the independence assumption, the occupancy-grid mapping algorithm assumes that the occupancy-grid map cell values are independent conditioned on the vehicle trajectory, and it can be formulated as follows:

$$p(\mathbf{m}|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) = \prod_{i=1}^{n} p(\mathbf{m}_i|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \tag{9}$$

The probability distribution of each occupancy-grid map cell, $\gamma_{i,t}$, is defined as follows:

$$\gamma_{i,t} = p(\mathbf{m}_i|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \tag{10}$$

Hence, the occupancy-grid map cells, $\mathbf{m}_i$ can be modeled as independent Bernoulli random variables as shown below:

$$\mathbf{m}_i = \begin{cases} 1 & \text{if Occupied with a probability of } \gamma_{i,t} \\ -1 & \text{if Free with a probability of } 1 - \gamma_{i,t} \end{cases} \tag{11}$$

Additionally, for the texture mapping, the problem is to for a vector $\mathbf{m}_{color} \in \mathbb{R}^{n \times 3}$ with the given occupancy-grid map, $\mathbf{m}$, where each row of the cell is RGB color pixel values of the ground.

2) Bayes Filtering for Localization

Bayes filtering is a probabilistic inference technique for estimating the state, $\mathbf{x}_t$ of the dynamical system of the vehicle that combines evidence from control inputs and observations using the Markov assumptions as listed above and Bayes rule [4]. The Bayes filtering algorithm requires two main steps, which are the prediction and update steps, to keep track of the updated probability density function (pdf), and the predicted pdf over a discrete state set X as defined below:

$$\textbf{Updated pdf}: p_{t|t}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) \tag{12}$$

$$\textbf{Predicted pdf}: p_{t+1|t}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:T}, \mathbf{u}_{0:T}) \tag{13}$$

To compute the predicted pdf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$, the vehicle motion model pdf p_f is applied with the given prior pdf $p_{t|t}$ over $\mathbf{x}_t$ and the control input $\mathbf{u}_t$ as formulated below:

$$p_{t+1|t}(\mathbf{x}) = \int p_f(\mathbf{x}|\mathbf{s}, \mathbf{u}_t) p_{t|t}(\mathbf{s}) d\mathbf{s} \tag{14}$$

To obtain the updated pdf $p_{t+1|t+1}$ over $\mathbf{x}_{t+1}$ by using the vehicle observation model pdf, p_h with the given predicted pdf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$ and the measurement $\mathbf{z}_{t+1}$ as formulated below:

$$p_{t+1|t+1}(\mathbf{x}) = \frac{p_h(\mathbf{z}_{t+1}|\mathbf{x}) p_{t+1|t}(\mathbf{x})}{\int p_h(\mathbf{z}_{t+1}|\mathbf{s}) p_{t+1|t}(\mathbf{s}) d\mathbf{s}} \tag{15}$$

where $\mathbf{s} \in X$.

3) Vehicle Configuration and Parameters

The vehicle is mounted and equipped with multiple sensors including the front 2-D LIDAR scanner, FOG, and encoders for SLAM as well as the stereo cameras for texture mapping as shown in Figure 1.

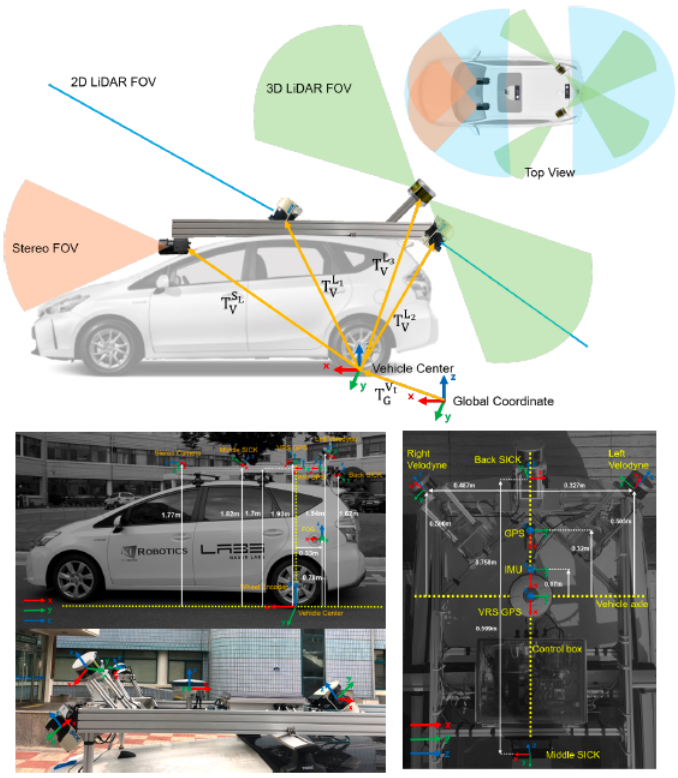

Fig. 1: Sensor Layout on the Autonomous Vehicle.

The encoders measures the instantaneous linear velocities of the vehicle, v_t, and FOG measures the instantaneous angular velocities of the vehicle, ω_t. The encoders data, $E := \{t_i, z_{left,i}, z_{right,i}\}_{i=1}^{116048}$, where $z_{left}, z_{right} \in \mathbb{R}$, consist of three entries, which are timestamps, t, left encoder counts, z_{left}, and right encoder counts z_{right}. With the encoders data, change in positions, $\Delta x \in \mathbb{R}$ and $\Delta y \in \mathbb{R}$ between $\mathbf{x}_{t+1}$ and $\mathbf{x}_t$ can be calculated as follows given that the left wheel diameter, $D_{left} = 0.623479$ m, and, the encoder resolution $res_{encoder} = 4096$ Hz:

$$\Delta x = \frac{\pi D_{left}(z_{left,t+1} - z_{left,t})}{res_{encoder}} \cos(\Delta\theta) \tag{16}$$

$$\Delta y = \frac{\pi D_{left}(z_{left,t+1} - z_{left,t})}{res_{encoder}} \sin(\Delta\theta) \tag{17}$$

The FOG data, $F := \{t_i, R_i, P_i, \Delta\theta_i\}_{i=1}^{1160508}$, where $R, P, \Delta\theta \in \mathbb{R}$ consists of timestamps, t, rolls of the vehicle, R, pitches of the vehicle, and yaws of the vehicle, $\Delta\theta$. Hence, the control input at time t can be defined as $\mathbf{u}_t = [\Delta x_t, \Delta y_t, \Delta\theta] \in \mathbb{R}^3$.

The 2-D LIDAR data, $L : \{t_i, G_i\}_{i=1}^{115865}$, consist of timestamps, t, and scanned points $G \in \mathbb{R}^{286}$. The field of view (FOV) is 190° ranging from -5° to 185° with an angular resolution of 0.666°. The maximum detection range of the 2-D LIDAR is 80 meters, and the minimum detection range of the 2-D LIDAR is 2 meters

The RGB stereo cameras captures two RGB images (i.e., left and right images) at each time step, and each RGB image contains RGB values at each pixel, denoted as $M \in \mathbb{N}_0^{560\times1280\times3}$.

All the sensors that are equipped by the vehicle worked independently. Therefore, it is necessary to sync the sensors' timestamps before performing particle filter SLAM and texture mapping.

III. Technical Approach

A. Vehicle Motion Model

The pose (i.e., positions and orientation) of the vehicle at discrete time t is defined as $\mathbf{x}_t = [x_t, y_t, \theta_t]^T$, where $\mathbf{x}_t \in \mathbb{R}^3$, x_t is the x-coordinate of the vehicle in a 2-D space, y_t is the y-coordinate of the vehicle in a 2-D space, and θ_t is the orientation of the vehicle. Both $x-y$ coordinates' unit is in meters, and the vehicle orientation, θ, is in radian. The initial pose of the vehicle, $\mathbf{x}_0$ is initialized as $[0,0,0]^T$. Since the Δx and Δy can be calculated from the encoders data as shown in Equation 16 and Equation 17, and $\Delta\theta$ can be directly obtained from the FOG yaw data, the differential-drive motion model of the vehicle can be defined as below:

$$\begin{aligned} \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \\ &= f(\mathbf{x}_t, \mathbf{u})_t + \mathbf{w}_t \\ \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} &= \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{bmatrix} + \mathbf{w}_t \end{aligned} \tag{18}$$

where $\mathbf{w}_t$ is a 3-D Gaussian motion noise, and $\mathbf{W} \in \mathbb{R}^{3\times3}$ is the variance matrix of the Gaussian motion noise as defined below:

$$\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{W} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}) \tag{19}$$

B. Vehicle Observation Model

In this project, the laser correlation model is applied for the observation model of the vehicle to construct the occupancy-grid map. Before calculating the probability density function of the observation model, $p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$, first, the 2-D LIDAR scanned points, $\mathbf{z}_t$ that are calculated with the 2-D LIDAR ranges data, G, from the measured 2-D LIDAR data, L, as defined in §II, must be converted to homogeneous points, $\hat{\mathbf{z}}_t \in \mathbb{R}^4$ and transformed from the vehicle frame to the world frame to determine the 2-D LIDAR homogeneous observation points in the world frame, $\hat{\mathbf{z}}_{t,w} \in \mathbb{R}^4$ as shown in the formulation below:

$$\hat{\mathbf{z}}_{t,w,i} = {}_w\mathbf{T}_v \cdot {}_v\mathbf{T}_l \cdot \hat{\mathbf{z}}_{t,i}; \forall i = 1, \cdots, 286 \tag{20}$$

where ${}_w\mathbf{T}_v$ is the transformation matrix from the vehicle frame to the world frame and ${}_v\mathbf{T}_l$ is the transformation matrix

from the 2-D LIDAR frame to the vehicle frame. Both ${}_w\mathbf{T}_v$, and ${}_v\mathbf{T}_l$ transformation matrices are defined below:

$$ {}_v\mathbf{T}_l = \begin{bmatrix} {}_v\mathbf{R}_l & {}_v\mathbf{p}_l \\ \mathbf{0} & 1 \end{bmatrix} \tag{21} $$

$$ {}_w\mathbf{T}_v = \begin{bmatrix} {}_w\mathbf{R}_v & {}_w\mathbf{p}_v \\ \mathbf{0} & 1 \end{bmatrix} \tag{22} $$

where ${}_v\mathbf{R}_l \in \mathbb{R}^3$ is the rotation matrix from the 2-D LIDAR frame to the vehicle frame, ${}_v\mathbf{p}_l \in \mathbb{R}^3$ is the position of the 2-D LIDAR in the vehicle frame, ${}_w\mathbf{R}_v \in \mathbb{R}^3$ is the rotation matrix from the vehicle frame to the world frame, and ${}_w\mathbf{p}_v \in \mathbb{R}^3$ is the position of the vehicle in the world frame. The 2-D LIDAR homogeneous observation point, $\hat{\mathbf{z}}_{t,i}$ can be determined with the 2-D LIDAR scanned data, G_i as shown below:

$$ \psi_i = \psi_{start} + \psi_{res} \cdot i; \forall i = 1, \cdots, 286 \tag{23} $$ [1]

$$ \hat{\mathbf{z}}_{t,i} = \begin{bmatrix} G_i \cos(\psi_i) \\ G_i \sin(\psi_i) \\ h_l \\ 1 \end{bmatrix}; \forall i = 1, \cdots, 286 \tag{24} $$

where ψ_i is the 2-D LIDAR angle, ψ_{start} is the 2-D LIDAR field of view starting angle, ψ_{res} is the 2-D LIDAR angle resolution, and h_l is the height of the 2-D LIDAR in the vehicle frame.

After the 2-D LIDAR observation points in the world frame are computed, the probability density function of the vehicle observation model, $p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$ can be determined by comparing the correlation between the environment of the occupancy-grid map, $\mathbf{m}$ and the 2-D LIDAR observation scanned euclidean points in the world frame, $\mathbf{z}_{t,w}$; then, the correlation between $\mathbf{m}$ and $\mathbf{z}_{t,w}$ is converted to probabilities using the softmax function as formulated below:

$$ \text{corr}(\mathbf{z}_w, \mathbf{m}) = \sum_i \mathbb{1}\{m_i = z_{w,i}\} \tag{25} $$

$$ p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) = \frac{\exp\left(\text{corr}(\mathbf{z}_{t,w}, \mathbf{m})\right)}{\sum_{\mathbf{z}_{t,w}} \exp\left(\text{corr}(\mathbf{z}_{t,w}, \mathbf{m})\right)} \tag{26} $$

where $\text{corr}(\mathbf{z}_w, \mathbf{m})$ is the correlation function of the laser correlation model.

C. Simultaneous Localization and Mapping (SLAM)

In this paper, simultaneous localization and mapping (SLAM) is implemented with the particle filtering approach to localize the vehicle and construct a 2-D occupancy-grid map of the environment. The SLAM algorithm consists of four major steps, which are the mapping step, prediction step, the update step, and the particle re-sampling step. The objective of the mapping step, prediction step, update step, and re-sampling step is to update the occupancy-grid map log-odds based on the 2-D LIDAR observations, predict the trajectory of the vehicle, correct the vehicle pose using the scan-grid correlation, and re-sample the particles if the number of effective particles is less than a certain threshold, respectively; the pseudocode for the SLAM algorithm is presented in Algorithm 1 below:

Algorithm 1 Particle Filter SLAM Algorithm

$\mathbf{x}_t \leftarrow Initialize\_Pose$
$\mathbf{m} \leftarrow Initialize\_Occupancy\_Grid\_Map$
RGB_map $\leftarrow Initialize\_RGB\_Occupancy\_Grid\_Map$
$\boldsymbol{\mu}, \boldsymbol{\alpha} \leftarrow Initialize\_Particles\_and\_Weight$
for *t in FOG_timestamps* **do**
  $t_{LIDAR}, t_{FOG}, t_{encoders} \leftarrow Sync\_Timestamps$
  $\mathbf{u} \leftarrow Calculate\_Inputs(E, F)$
  // Prediction Step
  $\mathbf{x}_{t+1} \leftarrow Predict\_Trajectory(\mathbf{x}_t, \mathbf{u})$
  // Update and Mapping Step
  if *t % 100 == 0* **then**
    // Update Step
    G, $\psi \leftarrow Get\_Valid\_LIDAR\_Data(L)$
    $\mathbf{z}_t \leftarrow Convert2Cartesian\_Coordinates(G, \psi)$
    $\mathbf{z}_{t,v} \leftarrow Transform2Vehicle\_Frame(\mathbf{z}_t)$
    $\mathbf{z}_{t,w} \leftarrow Transform2World\_Frame(\mathbf{z}_{t,v})$
    $\text{corr}(\mathbf{z}_w, \mathbf{m}) \leftarrow Get\_Map\_Correlation(\mathbf{z}_w, \mathbf{m})$
    $\boldsymbol{\mu} \leftarrow Update\_Particles2Local\_Max(\boldsymbol{\mu})$
    $\boldsymbol{\alpha} \leftarrow Update\_Particle\_Weights$
    // Mapping Step
    $\boldsymbol{\mu}_{best} \leftarrow Get\_Largest\_Particles(\boldsymbol{\mu}, \boldsymbol{\alpha})$
    $\mathbf{z}_t \leftarrow Convert2Cartesian\_Coordinates(G, \psi)$
    $\mathbf{z}_{t,v} \leftarrow Transform2Vehicle\_Frame(\mathbf{z}_t)$
    $\mathbf{z}_{t,w} \leftarrow Transform2World\_Frame(\mathbf{z}_{t,v})$
    $\mathbf{m} \leftarrow Update\_Map\_Log\_Odds(\mathbf{m}, \mathbf{z}_{t,w})$
    // Texture Mapping
    $d \leftarrow Compute\_Disparity(Stereo\_Images)$
    $depth \leftarrow Compute\_Depth(d)$
    $x_o, y_o \leftarrow Transform2Optical\_Frame(u, v)$
    $x_w, y_w, z_w \leftarrow Transform2World\_Frame(x_o, y_o, depth)$
    $x_w, y_w, z_w \leftarrow Thresholding\_Height(x_w, y_w, z_w)$
    $Rd, Gr, Bl \leftarrow Get\_RGB(x_w, y_w, z_w)$
    RGB_map $\leftarrow Update\_RGBmap(Rd, Gr, Bl,$ **RGB_map**$)$
    // Particle Resampling if $N_{eff} \leq N_{threshold}$
    $\boldsymbol{\mu}, \boldsymbol{\alpha} \leftarrow Resample(\boldsymbol{\mu}, \boldsymbol{\alpha}, N_{eff}, N_{threshold})$
  –

1) Occupancy-grid Mapping

The objective of the occupancy-grid mapping is to maintain and update the $\gamma_{i,t}$ over time. Since the map cells, $\mathbf{m}_i$, can be modeled as Bernoulli random variables as mentioned §II, the probabilistic occupancy-grip map can be formulated with Bayes Rule as shown below:

$$ \begin{aligned} \gamma_{i,t} &= p(\mathbf{m}_i = 1|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \\ &= \frac{1}{\eta_t} p_h(\mathbf{z}_t|\mathbf{m}_i = 1, \mathbf{x}_t)\gamma_{i,t-1} \end{aligned} \tag{27} $$

The occupancy-grid map can be updated over time by accumulating the log odds ratio, $\Delta\lambda_{i,t}$ over time as formulated below, where $\lambda_{i,t}$ is the log-odds of the Bernoulli random variable $\mathbf{m}_i$:

$$
\begin{aligned}
\lambda_{i,t} &= \lambda(\mathbf{m}_i, \mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \\
&= \log o(\mathbf{m}_i, \mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \\
&= \log \frac{p(\mathbf{m}_i = 1|\mathbf{z}_t, \mathbf{x}_t)}{p(\mathbf{m}_i = -1|\mathbf{z}_t, \mathbf{x}_t)} - \lambda_{i,0} + \lambda_{i,t-1} \\
&= \Delta\lambda i, t - \lambda_{i,0} + \lambda_{i,t-1} \\
\lambda_{i,t} &= \lambda_{i,t-1} + (\Delta\lambda i, t - \lambda_{i,0})
\end{aligned} \tag{28}
$$

In this project, the 2-D LIDAR observations, $\mathbf{z}_t$ indicates whether $\mathbf{m}_i$ is an obstacle or a freespace, and the log odds ratio, $\Delta\lambda_{i,t}$ of the inverse measurement model specifies the measurement "trust" of the 2-D LIDAR observation. The log odds ratio, $\Delta\lambda_{i,t}$ can be formulated as shown below:

$$
\begin{aligned}
\Delta\lambda i, t &= log \frac{p(\mathbf{m}_i = 1|\mathbf{z}_t, \mathbf{x}_t)}{p(\mathbf{m}_i = -1|\mathbf{z}_t, \mathbf{x}_t)} \\
\Delta\lambda i, t &= \begin{cases} +\log 4 & \text{if } \mathbf{z}_t \text{ indicates } \mathbf{m}_i \text{ is occupied} \\ -\log 4 & \text{if } \mathbf{z}_t \text{ indicates } \mathbf{m}_i \text{ is free} \end{cases}
\end{aligned} \tag{29}
$$

Next, the new 2-D LIDAR scan $\mathbf{z}_{t+1}$ is transformed to the world frame using the vehicle pose, $\mathbf{x}_{t+1}$. Then, the cells, $\mathbf{m}_i$ that the 2-D LIDAR Beams pass through are determined using Bresenham's line rasterization algorithm. For each observed cell i, the log-odds decreases if it was observed free; on the other hand, the log-odds increases if it was observed occupied:

$$
\lambda_{i,t+1} = \lambda_{i,t} \pm \log 4 \tag{30}
$$

To prevent overconfident estimation, an inequality constraint, $\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MAX}$ is defined to clip the log-odds value, $\lambda_{i,t}$ and to filter out the $\lambda_{i,t}$ outliers. Last but not least, given the $\lambda_{i,t}$ are estimated, the occupancy-grid map probability mass function $\gamma_{i,t}$ can be calculated from the log-odds $\lambda_{i,t}$ via the logistic sigmoid function as shown in the following:

$$
\begin{aligned}
\gamma i, t &= p(\mathbf{m}_i = 1|\mathbf{z}_{0:T}, \mathbf{x}_{0:T}) \\
&= \sigma(\lambda_{i,t}) \\
\gamma i, t &= \frac{\exp \lambda_{i,t}}{1 + \exp \lambda_{i,t}}
\end{aligned} \tag{31}
$$

2) Particle Filter Localization

Particle filters are also known as a sequential Monte Carlo methods, and it used to solve filtering problems in this project to localize the vehicle. A set of hypotheses, which are known as particles with the positions of the particles $\{\boldsymbol{\mu}^{(k)}\}_k \forall k = 1, \cdots, N$ and weights $\{\alpha^{(k)}\}_k, \forall k = 1, \cdots, N$, is used for the particle filter to represent the probability density functions $p_{t|t}$ and $p_{t+1|t}$. The particles specify the probability density function of the vehicle state, $p(\mathbf{x}_t|\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}, \mathbf{m})$, at discrete time t:

$$
\begin{aligned}
p_{t|t} &= p(\mathbf{x}_t|\mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}, \mathbf{m}) \\
p_{t|t} &\approx \sum_{k=1}^{N} \alpha_{t|t}^{k} \delta(\mathbf{x}_t; \boldsymbol{\mu}_{t|t}^{(k)})
\end{aligned} \tag{32}
$$

where the δ is the Dirac delta function:

$$
\delta(\mathbf{x}; \boldsymbol{\mu}) = \begin{cases} 1 & \mathbf{x} = \boldsymbol{\mu}^{(k)} \\ 0 & \text{otherwise} \end{cases} \forall k = 1, \cdots, N \tag{33}
$$

The particle filter can be derived by substituting the probability density functions in the Bayes filter prediction and update steps. The prediction and update steps should maintain the mixture-of-delta-functions of the probability density functions.

2.1) Prediction Step

As mentioned and defined in §II and Equation 14, the Bayes filtering computes the predicted pdf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$, the vehicle motion model pdf p_f is applied with the given prior pdf $p_{t|t}$ over $\mathbf{x}_t$ and the control input $\mathbf{u}_t$. To derive the particle filter's prediction step, Equation 32 is substituted into the Bayes filter prediction equation, which is Equation 14, as shown below, and the prediction step should maintain the mixture-of-delta-functions of the probability density function as mentioned previously:

$$
\begin{aligned}
p_{t+1|t} &= \int p_f(\mathbf{x}|\mathbf{s}, \mathbf{u}_t) \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} \delta(\mathbf{s} - \boldsymbol{\mu}_{t|t}^{(k)}) d\mathbf{s} \\
&= \sum_{k=1}^{N_{t|t}} \alpha_{t|t}^{(k)} p_f(\mathbf{x}|\boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{u}_t)
\end{aligned} \tag{34}
$$

To approximate the particle filter's prediction step as a delta-mixture probability density function, samples can be drawn from it since $p_{t+1|t}(\mathbf{x})$ us a mixture pdf with components $p_f(\mathbf{x}|\boldsymbol{\mu}_{t|t}^{(k)}, \boldsymbol{\mu}_t)$. The vehicle motion model from Equation 18 is applied to each $\boldsymbol{\mu}_{t|t}^{(k)}$ by drawing as shown in the following:

$$
\boldsymbol{\mu}_{t+1|t}^{(k)} \sim p_f(\cdot|\boldsymbol{\mu}_{t|t}^{(k)}, \mathbf{u}_t); \quad \alpha_{t+1|t}^{(k)} = \alpha_{t|t}^{(k)} \tag{35}
$$

$$
\boldsymbol{\mu}_{t+1|t}^{(k)} = \boldsymbol{\mu}_{t|t}^{(k)} + \mathbf{u}_t + \mathbf{w}_t \tag{36}
$$

Hence, the particle filter's prediction step, $p_{t+1|t}(\mathbf{x})$ can be approximated as follows:

$$
p_{t+1|t} \approx \sum_{k=1}^{N_{t+1|t}} \alpha_{t+1|t}^{(k)} \delta(\mathbf{x} - \boldsymbol{\mu}_{t+1|t}^{(k)}) \tag{37}
$$

Lastly, the vehicle state, $\mathbf{x}_{t+1|t}^{greatest}$ is predicted using the particle that has the greatest weight, $\alpha_{t+1|t}$ as shown below:

$$
\begin{aligned}
\mathbf{x}_{t+1|t}^{greatest} &= \boldsymbol{\mu}_{t+1|t}^{(k_{max})}); \\
k_{max} &= \arg\max(\alpha_{t+1|t}^{(k)}); \\
\forall k &= 1, \cdots, N
\end{aligned} \tag{38}
$$

2.1) Update Step

In the Bayes filter update steps, the updated pdf $p_{t+1|t+1}$ over $\mathbf{x}_{t+1}$ can be calculated by using the vehicle observation model pdf, p_h with the given predicted pdf $p_{t+1|t}$ over $\mathbf{x}_{t+1}$

and the measurement $\mathbf{z}_{t+1}$ as shown in Equation 15. To derive the particle filter update step with the particle filter predicted delta-mixture pdf in Equation 37, Equation 37 is substituted into the Bayes filter posterior in Equation 15 as shown in the formulation below:

$$\begin{aligned}\mathbf{p}_{t+1|t+1}(\mathbf{x}) &= \frac{p_h(\mathbf{z}_{t+1}|\mathbf{x})\sum_{k=1}^{N_{t+1|t}}\alpha_{t+1|t}^{(k)}\delta(\mathbf{x}-\boldsymbol{\mu}_{t+1|t}^{(k)})}{\int p_h(\mathbf{z}_{t+1}|\mathbf{s})\sum_{j=1}^{N_{t+1|t}}\alpha_{t+1|t}^{(j)}\delta(\mathbf{s}-\boldsymbol{\mu}_{t+1|t}^{(j)})d\mathbf{s}} \\ &= \sum_{k=1}^{N_{t+1|t}}\left[\frac{\alpha_{t+1|t}^{(k)}p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^{(k)})}{\sum_{j=1}^{N_{t+1|t}}\alpha_{t+1|t}^{(j)}p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^{(j)})}\right]\delta(\mathbf{x}-\boldsymbol{\mu}_{t+1|t}^{(k)})\end{aligned} \tag{39}$$

Equation 39 presents that the particle filter updated pdf is a delta-mixture pdf; hence, no approximation is necessary. The update step only changes the weights but not the particle poses. The particle weights are scaled by the observation model as presented below:

$$\boldsymbol{\mu}_{t+1|t+1}^{(k)} = \boldsymbol{\mu}_{t+1|t}^{(k)} \tag{40}$$

$$\alpha_{t+1|t+1}^{(k)} \propto p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{m})\alpha_{t+1|t}^{(k)} \tag{41}$$

After the vehicle observation model, $p_h(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m})$, and the 2-D LIDAR homogeneous observation points in the world frame, $\hat{\mathbf{z}}_{t,w}$ are calculated with Equation 26 and Equation 20, respectively, the laser correlation model from Equation 25 is applied to update the particle weights as shown below:

$$p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{m}) \propto \text{corr}(\mathbf{z}_{t+1,w}^{(k)}, \mathbf{m}) \tag{42}$$

As presented in Particle Filter SLAM Algorithm, the particles are updated by computing the local maximum map correlation with a grid of values (e.g., 9×9) around the current particle position to get a good correlation. Then, the particle poses with the position in the 9×9 grid are replaced with the greatest correlation values.

2.3) Stratified and Systematic Particle Resampling

The objective of the particle resampling is to avoid particle depletion, which is when the majority of the updated particle weights become close to zero due to the finite number of particles, N_{eff}, is insufficient, i.e., the observation likelihoods, $p_h(\mathbf{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^{(k)}, \mathbf{m})$ are small $\forall k = 1, \cdots, N$. The particle resampling creates a brand new particle set with equal weights from the given weighted particle set by adding many particles to the locations that had large weights and a few particles to the locations that had low particle weights. The particle resampling focuses on the representation power of the particles to likely regions with only a few particles. If the effective number of particles is less than a threshold, the resampling will be applied at time t as shown in the following equation:

$$N_{eff} = \frac{1}{\sum_{k=1}^{N}(\alpha_{t|t}^{(k)})^2} \tag{43}$$

In this project, stratified and systematic resampling is implemented for particle resampling. Stratified resampling guarantees that samples with large weights appear at least once and those with small weights - at most once. The stratified and systematic resampling algorithm is presented below:

Algorithm 2 Stratified and Systematic Resampling Algorithm

Input: Particle Set $\{\ \boldsymbol{\mu}^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
Output: Resampled Particle Set $\{\overline{\boldsymbol{\mu}}^{(k)}, \overline{\alpha}^{(k)}\}_{k=1}^{N}$

```
j ← 1, c ← α^(1)
for k = 1, ··· , N do
    u ~ U(0, 1/N)
    β = u + (k−1)/N
    while β > c do
        j = j + 1
        c = c + α^(j)
    {μ̄^(k), ᾱ^(k)}_{k=1}^N ← Add2New_Set(μ^(j), 1/N)
```

D. Texture Mapping

The objective of texture mapping is to utilize the stereo camera RGB images to add RGB texture to the 2-D occupancy-grid map. Since the vehicle's stereo camera RGB images are not calibrated, the images need to be undistorted by using the OpenCV Python package function, $cv2.undistort()$ given that the distortion coefficients, intrinsic camera matrix, $K \in \mathbb{R}^{3\times3}$ from both left and right cameras are provided. Then, a disparity image is computed, d, from stereo image pairs with another OpenCV Python package function, $cv.StereoBM_create().compute()$. With the disparity image, the depth, z_o in the optical frame can be estimated by using the equation below:

$$d = \frac{1}{z_o}fs_ub \tag{44}$$

where f is the focal length of the camera in meters, s_u and s_v are defined as the scaling from meters to pixels in [pixels per meter], and b is the baseline of the stereo camera. The intrinsic camera can be defined as follows:

$$K = \begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{45}$$

where c_u and c_v are the pixel coordinates of the principal point used to translate the image frame origin, and s_θ is the skew factor that scales non-rectangular pixels and is proportional to $\cot(\alpha)$ where α is the angle between the coordinate axes of the pixel array. After the depth is estimated, each pixel coordinates in the left stereo image, u_L and u_R are transformed into the pixel coordinates in the optical frame, $[x_o, y_o, z_o]^T$ by solving the system of equations below:

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & fs_\theta & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_ub \end{bmatrix} \frac{1}{z_o} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \tag{46}$$

Then, the pixel coordinates in the optical frame are transformed to the pixel coordinates in the world frame, $x_w \in \mathbb{R}^3$, as shown in the formulation below:

$$\begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} = {}_oR_{vw} R_v^T (x_w - \mathbf{p}) \\ x_w = ({}_oR_{vw} R_v^T)^{-1} \begin{bmatrix} x_o \\ y_o \\ z_o \end{bmatrix} + \mathbf{p} \tag{47}$$

where $\mathbf{p} \in \mathbb{R}^3$ is the vehicle position in the world frame, ${}_oR_v$ is the rotation matrix from the vehicle frame to the optical frame. After the pixel coordinates in the world frame, x_w is determined, the image RGB values are transformed to the world frame. Then, the determined plane corresponds to the occupancy-grid map in the transformed data by thresholding on the height, which is the height of the camera from the vehicle frame. The occupancy-grid map cells are colored with RGB values according to the projected points that belong to its plane.

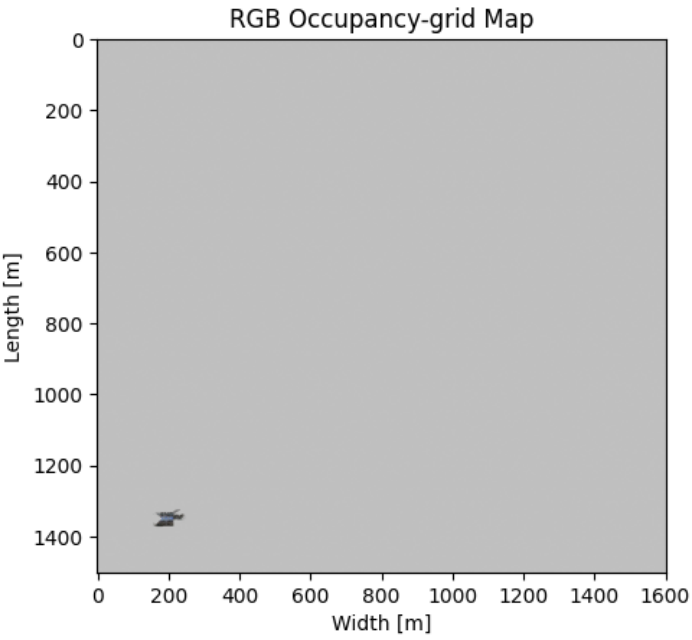

Fig. 3: Textured Map with Colors at $10,000^{th}$ FOG Timestamp

IV. Results and Discussion

The SLAM algorithm is tested with a dataset that was collected by a vehicle with integrated encoders, FOG, 2-D LIDAR scans, and a stereo camera. The hyper-parameters that are used in the SLAM algorithm are listed in Table I. The particle filter SLAM results are shown in the following figures:

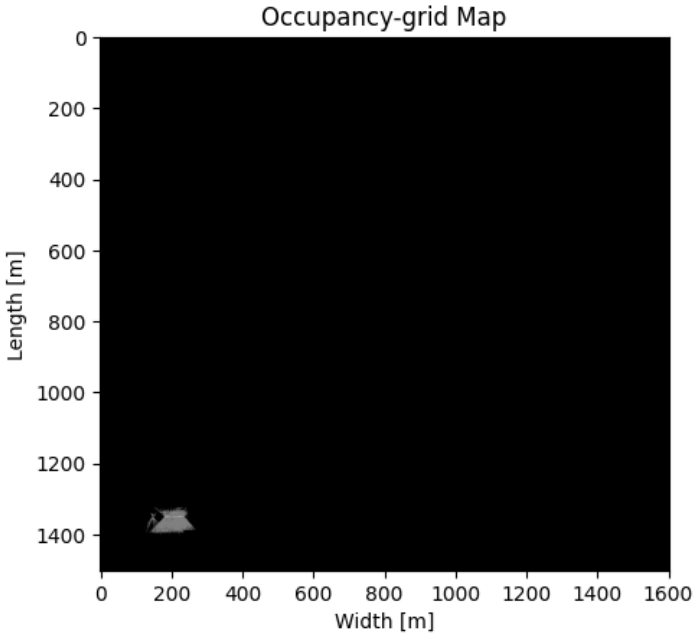

Fig. 2: Occupancy-Grid Map with Vehicle Trajectory at $10,000^{th}$ FOG Timestamp

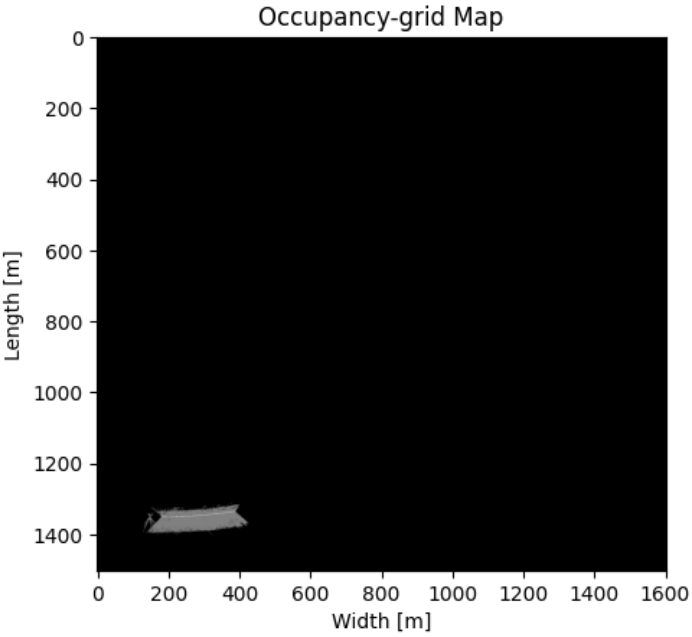

Fig. 4: Occupancy-Grid Map with Vehicle Trajectory at $100,000^{th}$ FOG Timestamp

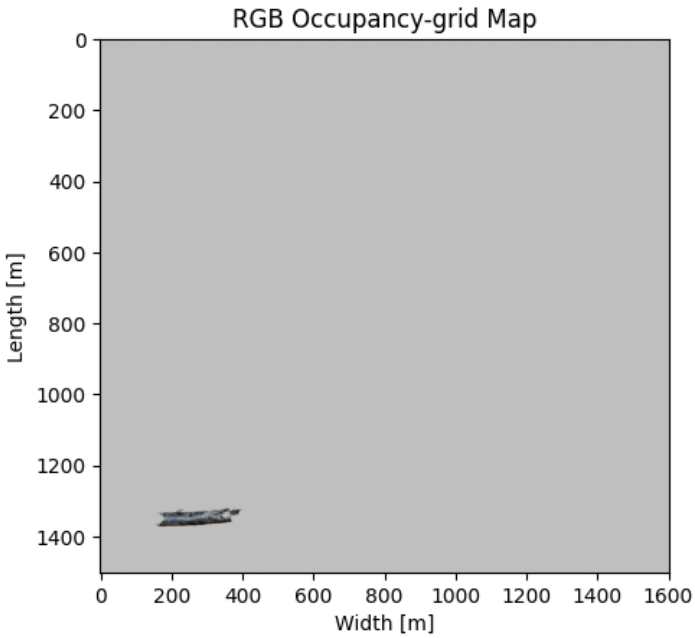

Fig. 5: Textured Map with Colors at $100,000^{th}$ FOG Timestamp

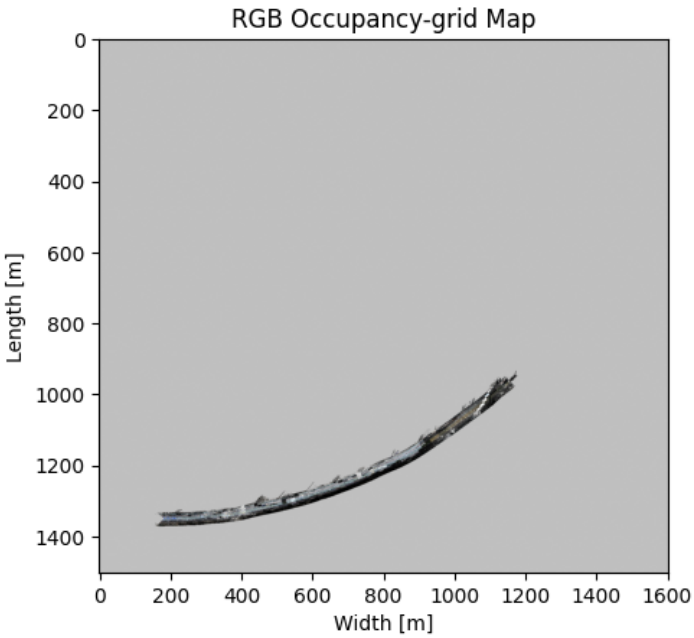

Fig. 7: Textured Map with Colors at $300,000^{th}$ FOG Timestamp

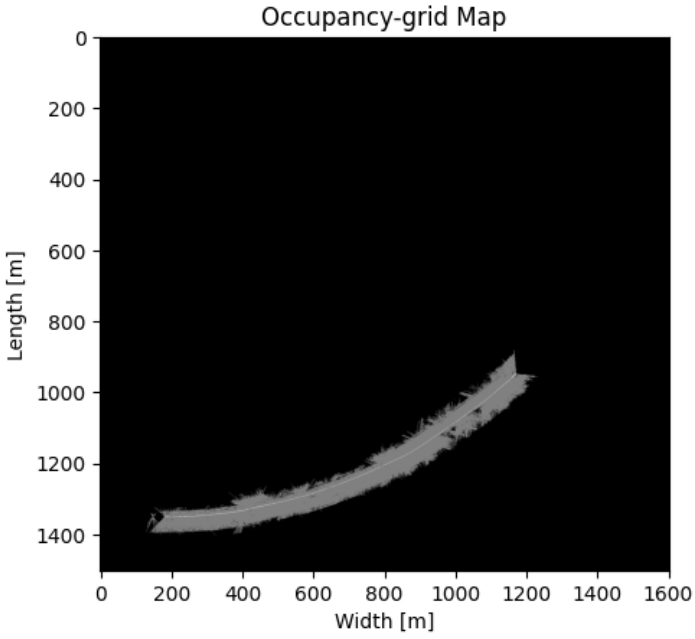

Fig. 6: Occupancy-Grid Map with Vehicle Trajectory at $300,000^{th}$ FOG Timestamp

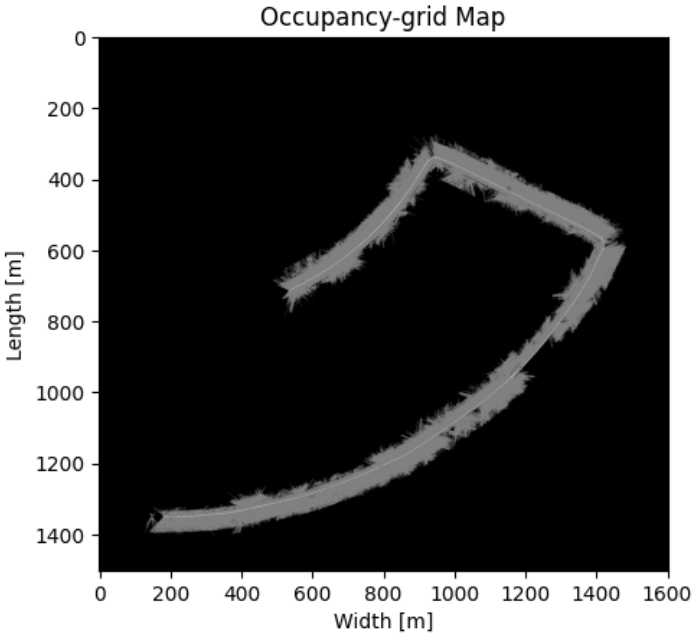

Fig. 8: Occupancy-Grid Map with Vehicle Trajectory at $500,000^{th}$ FOG Timestamp

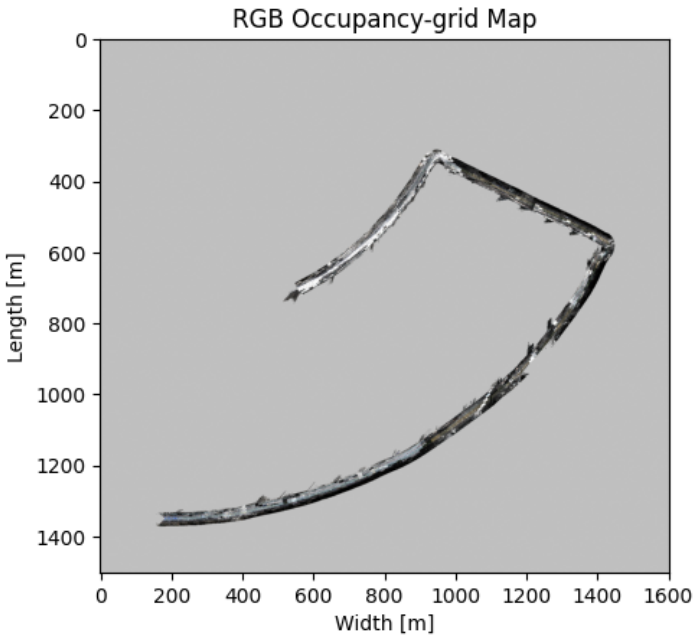

Fig. 9: Textured Map with Colors at $500,000^{th}$ FOG Timestamp

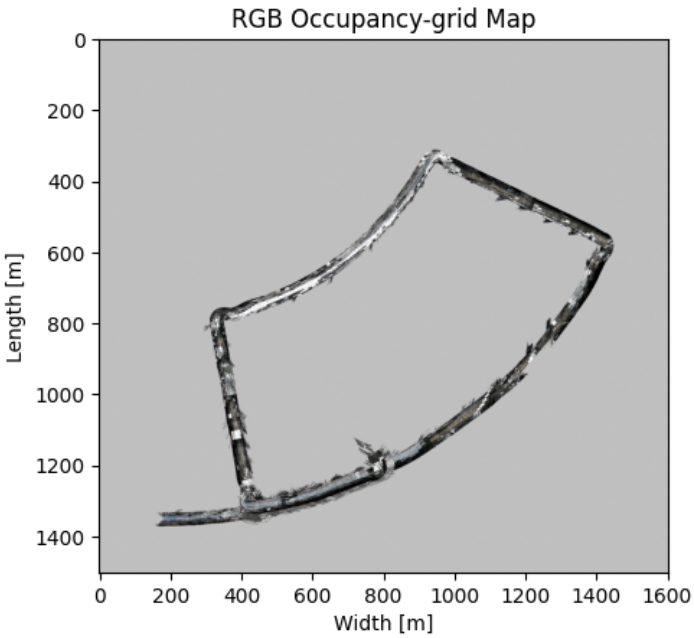

Fig. 11: Textured Map with Colors at $700,000^{th}$ FOG Timestamp

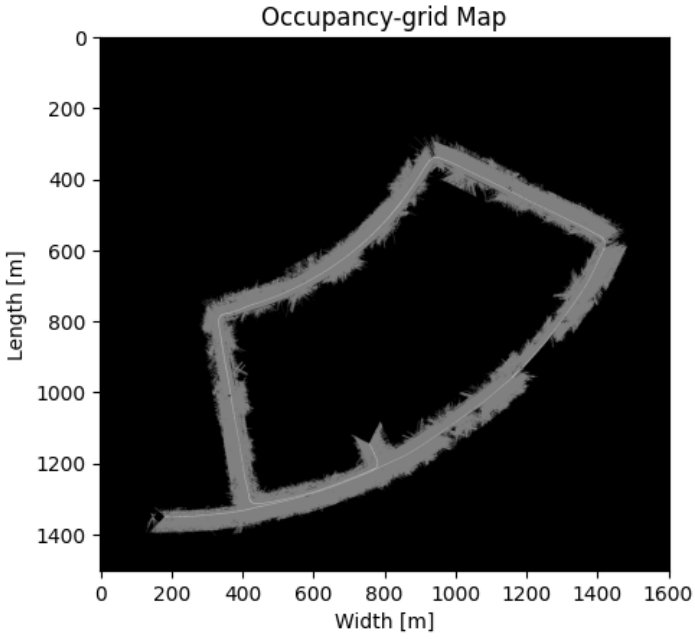

Fig. 10: Occupancy-Grid Map with Vehicle Trajectory at $700,000^{th}$ FOG Timestamp

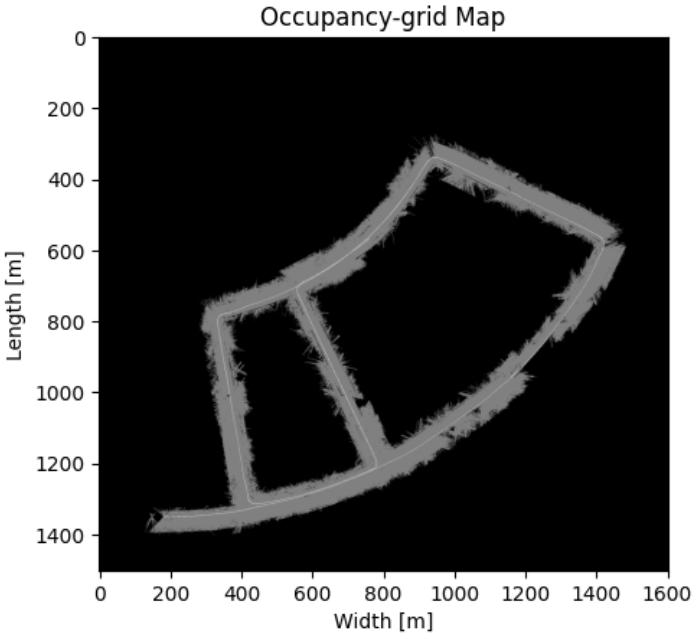

Fig. 12: Occupancy-Grid Map with Vehicle Trajectory at $1,000,000^{th}$ FOG Timestamp

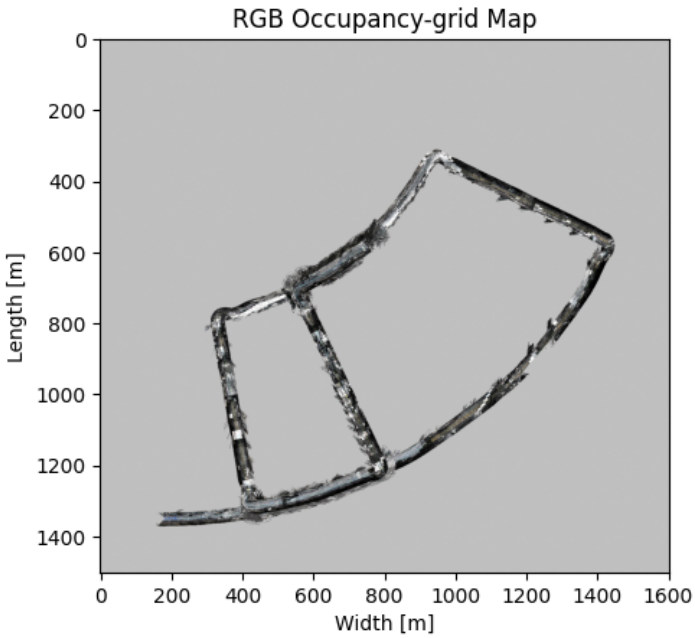

Fig. 13: Textured Map with Colors at $1,000,000^{th}$ FOG Timestamp

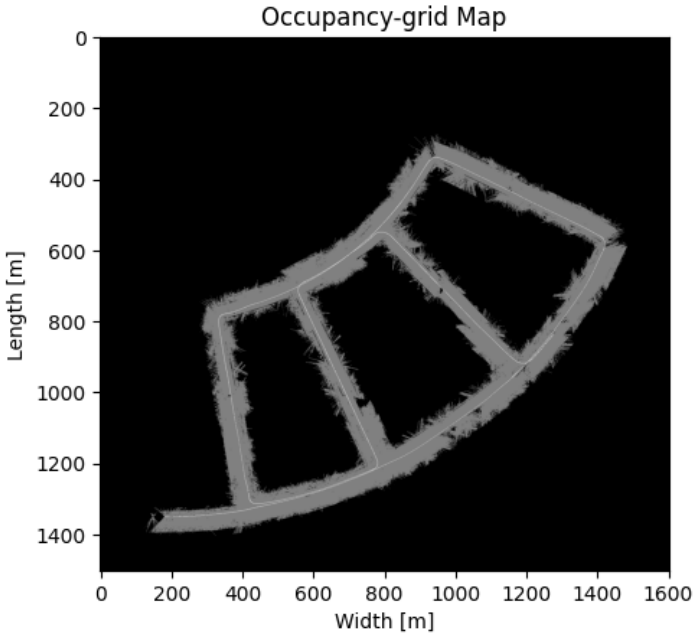

Fig. 14: Occupancy-Grid Map with Vehicle Trajectory at $1,160,507^{th}$ FOG Timestamp

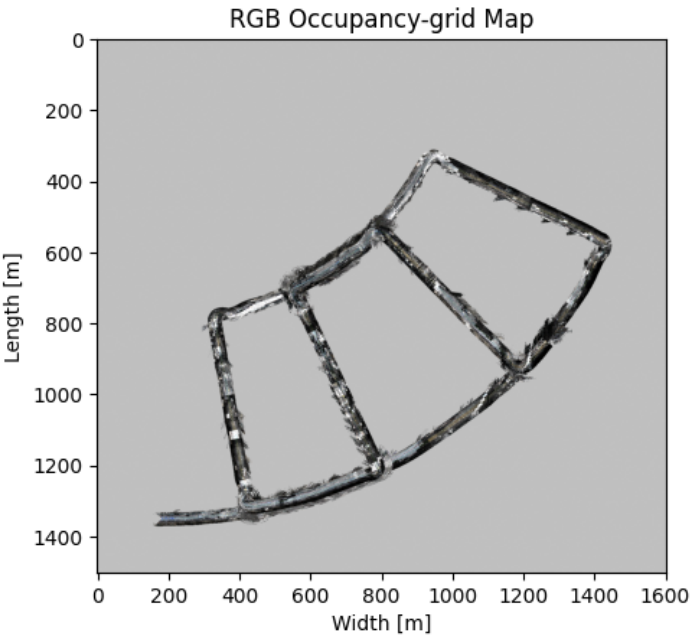

Fig. 15: Textured Map with Colors at $1,160,507^{th}$ FOG Timestamp

Figure 2, Figure 4, Figure 6, Figure 8, Figure 10, Figure 12, and Figure 14 presents the particle filter SLAM's occupancy-grid map results at different FOG timestamps. In Figure 2, Figure 4, Figure 6, Figure 8, Figure 10, Figure 12, and Figure 14, the white lines represent the vehicle trajectories, the grey pixels represent freespace in the occupancy-grid map, and the black pixels represent unexplored area or occupied area.

Figure 3, Figure 5, Figure 7, Figure 9, Figure 11, Figure 13, and Figure 15 illustrates the particle filter SLAM's texture mapping results at different FOG timestamps. In Figure 3, Figure 5, Figure 7, Figure 9, Figure 11, Figure 13, and Figure 15, the the grey pixels represent unexplored area or occupied area, and the RGB pixels, which are mostly the color of the road, represent freespace.

The most challenging part of this project is to sync the timestamps of different vehicle sensors and to optimize the particle SLAM algorithm run-time by tuning the hyper-parameters. However, overall, the implemented particle filter SLAM algorithm outputs smooth vehicle trajectories, and reasonable 2-D LIDAR scans in the occupancy-grid maps. Based on the occupancy-grid map figures, the vehicle maps out noticeable rough edges in the occupancy-grid maps because the surfaces that the 2-D LIDAR detected might not be smooth and flat. From the images that are captured by the vehicle stereo camera, the 2-D LIDAR detected a huge variety of complex surfaces, which includes humans, trees, bridges, buildings, buses, vehicles, etc. Besides that, there are a large variety of 2-D LIDAR ranges due to different distances between the detected objects and the 2-D LIDAR. Furthermore, the 2-D LIDAR instrument noise could cause noisy occupancy-grid map edges. The vehicle trajectories in the occupancy-grid map can be validated with Google Map

and the provided images from the stereo camera. The textured map figures show acceptable and reasonable results after stereo images are undistorted. The RGB pixels in the textured maps' freespaces are mostly the color of the road instead of other color pixels from images like sky blue, vehicle metallic grey, etc.

REFERENCES

[1] Kazunori Ohno, Takashi Tsubouchi, Bunji Shigematsu Shin'ichi Yuta (2004) Differential GPS and odometry-based outdoor navigation of a mobile robot, Advanced Robotics, 18:6, 611-635, DOI: 10.1163/1568553041257431

[2] Ferreira A, Matias B, Almeida J, Silva E. Real-time GNSS precise positioning: RTKLIB for ROS. International Journal of Advanced Robotic Systems. May 2020. doi:10.1177/1729881420904526

[3] Zipline. (n.d.). Technology with Purpose. Zipline. Retrieved February 24, 2022, from https://flyzipline.com/technology/

[4] Atanasov, N. (2022, February 10). ECE276A: Sensing amp; Estimation in Robotics Lecture 9: Bayesian Filtering. UCSD ECE276A: Sensing amp; Estimation in Robotics (Winter 2022). Retrieved February 24, 2022, from https://natanaso.github.io/ece276a/ref/ECE276A_9_BayesianFiltering.pdf

[5] Bagnell, D. (2014, September 11). Occupancy Maps. Retrieved February 24, 2022, from https://www.cs.cmu.edu/ 16831-f14/notes/F14/16831_lecture06_agiri_dmcconac_kumarsha_nbhakta.pdf

TABLE I: Hyper-parameters for the SLAM Algorithm

Hyper-parameter	**Description of the Hyper-parameter**	**Value**
$\Delta\lambda_{occupied}$	Occupancy-grid Map Log-odds Ratio of an Occupied Cell	$\log(4)$
$\Delta\lambda_{free}$	Occupancy-grid Map Log-odds Ratio of a Free Cell	-$\log(4)$
λ_{trust}	Percentage of Trust in the Instrument Measurement	80 %
$N_{particles}$	Number of Particles for the Particle Filter Localization	100
$N_{threshold}$	Number of Effective Particles, N_{eff} for the Stratified and Systematic Particle Resampling	0.2 $\cdot N_{particles}$
$\lambda_{\max}$	Upper-bound of the Occupancy-grid Map Log-odds Constraint	$\mathrm{N}_{particles} \cdot \Delta\lambda_{occupied}$
$\lambda_{\min}$	Lower-bound of the Occupancy-grid Map Log-odds Constraint	$\mathrm{N}_{particles} \cdot \Delta\lambda_{free}$
$\lambda_{threshold}$	The Occupancy-grid Map Log-odds Threshold Value that considered as an Occupied Grid Cell	2 $\cdot\Delta\lambda_{occupied}$