Particle Filter SLAM

Nathan Cusson-Nadeau

I. Introduction

In order for a mobile robot system to efficiently and safely carry out tasks in an unknown environment, it is necessary for this agent to be able to simultaneously localize and map its environment. As humans, we often take this task for granted when introduced into a new environment. We can easily use our senses to quickly detect solid objects and move effortlessly around new surroundings. For a robot this is not a trivial task. In order to know where an agent is in an unknown environment, it is necessary to have a map displaying obstacles. But to generate this map, an agent must know where it resides within it. This introduces a chicken-and-egg problem that seems very difficult to solve. However, through the use of Bayesian filtering techniques, this simultaneous localization and mapping (or SLAM) problem can be solved.

In this paper, we implement SLAM using a particle filter and previously recorded measurements from an autonomous car equipped with optometry, 2-D LiDAR and stereo camera sensors. From this data we generate a 2-D occupancy grid map of the environment as would have been generated at run-time for the car's trip. Then, this occupancy grid is textured with RGB images using stereo camera observations. The technical approach will still be described in theory but no results or algorithms will be discussed.

II. Problem Formulation

Consider an autonomous vehicle traversing an unknown environment $\mathbf{m}$ with state $\boldsymbol{x_t}$ that evolves over trip time horizon $t \in \{0, T\}$ with motion model:

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

where $\boldsymbol{w_t}$ is a random variable which represents the motion noise. The vehicle observes its state $\boldsymbol{x_t}$ and environment $\boldsymbol{m_t}$ using various sensors and updates its state using the observation model:

$$\boldsymbol{z_t} = h(\boldsymbol{x_t}, \boldsymbol{v_t}) \tag{2}$$

where $\boldsymbol{v_t}$ represents observation noise. We assume the environment map $\boldsymbol{m_t}$ to be completely occupied at time $t = 0$ and to update over the time horizon T according to the following case structure:

$$m_i := \begin{cases} 1, & Occupied \\ -1, & Unoccupied \end{cases}$$

Problem: Given observations z_t and states x_t, $\forall t \in \{0, T\}$ update the 2-D occupancy map $\boldsymbol{m_t}$ and plot the trajectory of the vehicle using (x_t, y_t) in m using dead-reckoning. We assume the elevation $\forall t$ remains constant. Additionally, create a 2-D RGB textured map $\mathbf{m}_{RGB}$:

$$\mathbf{m}_{RGB} := \begin{cases} m_{i,RGB} = (R, G, B), & m_i = 1 \\ m_{i,RGB} = (0, 0, 0), & m_i = -1 \end{cases}$$

where $R, G, B \in \{0, 255\}$ represent the red, green and blue pixel values respectfully.

III. Technical Approach

A. Particle Filter Initialization

To update m_t we elected to use the probabilistic inference technique of Bayesian filtering. In particular, we applied a special case of a Bayes filter called the particle filter. The particle filter was implemented with an initial N particles, $\{\mu^k_{t|t}, \alpha^k_{t|t}\}$, where $\mu^k_{t|t} \in \mathbb{R}^3$ represents a particle $k \in \{1, N\}$ hypothesized state $x_t = [x, y, \theta]^T$ at time t, and $\alpha^k_{t|t} \in [0, 1]$ represents the weight of this particle. The initial particle count was set to $N = 300$ and weights were initialized to be equal $\alpha^k_{t|t} = \frac{1}{N}$. The threshold for resampling was $N_{thresh} = 0.6N$. An initial fully occupied map $m_{t=0}$ where $m_{i,j} = 1 \; \forall i, j$ was created with dimensions given in Table I.

Occupancy Map Parameters	
Map Parameters	Values (meters)
Resolution	1
x-min	-300
y-min	-1300
x-max	1600
y-max	500

TABLE I: Occupancy Map Parameters

The set of particles $\{\boldsymbol{\mu}_{t|t}, \boldsymbol{\alpha}_{t|t}\}$ can be equivalently represented as a probability density function:

$$p_{t|t}(\boldsymbol{x}_t) = \sum_{k=1}^{N_{t|t}} \alpha^{(k)}_{t|t} \delta(\boldsymbol{x}_t - \boldsymbol{\mu}^{(k)}_{t|t}) \tag{3}$$

$$p_{t+1|t}(\boldsymbol{x}_t) = \sum_{k=1}^{N_{t+1|t}} \alpha^{(k)}_{t+1|t} \delta(\boldsymbol{x}_{t+1} - \boldsymbol{\mu}^{(k)}_{t+1|t}) \tag{4}$$

and will be occasionally referenced as such in future sections. The main algorithm for the particle filter can be found at algorithm 1.

B. Update Step

The update step of the particle filter takes the following form:

ALGORITHM 1

Particle Filter - Main Algorithm

Given, N, N_{th}, $\{\boldsymbol{\mu}_{t=0}, \boldsymbol{\alpha}_{t=0}\}$, α_{th}, T, $\boldsymbol{m}$
for t = 0 to T in steps of 30 **do**
  Update Step:
  $s_l \leftarrow [r\cos\theta_l, r\sin\theta_l, 0]^T$
  $s_b \leftarrow {}_bR_l s_l + p_l$
  if $t = 0$ **then**
    skip update
  else
    Get State Velocity
    $v_L \leftarrow \frac{\pi \times d \times \Delta z_{l,t}}{4096 \times \tau_e}$
    $v_R \leftarrow \frac{\pi \times d \times \Delta z_{r,t}}{4096 \times \tau_e}$
    $v_{(avg,t)} \leftarrow \frac{v_{R,t} + v_{L,t}}{2}$
    for j = $10 \times (t - stepSize)$ to $10 \times t$ **do**
      Get State Angular Velocity and Orientation
      $\omega_t \leftarrow \frac{\Delta\Psi_j}{\tau_f}$
      $\theta_t \leftarrow \theta_{t-1} + \Delta\Psi$
    end for
    Compute Rotation Matrix:
    ${}_wR_b \leftarrow \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  end if
  for k = 1 to N **do**
    $\boldsymbol{s}_w \leftarrow {}_wR_b s_b + p_b$
    for f = 1 to n_{valid} in steps of 2 **do**
      Get Cells Where Valid Laser f Passed Thru
      cells $\leftarrow$ **bresenham2D**($\boldsymbol{\mu}_t^k, \boldsymbol{s}_w^f$)
    end for
    $\boldsymbol{c}^k \leftarrow$ mapCorrelation($\boldsymbol{m}$,cells)
    $c_{max}^k \leftarrow$ index where $\boldsymbol{c}^k = max(\boldsymbol{c}^k)$
  end for
  Resampling Step:
  $\alpha \leftarrow \frac{\alpha \times c}{\sum_{k=1}^{N} \alpha^k \times c^k}$
  if $\alpha^k \leq \alpha_{th}, \alpha^k \leftarrow 0$
  $\alpha^{index,max} \leftarrow$ index of max(α)
  if $N_{eff} \leq N_{th}$ **then**
    Replenish particles
    $\mu_t^{deleted} \leftarrow \mu_t^{max} + \boldsymbol{\mathcal{N}}(0, \boldsymbol{\sigma}^2)$
  end if
  $\boldsymbol{m}_t \leftarrow \boldsymbol{\lambda}_t$
  Prediction Step:
  for k = 1 to N **do**
    if t = 0 **then**
      skip prediction
    else
      $\boldsymbol{\mu}_{t+1} \leftarrow \boldsymbol{\mu}_t + \tau_e \begin{bmatrix} v_t cos(\theta_t) \\ v_t sin(\theta_t) \\ \omega_t \end{bmatrix} + \boldsymbol{w}_t$
    end if
  end for
end for

$$p_{t+1|t+1}(x) = \sum_{k=1}^{N} \frac{\alpha_{t|t}^k p_h(\boldsymbol{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^k)}{\sum_{j=1}^{N_{t+1|t}} \alpha_{t|t}^j p_h(\boldsymbol{z}_{t+1}|\boldsymbol{\mu}_{t+1|t}^j)} \delta(\boldsymbol{x} - \boldsymbol{\mu}_{t+1|t}^k) \tag{5}$$

We begin the filter with the update step in order to generate a prior set of states:

$$\boldsymbol{x}_t|\boldsymbol{z}_{0:t} \rightarrow p_{t|t}(\boldsymbol{x}_t)$$

Observations $\boldsymbol{z}_t$ were acquired from odometry and LiDAR measurements from the respective sensor's frame $\{S\}$. In order to use this measurement data as an observation in the world frame $\{W\}$, it was necessary to conduct two transforms, first from sensor $\{S\}$ to body $\{B\}$ and another from body $\{B\}$ to world $\{W\}$. For the odometry measurements, this was a trivial transformation because the yaw angle Ψ from the FOG was purely rotational ${}_wR_b$ and encoder measurements were purely translational, ${}_bp_w$, and most importantly took place from the origin of the body frame $\{B\}$. Using the parameters from Table II and elapsed time from the prior measurement $\tau_{e,f}$, the following equations could be used to generate part of the observations and estimate the control input u_t.

$$\theta_t = \theta_{t-1} + \Delta\Psi \tag{6}$$

$$\omega_t = \frac{\Delta\Psi}{\tau_f} \tag{7}$$

$$v_L = \frac{\pi \times d_l \times \Delta z_{l,t}}{4096 \times \tau_e} \tag{8}$$

$$v_R = \frac{\pi \times d_r \times \Delta z_{r,t}}{4096 \times \tau_e} \tag{9}$$

$$v_{(avg,t)} = \frac{v_{R,t} + v_{L,t}}{2} \tag{10}$$

$${}_wR_b = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 \\ \sin\theta_t & \cos\theta_t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$${}_w\boldsymbol{p}_b = \begin{bmatrix} v_{avg}\cos\theta_t \\ v_{avg}\sin\theta_t \\ 0 \end{bmatrix}$$

Note that at $\theta_{t=0} = 0$ and $v_{avg,t=0} = 0$ because $\{W\} = \{B\}$ at $t = 0$ and we assume to start from rest. The z-coordinate of ${}_wp_b$ is 0 because we assume the vehicle elevation change is negligible for all time and is not necessary to create a 2-D occupancy map.

TABLE II: Encoder Parameters

Encoder Parameters	
Parameters	**Values**
Resolution (ticks/rev)	4096
L, Wheel Diameter (m)	0.623479
R. Wheel Diameter (m)	0.622806

TABLE III: LiDAR Parameters

Encoder Parameters	
Parameters	**Values**
FOV (degrees)	190
Start Angle (degrees)	-5
End Angle (degrees)	185
Resolution (degrees)	0.666
Max Range (meters)	80

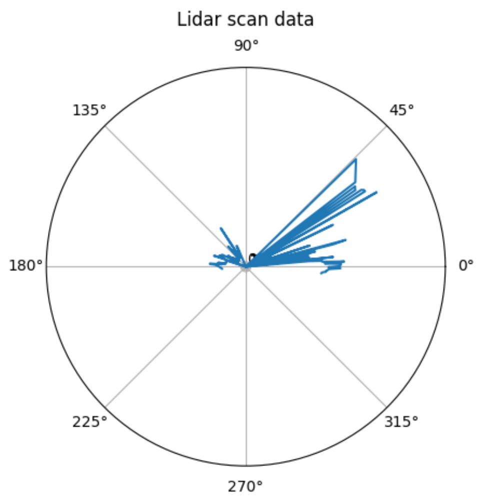

Fig. 1: First LiDAR Sweep

1) LiDAR Ray Tracing and World Frame Transformation: LiDAR measurements required extensively more work to be converted into usable form for the filter. LiDAR data at each time t was represented as a sweep of laser ranges in from angles $-5°$ to $185°$ as seen in Table III and Figure 1. To use these laser rays to determine which space was occupied or unoccupied, we implemented the Bresenham Ray Tracing Algorithm in 2D (see algorithm 2) at each particle for each time step t. This provided the coordinates s_l of occupied or unoccupied 1x1 meter cells in the LiDAR frame $\{S\}$. This then allowed the observed cell coordinates to be transformed into the world frame $\{W\}$ using the following series of transformation equations:

$$\mathbf{s}_b = {}_bR_l s_l + {}_b\mathbf{p}_l \tag{11}$$

$$\mathbf{s}_w = {}_wR_b s_b + {}_w\mathbf{p}_b \tag{12}$$

where ${}_bR_l$ and ${}_bp_l$ were given from the measuring the pose of the LiDAR sensor in relation to the origin of the body frame $\{B\}$:

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

$${}_bp_l = \begin{bmatrix} 0.8349 & -0.0126869 & 1.76416 \end{bmatrix}^T$$

ALGORITHM 2
Bresenham Ray Tracing in 2D

Given, start (sx,sy) and end (ex,ey) points of ray:
$\Delta x \leftarrow |ex - sx|$
$\Delta y \leftarrow |ey - sy|$
$steep \leftarrow |dy| > |dx|$
if $steep$ **then**
  $dx, dy = dy, dx$
end if
if $dy = 0$ **then**
  $q \leftarrow zeros(dx + 1, 1)$
else
  $q \leftarrow append(0, greaterOrEqual(diff(mod(arange(floor(dx/2), -dydx + floor(dx/2) - 1, -dy), dx)), 0))$
end if
if $steep$ **then**
  if $sy \leq ey$ **then**
    $y \leftarrow arange(sy, ey + 1)$
  else
    $y \leftarrow arange(sy, ey - 1, -1)$
  end if
  if $sx \leq ex$ **then**
    $x \leftarrow sx + cumsum(q)$
  else
    $x \leftarrow sx - cumsum(q)$
  end if
else
  if $sx \leq ex$ **then**
    $x \leftarrow arange(sx, ex + 1)$
  else
    $x \leftarrow arange(sx, ex - 1, -1)$
  end if
  if $sy \leq ey$ **then**
    $y \leftarrow sy + cumsum(q)$
  else
    $y \leftarrow sy - cumsum(q)$
  end if
end if
return $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$

The body to world transformation variables ${}_wR_b$ and ${}_w\mathbf{p}_b$ were required to be calculated online at each time step by using the estimated state $\boldsymbol{x}_{t+1}$. At $t = 0$ these values could be defined as ${}_wR_b = I$ and ${}_w\mathbf{p}_b = 0$ because the vehicle can be assumed to start at the origin of $\{W\}$.

2) Resampling Step: As an extension to the update step, we then resample the set $\{\boldsymbol{\mu}_{t|t}, \boldsymbol{\alpha}_{t|t}\}$ to adjust the weights of the probability mass function $\boldsymbol{\alpha}$ and create new particles $\boldsymbol{\mu}_{t|t}$ if necessary.

To adjust the weights, a map correlation function was used which compares the current map $\boldsymbol{y}^{(k)}$ of each particle based on z_t against the prior map $\boldsymbol{m}_{t-1}$. Using a correlation function:

$$c = corr(\mathbf{y}, \mathbf{m}) := \sum_i \mathbb{1}\{y_i = m_i\} \quad (13)$$

we computed each particle's correlation $c^{(k)}$ then correlations to find which particle correlated the best $c_{max} = max(\boldsymbol{c})$. Weights of each particle were then shifted to correspond with these new values by the following formula:

$$\alpha_t^{(k)} = \frac{\alpha_{t-1}^{(k)} c^{(k)}}{\sum_{k=1}^{N} c(k)} \quad (14)$$

If any weights were less than or equal to a threshold $\alpha_t^{(k)} \leq \alpha_{th}$ these particles would be deleted to lower computation time and to remove low correlation hypotheses. α_{th} was set to 0.05

To avoid particle depletion, the number of effective particles was computed at each time step:

$$N_{eff} := \frac{1}{\sum_{k=1}^{N} (\alpha_{t|t}^{k})^2} \leq N_{threshold} \quad (15)$$

If N_{eff} passed below the threshold value, new particles were distributed around high correlation in proportion to their current weight.

Note: This part was not coded in due to time constraints. This high-level description of the intended particle re-implementation methodology is all the grader will find. In code, N=3 and new particles besides heaviest were deleted and respawned in a normally distributed radius $X \sim \mathcal{N}(\mu, \sigma^2)$ with $\sigma^2 = 0.1\mu$ around highest weight particle's estimated location.

C. Log-Odds Update and Occupancy Map Generation

Using the highest weighted particle $\mu_t^{(kmax)}$, we then update the map by accumulating the log-odds ratio for each observed cell (see equation 16) around the particle. We assume the LiDAR sensor to be quite accurate and therefore assign a log-odds ratio $\Delta\lambda_{i,t} = \pm \log 9$

$$\Delta\lambda_{i,t} = \log \frac{p(m_i = 1|\boldsymbol{z}_t, \boldsymbol{x}_t)}{p(m_i = -1|\boldsymbol{z}_t, \boldsymbol{x}_t}) = \begin{cases} +\log 9, \boldsymbol{z}_t \rightarrow m_i \text{ occupied} \\ -\log 9, \boldsymbol{z}_t \rightarrow m_i \text{ free} \end{cases} \quad (16)$$

To avoid overconfidence in the odd-log map, we introduce constraints for each cell:

$$\lambda_{i,t}^{max} = +3\Delta\lambda_{i,t}$$

$$\lambda_{i,t}^{min} = -3\Delta\lambda_{i,t}$$

Using $\boldsymbol{\lambda}_t$ we construct $\boldsymbol{m}_t$ using the following formula:

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

D. Prediction Step

Using a predictive motion model obtained from an odometry based discrete-time differential-drive kinematic model, the trajectory of the vehicle was estimated at each time step t and applied to each hypothesis $\mu_{t|t}^{k}$.

$$\boldsymbol{\mu}_{t+1|t}^{(k)} = f(\boldsymbol{\mu}_{t|t}^{(k)}, \boldsymbol{u}_t + \boldsymbol{w}_t)$$

where the control input $\boldsymbol{u}_t$ is computed using the estimated values from the observation step $\boldsymbol{z}_t$.

The motion model used takes the form:

$$\boldsymbol{x_{t+1}} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} := \boldsymbol{x_t} + \tau \begin{bmatrix} v_t cos(\theta_t) \\ v_t sin(\theta_t) \\ \omega_t \end{bmatrix} + \boldsymbol{w_t} \quad (17)$$

Here, x_{t+1}, y_{t+1}, and θ_{t+1} represent the robot's predicted coordinates in the world frame at the next time step $t+1$. This new state is computed from adding the prior state $\boldsymbol{x_t}$ to the change in coordinates and angle multiplied by τ, the time between prediction steps. $\boldsymbol{w_t} \in \mathbb{R}^3$ represents Gaussian noise proportional to each state and with covariance σ and mean μ_N

$$w(\cdot) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\cdot - \mu_N)^2/2\sigma^2}$$

For our approach, $\sigma_{velocity}$ was 0.5% of v_{avg} and the σ_ω was 0.05% of ω_t

E. Generating Textured Map

Note: Due to time constraints the textured map coding portion could not be completed. A theoretical technical approach will be discussed here but will not coincide with any code.

Once the particle filtering is completed, the occupancy grid $\boldsymbol{m}$ is obtained. To texture this map, we first need to compute the disparity d between the two images Img_L and Img_R. This is done by choosing corresponding pairs of pixels u_L and u_R in both the left and right images and using the formula:

$$d = u_L - u_R = \frac{1}{z} f s_u b \quad (18)$$

From the disparity, this same equation can be used to estimate the depth of any set of pixels. From a measured baseline b and known fs_u, fs_v the world coordinate z_t of a pixel. Using the stereo camera model:

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_u b \end{bmatrix} \frac{1}{z} \begin{bmatrix} fs_u \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (19)$$

we can then compute the sensor frame coordinates s_S of a given pixel (u_L, v_L). Finally, the sensor frame coordinates can be converted to the world frame using the following equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = {}_oR_r R^T(\mathbf{s}_S - \mathbf{p}) \tag{20}$$

Where ${}_oR_r \in SO(3)$ represents the optical rotation matrix:

$${}_oR_r = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix}$$

and $R \in SO(3)$ is the measured extrinsic rotation matrix for the left stereo camera:

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

and $\mathbf{p}$ is the position in the body frame $\{B\}$ of the left stereo camera.

By converting the image to RGB the using the respective pixels RGB value can be obtained. By using a range of $z \in (-0.5, 0.5)$ meters, RGB pixel values of pixels with z world coordinates in that range can be projected onto $\boldsymbol{m}$ where $m_i = -1$ (i.e. is free).

TABLE IV: Stereo Camera Parameters

Stereo Camera Parameters	
Parameters	**Values**
Baseline (cm)	475.1436

IV. Results

A. Dead-Reckoning

To verify our prediction step worked, a plot of a particle's trajectory without any noise (dead-reckoning) was estimated and graphed over the time horizon. Figure 2 depicts the vehicle's trip over all time within the generated occupancy map $\boldsymbol{m}$.

Figures 3, 4, and 5 depict this same trajectory at each quarter of the time horizon.

B. LiDAR Scan

Next, to verify our update step worked properly, we plotted the LiDAR sweeps through time (i.e. compute the occupancy map $\boldsymbol{m}$), using an arbitrary particle's $\mu^{arbitrary}$ trajectory as the body frame $\{B\}$ coordinates in the Bresenham2D algorithm. See Figure 6.

1) Observations and Room for Improvement: Because the resampling step of our algorithm did not work, results depicting the occupancy map through time with multiple particles and noise are not available. However, plotting an arbitrary particle with noise and using it to generate an occupancy grid still yielded interesting results. In Figure 7 we see that the trajectory drifts quite far off course throughout time. This is why the resampling step is critical when z_t carries uncertainty.

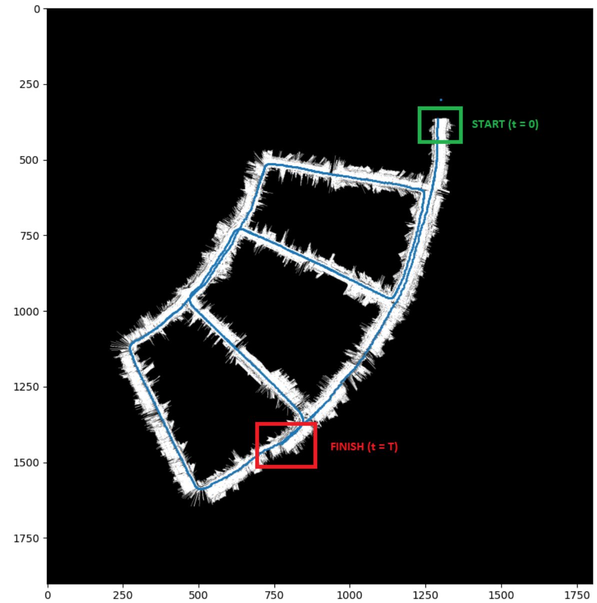

Fig. 2: Dead-Reckoning All Time

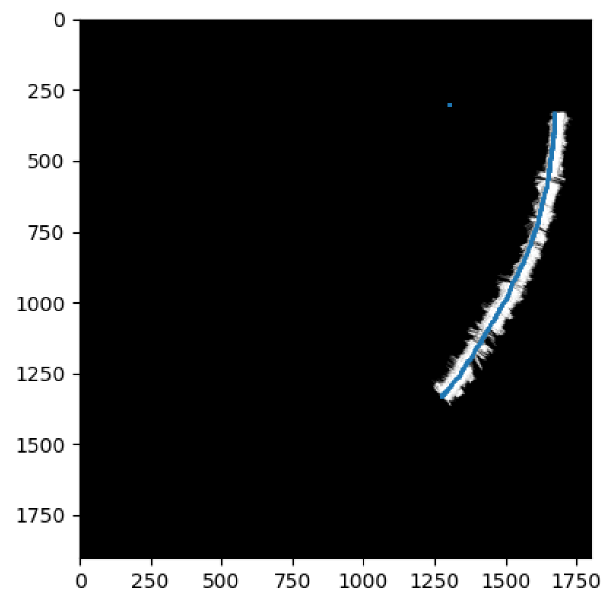

Fig. 3: Dead-Reckoning T/4

Through using multiple particles, the presence of noise is mitigated because even if the original highest weighted particle veers off-course, another particle has a high likelihood of being close to the true state $\boldsymbol{x_t}$ (given N is sufficiently large). In other words, the correlation of the plotted particle would drop, and by extension it's respective α_t and it would not have been used to compute the current map m_t

Lastly, this project has shown the importance of writing code and algorithms that are computationally efficient. As the particle size increased, computational times scaled exponen-

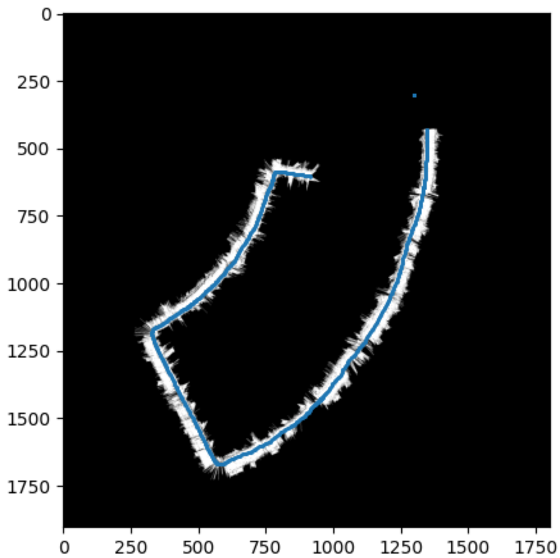

Fig. 4: Dead-Reckoning T/2

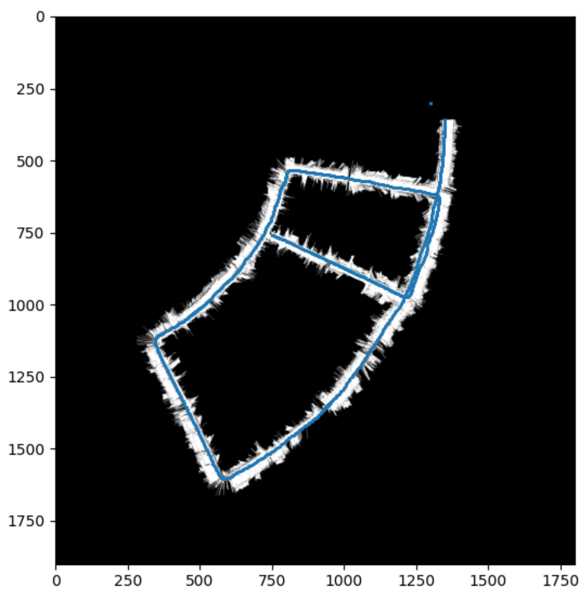

Fig. 5: Dead-Reckoning 3T/4

tially because of the way our code was written. This not only made completion of the project difficult to do in time but means that when implementing such an algorithm online the SLAM procedure would update very slowly - an undesirable result in any robot application where the environment must be quickly mapped.

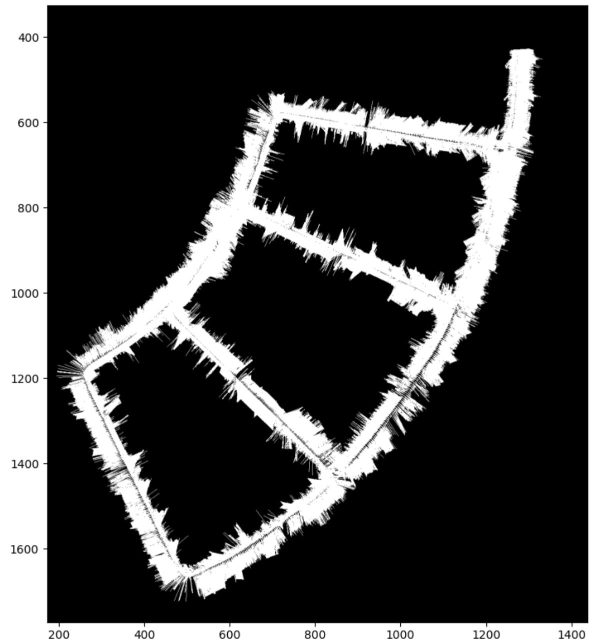

Fig. 6: Occupancy Map

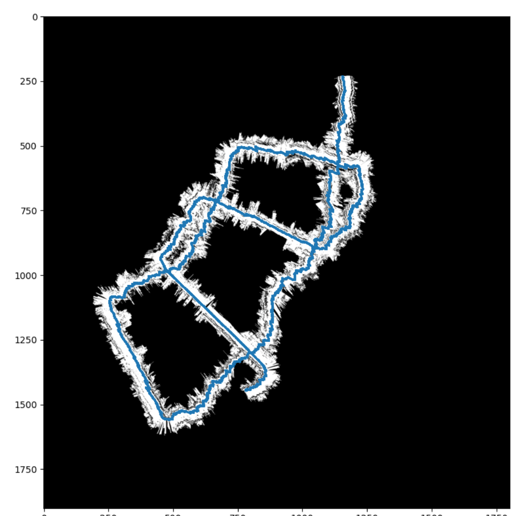

Fig. 7: All LiDAR Scans Without Resampling

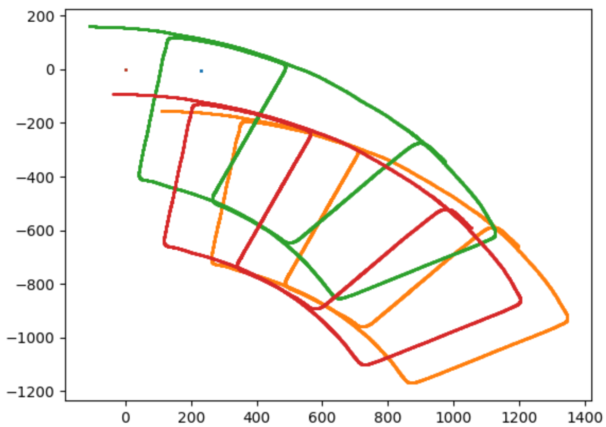

Fig. 8: 3 Particles with Guassian Noise

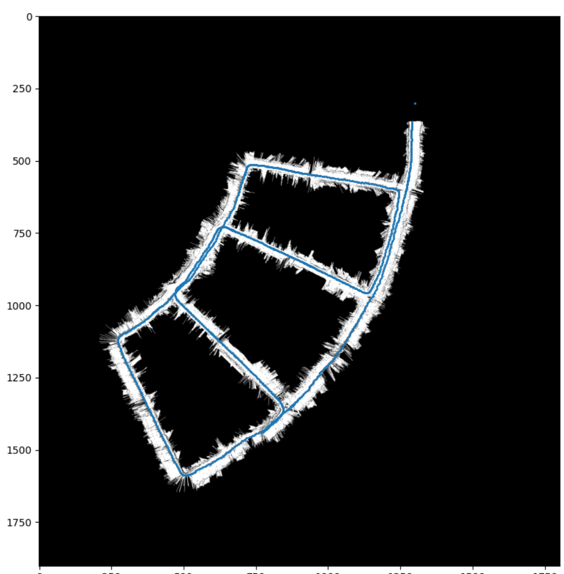

Fig. 9: Occupancy Map: 0.5 Percent Sigma Velocity, 0.05 Percent Sigma Omega

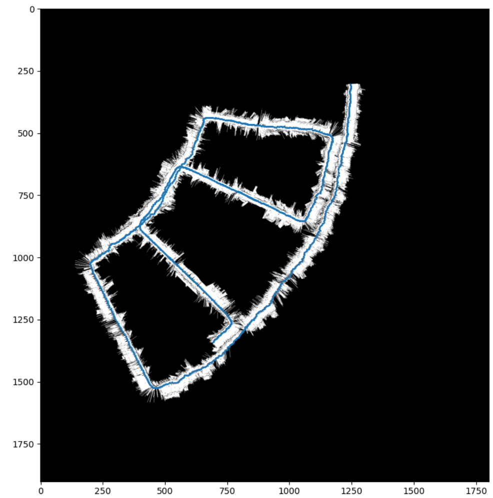

Fig. 10: Occupancy Map: 1 Percent Sigma Velocity, 0.1 Percent Sigma Omega

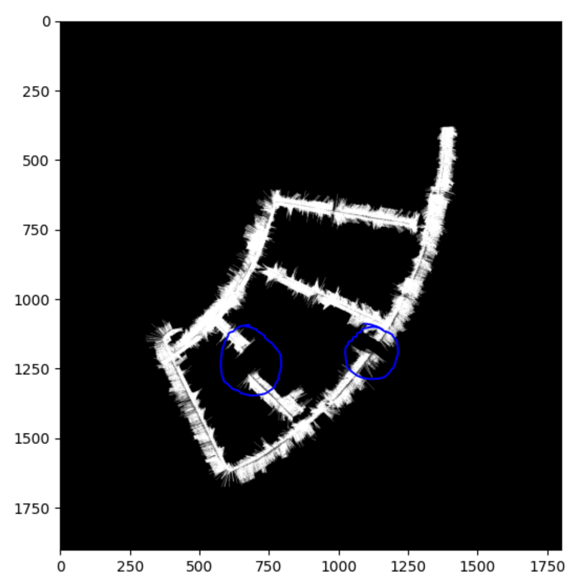

Fig. 11: Occupancy Map Using 3 Particles

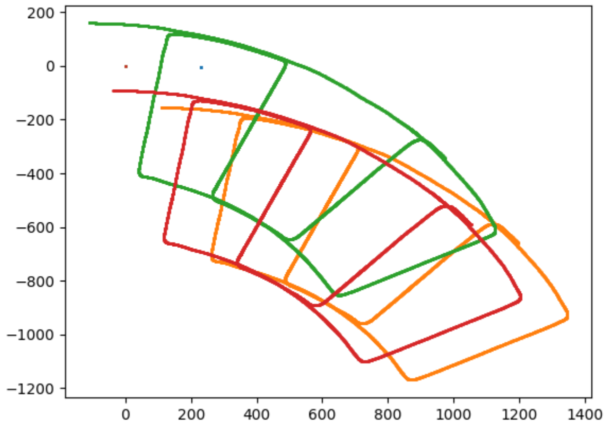

Fig. 12: 3 Particles with Guassian Noise

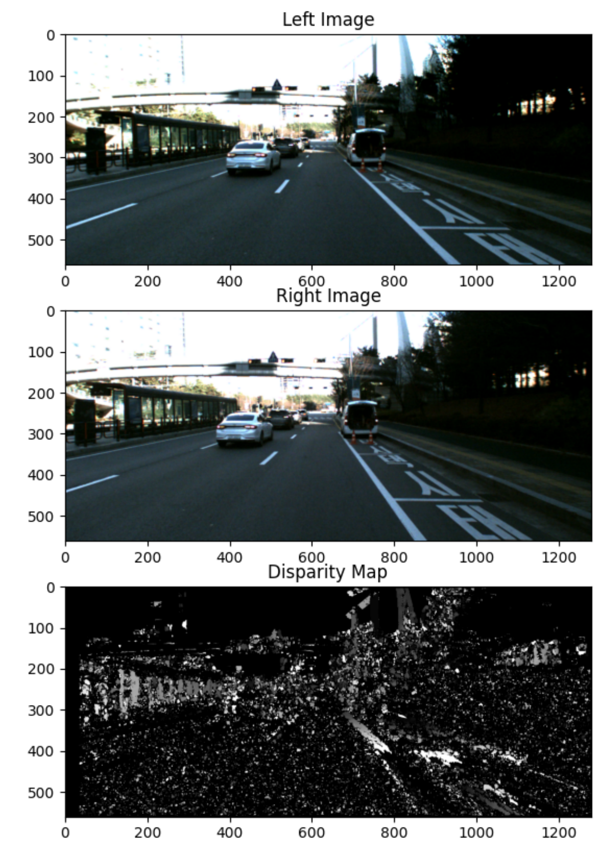

Fig. 13: Disparity Image of First Timestep