ECE276A Project 2 Report

Yuchen Zhang[1]

Abstract– **In this project, particle filter is employed to fuse on-board sensor information from LIDAR, FOG, encoders, and stereo camera into accurate localization and a partially colored map with 0.2m resolution. The code is optimized to run the SLAM portion in real time with 100 particles.**

I INTRODUCTION

Autonomous driving is desirable to prevent traffic accident due to human error. Autonomous driving must rely on accurate and robust localization.

Besides GPS, localization from the vehicle itself with its own sensors provide a backup when GPS signal is unstable, which is a common issue in cities. In this project, particle filter is employed to fuse on-board sensor information from LIDAR, FOG, encoders, and stereo camera into accurate localization and a partially colored map with 0.2m resolution.

II PROBLEM FORMULATION

With the assumption that the robot pose is identity at time $t = 0$, build and maintain at any time instance t

1. A probability distribution of the robot pose based on all available information, i.e., $p(x_t|x_{0:t-1}, u_{0:t-1}, z_{0:t})$ where $x_t \in \mathrm{SE}(2)$
2. A occupancy grid map $m \in \mathbb{R}^{M\times N}$ with each element, $\gamma_{(u,v)}$ represent the probability of a map cell (u, v) being occupied.
3. A colored grid map $m_c \in \mathbb{R}^{M\times N\times 3}$ with each RGB pixel at (u, v) represent the color of map cell (u, v).

Based on the following available information:

1. Encoder tick count of left and right wheels $E_L, E_R \in \mathbb{Z}$ at time instance $0 : t-1$ and their time stamps, diameter of both wheels D_L, D_R, wheelbase w_b
2. FOG measurement on orientation change $\Delta\theta \in \mathbb{R}$ at time instance $0 : t-1$ and their time stamps
3. LIDAR measurement of distance $r_i \in \mathbb{R}$ from a fixed set of angles $-5^\circ \sim +185^\circ$ with fixed interval of 0.66°
4. Static frame transformations ${}_{\{F\}}T_{\{L\}}$, ${}_{\{F\}}T_{\{S\}}$, ${}_{\{V\}}T_{\{F\}}$
5. Image $\mathbb{R}^{(H\times W\times 3)}$ from left and right camera of the stereo camera S up time instance t. Camera intrinsic matrix K and static transformation between the cameras represented by base_line.

III TECHNICAL APPROACH

In this project, camera information is not used in the SLAM part, i.e., texture mapping will not influence SLAM part. Therefore, I separated SLAM and texture mapping: SLAM will produce a final map and a history of best estimation of vehicle pose, texture mapping then utilize those information to color the final map.

I SLAM

The SLAM part consists three repeating steps: prediction, update, and mapping. prediction is based on motion model $p_f(x_t|x_{t-1}, u_{t-1})$ that update the belief to the best estimation at the current time instance. Update is based on observation model $P_h(z_t|x_t, m)$ that update the best estimation according to measurement z_t and forming a posterior. During mapping, a single pose is selected by the MAP method and the map is updated according to the inverse observation model. To prevent particle depletion, resampling is done before prediction step when needed.

I.1 Assumptions and State Representation

Assumptions

1. Markov assumption

$$p(x_t|x_{0:t-1}, u_{0:t-1}, m) = p(x_t|x_{t-1}, u_{t-1}, m)$$
$$p(z_t|x_{0:t}, u_{0:t-1}, m) = p(z_t|x_t, m)$$

2. Map cell independence given current state

$$p(m|x_t) = \prod_{(u,v)} p(m_{(u,v)}|x_t)$$

Where m_i represent the cell i in map m.

State Representation The probability distribution of robot state $p(x_t|x_{0:t-1}, u_{0:t-1}, m)$ is approximated by a discrete pmf represented by particles $\mu_i \in SE(2)$ and their probability, or weights, $w_i \in [0, 1]$. The robot pose is the transformation from the FOG frame to the Global frame.

At time instance t,

$$\mathbb{P}(x_t) = \begin{cases} w_i & , x_t = \mu_i \\ 0 & , \text{otherwise} \end{cases}$$

The map is represented as a $2D$ occupancy grid with equal resolution in the x, y direction. the relation between index (u, v) and true location (x, y) is:

$$(u, v) = \lfloor (x, y)/(\text{map\_res} = 0.2m) \rfloor$$

The probability distribution of the map, according to the assumptions, is represented as the product of individual map cells. The probability of $m_{(u,v)}$ is modeled as a bernoulli distribution. For simplicity define $+1$ as occupied and -1 as empty.

$$P(m_{(u,v)} = 1|x_{0:t}, z_{0:t}) = \gamma_{(u,v),t}$$
$$P(m_{(u,v)} = -1|x_{0:t}, z_{0:t}) = 1 - \gamma_{(u,v),t}$$

For implementation, the map state is saved as log-odds form, i.e. $\lambda_i = \log\left(\frac{P(m_{(u,v)}=1|x_t)}{P(m_{(u,v)}=-1|x_t)}\right) = \log\left(\frac{\gamma_{(u,v)}}{1-\gamma_{(u,v)}}\right)$. The probability can be recovered by $\gamma_{(u,v)} = \frac{1}{1+\exp(-\lambda_{(u,v)})}$

I.2 Timestamp Management

The data given in this project does not have their time stamp synchronized. In this project, control input u is assumed to be constant unless updated by the encoder or FOG information and time is dynamically discredited. Period between two discrete time instance is denoted as τ. The following prediction and update steps are triggered by sensor inputs.

I.3 Prediction-Motion Model

Sensor information processing When a Encoder or FOG information is available, the prediction step is triggered. Firstly, the control input $u = [v, \omega]$ is updated according to sensor input:

- Encoder information:

$$v := \frac{\pi * (D_L \Delta E_L + D_R \Delta E_R)}{2 * \text{CPR} * \Delta\text{ts\_enc}}$$

In which D_L, D_R are the diameter of the left, right wheels, $\Delta E_L, \Delta E_R$ are the change in encoder ticks compared to last encoder information received, Δts_enc is the time between current and last encoder information.

- FOG information:

$$\omega := \frac{\Delta\theta}{\Delta\text{ts\_fog}}$$

Where $\Delta\theta$ is directly provided by the FOG and Δts_fog is the time between current and last FOG information.

Prediction For each particle,

$$\mu_t = f(\mu_{t-1}, \tau_t(u + w)), \quad w \sim N(0, W = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix})$$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = f\left(\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix}, \begin{bmatrix} dl \\ d\theta \end{bmatrix}\right) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \cos(\theta_{t-1} + \frac{d\theta}{2}) \\ \sin(\theta_{t-1} + \frac{d\theta}{2}) \\ d\theta \end{bmatrix}$$

And the weights are unchanged.

$$w_t = w_{t-1}$$

Where $u = [v, \omega]$ that consists the linear and angular velocity, $w \sim N(0, W)$ is the motion noise, and τ *is the time between current and last prediction.* The prediction step, although not preformed with fixed time interval, always predict for same length of time in total, in accordance with real time in the data.

I.4 Update-Observation Model

The update step is triggered whenever a lidar scan is available.

Prediction Firstly, the predict step is called to predict to current time to obtain the best estimate of the robot state distribution.

Frame Transformation Given raw laser measurement $z_t = \{(\alpha_1, r_1), ..., (\alpha_k, r_k)\}$, only those with $2\text{m} < r_k < 75\text{m}$ are kept. For any robot state hypothesis pose (x, y, θ), the following formula is used to project the laser hit points to 2D and transform them to 2D global frame.

$$p_k^i = {}_{\{G\}}T_{\{F\}} \cdot P \cdot {}_{\{F\}}T_{\{L\}} \begin{bmatrix} \cos(\alpha_k) r_k \\ \sin(\alpha_k) r_k \\ 0 \\ 1 \end{bmatrix}$$

Where,

$$\begin{aligned} {}_{\{G\}}T_{\{F\}} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix} \\ P &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}_{\{F\}}T_{\{L\}} &: \text{see List of Static Variables} \end{aligned}$$

In other words, all 3D lidar hit points are first transformed to and projected along the z direction of the vehicle frame, then transformed to 2D global frame according to the state hypothesis.

Correlation Computation For each particle carrying state hypothesis μ^i, it is shifted according to a fixed grid in state space(SE2) to produce a set of shifted copy of particles $\mu_l^{(i)}$. Denote this perturbation grid as ordered list $G = \{(dx_l, dy_l, d\theta_l)\}_{l=1}^L$

I tried different grids, the final grid is to change the orientation by $-0.005, 0, +0.005$ rad and does not shift the x,y position. Each particle have three shifted copies.

Lidar observation z_t is transformed according to each shifted copy particle. Denote shifted hit points for a shifted particle $\mu_l^{(i)}$ as $\{p_{l,k}^{(i)}\}_{k=1}^K$.

Denote the cell index in which set of laser hit points $\{p_{l,k}^{(i)}\}_{k=1}^K$ lands as (u_k, v_k), correlation between and map m is defined as:

$$\begin{aligned} \operatorname{corr}(\{p_{l,k}^{(i)}\}_{k=1}^K, m) &= \sum_{k=1}^K \mathbb{P}(m_{(u_k,v_k)} = +1) \\ &= \sum_{k=1}^K \frac{1}{1 - \exp(-\lambda_{(u_k,v_k)})} \end{aligned}$$

Proposal Adjustment Based on the correlation result for shifted copies of particle $p^{(i)}$, adjustment is made to the proposal distribution to match the observation.

$$\begin{aligned} L^{(i)} &= \arg\max_l \operatorname{corr}(\{p_{l,k}^{(i)}\}_{k=1}^K, m) \\ \Delta\mu^{(i)} &= \frac{1}{|L^{(i)}|} \sum_{l \in L^{(i)}} G_l \\ \mu^{(i)} &:= \mu^{(i)} + 0.05\Delta\mu^{(i)} \end{aligned}$$

Correlation is calculated for each shifted copy and the highest correlation is kept for the original particle. Denote the highest correlation for all particles $C^{(i)}$.

$$C^{(i)} = \max_l \operatorname{corr}(\{p_{l,k}^{(i)}\}_{k=1}^K, m)$$

Update The observation model is defined as:

$$\begin{aligned} \mathbb{P}(z_t | x_t = \mu^{(i)}) &= \frac{1}{\eta} \exp(C^{(i)}/5) \\ \eta &= \sum_i \exp(C^{(i)}/5) \end{aligned}$$

And the particle weight is updated according to:

$$\begin{aligned} w_t^{(i)} &= \mathbb{P}(x_t = \mu^{(i)} | x_{0:t-1}, u_{0:t-1}, z_{0:t}, m) \\ &= \frac{1}{\eta} \mathbb{P}(z_t | x_t = \mu^{(i)}) w_{t-1}^{(i)} \\ &\propto \mathbb{P}(z_t | x_t = \mu^{(i)}) w_{t-1}^{(i)} \end{aligned}$$

where

$$\eta = \sum_i \mathbb{P}(z_t | x_t = \mu^{(i)}) w_{t-1}^{(i)}$$

I.5 Mapping

After updating the weights, the state hypothesis with largest weight is used to fuse the lidar scan into the map.

Let denote the best particle and the transformation is represents as $\mu^{(i^*)} \Leftrightarrow {}_{\{G\}}T_{\{F\}}$ and denote the transformed laser hit points $\{p_k^{(i^*)}\}_{k=1}^K$. Compute the lidar position p_{lidar} by

$$p_{lidar} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} {}_{\{G\}}T_{\{F\}} {}_{\{F\}}T_{\{L\}} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Transform all positions into map index, denote the transformed index of $\{p_k^{(i^*)}\}_{k=1}^K$ as $\{(u_k, v_k)\}_{k=1}^K$, denote the transformed index of lidar position by (u_{lidar}, v_{lidar}).

For each beam of laser ($\forall k = 1, ..., K$), OpenCV's drawContours method is employed to find all map cell that intersects the line originating from (u_{lidar}, v_{lidar})

and ends at (u_k, v_k). These cells, except the laser hit point, are "observed free" and the last cell in the beam, (u_k, v_k) is "observed occupied".

The map log-odds is updated accordingly with the inverse observation model:

$$\begin{aligned}
\lambda_{(u,v),t} &= \log\left(\frac{\mathbb{P}(m_{(u,v)} = 1|z_{0:t}, x_{0:t})}{\mathbb{P}(m_{(u,v)} = -1|z_{0:t}, x_{0:t})}\right) \\
&= \log\left[\frac{p_h(z_t|m_{(u,v)} = 1, x_t)}{p_h(z_t|m_{(u,v)} = -1, x_t)}\right] + \lambda_{(u,v),t-1} \\
&= \underbrace{\log\left[\frac{p_h(m_{(u,v)} = 1|z_t, x_t)}{p_h(m_{(u,v)} = -1|z_t, x_t)}\right]}_{\Delta log\_odds} \\
&\quad + \underbrace{\log\left[\frac{\mathbb{P}(m_{(u,v)} = 1)}{\mathbb{P}(m_{(u,v)} = -1)}\right]}_{prior} + \lambda_{(u,v),t-1} \\
&\text{Assume } \mathbb{P}(m_{(u,v)} = 1) = \mathbb{P}(m_{(u,v)} = -1) \\
&= \Delta log\_odds + \lambda_{(u,v),t-1}
\end{aligned}$$

Assume the lidar have true positive rate / false positive rate of 8.0, we have:

$$\Delta log\_odds_{(u,v)} = \begin{cases} -\log(8) & ,(u,v)\text{observed free} \\ +\log(8) & ,(u,v)\text{observed occupied} \\ 0 & ,\text{otherwise} \end{cases}$$

All observed map cells are updated accordingly.

I.6 Resampling

As the iteration goes, most weights are going to concentrate in few particles. Resampling prevents this waste of computation power by redistribute the particles to represent the same distribution while particles have more even weights.

Define number of effective particles as

$$N_{\text{eff}} = \frac{1}{\sum_i w_i^2}$$

A resampling occurs whenever $N_{\text{eff}} \leq 0.2N$, where $N = 1000$ is the number of particles. Systematic resampling is used to draw particles proportional to its weight w_i:

Algorithm 1 *systematic_resampling*$(\{\mu^{(i)}, w_i\}_{i=1}^N)$

```
 1: selection ← []
 2: draw $u \sim U(0, 1/N)$
 3: $c \leftarrow 0$
 4: $j \leftarrow 0$
 5: for $i = 1, ..., N$ do
 6:     th ← $u + \frac{i-1}{N}$
 7:     while $c <$ th do
 8:         $c \leftarrow c + w_i$
 9:         $j \leftarrow j + 1$
10:     end while
11:     selection ← [selection,$(\mu^{(j)}, 1/N)$]
12: end for
```

I.7 Compromise for Optimization

For the sake of runtime, efficient implementation in Python, and memory consideration, some compromise is made during implementation that differs slightly from the above description:

1. Log-odds increment $\Delta log\_odds_{(u,v)}$, $\pm\log(8)$ is approximated to ± 2 in order to store the log-odds map in int16 format.

2. Map cells observed by multiple laser beams are updated only once. This compromise is made in order to use the drawContours function of OpenCV. This mainly affects cells close to the origin of laser beam.

II Texture Mapping

After SLAM, a occupancy map and a history of best estimation of robot pose is passed to this texture mapping section. Based on these two information and images from the stereo camera, a colored map is created. Texture mapping consists of mainly three steps: Depth computation, Projection to color map, and occupancy masking.

Depth Computation From the left and right stereo image, OpenCV's StereoSGBM is used to perform block matching and the disparity $d = u_L - u_R$ is computed for matched blocks. The depth is compted by $D = \frac{s_u \text{base\_line}}{d}$. Only pixels with depth $\in [0, 70\text{m}]$ is kept. Then, the projected 3D point in the Optical frame(Denoted $\{S\}$) is computed as:

$$p = \frac{s_u \cdot \text{base\_line}}{d} K^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix}$$

Projection to color map Appending 1 to p to transform is into homogeneous form $p_{\{S\}}$. Transform is to global frame by:

$$p_{\{G\}} = {}_{\{G\}}T_{\{F\}}{}_{\{V\}}T_{\{S\}}p_{\{S\}}$$

Where ${}_{\{G\}}T_{\{V\}}$ is obtained from the history of robot pose produced by SLAM.

Next, thresholding is applied to the Z axis and only pixels with height $-0.5m \sim 1.5m$ are kept.

Then, their index on the color map is computed and the colored map is updated to color of *one of* the pixels that projected to that cell.

Occupancy Masking Finally, the color map is masked by the occupancy map to produce the final texture mapped map. Precisely, map cells with log-odds < 0 are colored according to the color map. The default color is white.

III List of Static Variables

Variable	Value
${}_{\{F\}}T_{\{L\}}$	$\begin{bmatrix} 0.0013 & 0.7961 & 0.6052 & 1.1699 \\ 1.0000 & -0.0004 & -0.0016 & 0.0223 \\ -0.0010 & 0.6052 & -0.7961 & 0.9842 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
${}_{\{F\}}T_{\{S\}}$	$\begin{bmatrix} -0.0068 & -0.0153 & 0.9999 & 1.9774 \\ -1.0000 & 0.0003 & -0.0068 & 0.2824 \\ -0.0002 & -0.9999 & -0.0153 & 0.8041 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
D_L	0.622806m
D_R	0.623479m
CPR	4096
σ_v	0.50 m/s
σ_ω	0.04 rad/s
map_res	0.2m
N	1000
base_line	475.143600050775 mm
K	$\begin{bmatrix} 816.9038 & 0.5051 & 608.5073 \\ 0.0000 & 811.5680 & 263.4760 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
s_u	816.9038

IV RESULTS

I Overview

Overall, the SLAM and Texture mapping successfully mapped and colored available portion of the map.

* Videos of occupancy and texture mapping, full resolution of both the occupancy and colored map are included in the submission. Link to videos: `https://drive.google.com/drive/folders/1bLR7G0qYp54NdamVCO04dIIwr_xi_17Z`, Images are located at "data_output/images"

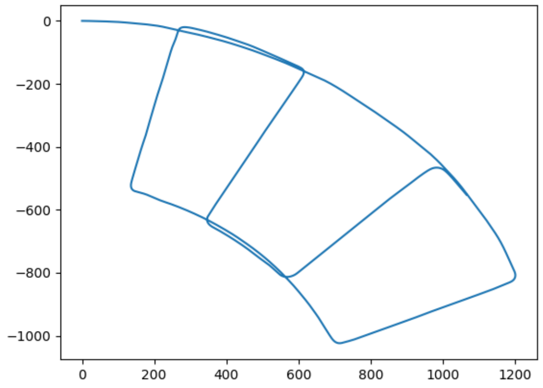

Figure 1: Vehicle Path (FOG)

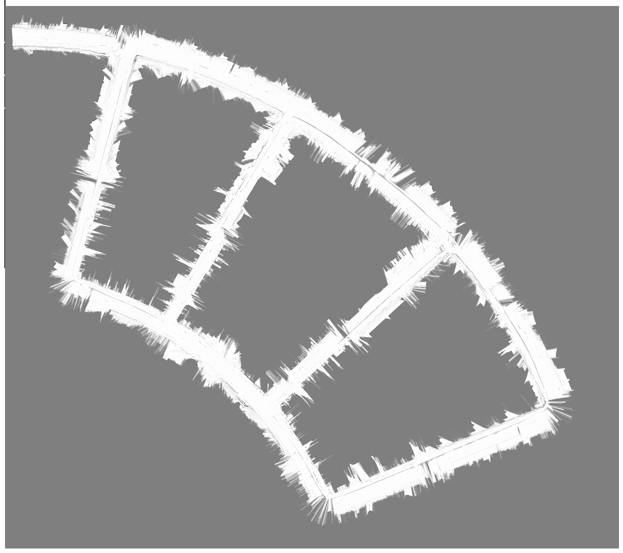

Figure 2: Final occupancy map

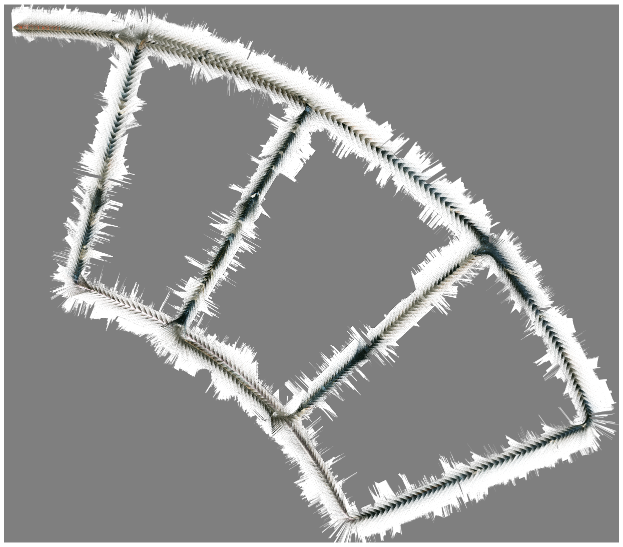

Figure 3: Final textured map

II SLAM

The SLAM portion works well for most portion of local mapping.

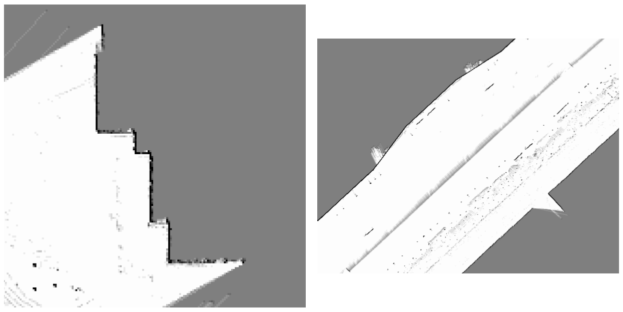

Figure 4: Local mapping well

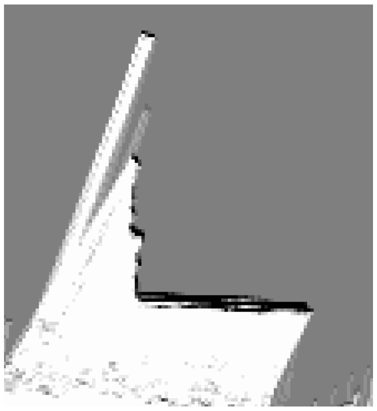

Figure 5: Local mapping doubled

From the local mappings, I can clearly see that the particle filter and the update step in function correctly. The noise assigned on the linear velocity have std of $0.5m/s$, The vehicle would drift far beyond several grid cells if the update step were not functioning.

Still, there are evidence of mismatch after a large loop.

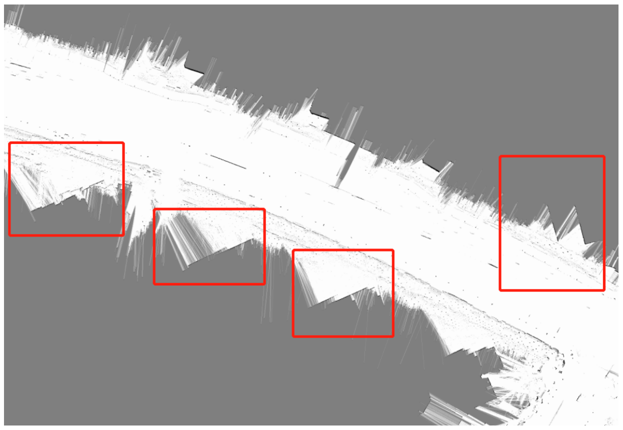

Figure 6: Map mismatch (boxed)

This shows that particle filter and the use of one global map is not appropriate for detecting and adjusting according to loop closure. For the sake of future improvement, a alternative representation of the map should be employed and a dedicated loop closure module should be added.

III Texture Mapping

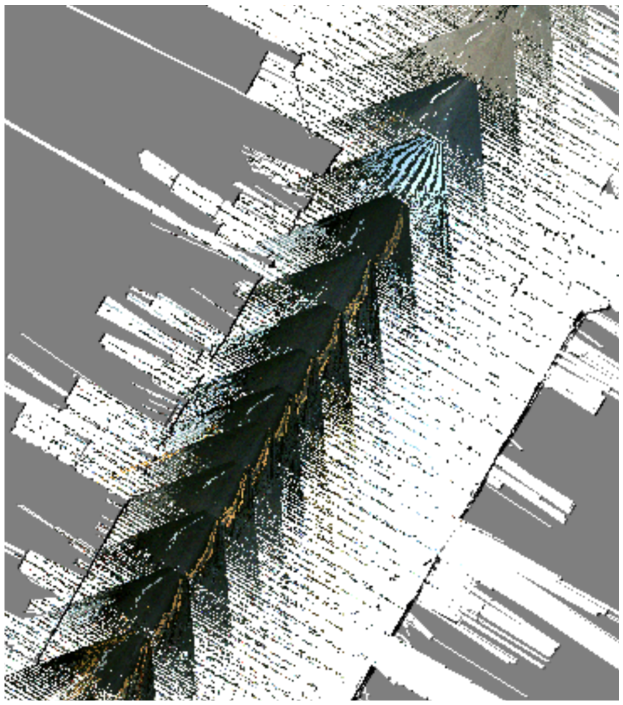

Figure 7: Local texture mapping

Globally, texture mapping works well. I can Identify bright and shaded regions on the ground from the textured map, and the lanes. Locally, texture mapping is not very accurate. From the above figure, we can see that the lanes all bent inwards when it is close to the camera. This is due to non-perfect block matching between left and right stereo images. The lanes are mostly a solid color without much variation, which is hard to match accurately. Also, limited by the 1fps framerate of the image data, most of the occupancy map cannot be assigned a color.

IV Runtime

The above result is obtained with $N = 1000$ particles with the perturbation grid making 3 copies of every particle, using all available data. Under this setting, the SLAM portion (not including visualization and texture mapping) runs at $\sim 10\%$ speed compared to real time on my computer (Dell Precision 5530, i9 8950HK). To SLAM at realtime, only $80 \sim 100$ particles can be used.