Particle filter based SLAM and texture mapping

Yusi Chen

October 9, 2021

Contents

1 Introduction

This project aims to implement simultaneous localization and mapping (SLAM) using odometry, inertial, 2-D laser range and RGBD measurements from a differential-drive robot. In addition to that, RGBD information was also provided to perform texture mapping of the map (the ground floor in this case).
SLAM was tackled using particle filter, which involves two main steps: prediction and update.
Texture mapping was performed using a RGBD camera model and coordinate transformation.

2 Problem Formulation

2.1 SLAM and Bayesian filter

SLAM is such an estimation problem that given external output $u_{0:T}$ and observation $z_{0:T}$, we want to estimate the best current robot state x_T and external map m_T. Under Markov assumption, we could expression the joint pdf as Equation **??** where p_h and p_f are observation model and motion model respectively. To maximize the data likelihood given parameters (MLE), we aim to solve the optimization problem of $\max_{x_{0:T},m} \log p(x_{0:T}, m, z_{0:T}, u_{0:T-1})$. According to Equation 1, we want to find $x_{0:T}$ and m that maximize $p_h(z_t|x_t, m)$ and $p_f(x_t|x_{t-1}, u_{t-1})$ at every step t under a given observation and motion model.

$$\begin{aligned} p(x_{0:T}, m, z_{0:T}, u_{0:T-1}) &= p_{0|0}(x_0, m) \prod_{t=0}^{T} p_h(z_t|x_t, m) \prod_{t=1}^{T} p_f(x_t|x_{t-1}, u_{t-1}) \\ \log p(x_{0:T}, m, z_{0:T}, u_{0:T-1}) &= \sum_{t=0}^{T} \log p_h(z_t|x_t, m) + \sum_{t=1}^{T} \log p_f(x_t|x_{t-1}, u_{t-1}) \end{aligned} \tag{1}$$

For a Bayesian filter (particle filter in this case), we keep track of $p_{t|t}(x_t) := p(x_{t+1}|z_{0:t}, u_{0,t-1})$ so that we could find the best x_t and m. Under Markov assumption and Bayes rule, we could keep track of it by Equation **??**. In addition, this equation could be decomposed into prediction (the integral term) and update (the whole term). So we could compute $p_{t+1|t}(x)$ and $p_{t+1|t+1}(x)$ iteratively to do that.

$$\begin{aligned} p_{t+1|t+1}(x_{t+1}) &:= p(x_{t+1}|z_{0:t+1}, u_{0:t}) \\ &= \frac{p_h(z_{t+1}|x_{t+1})}{p(z_{t+1}|z_{0:t}, u_{0:t})} \int p_f(x_{t+1}|x_t, u_t) p_{t|t}(x_t) dx_t \\ \text{Prediction} : p_{t+1|t}(x) &:= \int p_f(x_{t+1}|x_t, u_t) p_{t|t}(x_t) dx_t \\ \text{Update} : p_{t+1|t+1}(x) &:= \frac{p_h(z_{t+1}|x_{t+1})}{p(z_{t+1}|z_{0:t}, u_{0:t})} p_{t+1|t}(x) \end{aligned} \tag{2}$$

In this project, we used particle filter where the distributions are represented by the summation of delta functions. Let's say we have N particles with weight $\alpha^{(k)}$ and delta function

$\delta(\mu^{(k)})$.

$$\begin{aligned}
p_{t+1|t}(x_{t+1}) &= \int p_f(x_{t+1}|x_t,u_t)p_{t|t}(d)ds = \int p_f \sum_{k=1}^{N} \alpha_{t|t}^{(k)}\delta(s,\mu_{t|t}^{(k)})ds \\
&= \sum_{k=1}^{N} \alpha_{t|t}^{(k)} p_f(x_{t+1}|\mu_{t|t}^{(k)},u_t), \text{(Bootstrap approximation)} = \sum_{k=1}^{N} \alpha_{t+1|t}^{(k)}\delta(x_{t+1},\mu_{t+1|t}^{(k)}) \\
p_{t+1|t+1}(x_{t+1}) &= \frac{p(z_{t+1}|x_{t+1})p(x_{t+1})}{p(z_{t+1})} = \frac{p_h(z_{t+1}|x_{t+1})\sum_{k=1}^{N}\alpha_{t+1|t}^{(k)}\delta(x_{t+1},\mu_{t+1|t}^{(k)})}{\int p_h(z_{t+1}|s)\sum_{k=1}^{N}\alpha_{t+1|t}^{(k)}\delta(s,\mu_{t+1|t}^{(k)})ds} \\
&= \sum_{k=1}^{N} \frac{p_h(z_{t+1}|\mu_{t+1|t}^{(k)})\alpha_{t+1|t}^{(k)}}{\sum_{k=1}^{N} p_h(z_{t+1}|\mu_{t+1|t}^{(k)})\alpha_{t+1|t}^{(k)}}\delta(x_{t+1},\mu_{t+1|t}^{(k)})
\end{aligned} \tag{3}$$

2.2 Texture mapping

Texture mapping is to obtain a more detailed representation of m given information captured by an RGBD camera located on the robot. The key is to transform projected pixel coordinates (u, v) into world frame coordinates (X_w, Y_w, Z_w) based on rotation model and camera intrinsic parameters.

For a RGBD camera used in this project, it has two rigidly-connected horizontal cameras. For a point $m = (X_w, Y_w, Z_w)$ with pixel coordinates (u_L, v_L) and (u_R, v_R) for left camera and right camera, the transformation could be described as Equation **??** according to a stereo camera model where (X_o, Y_o, Z_o) is the coordinates in optical plane; R_{oc} is the camera orientation in optical plane, R_{wc} is camera in world frame and p is the camera pose in world frame.

$$\begin{aligned}
\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} &= \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ 0 & 0 & 0 & fs_u b \end{bmatrix} \frac{1}{Z_o} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} \\
\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} &= R_{oc}R_{wc}^T(m-p)
\end{aligned} \tag{4}$$

3 Technical Approach

3.1 Setup

3.1.1 Time stamp alignment

Since we have multiple measurements, including encoder (40Hz), lidar (40Hz), IMU (100Hz) and camera (20Hz), it's important to align them and identify the right paired measurements. In this project, I used encoder measurements as the standard time train and all dt equals to encoder time interval. For lidar and camera, I adopted the closest measurements while for IMU data, I averaged the measurements between two encoder time stamps.

3.1.2 Body frame

For convenience, I chose the lidar position as my robot center. x axis is to the front, y axis to the left and z axis towards the sky. Based on lidar setup, left turn is related to positive yaw angle change. For SLAM problem, which doesn't involve z axis, this center is almost equivalent to front wheel center. For texture mapping, the disparity camera pose relative to robot is $p_{rc} = (0.09, 0.005, -0.154)$ and angles (roll, pitch, yaw) relative to robot is $(0, 0.36, 0.021)$rad. World frame origin was set at the robot position when $t = 0$.

3.1.3 Grid map

To simplify map representation, I used an occupancy grid map to store the map. World frame origin is at the center of the grid. The grid resolution is 0.1 meters/grid and the grid ranges from (-55,55) meters both x and y axis. $m_i = 1$ means the grid is occupied and $m_i = 0$ means the grid is free.

3.2 Differential-drive robot motion prediction

The movement of this robot was modeled as differential-drive wheels. Let the robot state $\mu_t = (x_t, y_t, \theta_t)$ and external force $u_t = (v_t, w_t)$ then the robot movement could be described by Equation **??**.

$$\begin{aligned}
\theta_{t+1} &= \theta_t + w_t dt \\
x_{t+1} &= x_t + v_t \int_t^{t+1} \sin(\theta_s) ds = x_t + \frac{v}{w}(\sin(\theta_{t+1}) - \sin(\theta_t)) \\
y_{t+1} &= y_t + v_t \int_t^{t+1} \sin(\theta_s) ds = y_t - \frac{v}{w}(\cos(\theta_{t+1}) - \cos(\theta_t))
\end{aligned} \tag{5}$$

For particle filter, $p_{t+1|t}(x)$ is equivalent to apply motion model to each particle and retain their weight α.

3.3 Map correlation-based update and stratified resampling

In this project, I used laser correlation model as the observation model. In Equation **??**, m is the current map, y_k is the map from particle k. Map is inherited from the best particle at the previous step. In practice, small vibrations were added to y_k to counter observation noise.

$$\begin{aligned}
\mathbf{corr}(y_k, m) &:= \sum_i \mathbf{1}(m_i = y_i) \\
p_h(z_t|\mu_{t,k}, m) &= \exp(\mathbf{corr}(y_k, m) - \max_k \mathbf{corr}(y_k, m)) \in (0, 1) \\
\alpha_{t+1} &= \alpha_t p_h(z_t|\mu_{t,k}, m) \quad \text{with normalization}
\end{aligned} \tag{6}$$

Because we are using a grid map, in practice, an accumulated log odds ratio map $\lambda(m_i|z_{0:t}, x_{0:t})$ was kept during iterations. It was updated according Equation **??**

$$\begin{aligned}\lambda(m_i|z_{0:t}, x_{0:t}) &:= \log o(m_i|z_{0:t}, x_{0:t}) \\ &= \lambda(m_i|z_{0:t-1}, x_{0:t-1}) + \log g_h(z_t|m_i, x_t) \\ g_h(1|m_i, x_t) &= \frac{p_h(z_t = 1|m_i = 1, x_t)}{p_h(z_t = 1|m_i = 0, x_t)} = 4, \quad g_h(0|m_i, x_t) \\ &= \frac{1}{4}\end{aligned} \tag{7}$$

A grid map m could be retrieved by sampling from a binomial distribution where $p_i = 1 - \frac{1}{1+\exp(\lambda_{i,t})}$. To avoid particle depletion, which is defined as $N_{eff} < N/2$ where $N_{eff} = \frac{1}{\sum \alpha_k^2}$, stratified importance was adopted.

3.4 Camera reverse projection

In this problem, we are given u_L, v_L, d and want to find its world frame pixel m. For a pixel $z = [u_L, v_L]$ in the disparity camera, m could be calculated as Equation **??** where Z_o is the depth which could be calculated from the equations given and $R_{wc} = R_{wr}R_{rc}$ and $p_{wc} = p_{wr} + p_{rc}$. R_{rc} and p_{rc} could be found in the setup session and R_{wr} and p_{wr} are two varying matrices which corresponding to the weighted robot states from all particles. The RGB color of point z could also be calculated from the equations given.

$$m = R_{wc}R_{oc}^T(K^{-1}zZ_o) + p_{wc} \tag{8}$$

Since we are only coloring the ground, I applied height threshold = 0.3 meters to m. Only low-height pixels were added to the grid map. And all pixels were averaged to form the color of certain grid.

4 Results and Discussion

4.1 Single particle prediction

In this part, I only used one particle to do prediction-only localization. That is to say, the robot trace totally depends on encoder and IMU readings. Results are shown in Fig. 4.1, 4.1 and 4.1.

4.2 Update results and log-odds ratio map

For the whole process of update, please refer to the video attached in the programming assignment. Here, I'm only showing the final results of SLAM.
First of all, the traces are more accurate in the single particle prediction results. For example, in dataset 20, from the video, we know that the robot came back to its starting place at the end. In Figure 4.1, the starting and ending point is about 5 meters apart while in Figure 4.2, the starting point and ending point is located at the same place.

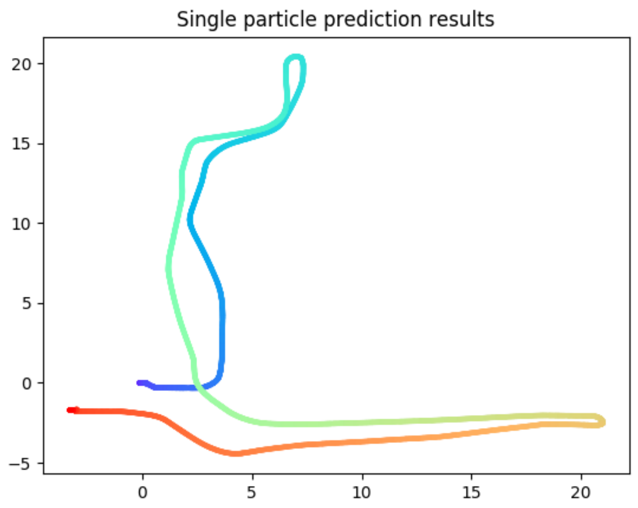

Figure 1: Single particle prediction-only trace for dataset20: Showing the physcal location of the particle and the unit is meter. Time is color-coded in the rainbow color, from blue to red

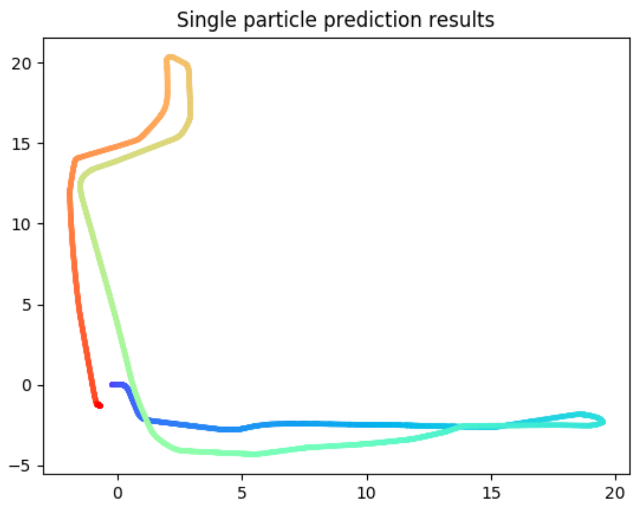

Figure 2: Single particle prediction-only trace for dataset21: Showing the physcal location of the particle and the unit is meter. Time is color-coded in the rainbow color, from blue to red

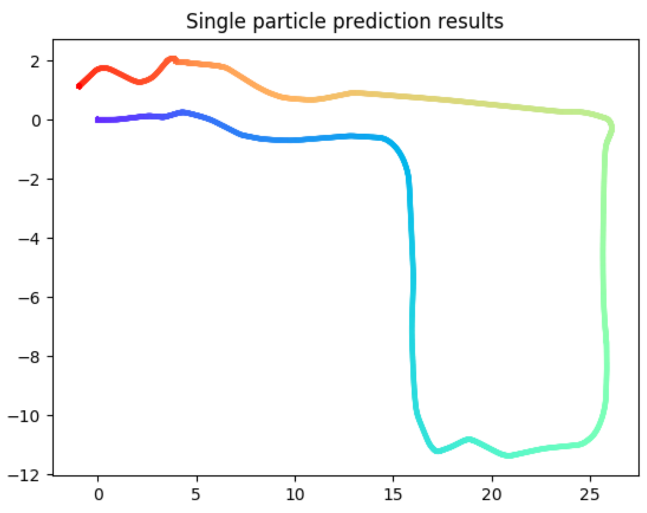

Figure 3: Single particle prediction-only trace for dataset23: Showing the physcal location of the particle and the unit is meter. Time is color-coded in the rainbow color, from blue to red

On the other hand, the SLAM results are not perfect because there are obviously duplicated structure in the accumulated odds ratio, which is due to inaccuracy of localization. For example, the corner part in dataset 21 (Figure 4.2) and the vertical corridor in dataset 23 (Figure 4.2). I tried to tune the noise and vibration level in the model, but I haven't found perfect parameters for dataset 21 and dataset 23.

4.3 Texture mapping

For texture mapping, the trace is slightly distorted but we could still identify some features or colors of the environment. In the middle panel of Figure 4.3, right part is gray stairs; upper left is a white floored room with many colored tools. In the middle panel of Figure 4.3, the right part is gray stairs; there are two slopes and some dark areas in the left vertical corridor.

4.4 Noise and particle dispersion

It is very hard to choose the appropriate motion noise so that the filter could balance between flexibility and randomness. I investigated the different scenario of noise and the results are shown below. Let's say both linear and angular velocity have Gaussian noise and variance is σ_v and σ_w.

- Stationary v.s. non-stationary noise: for stationary noise, σ is invariant throughout the time while for non-stationary noise, σ is a time-varying variable. Fig 4.4 shows the results of dataset20 given stationary noise (about 10% for linear velocity and about 3% for angular velocity). Fig4.4 shows the results of the same dataset given almost

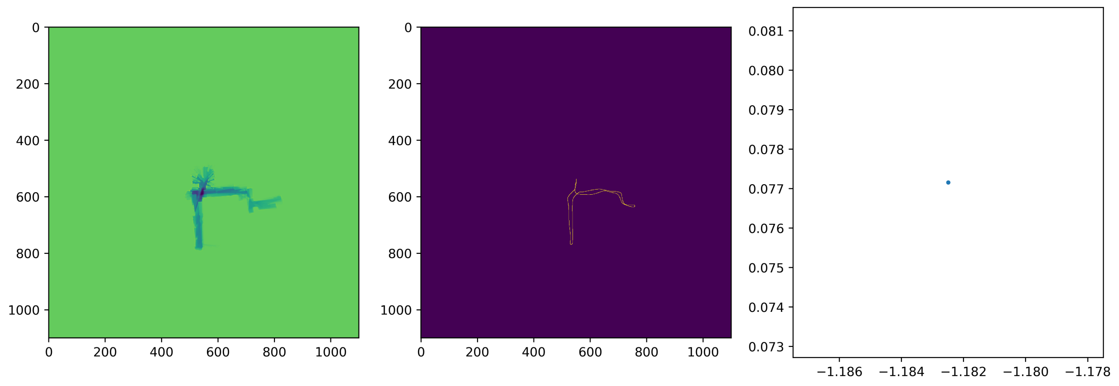

Figure 4: SLAM results for dataset20 (N=10). left panel: accumulated log-odds ratio; middle panel: estimated robot trace in the map grid; right panel: particle physical position in world frame

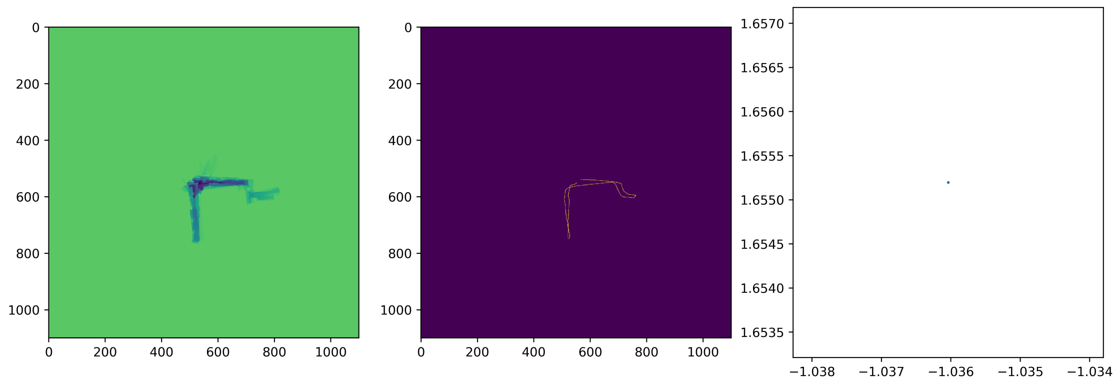

Figure 5: SLAM results for dataset21 (N=50). left panel: accumulated log-odds ratio; middle panel: estimated robot trace in the map grid; right panel: particle physical position in world frame

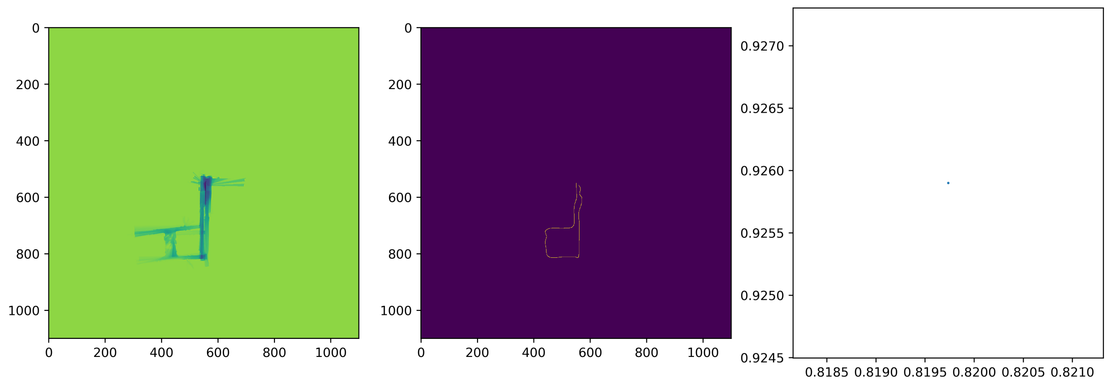

Figure 6: SLAM results for dataset23 (N=100). left panel: accumulated log-odds ratio; middle panel: estimated robot trace in the map grid; right panel: particle physical position in world frame

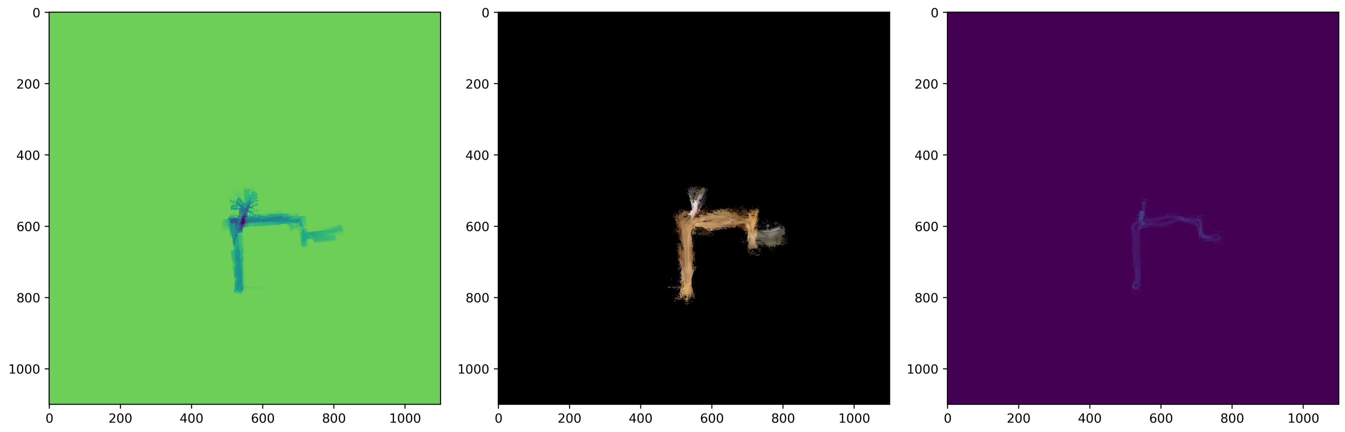

Figure 7: Texture mapping results for dataset20 (based on Figure 4.2 results). left panel: accumulated log-odds ratio; middle panel: texture mapping grid; right panel: accumulated number of pixels in each grid

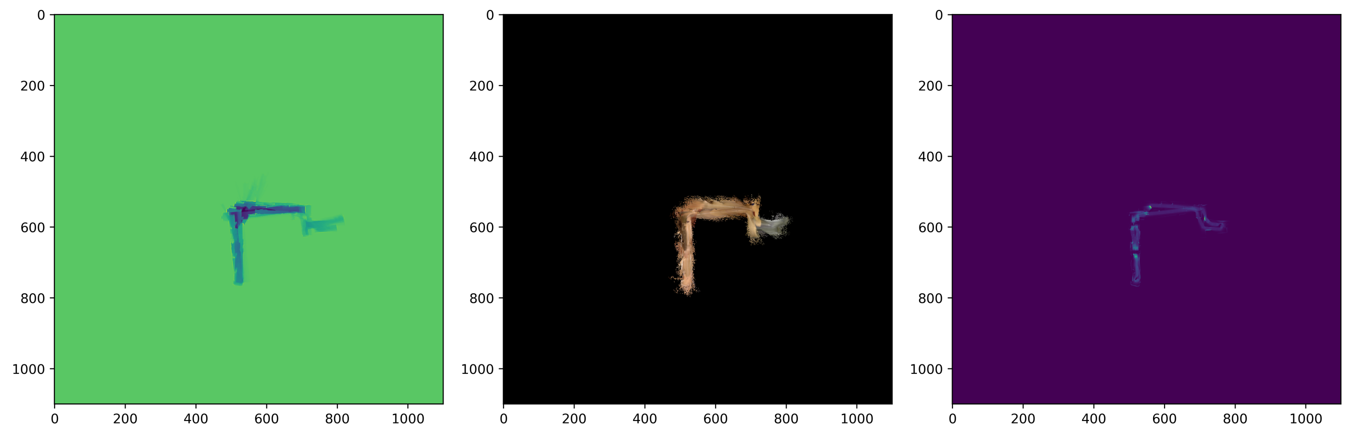

Figure 8: Texture mapping results for dataset21 (based on Figure 4.2 results). left panel: accumulated log-odds ratio; middle panel: texture mapping grid; right panel: accumulated number of pixels in each grid

similar percentage of non-stationary noise. Clearly, non-stationary noise is better in this case.

- Big Noise: big noise could be beneficial because it successfully handle the slope problem in dataset21 Comparing Figure 4.4 (big noise) and Figure 4.2, the big noise version has a better estimation of the lower vertical corridor but it failed to recover the right horrizontal corridor.

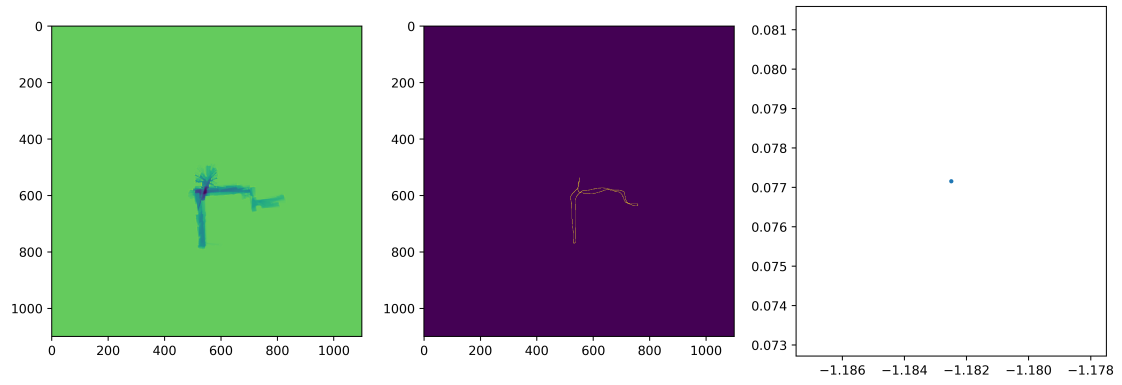

Figure 9: SLAM results for dataset20 (N=10)under non-stationary noise. left panel: accumulated log-odds ratio; middle panel: estimated robot trace in the map grid; right panel: particle physical position in world frame

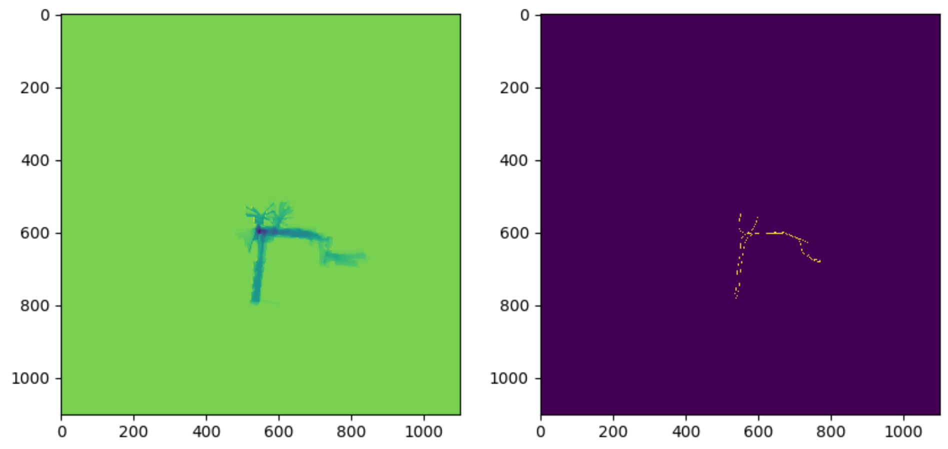

Figure 10: SLAM results for dataset20 (N=10) under stationary noise. left panel: accumulated log-odds ratio; right panel: estimated robot trace in the map grid

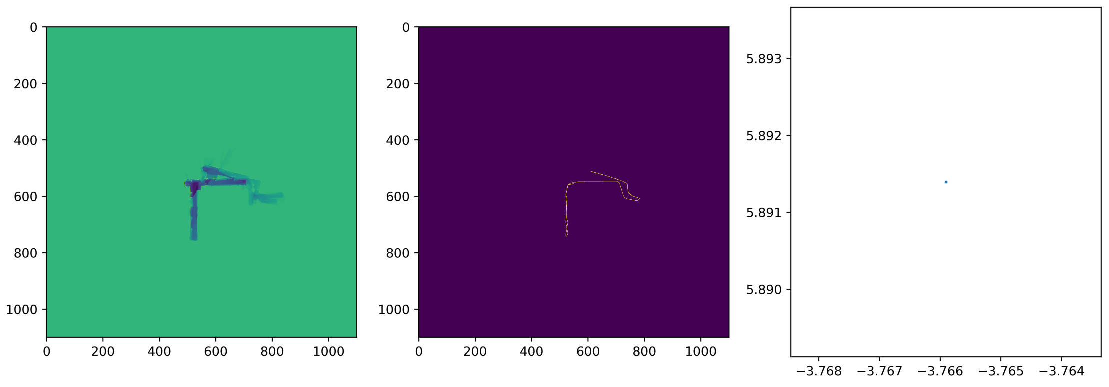

Figure 11: SLAM results for dataset21 (N=50)under big non-stationary noise (variance is about 20% for linear velocity and 10% for angular velocity). left panel: accumulated log-odds ratio; middle panel: estimated robot trace in the map grid; right panel: particle physical position in world frame