Simultaneous Localization and Mapping using Particle Filter

Hongbo Qian
Department of Electrical and Computer Engineering
University of California, San Diego
hoqian@ucsd.edu

***Abstract*—This project aims to use data captured from a variety of sensors including Encoder, FOG and LiDAR on the vehicle to map and localize its surrounding environment. The method to achieve this is Simultaneous Localization and Mapping(SLAM), which is the most popular method in this topic. This enables the vehicle to to perceive the external environment, which can be helpful for autonomous driving.**

***Index Terms*—SLAM, Particle Filter**

I. Introduction

In recent years, the field of autonomous driving has been very popular, and cars that can automatically drive on the road have been produced on the market. Although people can walk freely, it is not easy to give vehicles the ability to walk on the road. Not only do we need to provide the vehicle with the ability to perceive the external environment, such as providing various sensors, we also need to use the data returned by the sensor to construct a map, that is, let the vehicle know where it is. Most of the sensors required for autonomous driving are used in this project, including Encoder, FOG and LiDAR and using the most popular SLAM algorithm.

This project is divided into four parts in total. The first is "prediction," which predicts where the vehicle will be in the future. The second is "Particle Filter Update", which updates the location on the map based on the LiDAR data. The third is to output the final result to the map and display it in the form of an image. The fourth is "Adding Texture" which adds color to the final result.

II. Problem Formulation

In this part, the material is similar to the lecture slides. The occupancy grid map is one of the simplest and most widely used representations. The environment is divided into a regular grid with n cells. The occupancy grid is unknown and needs to be estimated given the robot trajectory and a sequence of observations. Since the map is unknown and the measurements are uncertain, it maintains a pmf $p(m|z_{0:t}, x_{0:t})$.

Estimating the pmf of m_i conditioned on $z_{0:t}$ and $x_{0:t}$ is equivalent to accumulating the log-odds ratio $\Delta\lambda_{i,t}$ of the inverse measurement model:

$$\Delta\lambda_{i,t} = \Delta\lambda_{i,t-1} + (\Delta\lambda_{i,t} - \lambda_{i,0}) \tag{1}$$

Assuming that z_t indicates whether m_i is occupied or not, the log-odds ration $\Delta\lambda_{i,t}$ of the inverse measurement model specifies the measurement trust.

$$\Delta\lambda_{i,t} = log\frac{p(m_i = 1|z_t, x_t)}{p(m_i = -1|z_t, x_t)} \tag{2}$$

$$\Delta\lambda_{i,t} = \begin{cases} +log4 & \text{z}_t \text{ indicates } m_i \text{ is occupied} \\ -log4 & \text{z}_t \text{ indicates } m_i \text{ is free} \end{cases} \tag{3}$$

Given a new laser scan z_{t+1}, transform it to the world frame using the robot pose x_{t+1}. Determine the cells that the lidar beams pass through, for example, using Bresenham's line rasterization algorithm. For each observed cell, decrease the log-odds if it was observed free or increase the log-odds if the cell was observed occupied:

$$\lambda_{i,t+1} = \lambda_{i,t} \pm log4 \tag{4}$$

Constrain $\lambda_{min} \leq \lambda_{i,t} \leq \lambda_{max}$ to avoid overconfident estimation. The map pmf can be recovered from the log-odds $\lambda_{i,t}$ via the logistic sigmoid function:

$$\gamma_{i,t} = \sigma(\lambda_{i,t}) = \frac{exp(\lambda_{i,t})}{1 + exp(\lambda_{i,t})} \tag{5}$$

In the update step, the particle poses will not be changed, but we need to consider the weights of each particle.

$$\mu^{(k)}_{t+1|t+1} = \mu^{(k)}_{t+1|t} \tag{6}$$

$$\alpha^{(k)}_{t+1|t+1} \propto p_h(z_{t+1}|\mu^{(k)}_{t+1|t}, m)\alpha^{(k)}_{t+1|t} \tag{7}$$

So we need to use the data from LiDAR to get the observation model $p_h(z|x, m)$. Here we need to introduce the Laser Correlation Model, which is a model for a laser scan z obtained from sensor pose x in occupancy grid m based on correlation between z and m.

Transform the scan z_{t+1} to the world frame using $\mu^{(k)}_{t+1|t}$ and find all cells $y^{(k)}_{t+1}$ in the grid corresponding to the scan. Update the particle weights using the laser correlation model:

$$p_h(z_{t+1}|\mu^{(k)}_{t+1|t}, m) \propto corr(y^{(k)}_{t+1|t}, m) \tag{8}$$

The laser correlation model sets the likelihood of a laser scan z proportional to the correlation between the scan's world-frame projection $y = r(z, x)$ via the robot pose x and the occupancy grid m.

$$p_h(z|x,m) \propto corr(r(z,x),m) \tag{9}$$

Finally, we need to compute the scan-grid correlation. Transform the scan z from the laser frame to the world frame using the robot pose x. Find all grid coordinates y that correspond to the scan. Let $y = r(z,x)$ be the transformation from a lidar scan z to grid cell indices y. Definite a similarity function $corr(r(z,x),m)$ between the transformed and discretized scan y and the occupancy grid m:

$$corr(y,m) = \sum 1\{y_i = m_i\} \tag{10}$$

III. Technical Approach

In this part, we mainly talk about how to solve the problem in detail.

A. Data and Transformation

To start the project, we first need to load all of the data from the sensors including encoder, FOG, and LiDAR. We also need to load the parameters such as encoder resolution, encoder wheel diameters, lidar to vehicle frame rotation matrices, stereo camera to vehicle frame rotation matrices, etc. The transformation matrix is shown below.

$$T = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \tag{11}$$

We always need to transform the data from the sensor frame to the world frame. We need to finish two steps. The first one is transforming from sensor frame to body frame. And then transform from body frame to world frame.

$$_WT_L = _W\ T_V\ _VT_L \tag{12}$$

Another problem for this project is that the timestamps for the sensors is not consistent. To solve this problem, I need to judge the timestamp of each sensor to achieve synchronization of sensor data. The encoder data has 116,048 timestamps data, the lidar data has 115,865 timestamps, and the FOG data has 1,160,508 timestamps. We can find that the number of FOG data is ten times than the number of encoder data and lidar data. In order to solve the problem that the timestamp is not synchronized, my solution strategy is to compare the timestamps of 3 sensors, mainly the timestamp of the FOG, and if its timestamp is larger than the other two, use their data to correct it.

B. Prediction

We use the motion model of the differentiate drive model to the vehicle. Where x is the pose of each particle and ω_t is the angular velocity given by FOG sensor.

$$x_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = x_t + \tau \begin{bmatrix} v_t cos(\theta_t) \\ v_t sin(\theta_t) \\ \omega_t \end{bmatrix} \tag{13}$$

We get v_t from the encoder sensor. The equation for the encoder sensor is shown below.

$$meters\ per\ tick = \frac{\pi(wheel\ diameter)}{ticks\ per\ revolution} \tag{14}$$

In our project scenario, we should use the equation below. We compute this for both left wheel and right wheel. Use the mean velocity of these two wheels as the final velocity.

$$\tau v = \frac{\pi dz}{4096} \tag{15}$$

As for the variable θ_t, it represents the yaw angle. At first, we set it as 0. When we update our pose data, its value becomes the third value of pose.

However, in practice, the data will carry a certain amount of noise, in this project, we used the data without noise to predict the trajectory of the movement, but also tried to add noise, using the method of particle filter, the map was estimated.

C. Update

In the update step, the main goal is to compute the α, which represents the weight of each particle. We achieve this using the laser correlation model. We talked about this in the previous part.In order to carry out this step of the particle filter. I first looped through all the FOG and encoder data, predicting the pose information of vehicle at every timestamp. And use the map correlation to get the weights, picking the highest weighted particle as the final pose of the vehicle.

We use the lidar data for each particle with a surrounding 9x9 grid around it. This means that for each particle, we move it in a 9x9 grid around its center, which determines how closely a new lidar scan end points allign with previous scans. We use the pose of the highest weight particle set as the vehicle pose. We use Bresenham2D algorithm to calculate which cells are corresponding to the lidar scan. Plot them on the map if they have a higher weights.

Every 1000 cycles, the drawn map is saved as a picture. The specific results of it, I will also show in the next part.

D. Texture Mapping

Actually, I didn't do texture mapping in this project. The main idea of this is to use stereo camera data with stereo camera model. This can be achieved by using stereo images in order to find the disparity between the two points of view.

$$z = u_L - u_R = \frac{1}{z} f s_u b \tag{16}$$

f is the given focal length, and b is the offset of the cameras. Once we have obtained the 3D coordinates of each pixel in the optical frame, we can transform it into the world frame by the parameters. By applying the same way of this measurement as we did for the lidar, we can get RGB values for the same coordinates as we did for lidar. And we also need to combine the RGB values with the values in the map. Finally, we can get a colourful map image.

IV. Result

A. Mapping

Try mapping using the first LiDAR scan and display the map to make sure the transformations are correct. The results are shown below. Since the results of the occupancy grid map are very unclear, if you zoom in the pdf file, you can still observe the data in the upper left corner.

B. Predict only particle filter

Implement a prediction-only particle filter at first. In other words, use the encoders and the FOG data to compute the linear and angular velocities. Estimate the robot trajectory via the differential drive motion model.

C. 30 particle filter

The vehicle trajectory in the result looks smooth, and the lidar scans drawn seem reasonable. However, it can be found that there is a lot of noise at the edges of map trajectory. This should be normal, but how much noise should be generated is a matter of further study. I think the reasons why its edges aren't smooth may be the following. First of all, the external environment is not very flat, for example, there may be some obstacles on the path. Second, we added some noise to the data, the trajectory looked correct, and the noise affected the detected profile. Perhaps it also has something to do with the number of examples we use, but in my attempts, I have found that the greater the number of examples, the more obvious the unsmooths that may result.

V. Discussion

I really spent a lot of time on this project. Mainly SLAM-related theories were very foreign to me, which made it very difficult to write code. I think the bigger problem is that the calculations are too slow, and it can take several hours to use the particle filter. This will lead to no help for actual production, what we may need is to be able to display data in real time, so that vehicle can obtain the current location information. If the speed is so slow, it cannot be applied to the actual situation. I reflected on what I had done, and I think the point that can be modified is that the number of updates can be reduced, no longer after each preview but after a certain number of previews, which will help to increase the speed of calculations and reduce the calculation time. But I don't think this change is going to work for real-time data, and it needs to be further modified to improve the code and algorithms.

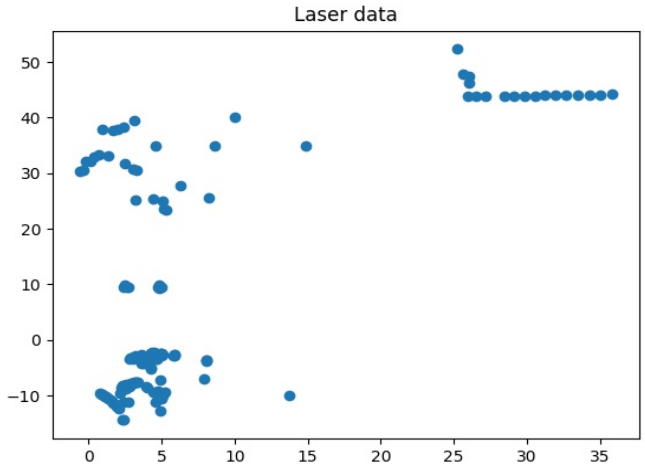

Fig. 1. laser data

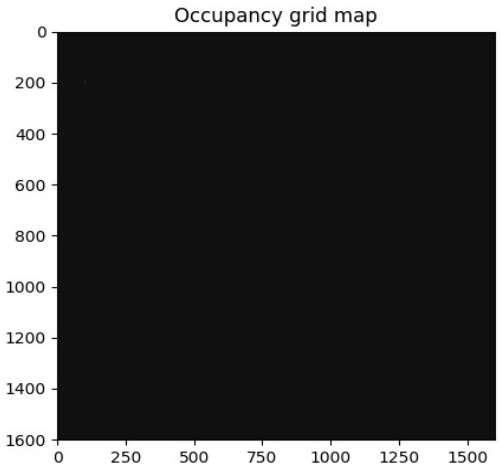

Fig. 2. occupancy grid map

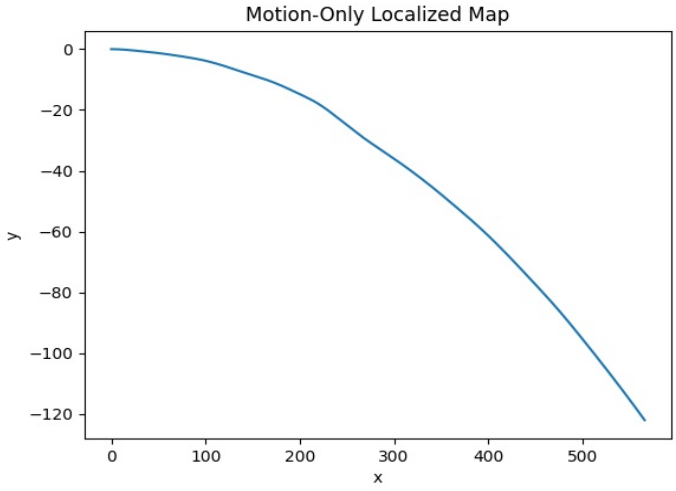

Fig. 3. motion only localized map 2

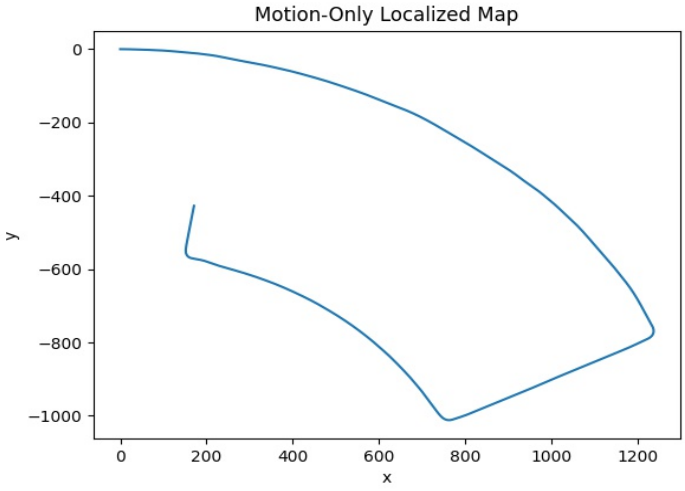

Fig. 5. motion only localized map 4

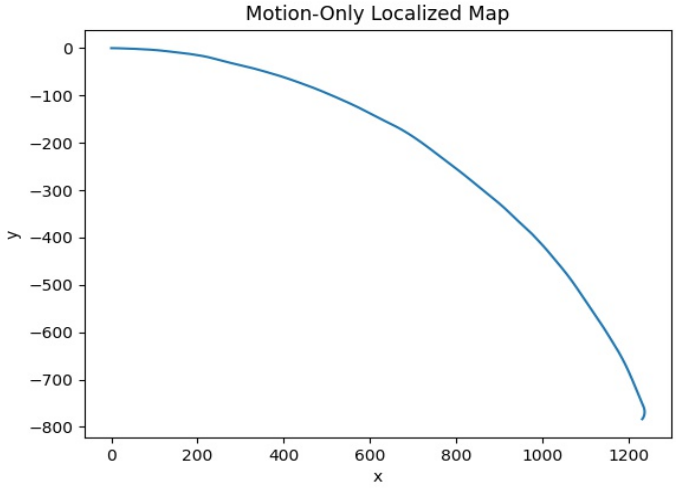

Fig. 4. motion only localized map 3

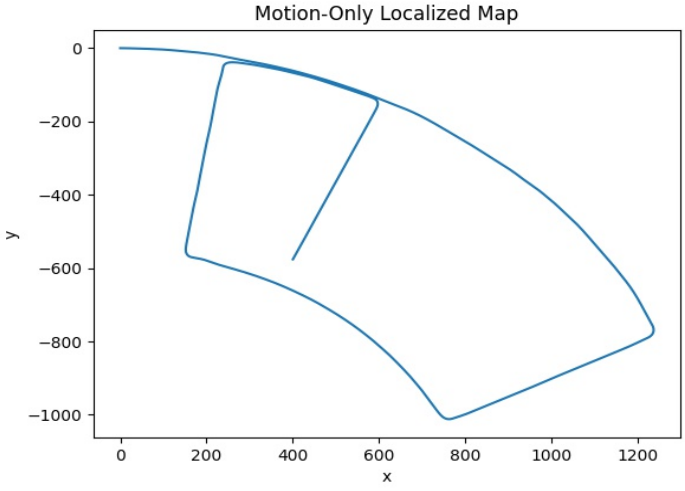

Fig. 6. motion only localized map 5

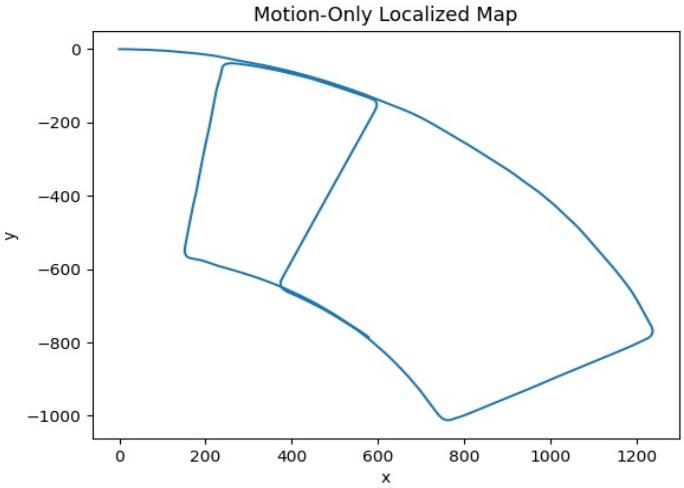

Fig. 7. motion only localized map 6

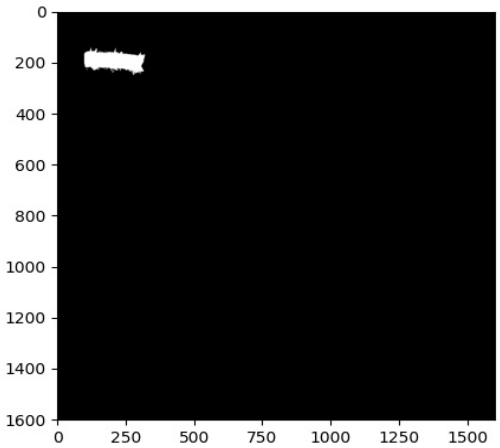

Fig. 9. test 150000

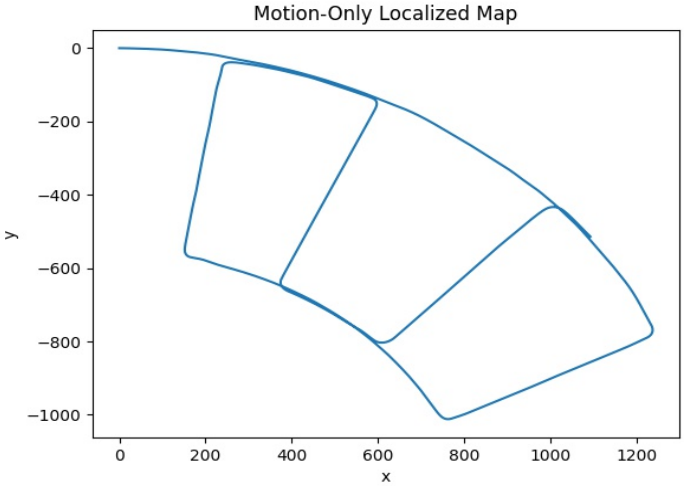

Fig. 8. motion only localized map 7

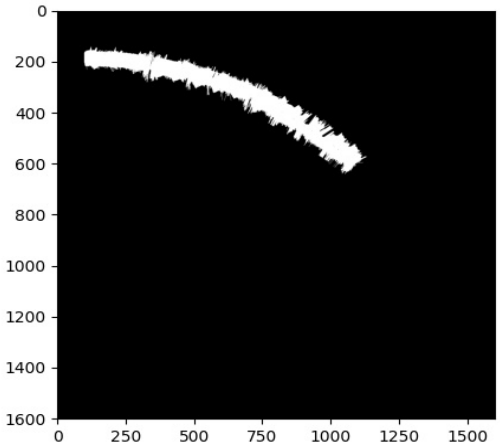

Fig. 10. test 300000

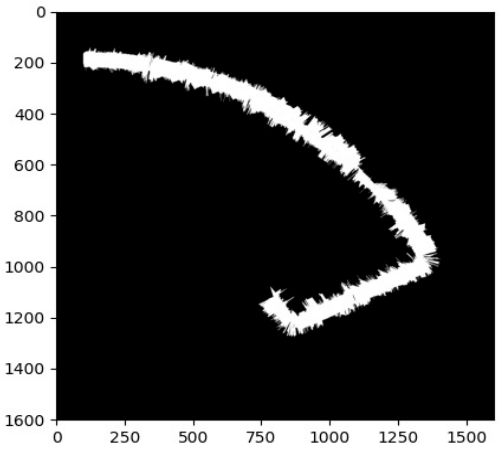

Fig. 11. test 450000

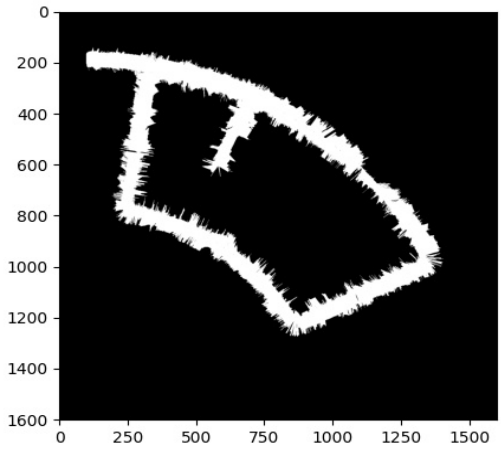

Fig. 13. test 750000

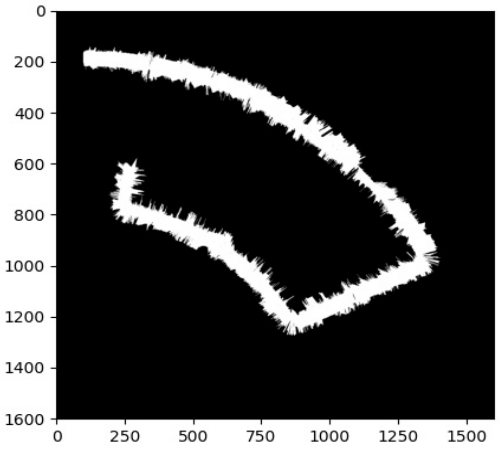

Fig. 12. test 600000

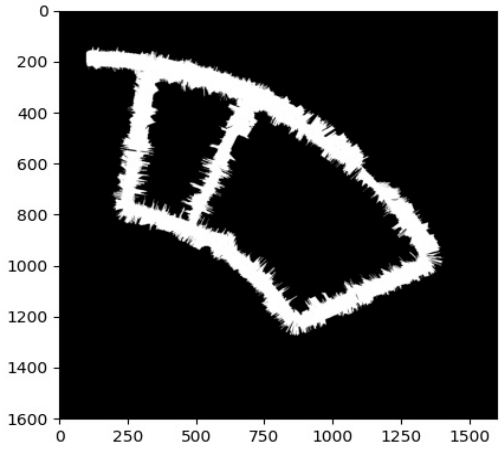

Fig. 14. test 900000

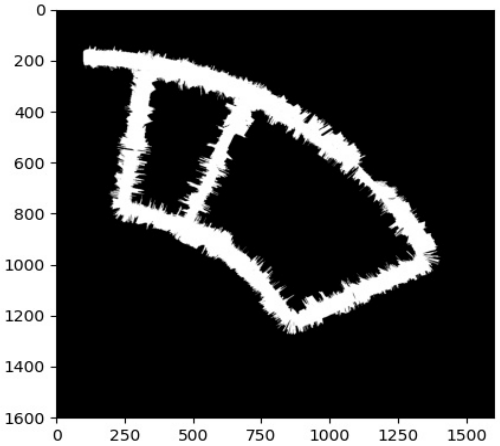

Fig. 15. test 1050000

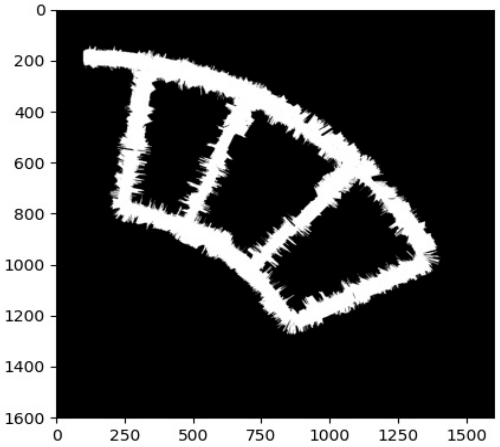

Fig. 16. test 1150000