Project 4: Simultaneous Localization And Mapping(SLAM)

Chi Zhang

I. INTRODUCTION

In this project, we implement the structure of mapping and localization in an indoor environment from an IMU and range sensors. We are provided with IMU data and odometry information from a walking humanoid with a 2D laser range scanner(LIDAR) as well as RGBD images. The Simultaneous Localization And Mapping(SLAM) algorithm is based on Monte Carlo particle filter algorithm.

II. SLAM

A. Data Preprocessing

Principally, the SLAM algorithm make use of the lidar data and robot IMU data to doing both mapping and localization. The first step should be data preprocessing since the scan data is expressed in head lidar frame. We need to transform the scan data to body frame. Then transform it to global frame using IMU data.

$$z_t^{(g)} = T_b^g T_h^b \cdot z_t^{(h)}$$

Then another issue we should consider is eliminating the data which is too far or too near the lidar. In this project, I eliminate the data whose distance is lower than 0.5 meters or higher than 30 meters.

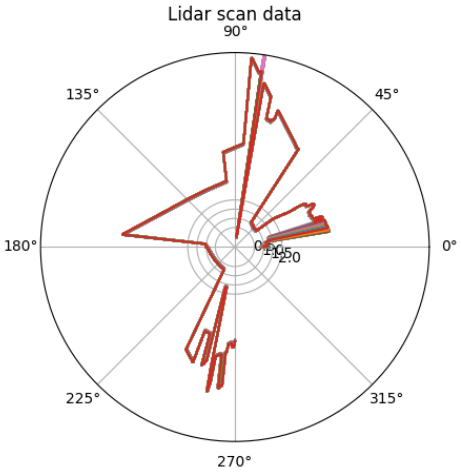

Fig. 1. Raw Lidar data

B. Occupancy Grid Mapping

The algorithm of mapping is pretty simple. In generally, there are grid maps use arrays with discretized cells to represent a real world. Each cell has a confidence of if it is occupied. With use of the scan data from lidar in each time step, the robot can update the confidence of grid cells. If the confidence hits a higher bound, then we can make sure that corresponding grid is an occupied grid.

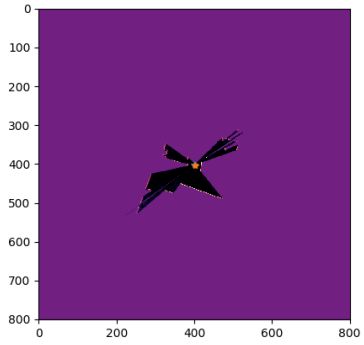

Fig. 2. First update of grid map

Pseudo-code

1. Transform $z_t = z_t^{(h)}$ via $T_b^g T_h^b$ to the global frame
2. Remove scan points that are too close, too far, or hit the ground
3. Obtain the cell locations y_t^o that are occupied according to the laser and y_t^f that are free according to the laser
4. Increase the log-odds in m_t of the occupied cells y_t^o and decrease the odds on the free cells y_t^o to obtain m_{t+1}

C. Monte Carlo Localization

Monte Carlo Localization, also known as particle filter, is used as a filter to localize robots. The algorithm assumes that the robot has no information about where it is, and thus make it start with a uniform random distribution of particles. When the robots moves, each random particle update its motion based its local frame. At the same time, the weight of each particle can be updated with respected of the correlation between map sensed by each particle and previous best map.

Pseudo-code

1. Compute odometry for each particle in local frame, the update the odometry on each particle and transform it in global frame.
2. Compute the correlation between particle map and previous best map.
3. Update each particle's weight and then normalize it.
4. If the $N_{eff} < N_{threshold}$, resample the particles using Low-Variance Sampling.

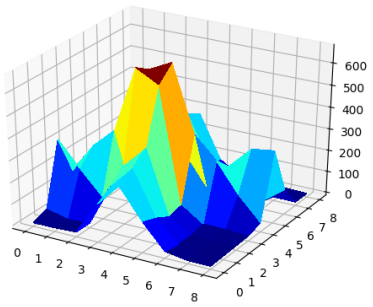

Fig. 3. Correlation Map

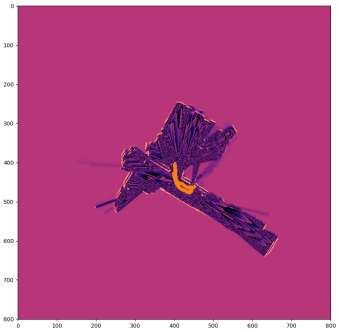

Fig. 4. Map0

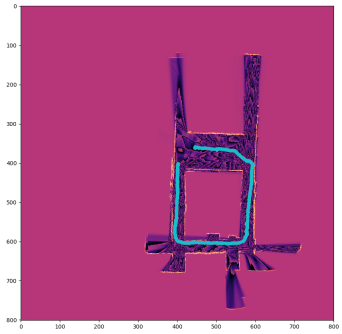

Fig. 5. Map1

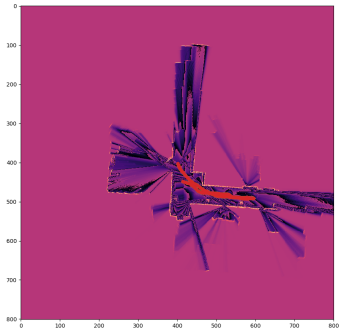

Fig. 6. Map2

D. Results

Then let me show my results of SLAM.
Parameters:
1. Noise: multi-Gaussian with covariance of

$$W = \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.5\pi/180/100 \end{pmatrix}$$

2. Particle number: 100
3. Neff : 10
4. log occupacy increase : 3
5. log free decrease: -1
6. log occupacy threshold: 6

III. Textures Map

In this project, we are also provided with RGB images and Depth images. These data are captured by Kinect, which has both a RGB camera and a Depth camera. When we have these images, the straight forward thinking is match these two kind of images. This can be achieved simply by using the transformation matrix and intrinsic coefficients of RGB camera and Depth camera. After doing this, we can read the rgb color from each pixel of depth camera.

The second part should be aligning the rgb color with our grid map from SLAM. By using the image points in camera frame, we can then transform them in global frame, which is used by our grid map. Then just discrete the grid map, we can have the texture 2D map.

Pseudo-code:
1. Transform I_t and d_t via $T_b^g \cdot T_h^b$
2. Find the ground plane in the transformed data via RANSAC or simple thresholding on the height
3. Color the cells in tm_t using the points from I_t and d_t that belong to the ground plane

A. Results

Then let me show my results of texture map.

IV. CONCLUSIONS

There is some problems for the test map. It cannot connect to the original point when it finish all the walking. Basically, it may be because the robot is adding up the errors when walking through the environment. It is a problem of loop closure consistency.

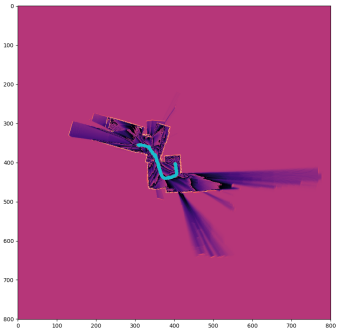

Fig. 7. Map3

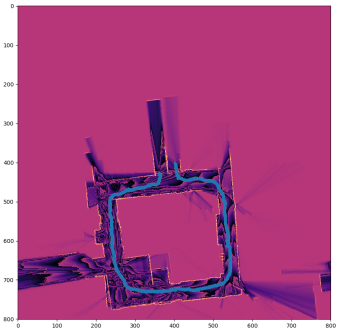

Fig. 8. Test Map

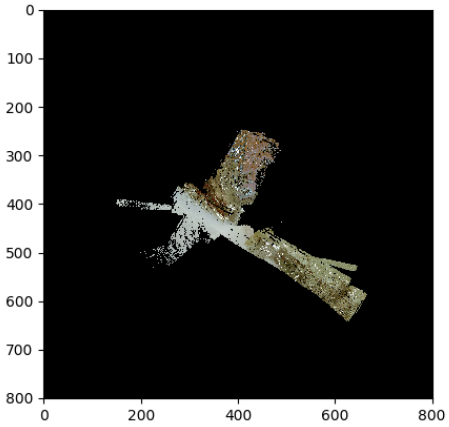

Fig. 9. Texture Map0

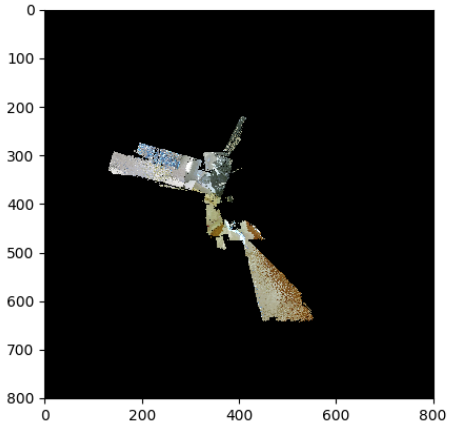

Fig. 10. Texture Map3

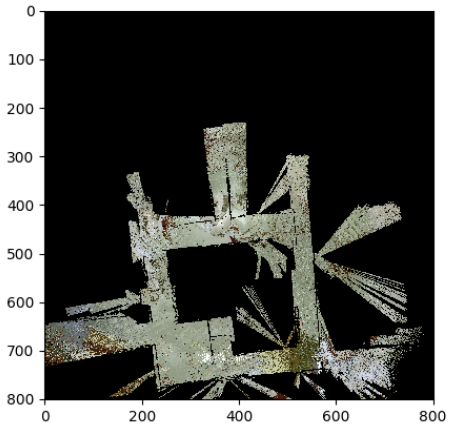

Fig. 11. Texture Map test