Lianming Shi

PID: A99097650

ECE 276

Project 3

Implement SLAM On Robot Thor

Introduction

Simultaneous Localization and Mapping, or SLAM, is arguably one of the most important algorithms in Robotics. Visual SLAM algorithms are able to simultaneously build 3D maps of the world while tracking the location and orientation of the camera. SLAM algorithms are complementary to Deep Learning, which is the master of perception recognition problems. It focuses on geometric problems. Today's SLAM systems help machines geometrically understand the immediate world. In this project, we are supposed to implement indoor localization and occupancy grid mapping using odometry, inertial, laser range, and RGBD measurements from a humanoid robot.

Problem Formulation

Given the data of a humanoid robot, the goal is to use the odometry, and laser measurements to localize the robot and build a 2-D occupancy grid map of the environment, and use the RGBD information to color the floor of my resulting 2-D map. Hence, I divided the project into several parts:

I. Initializing Data

a. Convert the lidar reading data into x y coordinate

b. Convert from lidar frame into robot body frame

c. Convert from robot body frame into world frame

II. Particle filter

a. Implement a particle filter with noise and find the best particle in large samples

b. Use the laser scan from each particle to compute map correlation (via getMapCorrelation) and update the particle weights

c. Choose the best particle among the particles (largest weight). if $N_{eff}<N_{threshed}$, then do stratified resampling

III. Mapping (SLAM)

a. Create the initial map

b. Convert the best particle scan to cells and update the map log-odds

c. Project the laser scan of the best particle

d. Use Bensenham algorithm to find free cells

e. Update wight

f. Update the map log-odds

IV. Texture Map

Use RGBD images from the best particle pose to assign colors to the occupancy grid cells

Technical Approach

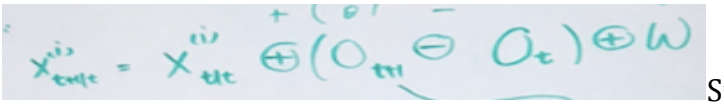
A. Initializing Data

At first, do time alignment for joint and lidar by selecting the joint data that are close to lidar data. I removed the laser scan point that are too close, too far, or hit the ground. Then I converted the lidar measurement into x y coordinate (Polar to Cartesian). Then I used the head and neck angles in joint measurement and robot height to compute lidar to body transformation ${}_bT_l$. Transform lidar measurement into body frame, Y_{body} can be calculated by multiplying ${}_bT_l$ with converted lidar measurement $[x\ y\ 0\ 1]^T$. Then transform lidar measurement into world frame, Y_{world} can be computed by multiplying ${}_wT_b$ with Y_{body}, where ${}_wT_b$ is obtained from the particle filter with the

largest weight.

B. Particle filter

Initialize particle filter with 100 particles with initial condition $u_{t|t}$=[0,0,0] and weight $a_{t|t}$ = 1/100. From the lidar pose, I got the Global frame odometry measurement Ot and Ot+1, which can be computed by multiplying ${}_{w}T_{l}$, which is obtained from lidar pose, with the inverse of lidar to body transformation $({}_{b}T_{l})^{-1}$.

In prediction step, I predicted $X_{t+1|t}$ by using odometry motion model and noise  Since we are in SE2 space, I just used plus and minus for odometry model and noise instead of smart plus and smart minus. I chose 0.01I for the motion model noise, and it worked pretty well. The weight is not changed in this step.

In update step, I used the laser scan from each particle to compute map correlation via getMapCorrelation given in the p3_uitls. Use mapCorrelation with a grid of 9x9 around the current particle position to get a correlation matrix, and find the maximum of the 9x9 by soft max function and move the particle to the maximum correlation position. Ph can be calculated by softmax(corr(z,m)−maxcorr(z,m)), and then it is obvious that each one is not bigger than 0 so the exponential is small than 1.

$$p_h(z_t|x,m) = \frac{exp(corr(z_t,m))}{\sum_z exp(corr(z,m))} = \text{softmax}(corr(z,m))$$

Update particle weight by

$$\alpha^{(k)}_{t+1|t+1} = \frac{\alpha^{(k)}_{t+1|t} p_h(z_t|x,m)}{\sum_j \alpha^{(j)}_{t+1|t} p_h(z_t|x^j,m)}$$

Find the particle with largest weight. Then computed N_{eff}, which is representing how many particles are with effective weight. $$N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} \left(\alpha^{(k)}_{t|t}\right)^2}$$

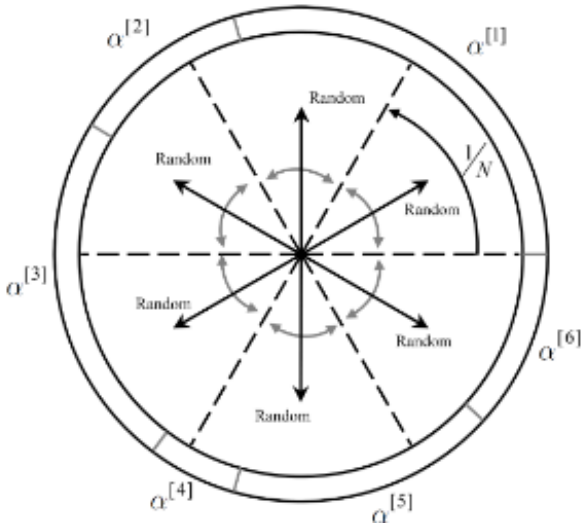
If $N_{eff}<T_{threshold}$ (I choose $N_{threshold}$ =5), do the stratified resampling:

INPUT: particles get from prediction step, weight get from update step, and the total number of particles.

OUTPUT: resampled new X particles and new weight.

Procedure: First, create a uniform distribution u, initialize j=1 and c=weight_1. Scan each part in the circle. If Beta is larger than c for j times, c=c+weight_j: increasing the decision section length, and add this repeated particles_j and 1/N in the new set. Repeat this for all particles.

1: **Input**: particle set $\{\mu^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
2: **Output**: resampled particle set
3: $j \leftarrow 1,\ c \leftarrow \alpha^{(1)}$
4: **for** $k = 1, \ldots, N$ **do**
5: $\quad u \sim \mathcal{U}(0, \frac{1}{N})$
6: $\quad \beta = u + \frac{k-1}{N}$
7: $\quad$ **while** $\beta > c$ **do**
8: $\quad\quad j = j + 1,\ c = c + \alpha^{(j)}$
9: $\quad$ add $(\mu^{(j)}, \frac{1}{N})$ to the new set



If $N_{eff}>T_{threshold}$, I was able to construct body to world transformation ${}_{w}T_{b}$ by using x y position and angle of the best particle. Use the ${}_{w}T_{b}$ to calculate the Y_{world} back to the first step.

III. Mapping (SLAM)

Initialize the map with zeros (scale, scale). I converted Y_{world} (Beam) to cell by using numpy cell function. The goal is to find the cells in the grid map that these beams go through. Input starting x y point (best particle x y coordinate) into ending x y point (Y_{world} x y coordinate) to bresenham algorithm to get the free cells that the beams go through.

Threshold the map: cell is occupied if p(mi=1)>0.8 , cell is free if p(mi=1)<0.2. Update log-odds map: Increase the log-odds of the occupied cell indices, and decrease the log-odds of the free cell

indices.

mi =mi+log(b) free cell

mi =mi+log(1/b) occupied cell

Then recover the map:

$$P(m_i = 1) = 1 - \frac{1}{1+exp(m_i)}$$

Hence, with occupied cell=1 and free cell=-1 and unknown=0, I was able to plot the final map.

IV. Texture Map

Read data in RGB and depth. After that, I did the time alignment for lidar measurement time comparing with the texture time. Then I wrote the transformation from IR to RGB frame, and RGB frame to body frame. Then I use the current best particle to get the transformation from body to world. Right now, I was able to transform RGB image and depth image into world frame. Then I need to find the ground plane in the transformed data via RANSAC, which I did not finish. The next step is to color the cells in texture map using the points from RGB image and depth image that belong to the ground plane.

Results and Discussion

Here are the results for training data. Overall, the results are great, and the shape of each map is plotted clearly. However, Fig1 has a small vibration on its right side and is less clear than other maps. I guess the reason is that dataset1 has much more data than other dataset, which increases the error in plotting. Moreover, I found that the accuracy of the map depends on lots of factors: map resolution, map threshold, noise matrix on prediction step, total number of particles etc. After trying to balance them, Fig0-Fig3 are the best running results with a time-step 100. I called them the best running results because there is a variation of the map each time you run. In addition, I did not varying yaw in the prediction step of the particle filter in every cycle, so I think it may be one of the reason that cause the blur of those maps.

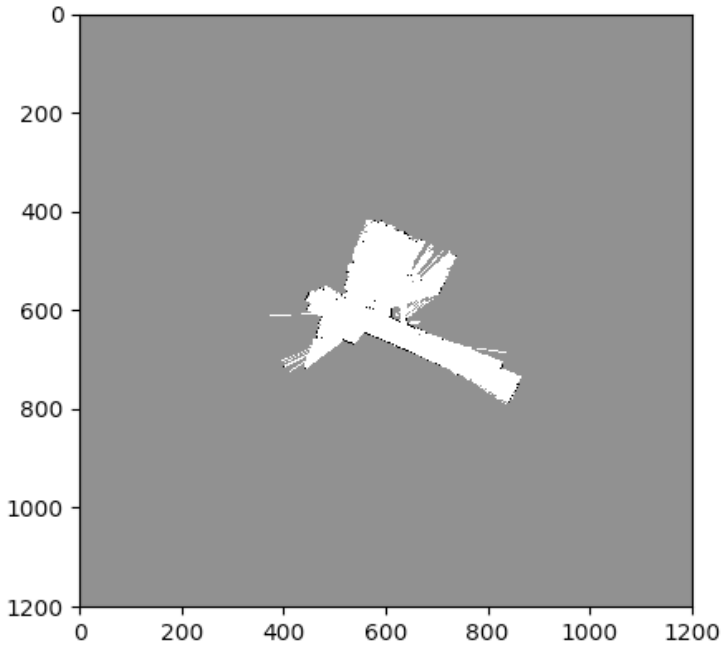


Fig0. Training set 0

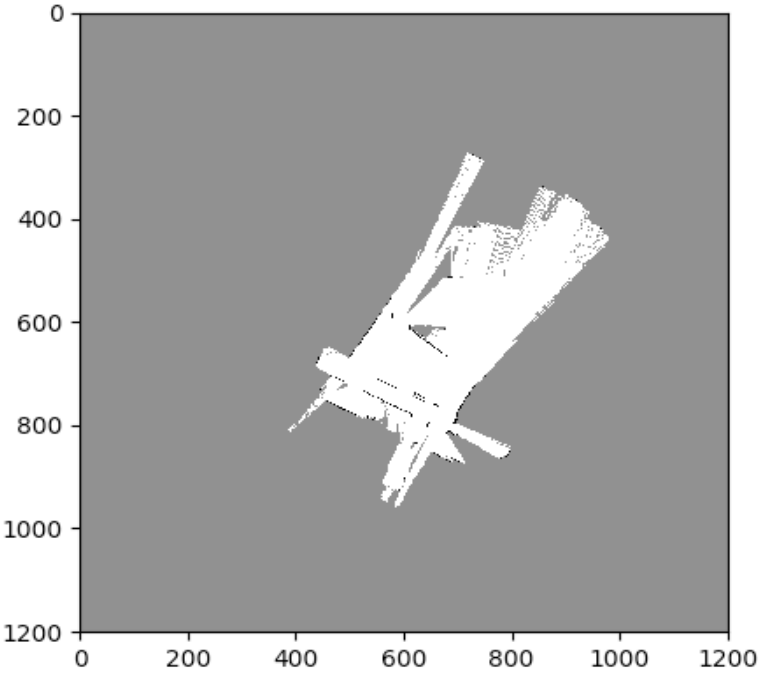


Fig1. Training set 1

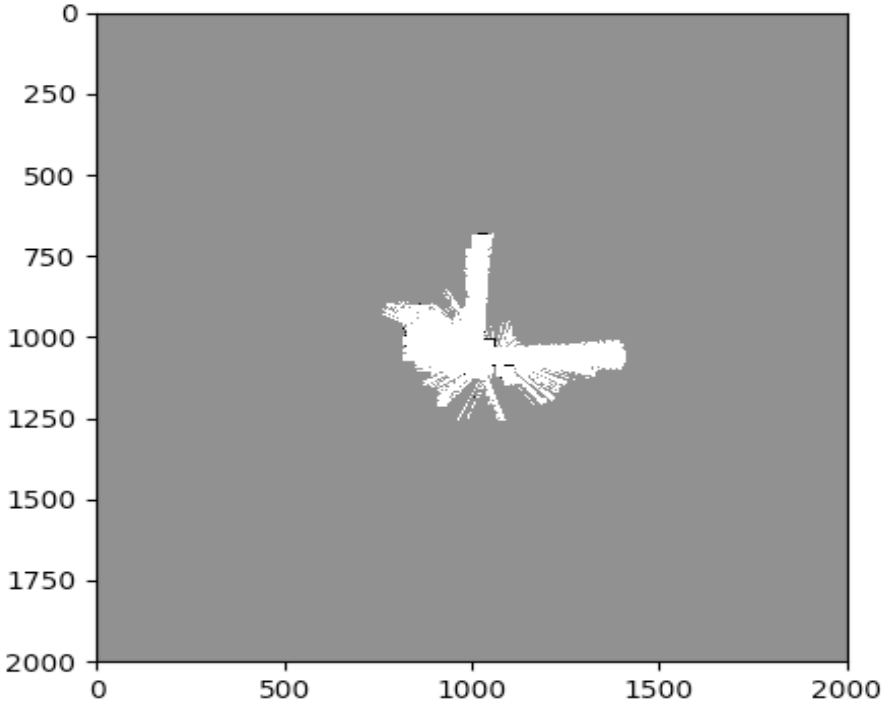


Fig2. Training set 2

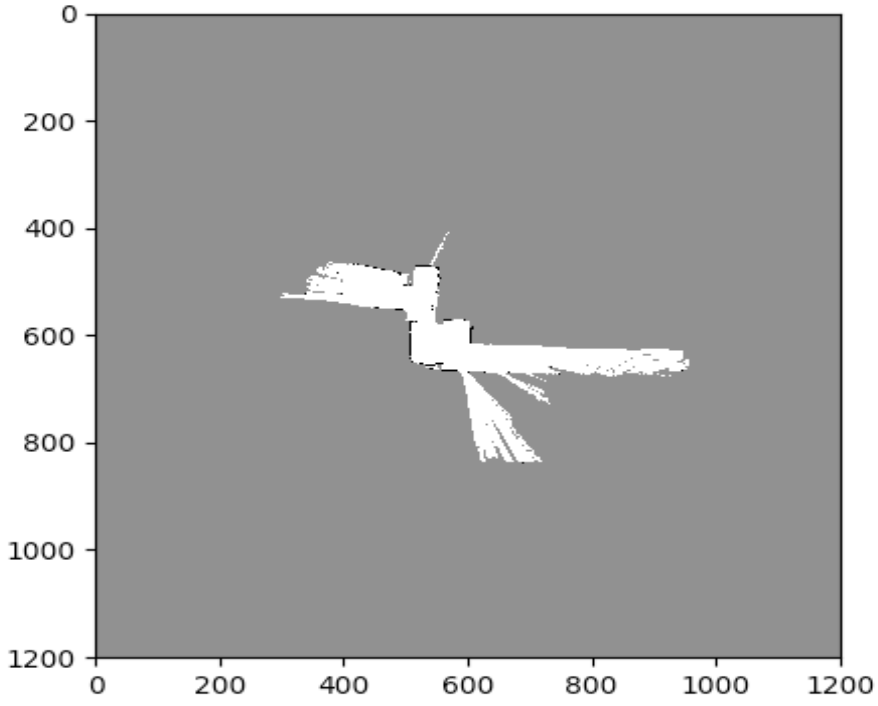


Fig3. Training set 3

TESTSET map:

In order to get a much more clear map, I changed the map resolution and occ-free cell threshold. Having done those changes, I got the Fig 4 and Fig 5 which are the top 2 most clear among all the trials. In Fig 4, I labeled two parts that I want to explain. For label 1, the beam seem did not touch the wall, it shoot might go through a window or mirror that cannot reflect back. For label 2, I think that it may be a balcony or just the product of noise.

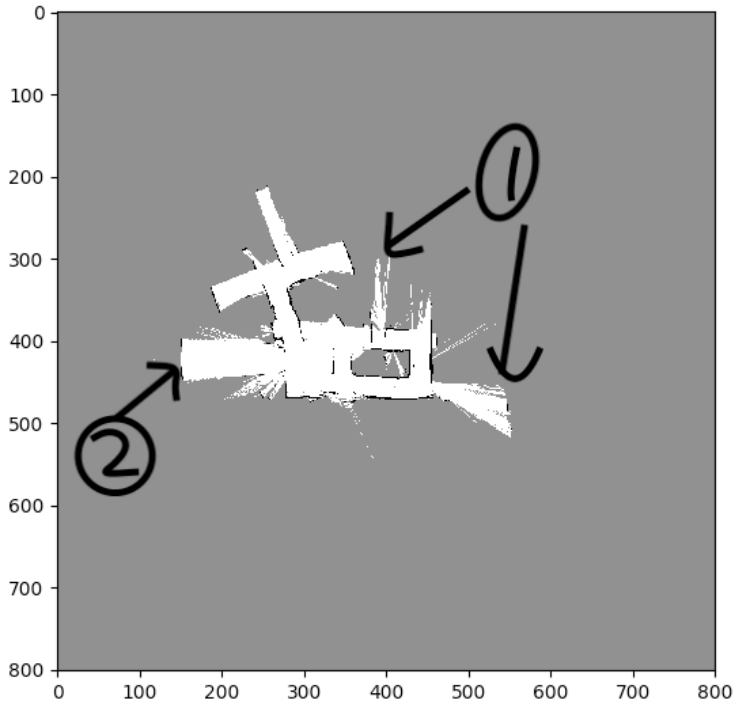


Fig 4. Test Set Trial 1

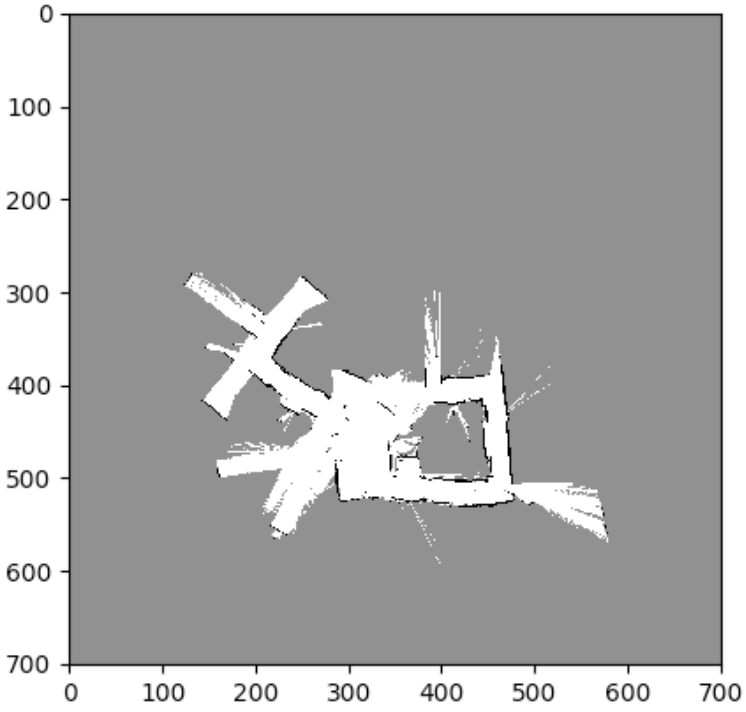


Fig 5. Test Set Trial 2