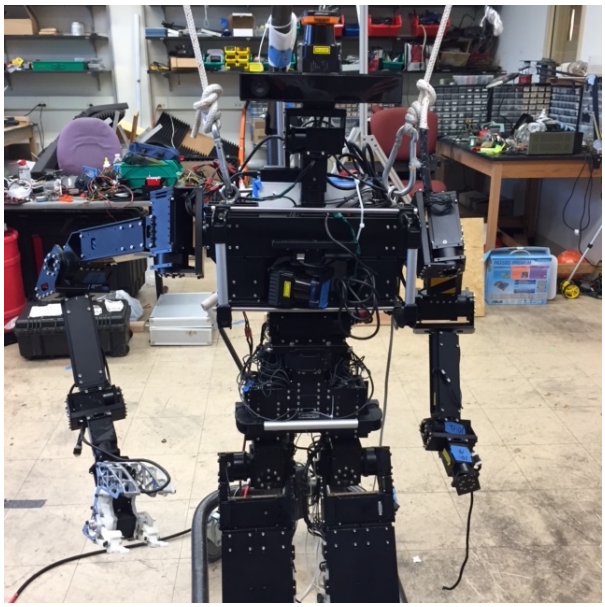
Particle-filter based Simultaneous Localization And Mapping (SLAM)

Yiren Lu
GRASP Lab, University of Pennsylvania
Mar 2016

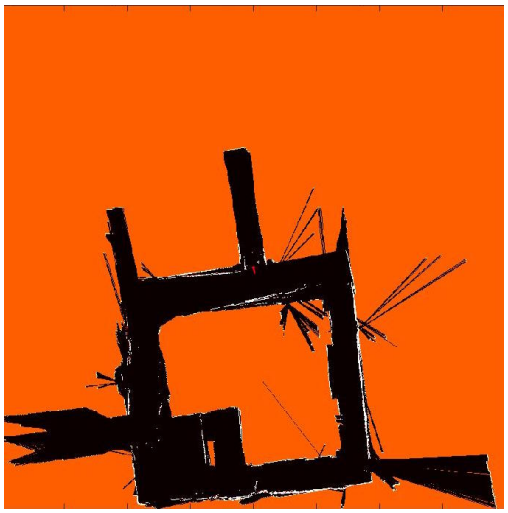
1 Introduction

In ESE 650 4th project, we did simultaneous localization and mapping (SLAM) of humanoid robot THOR based on Monte Carlo particle filter algorithm [2].



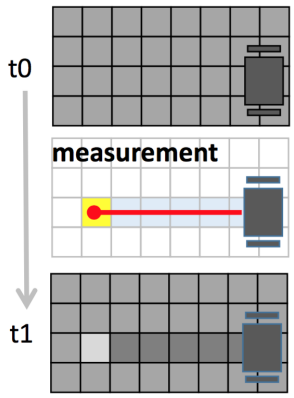
2 Dataset and Preprocessing

We have data from multiple sensors: Encoder's odometry data, IMU, Kinect, Lidar. In this project, I first simulate the robot path according to the robot's odometry data. Since the encoder's yaw data is less accurate than IMU's yaw, I adopted IMU's yaw and recomputed the incremental motion data. The map reconstructed from the odometry data with IMU's yaw works reasonably well. I also tried compensate head yaw and pitch. For train set 3, adding head's yaw effectively improve the performance. Head's pitch angle does not influence the performance too much. The following is the result only using the odometry data for test set



3 Occupancy Grid Mapping [1]

The occupancy grid is a map representing the belief of whether the position x, y is occupied or free $m_{x,y}$ by accumulating the measurement($z = 1, -1$)'s log-odd as follows:



$$\begin{aligned}\text{logodd} =& log\frac{p(z|m_{x,y}=1)p(m_{x,y}=1)}{p(z|m_{x,y}=0)p(m_{x,y}=0)}\\ =& log\frac{p(z|m_{x,y}=1)}{p(z|m_{x,y}=0)}+log\frac{p(m_{x,y}=1)}{p(m_{x,y}=0)}\end{aligned}$$

$$\begin{aligned}\text{logodd}_{occ} =& log\frac{p(z=1|m_{x,y}=1)}{p(z=1|m_{x,y}=0)}\\ \text{logodd}_{free} =& log\frac{p(z=-1|m_{x,y}=0)}{p(z=-1|m_{x,y}=1)}\end{aligned}$$

$$logodd = \begin{cases} logodd + logodd_{occ}, & \text{if z=1} \\ logodd + logodd_{free}, & \text{if z = -1} \end{cases}$$

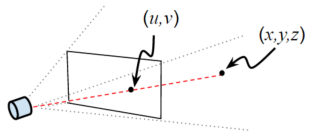
4 Particle Filter based Localization

I implemented Monte Carlo particle filter for robot localization. The procedure is:

- Initialize N particles
- For each timestamp, process the particles with odometry data and add noise
- Comparing map correlation and reweight the particles according to their correlation level to the existing map
- Calculate effectiveness factor $Neff = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}$, if it's less than αN, then resample according to importance. For my implementation, $\alpha = 0.7$
- Collect sensor's data, do mapping using the particle with largest weights
- Repeat

5 Ground Detection

I implemented Random sample consensus (RANSAC) algorithm to fit ground plane. However in this project, there is an easier way. Here is the depth image to position (x,y,z) mapping:



$$\frac{u}{f_u} = \frac{x}{z}$$
$$\frac{v}{f_v} = \frac{y}{z}$$

A ground point should simply satisfies:

$$|\frac{v}{f_v} - \frac{h}{z}| < \varepsilon$$

Where, h is the height of camera and epsilon is the threshold.

6 Results and Discussion

- I tuned my parameters a lot but here is the best results I have currently as shown in the following images. Due to time limitation, I haven't finished the RGB mapping yet.
- Surprisingly, on the test set, the map reconstruction from odometry data is slightly better than with particle filter.

Acknowledgement

Thanks Dr. Lee for designing and preparing this project and thanks TAs for timely responses to our questions.

References

[1] THRUN, S., ET AL. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium 1* (2002), 1–35.

[2] THRUN, S., FOX, D., BURGARD, W., AND DELLAERT, F. Robust monte carlo localization for mobile robots. *Artificial intelligence 128*, 1 (2001), 99–141.

Figure 1: Train set 0

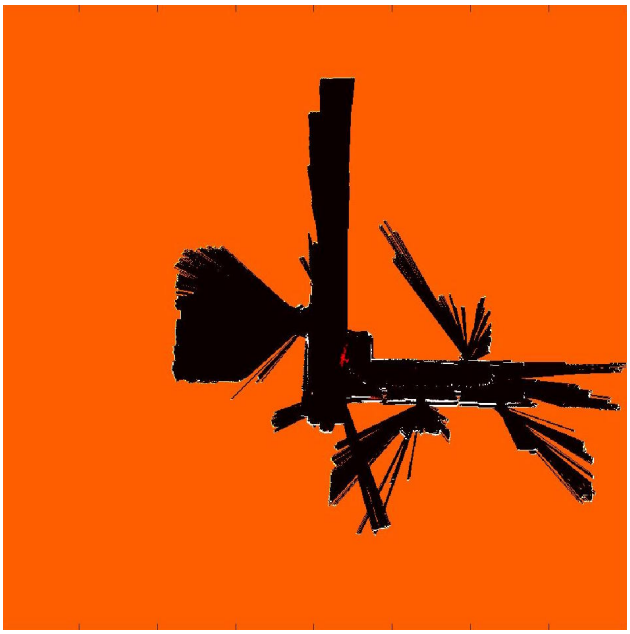


Figure 3: Train set 2

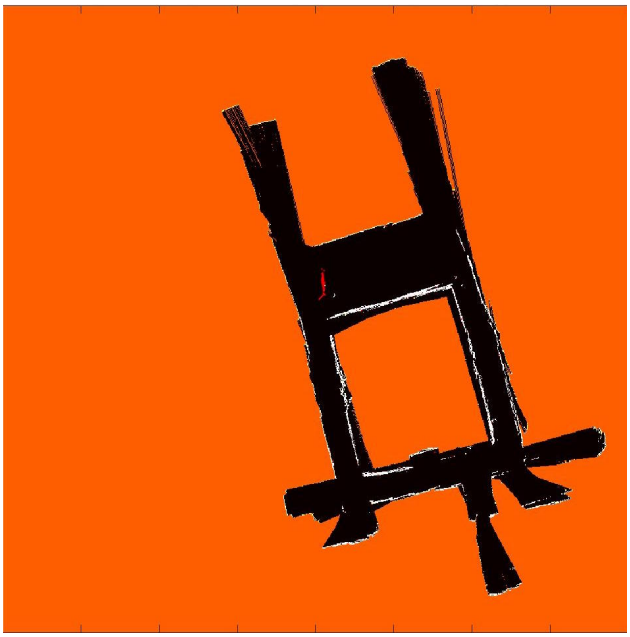


Figure 2: Train set 1

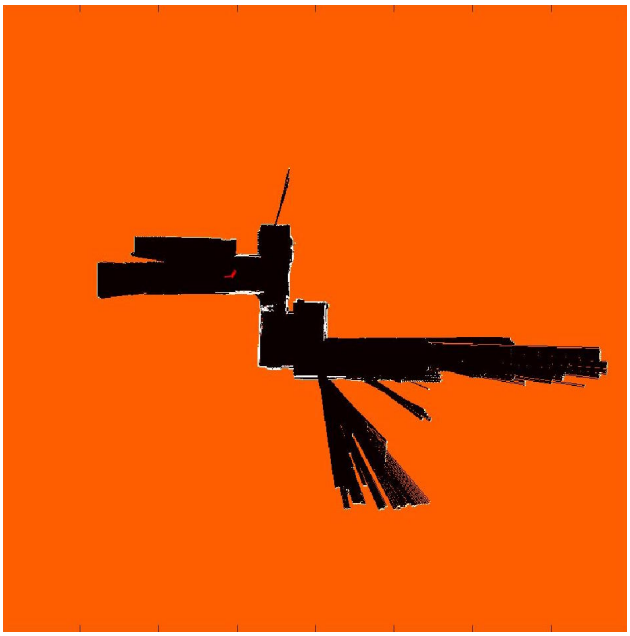


Figure 4: Train set 3

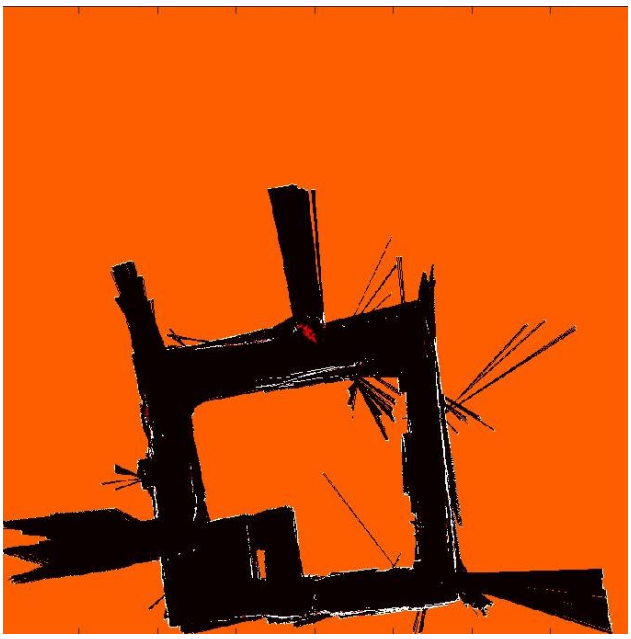


Figure 5: Test set

Figure 7: Ground detection for train set 0 mapping to rgb image

Figure 6: Ground detection for train set 0

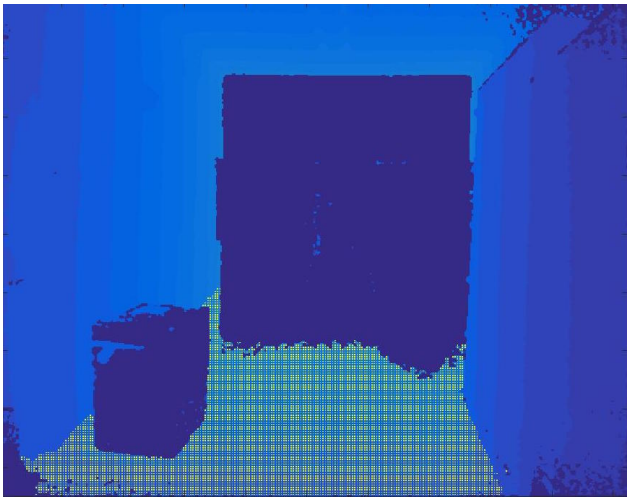


Figure 8: Ground detection for test set