# Project 2: Particle Filter SLAM

Joseph Chang
Department of Electrical Computer Engineering
University of California, San Diego
jdchang@ucsd.edu

## I. Introduction

Simultaneous Localization and Mapping (SLAM) is a Computer Vision algorithm used to track a robot as it moves in the world. It creates a map grid of its surroundings and continually locates and updates the robot's location within it. This technique is extremely useful for many applications dealing with unknown environments such as navigating disaster zones, underwater terrain, or acquiring intelligence in enemy territory. It is also used in consumer electronics such as cleaning robots and in medical devices for internal body mapping.

At first glance, SLAM appears to solve a chicken-and-egg problem since we want to build a map of the environment given the robot's state trajectory, but also need to localize the robot and estimate its trajectory given a map of the environment. There are numerous variations of SLAM developed to deal with this problem including the particle filter, Kalman filter, Factor Graphs SLAM, Fast SLAM, and Kinect Fusion.

## II. Problem Formulation

In our experiment, a humanoid robot named THOR navigates various hallways. It is equipped with a Kinect v2 which collects odometry data and captures RGBD frames. The Kinect is 0.15m above the head. It also holds a Lidar which takes 2D laser scans measurements of the environment. A Hokuyo Lidar sensor is 0.07m above the head. Both sensors have fixed translation and no rotation. The head frame is 0.33m above the center of mass located on the body and can rotate in two directions. The center of mass is 0.93m above the ground.

The Kinect captures RGB images, depth images in millimeters, and their timestamps. It also records the RGB and IR intrinsic and extrinsic camera parameters. The odometry data, also known as joint data, records the robot's head angles, neck angles, and their timestamps. If the head rotates left, yaw is negative, and the head angle is positive. If the head rotates down, pitch is positive, and the neck angle is positive. The Lidar collects depth scans of the room, the robot's change in pose, and their timestamps. Scan data is 1x1081, has a maximum range of 30m, and only detects within -135° to 135° with 0° being where the robot faces forward. The change in pose, also known as delta pose, contains the x-movement, y-movement, and change in angle $\theta$.

The goal is to implement SLAM using the robot's odometry, 2D laser scans, and RGBD measurements. Odometry and laser scans are used to localize the robot and build a 2D occupancy grid of the environment. The occupancy grid is a discretization of space into cells which is 30m×30m and has a resolution of 0.05m. Once the grid is created, we want to transform the lidar scans $z_t$ onto the grid as trajectory lines. The accumulation of lines will eventually create a map of the environment walls as log-odds

$\lambda_{i,t+1}$ which can be mapped as a probability mass function $\gamma_{i,t}$ and saved as an image or read by the robot. RGBD information is used to color the floor of the 2D map and create a texture map.

## III. Technical Approach

We begin by preprocessing the Lidar data to remove scan points that are closer than 0.5m or further than 20m. These data contain a lot of noise because they are near the sensor's detection range. Scans that hit the ground are also removed. To calculate the Lidar's z-coordinate $L_z$ in the world, we use the following formulation where 1.275m is the Lidar sensor's height above ground. In all cases where $L_z \leq 0$ the lidar scan, delta pose, and timestamp are removed and the total number of lidar scans decreases.

$$\text{joint\_head\_angle} > 0 \rightarrow \mathrm{L_z} = 1.275 - z_t * sin(joint\_head\_angle)$$
$$\text{joint\_head\_angle} < 0 \rightarrow \mathrm{L_z} = 1.275 + z_t * sin(joint\_head\_angle)$$

After preprocessing, we must match the joint data with corresponding Lidar scans. Their timestamps are compared and the joint data at the closest timestamp is placed a new array. We create an empty occupancy grid map where we will update our log-odds and begin implementing a particle filter. An initial particle set is initialized with 40 particles in the form (x-coordinate, y-coordinate, theta) and equally distributed weights of 1/40.

$$\boldsymbol{\mu}_{0|0}^{(k)} = (0,0,0)^\top \text{ with weights } \alpha_{0|0}^{(k)} = \tfrac{1}{N}$$

First, we apply an odometry motion model to predict each particle's pose $\mu_{t+1}$ by adding the robot's delta pose to it. Gaussian noise is also added because the delta pose has a degree of error. Then, we update the particle filter's weights using a laser correlation model. The Lidar scan is transformed to the real world using each particle's pose hypothesis from the predict step. We calculate the correlation between a 9x9 grid around each lidar scan in the world frame and the current occupancy grid which returns a matrix representing the number of correlations at each grid cell. These values are passed through a softmax function which normalizes them into a probability distribution proportional to the exponentials of the input values. These values are passed through a softmax function which normalizes them into a probability distribution proportional to the exponentials of the input values.

$$\mathbf{corr}(y, m) := \sum_i \mathbb{1}\{m_i = y_i\}$$

$$p_h(\mathbf{z} \mid \mathbf{x}, \mathbf{m}) = \frac{e^{\mathbf{corr}(\mathbf{y},\mathbf{m})}}{\sum_{\mathbf{v}} e^{\mathbf{corr}(\mathbf{v},\mathbf{m})}} \propto e^{\mathbf{corr}(\mathbf{y},\mathbf{m})} \quad \text{softmax}$$

The maximum correlation corresponds to the most accurate particle pose. We update the robot's trajectory by adding this new pose. To update the log-odds map, we transform the laser end points from the lidar frame to the world frame. This is not necessary for the laser start point as it is constant. The bresenham2D function is used to find the indices of cells between the laser's start and end point. For each observed cell i, the log-odds $\lambda_{i,t+1}$ is decreased if it is observed free and increased if it is observed occupied.

The map is constrained between $\lambda_{MIN}$ and $\lambda_{MAX}$ to prevent overconfident estimation. Finally, the map probability mass function $\gamma_{i,t}$ is recovered from the log-odds map and saved.

$$\lambda_{i,t+1} = \lambda_{i,t} + \log g_h(\mathbf{z}_{t+1} \mid m_i, \mathbf{x}_{t+1})$$

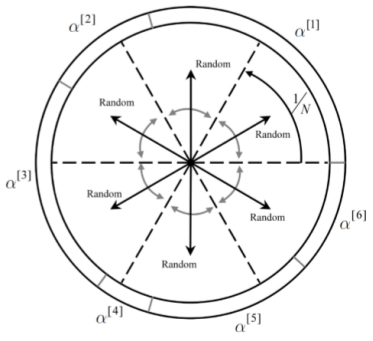
$$\lambda_{MIN} \leq \lambda_{i,t} \leq \lambda_{MAX}$$

$$\gamma_{i,t} = p(m_i = 1 \mid \mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = 1 - \frac{1}{1 + \exp(\lambda_{i,t})}$$

After each iteration, we calculate the number of effective particles. If $N_{eff} < 5$, the particle weights are resampled using stratified resampling. Weights are added along the circumference of a circle. The circle is divided into N=40 equal pieces a uniform is sampled on each piece. Samples with large weights appear at least once and samples with small weights appear at most once.

Stratified (low variance) resampling

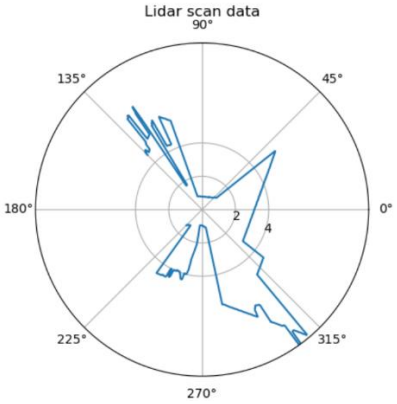
1: **Input**: particle set $\{\mu^{(k)}, \alpha^{(k)}\}_{k=1}^{N}$
2: **Output**: resampled particle set
3: $j \leftarrow 1$, $c \leftarrow \alpha^{(1)}$
4: **for** $k = 1, \ldots, N$ **do**
5: $u \sim \mathcal{U}\left(0, \frac{1}{N}\right)$
6: $\beta = u + \frac{k-1}{N}$
7: **while** $\beta > c$ **do**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

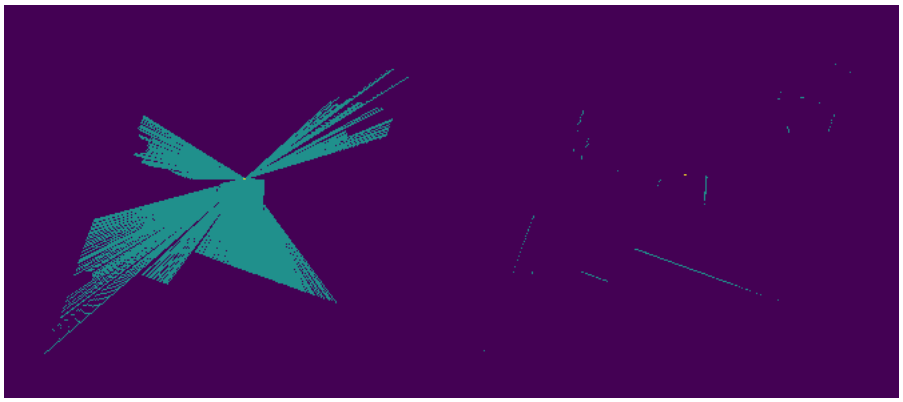



Texture mapping is unsuccessfully attempted. The goal is to project colored points from the RGBD data onto the occupancy grid using the best particle's pose. We begin by matching rgb and depth frames using their timestamps. The depth of each RGB pixel is determined from the depth image and then transformed into the world frame. We find the ground plane in the transformed data by thresholding the height. Then, the occupancy grid is colored using the projected points on the ground plane.

## IV. Results

Our Particle Filter SLAM implementation is run on three datasets containing .png depth images, .jpg RGB images, .mat joint data with head and neck angles, and .mat Lidar data with laser scans and delta pose. We begin by testing if the Lidar data follows the sensor's description. The first scan shown below is accurate as it ranges from -135° to 135° with 0° being where the robot faces forward. We then compute and display the first scan map and wall map to ensure our transforms are correct before estimating the robot's pose. Comparing the lidar data and scan map, they are identical so our algorithm is working. Walls are starting to appear at the correct locations in the wall map.

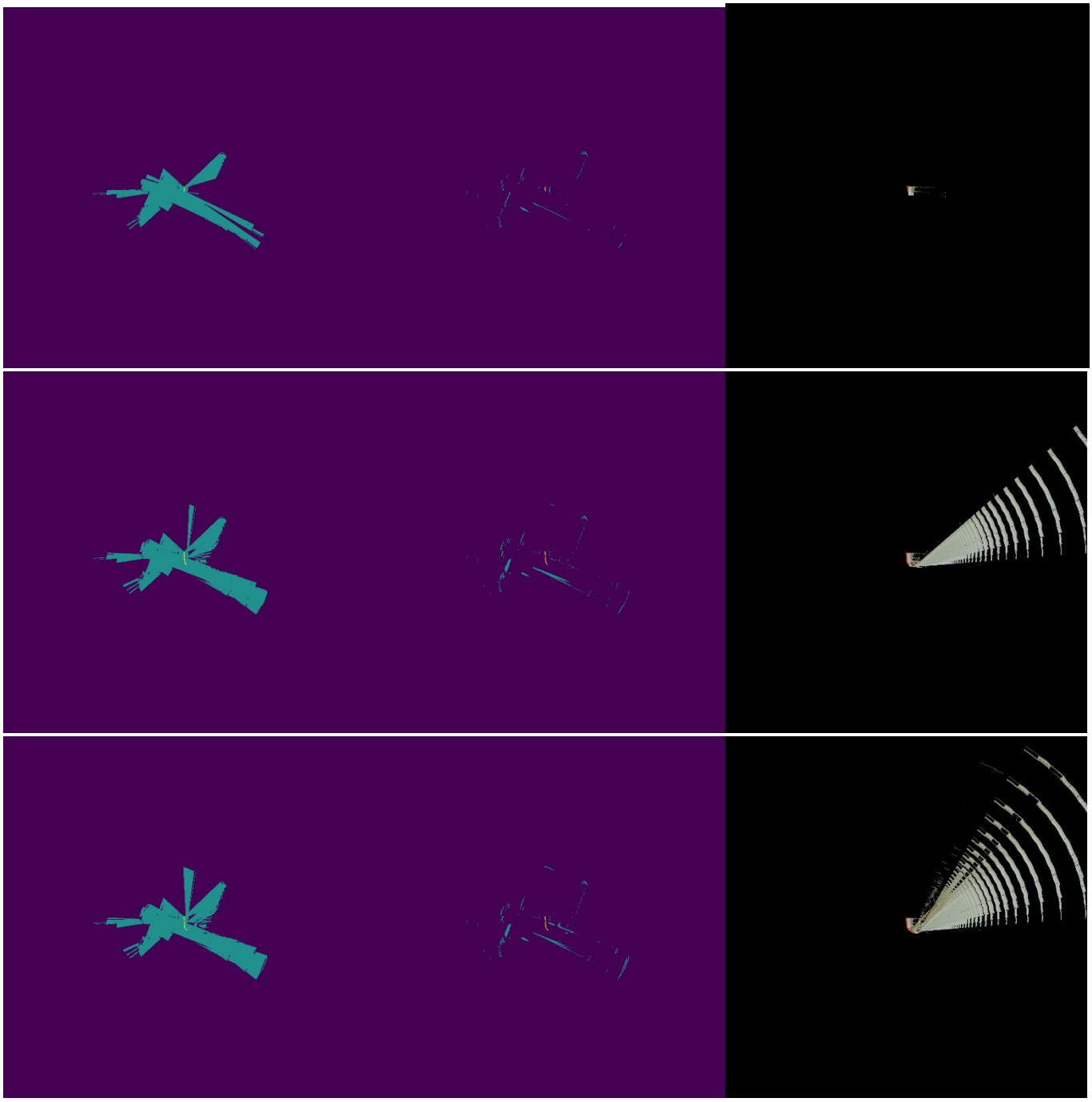


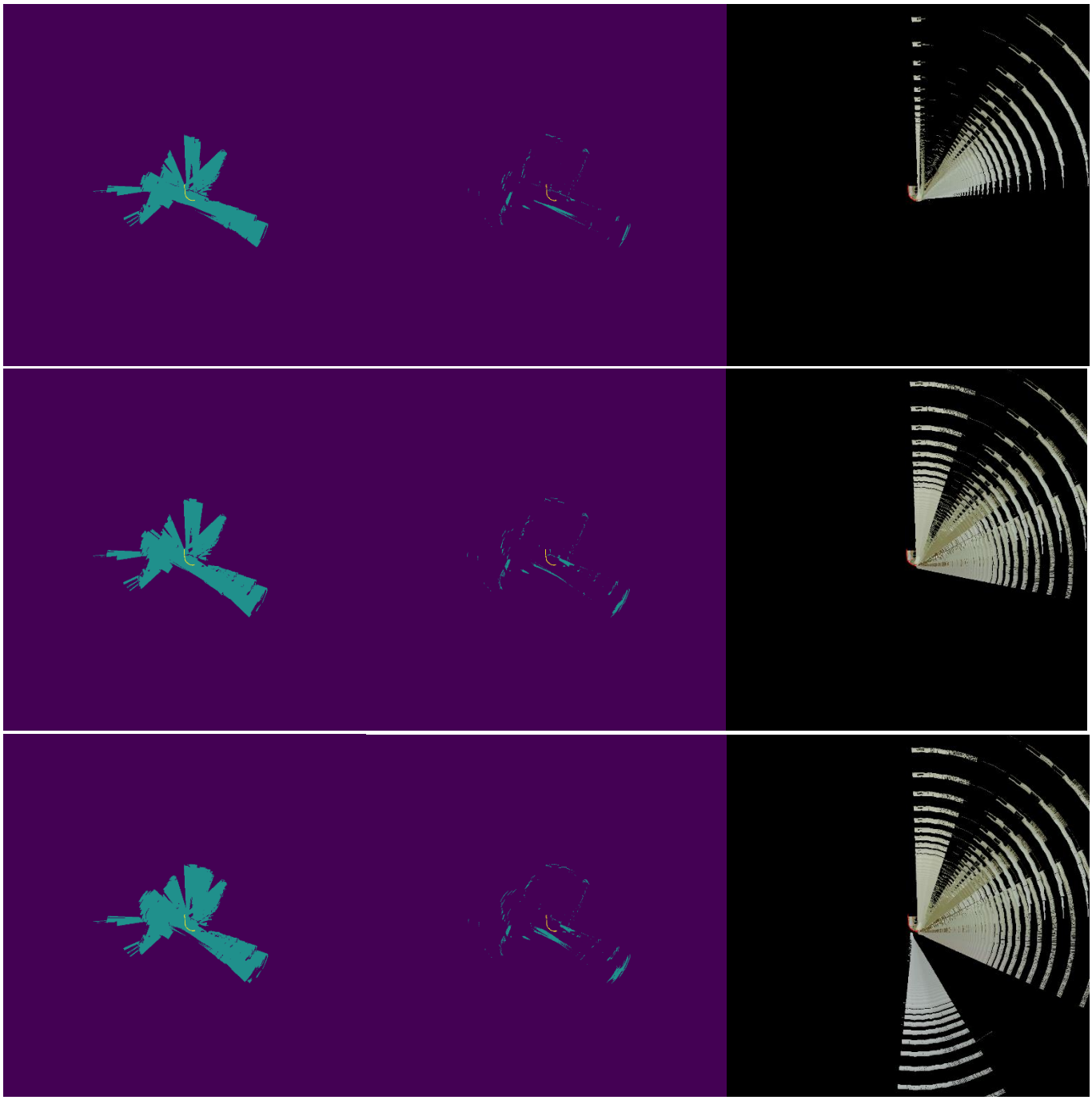



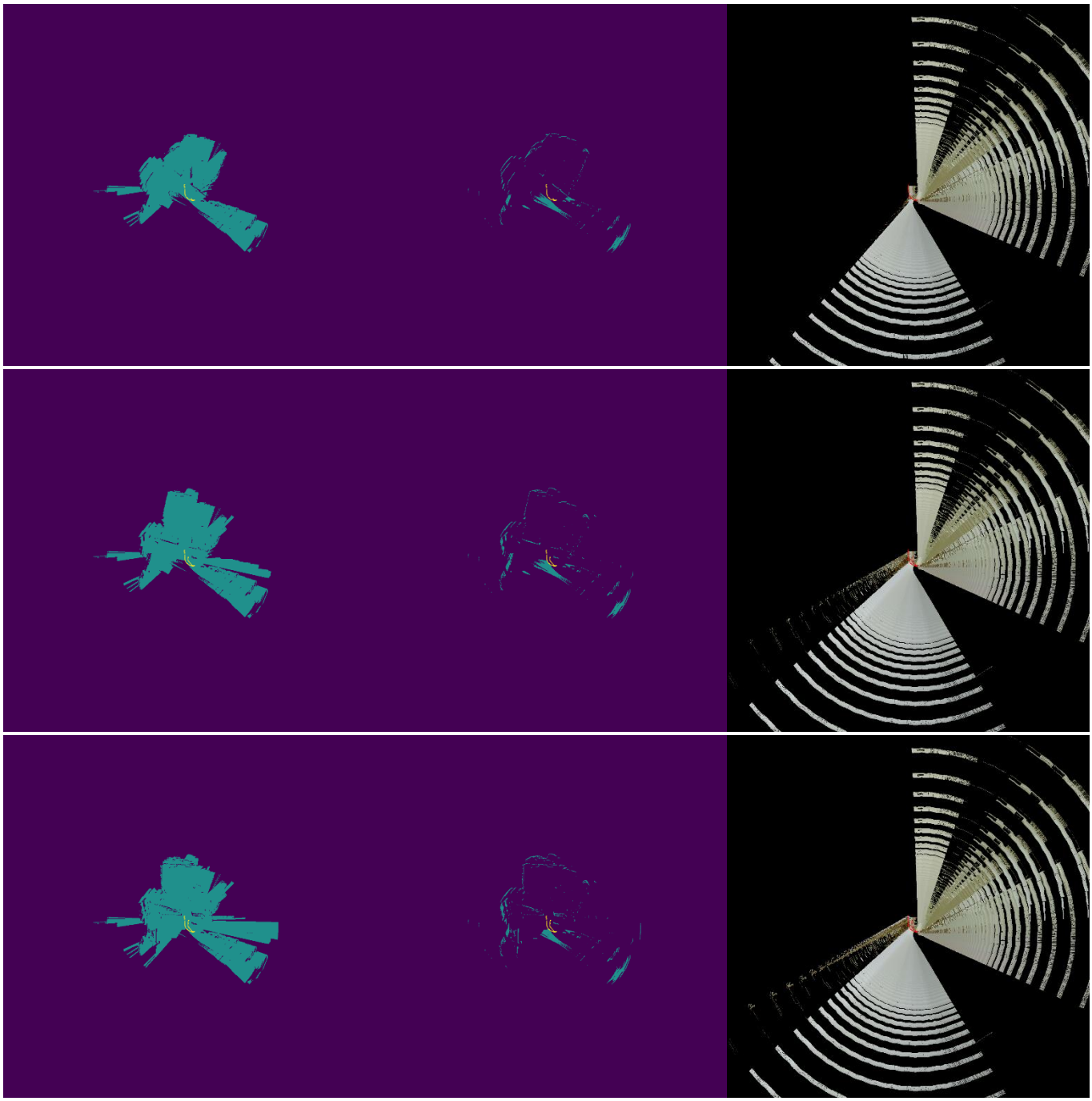
We proceed by running on three datasets 0, 3, and 4. The outputs are the scan map, wall map, and texture map which is not accurate. The yellow and red paths are the robot's trajectory through the indoor environment. Videos of the rgb frames are provided in the reference section for comparison.

**RGB_0, DEPTH_0, train_joint0, train_lidar0**

Iteration 0-11000 in increments of 1000



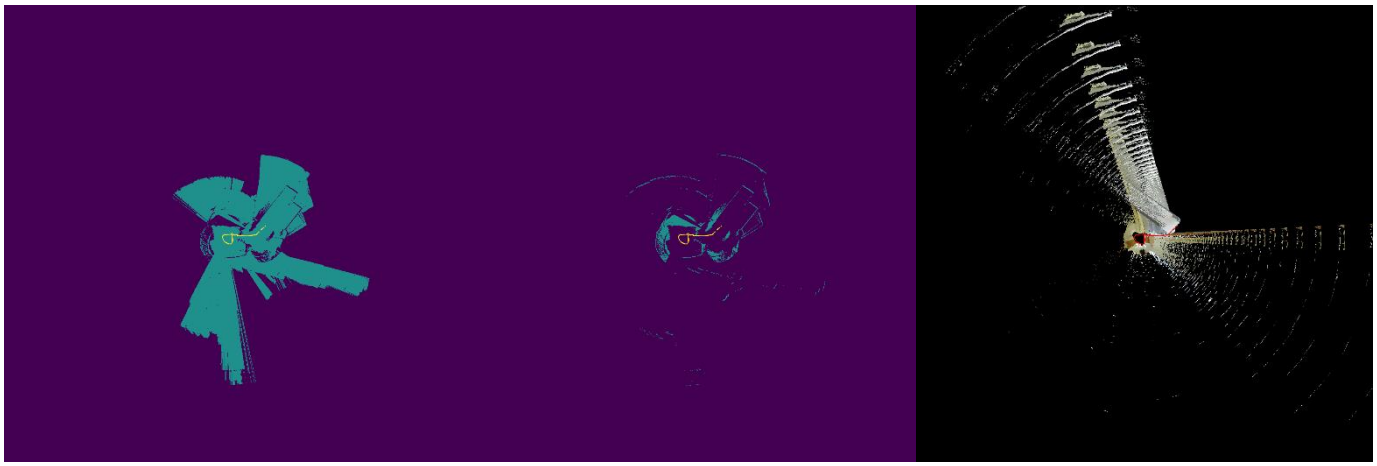




The map and wall results are accurate to the hallway the robot is navigating through. Initially, the robot hardly moves. Then, it makes a left turn around a corner wall, makes a U-turn, and returns to its original position. However, the texture map has a lot of radial distortion which is likely caused by an error in transforming the pixels to the ground plane. Although the texture map is poor in dataset 0 and 3, it runs fine on dataset 4 until past iteration 25000 where radial distortion appears. Some images are selected to understand the results. For complete videos of scan, wall, or texture maps of all three datasets, refer to the results_video folder. To view all image results, refer to the results_image folder.

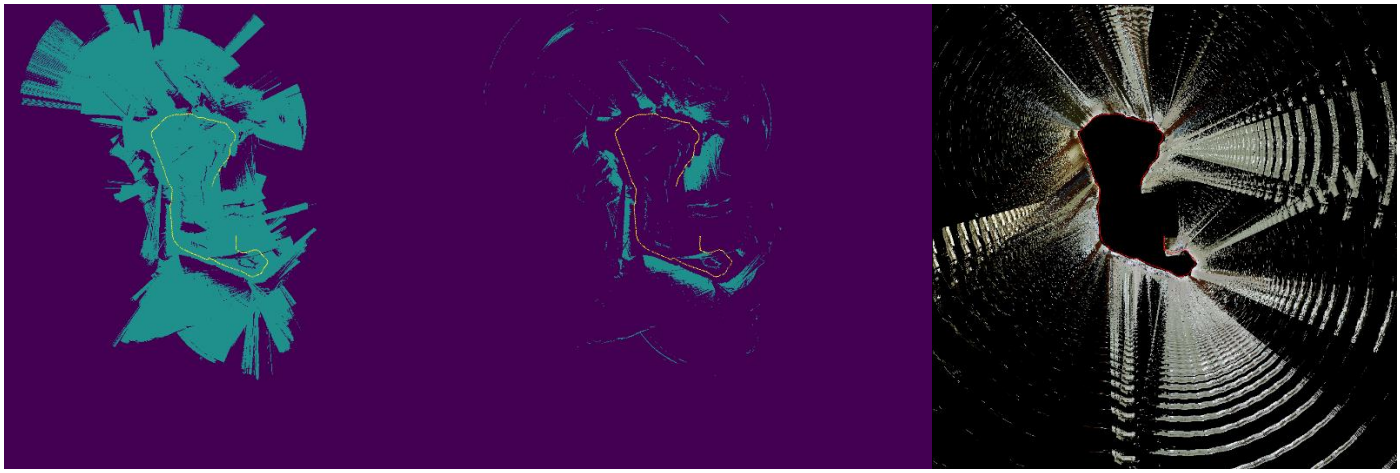
**RGB_3, DEPTH_3, train_joint3, train_lidar3**

Final iteration ~12000

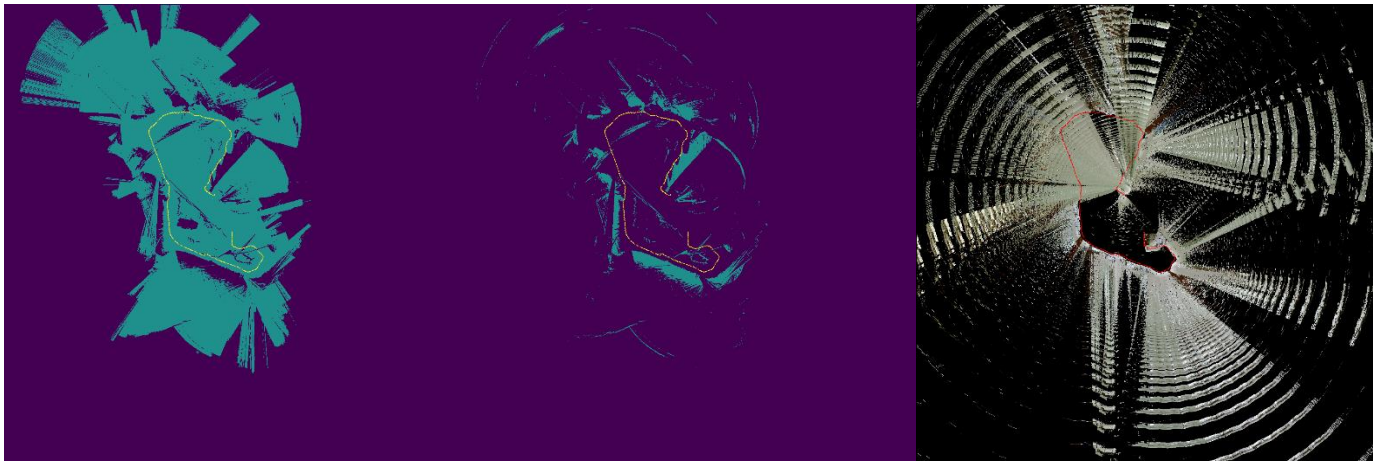


**RGB_4, DEPTH_4, train_joint4, train_lidar4**

Iteration 25000



Final iteration ~27400



## V. Conclusion

This project successfully implements Particle Filter SLAM for a humanoid robot equipped with cameras, odometry sensors, and Lidar. It is an effective technique for mapping an environment. Although this implementation reads pre-collected data, the algorithm could easily be tweaked to receive live data as the robot moves around so it can see obstacles and react. The texture mapping is inaccurate, but it does not affect mapping reliability since it is a purely visual addition. Future works would include fixing the transforms so RGB values are correctly mapped to the ground plane.

## VI. References

Fong, Simon, RGB 0 | Image Number Caption | Unflipped | Timestamp
https://www.youtube.com/watch?v=BnxvbZ943w4&feature=youtu.be

Fong, Simon, RGB 3 | Image Number Caption | Unflipped | Timestamp
https://www.youtube.com/watch?v=qAKAOcZkKYc&feature=youtu.be

Fong, Simon, RGB 4 | Image Number Caption | Unflipped | Timestamp
https://www.youtube.com/watch?v=l8oJmmFrPJU&feature=youtu.be