ECE 276A Project 3: SLAM

Brian Wilcox, University of California at San Diego, MS, Electrical and Computer Engineering
bpwilcox@ucsd.edu

I. Introduction

There has been great interest and progress in the robotics field to use robots (typically mobile) to explore and navigate the environment. From search and rescue, mapping a building, or finding an object, a variety of interesting tasks require that the robot know where it is in a given environment, while other tasks may desire the robot to produce a map of the area it has explored. Often it's the case that robots need to simultaneously locate where it is in an environment while also mapping it. This is the problem of SLAM (simultaneous localization and mapping). In this project, we use grid-based SLAM with particle filters to predict and update the state of the robot and an occupancy grid of the environment map.

II. Problem Formulation

A. Simultaneous Localization and Mapping

Given observations from a lidar sensor $\mathcal{Z} = \{z_1, z_2, ..., z_n\}$, odometry information, $\mathcal{O} = \{o_1, o_2, ..., o_n\}$, and configurations (or composition of configurations) of the lidar relative to the robot center of mass $\mathcal{C} = \{C_1, C_2, ..., C_n\}$, we aim to simultaneously localize the robot center of mass, $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, and update a grid-map of the environment $\mathcal{M} = \{m_1, m_2, ..., m_n\}$ such that

$$x_{k+1}, m_{k+1} = SLAM(x_k, m_k, z_{k+1}, o_k, o_{k+1}, C_{k+1})$$

i.e. find the next estimated state location and map given the previous estimates and the current sensor information.

B. Texture Mapping

Given the localized robot poses, $\mathcal{X}$, and the updated grid map, $\mathcal{M}$, produced from SLAM, RGB camera images $\mathcal{I} = \{I_1, I_2, ..., I_n\}$, and depth camera images $\mathcal{D} = \{D_1, D_2, ..., D_n\}$, we aim to generate a color texture map of the ground plane in $\mathcal{M}$, such that

i.e. update the current texture map with the next robot pose and RGB-D images.

III. Technical Approach

In this project, we performed SLAM using occupancy grids for representation of the environment map and particle filters for localization of the robot pose in the environment. Lidar scans and odometry information allowed for successful implementation of the SLAM algorithm After the pose and map are obtained, RGB-D images from a Kinect sensor are used to find the ground plane and color the map.

A. Robot Configurations

In order to facilitate the computations of the robot pose, which is measured in the body frame, we use the configuration of the sensors with respect to the different frames of the robot. The lidar sensor is located above the head, where a homogeneous transformation from lidar to body frame can be found as

$$_bT_l = {}_bT_h * {}_hT_l$$

$${}_bT_h{}_hT_l = \begin{bmatrix} R_z(neck) * R_y(head) & p_{head} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3X3} & p_{lidar} \\ 0 & 1 \end{bmatrix}$$

where we know the relative position of the head from the body center of mass p_{head},the neck and head angle rotations(as a function of time) of the robot, and the relative position of the lidar from the head, p_{lidar}. The pose of the robot's body center of mass at a time t, the state which we are localizing, is $\mathbf{x}_t = [x_t, y_t, \theta_t]^T$ where we can also define a homogenous transformation as such:

$$_wT_b(t) = \begin{bmatrix} R_{pose}(t) & p_{pose}(t) \\ 0 & 1 \end{bmatrix}$$

where $p_{pose}(t) = [x_t, y_t, h_{cm}]^T$ and $R_{pose}(t) = R_z(\theta_t)$.

B. Grid SLAM with Particle Filter

In SLAM, we recursively solve for the robot's location and the current estimate of the environment map. For our particle filter model, we begin with an initial estimate of the N particles,

$$\mu_{0|0}^{(i)} = [0, 0, 0]^T$$

and weights

$$\alpha_{0|0}^{(i)} = \frac{1}{N}$$

for all $i = 1, 2, ...N$. We also begin by initializing a log-odds map that corresponds to the occupancy grid with a size (M,M)

$$\lambda(m)_{ij} = 0$$

$foralli, j$. The grid map size (M,M) is determined by a range $[-x : x, -y : y]$ scaling with a resolution res which covers the physical size of the occupancy grid map.

1) Prediction: In the prediction step, we use the odometry information from the lidar $\mathcal{O} = \{o_1, o_2, ..., o_n\}$ to predict the next pose for each of the N particles. Here we note that we transform all the particles using the transformation matrix

$$\mu_{t|t}^{(i)} \rightarrow {}_wT_b^{(i)}(t)$$

For prediction we use an odometry motion model with noise $T_{noi}^{(i)}$ sampled from a Gaussian distribution such that

$$T_{noi}^{(i)} = \begin{bmatrix} R_z(\sigma_\theta^2) & \begin{pmatrix} \sigma_x^2 \\ \sigma_y^2 \\ 0 \end{pmatrix} \\ 0 & 1 \end{bmatrix}$$

The odometry model for the prediction goes as follows

$${}_wT_b^{(i)}(t+1) = {}_wT_b^{(i)}(t) * T_{noi}^{(i)} * ({}_bT_l * {}_wT_l(t)^{-1} * {}_wT_l(t+1) * {}_bT_l^{-1})$$

$$\mu_{t+1|t}^{(i)} \leftarrow {}_wT_b^{(i)}(t+1)$$

where the ${}_wT_l(t)$ and ${}_wT_l(t+1)$ transformations come from the odometry observations of the current and next time step. Not that we must transform our odometry states which were lidar to world frame to the body frame via ${}_bT_l$. Now, these predictions can be passed to the update step.

2) Update: In the update step, we aim to perform scan-matching, to correlate our current map to the lidar scan data. We first must remove scan rays which are either too close or too far such that

$$z_t = [z_t | z_t > 0.1 \text{ and } z_t < 30]$$

We can convert the scans (1x1081)into polar coordinates by using

$$\theta_i = (\frac{270}{1080} * i - 135)\frac{\pi}{180}$$

and then convert into Cartesian coordinates via

$$z_x^i = z_t \cos\theta_i$$

$$z_y^i = z_t \sin\theta_i$$

$$z_z^i = 0$$

We'd like to convert these scan coordinates into the world frame by performing the transformation:

$$z_t^{world} = {}_wT_b * {}_bT_l * \begin{bmatrix} z_x^{1:n} \\ z_y^{1:n} \\ 0 \\ 1 \end{bmatrix}$$

Now that the scans are in the world frame, we remove the ground plan by keeping the scans which are above a threshold in the z direction.

$$z_t^{world} = [z_t^{world} | z_t^{world}[z] > 0.1]$$

We then want to consider a 9x9 grid of values correlations around the center scan so that we can find the particle position with the maximum correlation. To simplify notation and methodology, we'll just call this 9x9 grid a transformation such that

$$z_t^{world} \rightarrow z_t^{'world}$$

Since we have our current log-odd map of size (M,M), we want to threshold the map to form our occupancy grid map such that

$$m_{ij} = \begin{cases} 1, & \lambda(m)_{ij} > 0 \\ 0, & \lambda(m)_{ij} < 0 \end{cases}$$

We can then pass our occupancy grid and transformed current lidar scan in the world frame to obtain the map correlation. We'll note that the output is a correlation over the 9x9 grid around the center scan, therefore we choose the best one (max) out of the grid $corr_{max}$ as well as find the transformation shift to that best position T_{corr}.

$$(corr_{max}, T_{corr}) \leftarrow corr(z_t^{'world}, m)$$

Each of the N particle states are updated by this shift such that

$$\mu_{t+1|t+1}^{(i)} \leftarrow (T_{corr}^{(i)} * {}_wT_b^{(i)}(t+1))$$

Now, Using the map correlation between the current map and the lidar scan, we update the N particle weights by defining the laser scan measurement likelihood in terms of the correlation with the occupancy grid as follows

$$p_h(z_t|x, m) = \frac{exp(corr(z_t^{'world}, m))}{\sum_z exp(corr(z_t^{'world}, m))}$$

and then updating the weights via

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

Now that we have both updated particles and weights, we will select the best particle that will be passed on to the mapping step as the particle having the largest weight.

$$\mathbf{x}_{t+1} = \mu_{t+1|t+1}^{*} = \mu_{t+1|t+1}^{\text{argmax}_i \alpha_{t+1|t+1}}$$

3) Mapping: In this step, we aim to update the current log-odds map which corresponds to the occupancy grid map of the environment.

Like in the update step, we take the lidar scans transformed into the world coordinate. In order to update the grid, however, we must discretize the lidar ray scans into cells which correspond with the size and resolution of our occupancy grid.

$$zcell \leftarrow \ getcell(z_t^{world})$$

Likewise, we take the current best particle state and transform it into the same frame as the lidar to world frame and obtain its corresponding cell in the grid,

$$\mathbf{x}_{t+1} \rightarrow {}_wT_b(t+1)$$

$$\mathbf{x}_{t+1}^{'} \leftarrow {}_wT_b(t+1) * {}_bT_l$$

$$xcell \leftarrow \ getcell(\mathbf{x}_{t+1}^{'})$$

Given we have the discretized cells of the occupied and free lidar scans and the position of the robot center of mass, we can use Bresenham's line algorithm to find the cells in our occupancy grid which are occupied and free:

$$(occupied, free) = bresenham2D(xcell, zcell)$$

With the known locations of free and occupied cells in our map, we now can update the log-odds of or map such that

$$\lambda(m_{t+1}^{free}) = \lambda(m_t^{free}) + \log g(0)$$
$$\lambda(m_{t+1}^{occ}) = \lambda(m_t^{occ}) + \log g(1)$$

where

$$g(1) = \frac{p_h(z|m_i = 1)}{p_h(z|m_i = 0)}$$

This is a simple observation model for the lidar, specifying our trust in its measurements. Here we use g(1) = 9 and g(0) = 1/9. Now our map has been updated for the next time-step.

4) Resampling: Because we want to avoid particle depletion, where most of the particle weights are close to zero, we exercise resampling if the effective number of particles is below a certain threshold. This measure of effective particles, N_{eff}, is defined as

$$N_{eff} := \frac{1}{\sum_{k=1}^{N_{t|t}} (\alpha_{t|t}^k)^2}$$

If this N_{eff} goes below a selected threshold, then we choose to resample the particles. In this project we used the Stratified resamplng procedure.

C. Texture Mapping

Given a localization of our robot pose, we can find a transformation from our camera images to obtain the pixel colors of the ground plane. The basic approach given RGB-D images is as follows: Transform depth images to Cartesian frame

$$D_t \rightarrow D_t^{xyz}$$

Transform from cartesian to the camera rgb frame

$$D_t^{xyz} \rightarrow D_t^{rgb}$$

Transform from the camera frane ti the optical frame

$$D_t^{rgb} \rightarrow D_t^o$$

Transform from the optical frame to the world frame using the current best robot pose $\mathbf{x}_t$

$$D_t^o \rightarrow D_t^w$$

Isolate the ground floor:

$$D_t^w = [D_t^w | D_t^w[z] < 0.1]$$

Transform back to the optical frame

$$D_t^W \rightarrow D_t^o$$

Use calibration matrix (given from data) and canonical projection to transform back to pixels.

$$D_t^W \rightarrow D_t^P$$

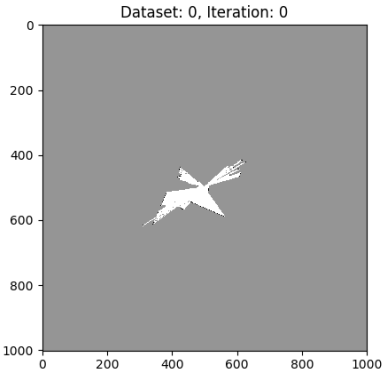
Now find the points in the RGB Image the points that correspond to the ground plane. Set these points into RGB points in a texture map m_t. With a texture map m_t, we can then overlay the pixel values over the occupancy grid found with SLAM.

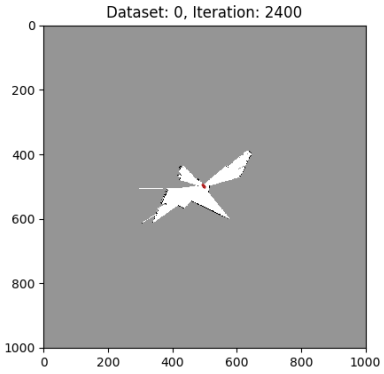
IV. Results

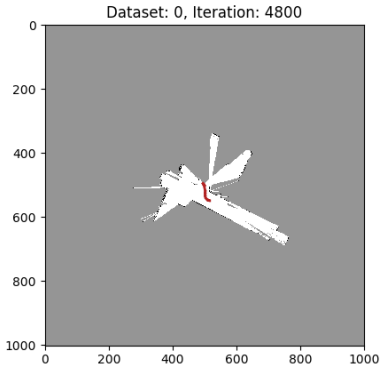
In this section we present results for 4 training datasets and 1 test set for SLAM and Texture Mapping. The number of particles used for all datasets was N = 25, and the map grid-size was (1001,1001) with a resolution of either 0.05m or 0.06m. The update step mapping step was performed ever 100 iterations through the dataset.

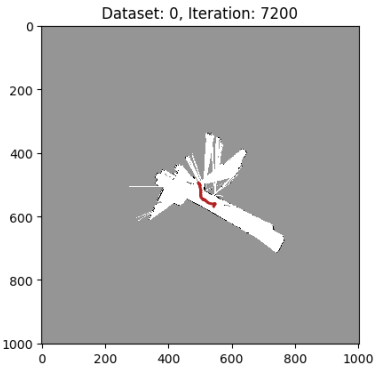
A. Training Results - SLAM

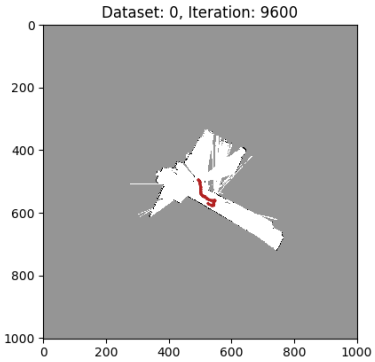
Below are the results for SLAM with the training datasets. Plots show the progression of each dataset over time for select iterations.

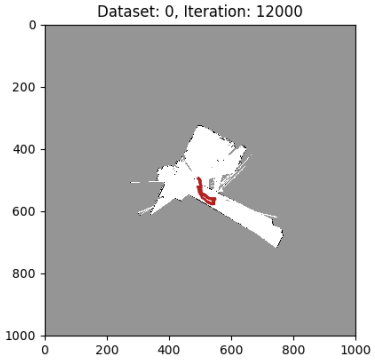












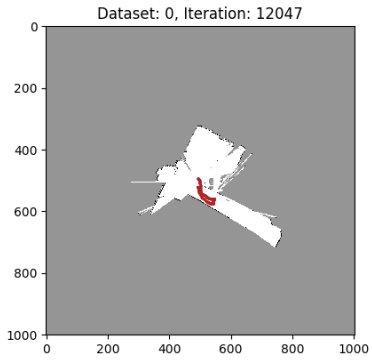
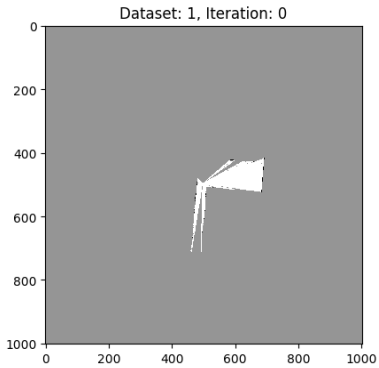
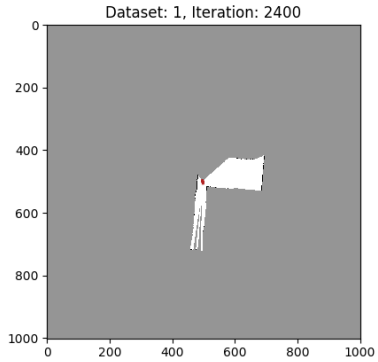
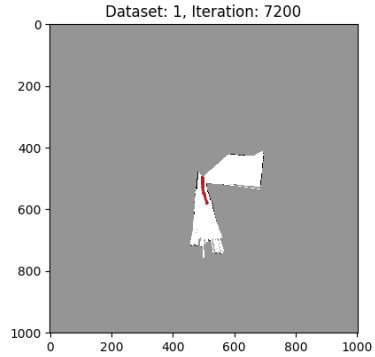
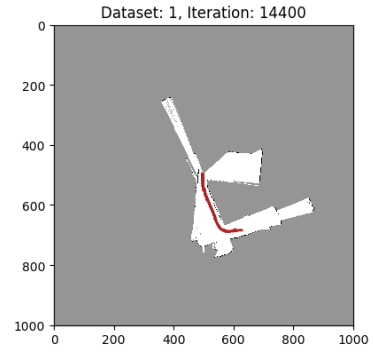


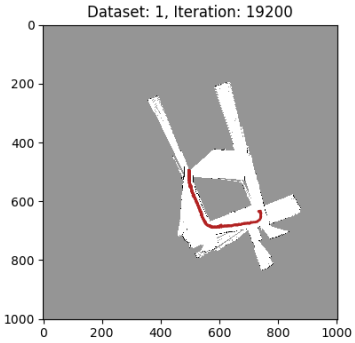
Fig. 1: SLAM Results: Dataset 0

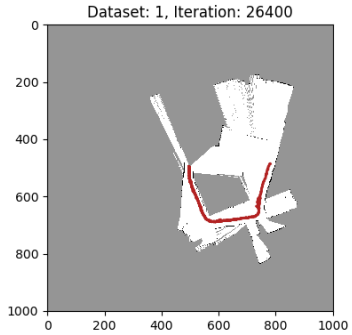


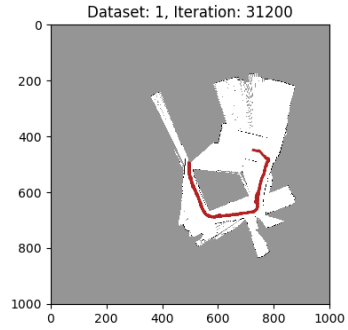












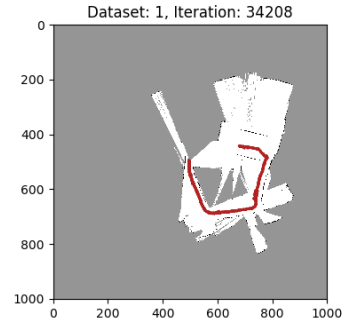
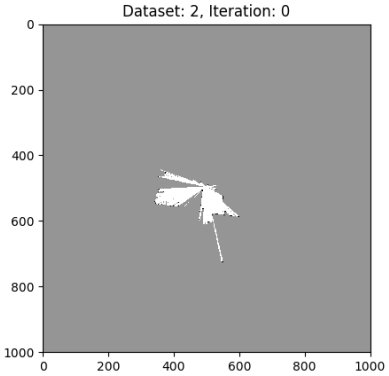
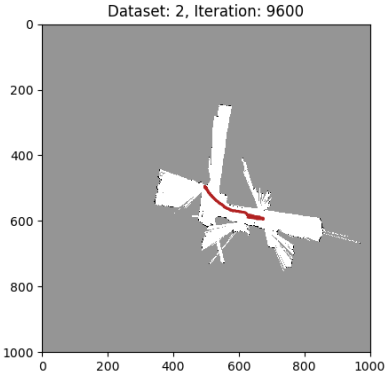
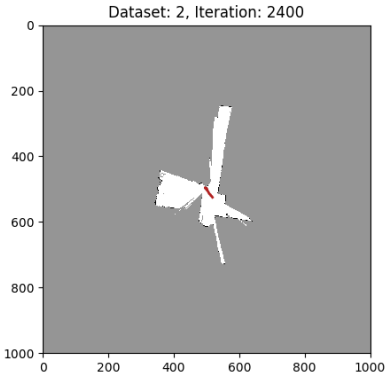
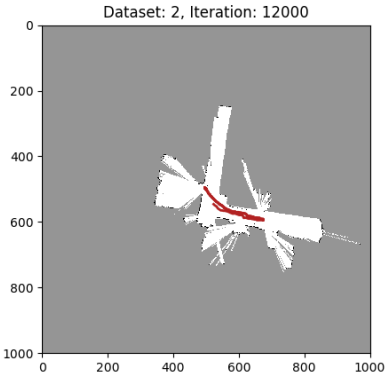


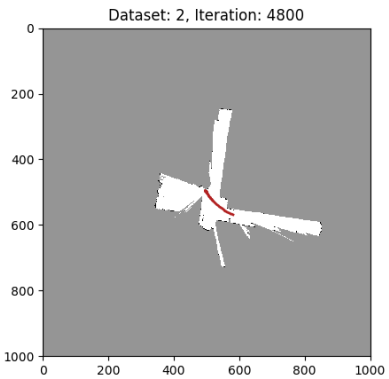
Fig. 2: SLAM Results: Dataset 1

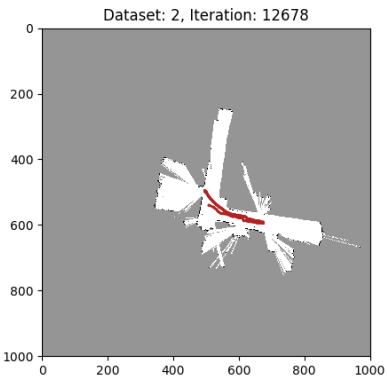












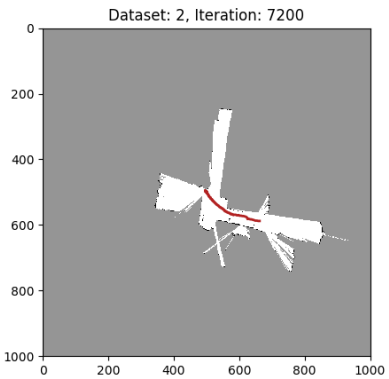
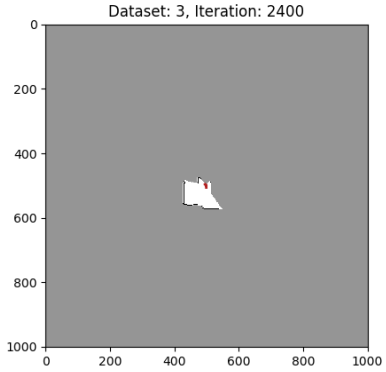
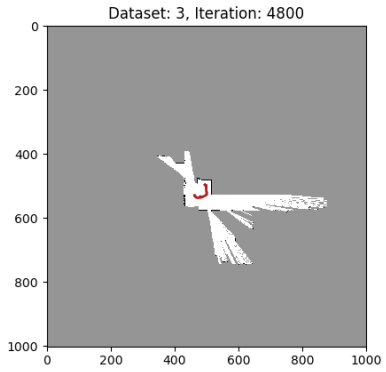
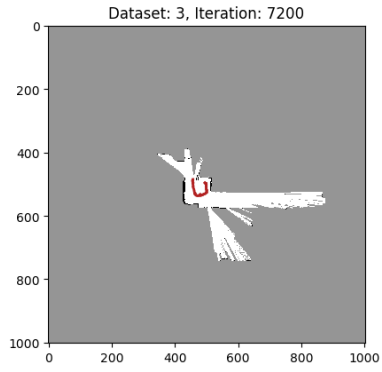
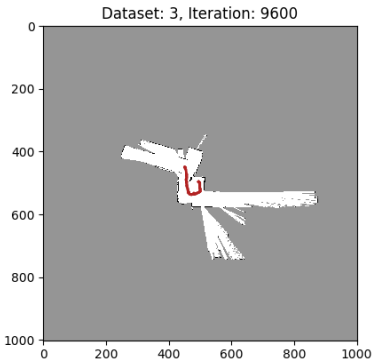


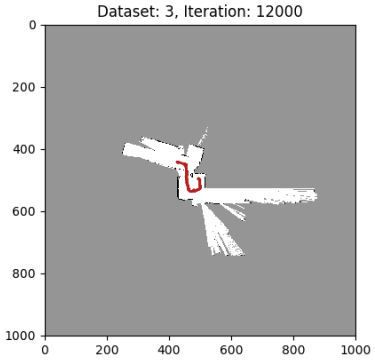
Fig. 3: SLAM Results: Dataset 2











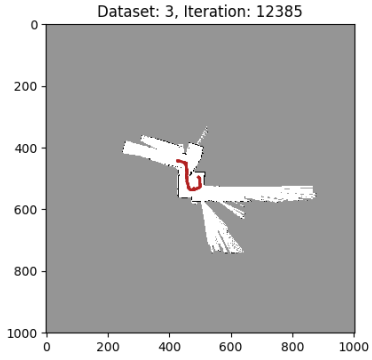
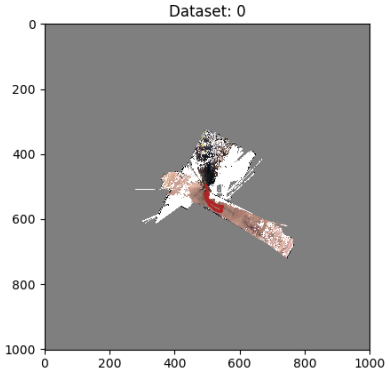


Fig. 4: SLAM Results: Dataset 3

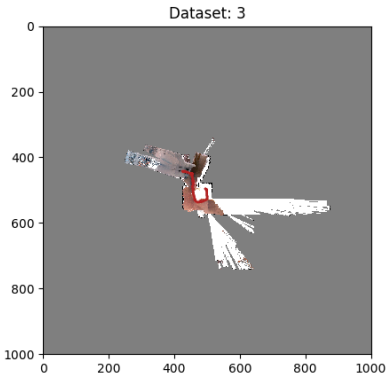
Overall, the maps generated over time for each dataset appear to develop quite well and the localization appears reasonable given the map.

B. Training Results - Texture Mapping

The following figure are the texture maps overlaying the occupancy grids for the two available training datasets with RGB-D Images.

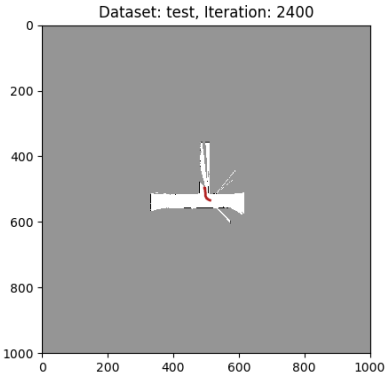


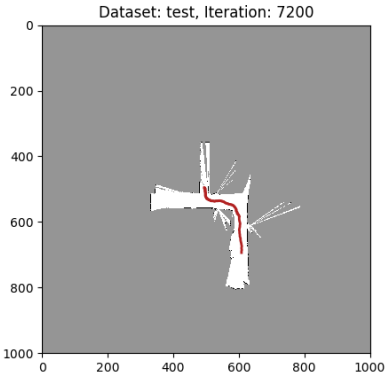
Texture Mapping Results: Dataset 0

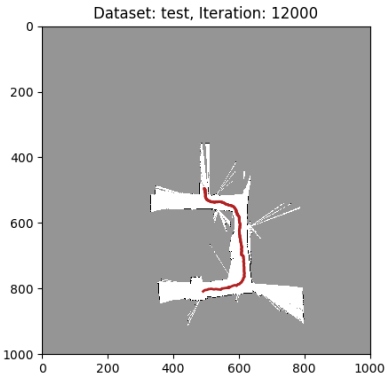


Texture Mapping Results: Dataset 3

The texture maps for the training datasets actually looks fairly reasonable.







C. Test Results - SLAM

In this section we present the SLAM results for the test dataset

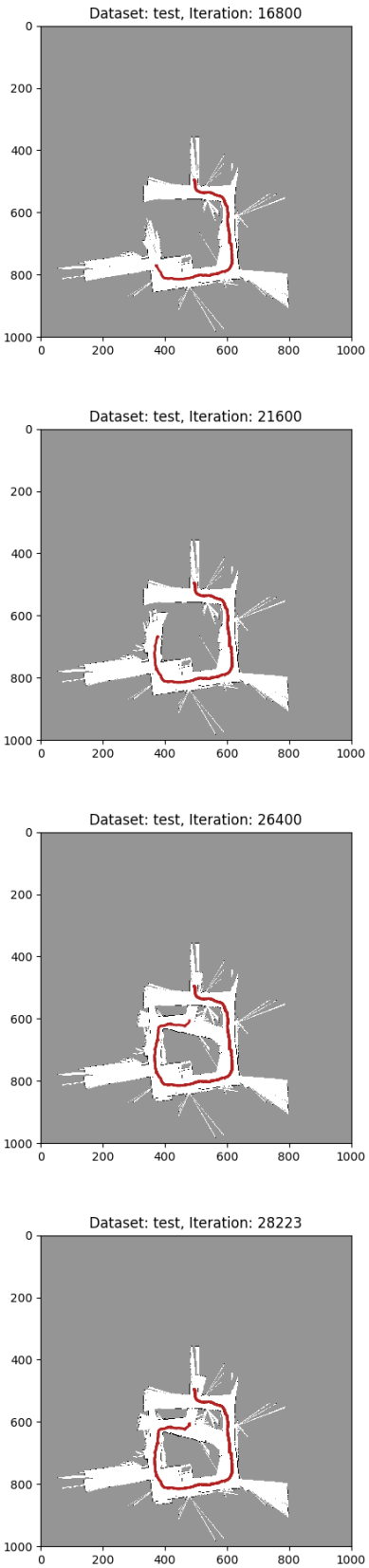
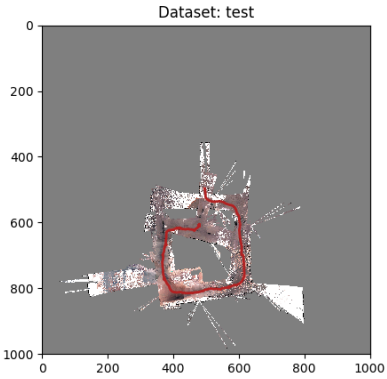


Fig. 5: SLAM Results: Dataset test

As can be seen, the map develops very well for the majority through the test, though towards the end it appears to be making a loop and it has trouble with localizing itself upon closing the loop. Though the map still performs well, perhaps using the yaw corrections from the IMU would help in this case.

D. Test Results - Texture Mapping



Texture Mapping Results: Test

The texture map in the test set covers very well the ground plane, though again the closing of the loop seems to be a struggle.