Simultaneous Localization And Mapping using Particle Filter

Surya Lakshmi Subba Rao Pilla
Electrical and Computer Engineering
Course: Sensing and Estimation in Robotics

I. Introduction

SLAM (Simultaneous Localization and Mapping) is an essential technique in robotics that allows robots to create maps of unknown environments and simultaneously determine their position within these maps. SLAM enables robots to navigate autonomously in unknown environments, create maps of their surroundings, determine their position with high accuracy, handle noisy sensor data, uncertainties, and errors, and achieve real-time performance. The objective of this project is to implement SLAM using IMU odemetry, encoder, 2-D LiDAR scans, and RGBD measurements from a Differential drive robot. The odometry and LiDar measurements are used to obtain the occupancy grid and localize the robot. RGBD images are used to assign colors to the 2-D map of the floor.

II. Problem Formulation

The goal of this project is to implement simultaneous localization and mapping (SLAM) using encoder and IMU odometry, 2-D LiDAR scans, and RGBD measurements from a differential-drive robot. There are two parts of the project:

1) **Particle Filter SLAM:** Apply particle filter with a differential-drive motion model and scan-grid correlation observation model for simultaneous localization and occupancy-grid mapping. Particle filter is a special case of Bayes filter in which probability density functions for prediction and update step are discrete distributions with N possible values called particles.

   Bayes Filter:: The Bayes filter keeps track of predicted pdf and updated pdf using Markov assumptions (Current state x_{t+1} depends only on the previous state x_t and previous input u_t), conditional probability and Bayes Rule. The prediction and update keeps track of pdfs given below:

   $$\text{Prediction pdf: } p_{t+1|t}(x_{t+1}) := p(x_{t+1}|z_{0:t}, u_{0:t}) \quad (1)$$

   $$\text{Update pdf: } p_{t+1|t+1}(x_{t+1}) := p(x_{t+1}|z_{0:t+1}, u_{0:t}) \quad (2)$$

   Prediction Step:

   $$p_{t+1|t}(x_{t+1}) := \int p_f(x|s, u_t) p_{t|t}(s) ds \quad (3)$$

   Update Step:

   $$p_{t+1|t+1}(x) := \frac{p_h(z_{t+1}|x) p_{t+1|t}(x)}{\int p_h(z_{t+1}|s) p_{t+1|t}(s) ds} \quad (4)$$

   Particle Filter:: The probability mass function $\alpha[1]$,...,$\alpha[N]$ over N values $\mu[1]$,...,$\mu[N]$ can be viewed as a continuous-space probability density function.

   $$p(x) = \sum_{k=1}^{N} \alpha[k] \delta(x - \mu[k]) \quad (5)$$

   where δ is a Dirax delta function.
   The particle filter used thes particles with locations $\mu[k]$ and weights $\alpha[k]$ for k = 1,...,N to represent the pdfs of $p_{t|t}$ and $p_{t+1|t}$:

   $$p_{t|t}(x_t) := \sum_{k=1}^{N} \alpha_{t|t}[k] \delta(x_t - \mu_{t|t}[k]) \quad (6)$$

   $$p_{t+1|t}(x_{t+1}) := \sum_{k=1}^{N} \alpha_{t+1|t}[k] \delta(x_{t+1} - \mu_{t+1|t}[k]) \quad (7)$$

   These pdfs are substituted in Bayes filter prediction and update steps to get:

   Prediction Step:

   $$p_{t+1|t}(x_{t+1}) = \sum_{k=1}^{N} \alpha_{t|t}[k] \delta(x_{t+1} - \mu_{t+1|t}[k]) \quad (8)$$

   Update Step:

   $$p_{t+1|t+1}(x_{t+1}) := \sum_{k=1}^{N} \alpha_{t+1|t+1}[k] \delta(x_{t+1} - \mu_{t+1|t+1}[k]) \quad (9)$$

2) **Texture Map:** The objective of this part is to use RGBD images and the estimated robot trajectory to produce a 2D color map of the floor surface. The process involves acquiring the depth of each RGB pixel and subsequently mapping the colored points, which are located near the floor, from the depth camera frame to the world frame.
   Estimating depth and pixel location:: The depth and pixel locations are obtained by the below equations: Given d at pixel (i,j):

   $$\begin{aligned} dd &= (-0.00304d + 3.31) \\ depth &= \frac{1.03}{dd} \\ rgbi &= \frac{526.37i - 4.5 \times 1750.46dd + 19276.0}{585.051} \\ rbgj &= \frac{526.37j + 16662}{585.051} \end{aligned} \quad (10)$$

Once the depth and pixel locations are obtained these coordinates are converted to body frame and then to the world frame to extract the color from image that are close to the floor and add texture to the map.

III. Technical Approach

Achieving sensor fusion is the foremost and critical aspect of the data processing workflow. The data obtained from sensors is unsynchronized, which necessitates synchronization. To this end, encoder data with a lower sampling rate is included in the dataset, enabling synchronization of all sensor readings with respect to encoder timestamps. Once the synchronization process is complete, subsequent steps are carried out on the synchronized data.

A. *Occupancy grid map generation:*

The robot moves around the environment and scan the environemnt using LiDAR. The given LiDAR has a field of view 270°. The pose of the robot obtained from IMU data and encoder data provides the orientation of robot at time t. The distance travelled in τ_t for a given encoder count z_t, wheel diameter d, and 360 ticks per revolution is:

$$\tau_t v_t \approx \frac{\pi d z_t}{360} \tag{11}$$

Motion Model of Differential-drive robot*:*

$$f_d(x_t, u_t) = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t + \tau_t v_t \cos(\theta_t) \\ y_t + \tau_t v_t \sin(\theta_t) \\ \theta_t + \tau_t \omega_t \end{pmatrix} \tag{13}$$

The LiDAR scan points are converted to the world coordinates using rotation matrix R_t.

$$R_t = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & x_t \\ \sin\theta_t & \cos\theta_t & y_t \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

The given problem is initialized with initial states and angles to be zero and then at each time stamp the pose angle is updated and LiDAR scans are used to populate the grid using the below mentioned occupancy grid mapping technique.

The occupancy grid map is obtained probabilistically using Log-odds occupancy.
Assuming that z_t indicates whether m_i is occupied or not, the log-odds ratio $\Delta\lambda_{i,t}$ of the inverse measurement model specifies the measurement "trust". Considering 80% correct sensor:

$$\begin{aligned} \Delta\lambda_{i,t} &= \log \frac{p(m_i = 1|z_t, x_t)}{p(m_i = -1|z_t, x_t)} \\ &= \begin{cases} +\log 4 & \text{if } z_t \text{ indicates } m_i \text{ is occupied} \\ -\log 4 & \text{if } z_t \text{ indicates } m_i \text{ is free} \end{cases} \end{aligned} \tag{15}$$

The map pmf is later recoverd by logistic sigmoid function:

$$\gamma_{i,t} = p(m_i = 1|z_{0:t}, x_{0:t}) = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})} \tag{16}$$

By applying log-odds occupancy technique at each time stamp, occupancy grid is obtained as discussed in results section. Next step is to apply particle filter for localization.

B. *Applying Particle filter prediction step:*

The differential-drive motion model predicts the motion of a robot using the linear velocity obtained from encoders and the yaw rate obtained from the IMU.
Each particle $\mu_{t|t}[k] \in \mathbb{R}^3$ represents a possible 2-D position (x,y) and orientation θ.

$$\mu_{t+1|t}[k] = f(\mu_{t|t}[k], u_t + \epsilon_t) \tag{17}$$

$$\alpha_{t+1|t}[k] = \alpha_{t|t}[k] \tag{18}$$

f(x,u): the differential-drive model model
u_t = (v_t, ω_t): linear and angular velocity
ϵ_t is the 2-d gaussian motion noise

$$\begin{bmatrix} \epsilon \mathbf{v}_t \\ \epsilon \boldsymbol{\omega}_t \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \sigma_v^2 & \mathbf{0} \\ \mathbf{0} & \sigma_\omega^2 \end{bmatrix} \right) \tag{19}$$

The u_t is obtained from IMU sensor(ω_t) using yaw angle measurement and encoder (v_t). In the problem, N random particles are selected with initial pose and orientation as zero and at each timestamp the pose of the particles is updates by adding random noise to velocity and angle as discussed above. This is only a prediction step and there is no estimation of best particle. The next step is to update the pose of the trajectory with best particle.

C. *Applying Particle filter update step:*

In the update step the pose remains unchanged but the weights are scaled by the observation model:

$$\mu_{t+1|t+1}[k] = \mu_{t+1|t} \tag{20}$$

$$\alpha_{t+1|t+1}[k] \propto p_h(z_{t+1}|\mu_{t+1|t}[k], m)\alpha_{t+1|t}[k] \tag{21}$$

The update step observes the environment at that time instance and updates the weights of the particles based on the correlation to the map obtained till previous state.
LiDAR correlation model: likelihood model $p_h(z|x, m)$ for LiDAR scan z obtained from sensor pose x in occupancy grid m. LiDAR scan is converted to world frame y=r(z,x) using rotation matrix, and this is used to find correlation with the previous map.

$$p_h(z|x, m) \propto corr(r(z, x), m) \tag{22}$$

The correlation for all the N particles is determined and the initial weights are updated using the new correlation. In the given problem, the weights are initialized with equal probability. After each time instance the weights are updated

based on the correlation. After several iterations the weights of few particle that are unimportant go down and at a certain threshold the re-sampling technique is applied to remove the points with low weights and add new points to maintain the number of particles.

Resampling: If the effective number of particles $N_{eff} := \frac{1}{\sum_{k=1}^{N}(\alpha_{t|t}[k])^2}$ is less than a threshold then resampling is done. In the implementation a normal resampling is done which is to select random particles with the distribution based on weights and then updating the weights to have equal probability for the new particles.

The final trajectory obtained is a better representative of robot movement as the update step corrects the path over time by correlating the observation at each time stamp to the previous map. The results are discussed in the results sections.

D. *Texture Mapping:*

The second part of the project is to utilize the trajectory obtained and the data from RGBD camera to add floor color to the map. As discussed in the problem formulation part the depth and the pixel coordinates of the image are obtained using the disparity image. Further these pixels need to be converted to the sensor frame and then to the world frame.

Conversion of pixels to sensor frame:

- u, v are the image coordinates in pixels
- fsu, fsv are the focal lengths in pixels
- cu, cv are the principal point coordinates in pixels
- s_θ is the skew factor that scales non-rectangular pixels
- z is the distance from the camera to the object in millimeters
- $X_\circ, Y_\circ, Z_\circ$ are the object coordinates in millimeters

In the problem statement the u,v pixels are given and : $X_\circ, Y_\circ, Z_\circ$ need to be computed using below relation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fs_u & fs_\theta & c_u \\ 0 & fs_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \frac{1}{Z_\circ} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_\circ \\ Y_\circ \\ Z_\circ \\ 1 \end{bmatrix} \tag{23}$$

$$\begin{bmatrix} X_{sensor} \\ Y_{sensor} \\ Z_{sensor} \end{bmatrix} = \begin{bmatrix} X_\circ \\ Y_\circ \\ Z_\circ \end{bmatrix} + \begin{bmatrix} 0.18 \\ 0.005 \\ 0.36 \end{bmatrix} \tag{24}$$

Conversion of sensor coordinates to world coordinates: This is done by multiplying the transformation matrix involving pose and orientation obtained from the current state of the robot after the update step.

$$R_t = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 & x_t \\ \sin\theta_t & \cos\theta_t & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} X_{world} \\ Y_{world} \\ Z_{world} \\ 1 \end{bmatrix} = R_t \begin{bmatrix} X_{sensor} \\ Y_{sensor} \\ Z_{sensor} \\ 1 \end{bmatrix} \tag{26}$$

IV. Results

A. *Trajectory:*

The trajectory is obtained from dead reckoning and their evolution over time is shown in Figure 1 and Figure 2. The Figure 4,5 and Figure 5,6 compare the trajectories with dead reckoning only and with update step of particle filter. For bettere visibility the plot show only 3 points but the particle filter in the code uses 100 particles for the best trajectory.

Link to GIF is given here:
https://drive.google.com/drive/folders/184UBGCF0dGuUFG5Er9vUILGCMJ1xsqk2?usp=sharing

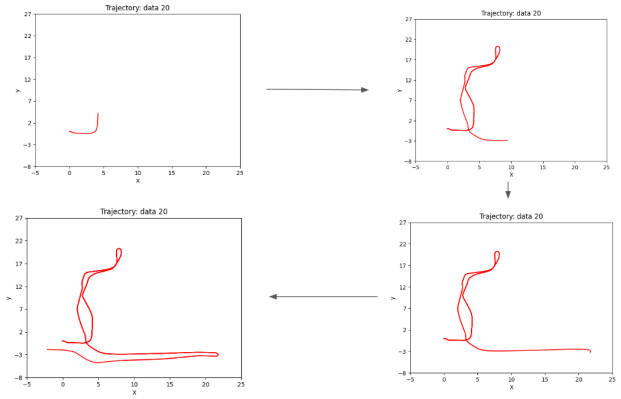

Figure 1: Evolution of trajectory of dataset 20

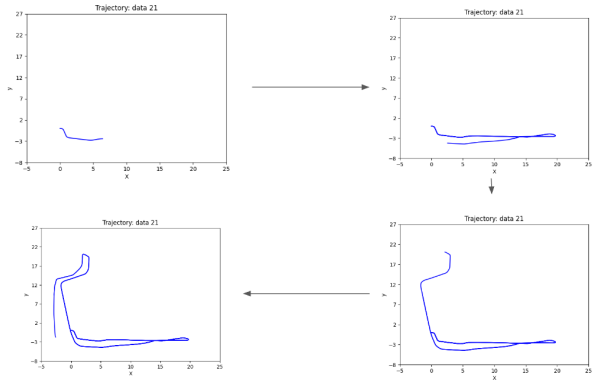

Figure 2: Evolution of trajectory of dataset 21

B. *Occupany Map:*

The occupancy map of dataset 20 and 21 have been shown in figure 7,8,9 and 10. These occupancy maps correspond to the best trajectories obtained through particle filter from 100 points and 3 points. It can be observed that 100 point particle filter map has better boundaries compared to the 3 point particle filter.

C. *Texture Map:*

The texture map of dataset 20 and 21 have been shown in figure 10, 11 respectively. These texture maps correspond to the best trajectories obtained through particle filter from 100 points and resampling.

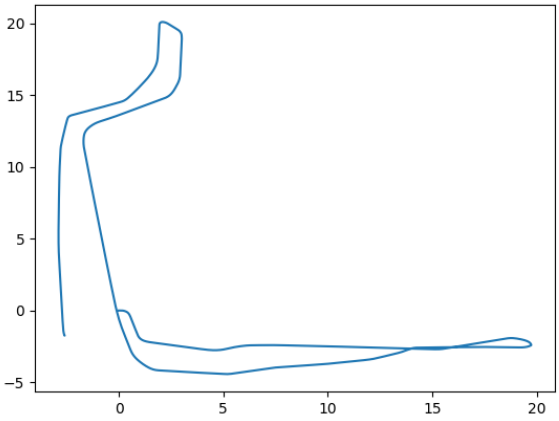

Figure 3: Trajectory of dead reckoning for dataset 21

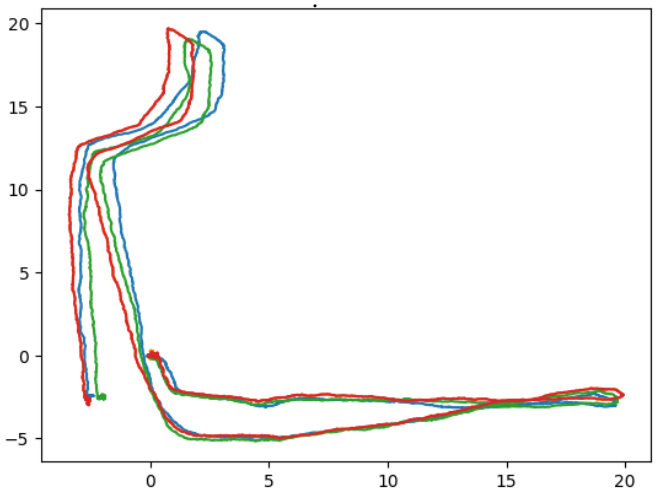

Figure 4: Trajectory of 3 points with resampling for dataset 21

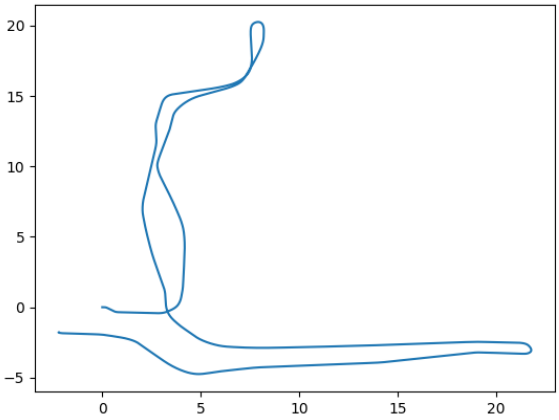

Figure 5: Trajectory of dead reckoning for dataset 20

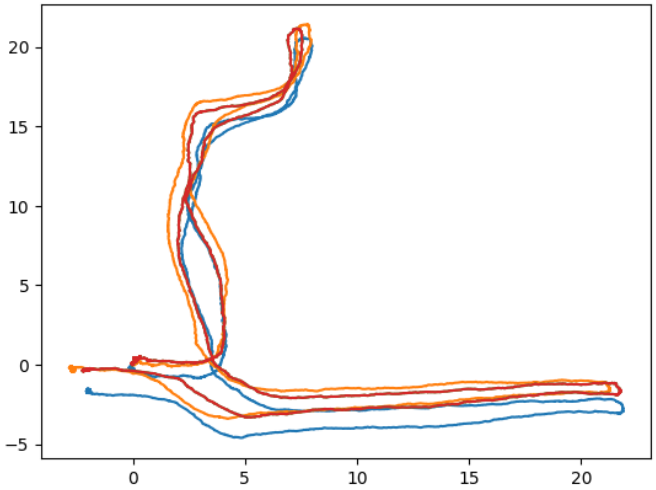

Figure 6: Trajectory of 3 points with resampling for dataset 20

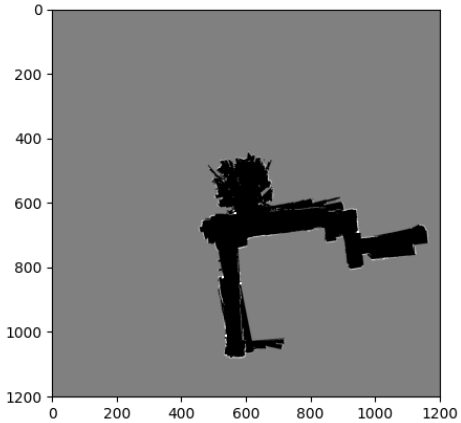

Figure 7: Occupancy map of dataset 20 with 3 points

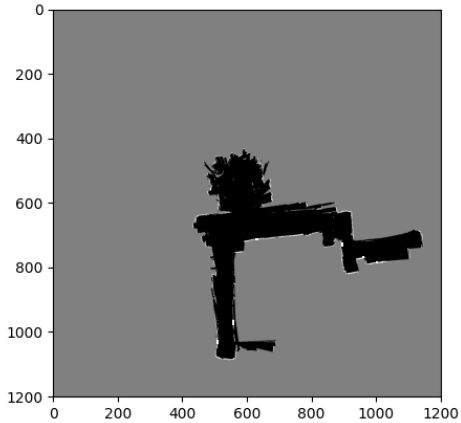

Figure 8: Occupancy map of dataset 20 with 100 points

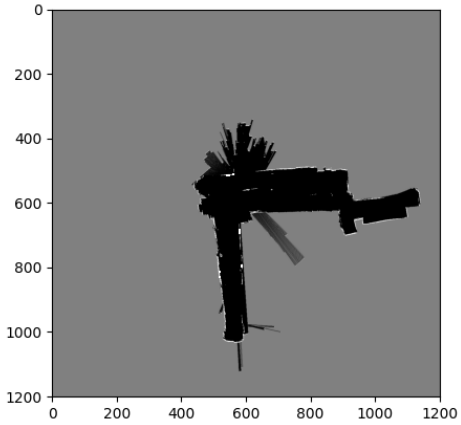

Figure 9: Occupancy map of dataset 21 with 3 points

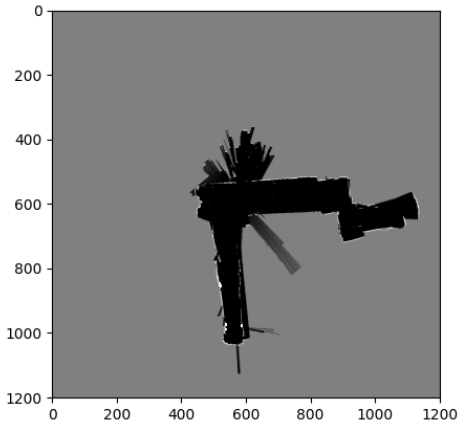

Figure 10: Occupancy map of dataset 21 with 100 points

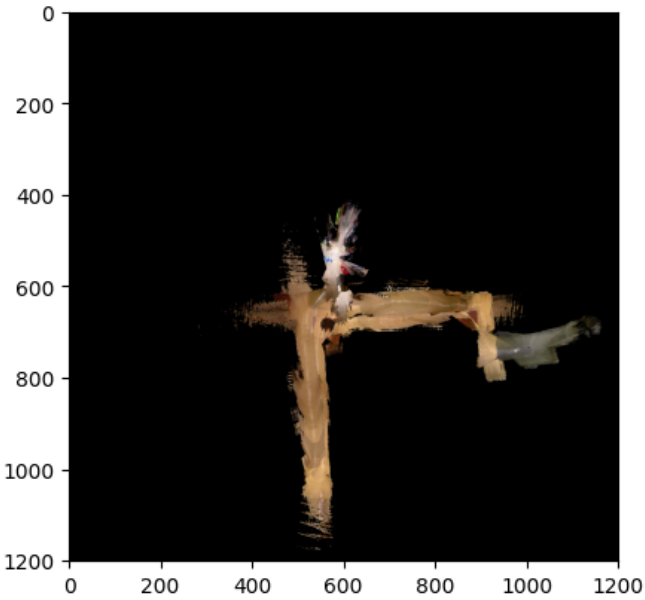

Figure 11: Texture Map of dataset 20

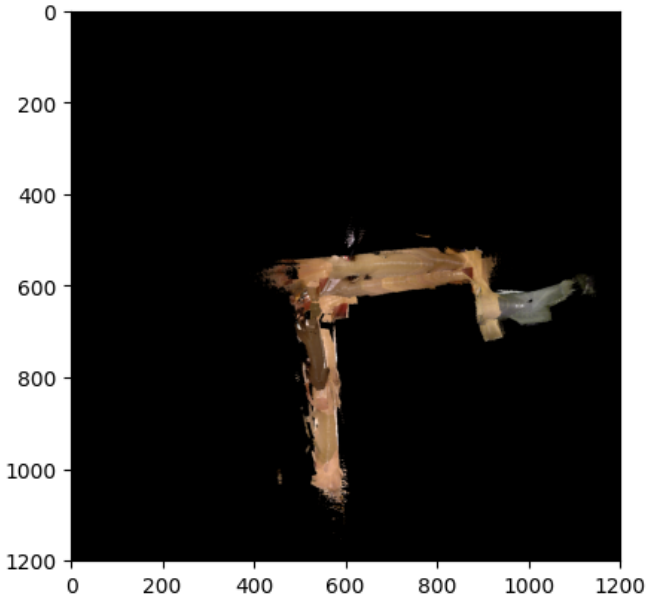

Figure 12: Texture Map of dataset 21

.