Particle Filter SLAM

Narayanan Elavathur Ranganatha
University of California San Diego
nelavathurranganatha@ucsd.edu

***Abstract*—The following report formulates the problem of *Simultaneous Localization and Mapping (SLAM)* using data provided by a Encoder, Inertial Measurement Unit (IMU), 2-D LiDAR scans from a Hokuyo UTM-30LX. The algorithm used for the above is the Particle Filter. The location estimated by this pipeline is then used to create a texture map of the scene by extracting the floor via height thresholding from RGB-D measurements provided by a Kinect Sensor.**

I. Introduction

A. Problem Statement

The objective of the problem is to simultaneously create a map of the surroundings while also localizing the robot in this map by using the encoders, IMU and the 2-D LiDAR. Then using our current location, we have to use the RGB-D measurements to accurately colour the floor of the entire map traversed.

B. Motivation

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robot autonomy. This provides the basic information needed by any robot to perform any task at hand. A robot has to know where it is (Localization) and what is around it (Mapping). If a map is known apriori, then the task of localization can be treated as a feature matching task based on current observations, but this is usually not the case for real-world robots. Most robots are required to have the ability of exploring the environment and map it for future use, for example, robots used for surveying caves, mines or areas affected by natural disasters. Doing this mapping is not possible without knowing where the robot is in the map at each instant. This also can't be directly computed via control inputs and motion models due to noise induced by real-world actuators and external factors, therefore this process has to be bootstrapped with temporal consistency of observations.

C. Approach

We use a Particle Filter to solve the problem of SLAM. The Particle Filters simulates the noise of the motion model by explicitly adding noise to the motion model and the uses the correlations between the created map and the observation to weight the current set of hypothesis and update their probabilities. Using the location estimates provided by the Particle Filter and the already know location of the Kinect with respect to the robot, the RGB-D measurements are converted to the world frame and the floor is extracted based on height thresholding to create the texture map.

II. Problem Formulation

We are trying to solve the SLAM and Texture Mapping Problems which can be stated as below:

A. Simultaneous Localization and Mapping (SLAM)

SLAM Problem : Given sensor measurements $\mathbf{z}_{0:T}$ and control inputs $\mathbf{u}_{0:T-1}$, estimating the robot state trajectory $\mathbf{x}_{0:T}$ and build a map of the environment $\mathbf{m}$. Mathematically we can say, Given a initial state $\mathbf{x}_0$, sensor measurements $\mathbf{z}_{0:T}$ with observation model h, control inputs $\mathbf{u}_{0:T-1}$ with motion model f, estimate the robot state trajectory $\mathbf{x}_{0:T}$ and build a map of the environment $\mathbf{m}$ such that:

$$\min_{\mathbf{x}_{0:T},\mathbf{m}} \sum_{t=1}^{T} ||\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{m}, \mathbf{v}_t)||_2^2 + \sum_{t=0}^{T-1} ||\mathbf{x}_{t+1} - f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)||_2^2 \tag{1}$$

Where the observation model h maps a state $\mathbf{x}_t$ and the map $\mathbf{m}$ to a observation $\mathbf{z_t}$ in the presence of the observation noise $\mathbf{v}_t$ and the motion model f describes the state $\mathbf{x}_{t+1}$ resulting from applying input $\mathbf{u}_t$ at state $\mathbf{x}_t$ in the presence of motion noise $\mathbf{w}_t$.

B. Texture Mapping

Given a RGB image I_t with the associated depth image D_t at time t, we would like to find all points on the image that are at a height of $0m$ from the ground. Estimating the robots location at time t as $\mathbf{x}_t$, for each pixel at (i, j) in D_t denoted by $D_t^{i,j}$, we would like to compute the transformation from Depth image to Robot frame denoted by ${}_{\{R\}}T_{\{I\}}$ and then the transformation from Robot frame to World frame denoted by ${}_{\{W\}}T_{\{R\}}$ and then use them as following:

$$P_R^{i,j} = {}_{\{R\}}T_{\{I\}} P_I^{i,j}$$
$$P_W^{i,j} = {}_{\{W\}}T_{\{R\}} P_R^{i,j}$$

Where $P_I^{i,j}$ is the 3-D coordinate obtained via $D_t^{i,j}$ and camera intrinsic matrix K and the pinhole camera model.

III. Technical Approach

We use a Bayes Filter, specifically the Particle Filter and the Probabilistic Occupancy Grid mapping to solve the SLAM problem and the pinhole camera model equations to solve the texture mapping problem.

A. Bayes Filter

The Bayes filter is a probabilistic inference technique for estimating the state $\mathbf{x}_t$ by combining the effects of control input $\mathbf{u}_t$ and observations $\mathbf{z}_t$ while exploiting the Markov Assumptions, conditional probability, total probability and Bayes rule. It keeps track of two two Probability Distribution Functions (pdfs):

$$\textbf{Predicted pdf : } p_{t+1|t}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}) \quad (2)$$

$$\textbf{Updated pdf : } p_{t+1|t+1}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t+1}, \mathbf{u}_{0:t}) \quad (3)$$

1) Markov Assumptions:

- The state $\mathbf{x}_{t+1}$ only depends on previous input $\mathbf{u}_t$ and state $\mathbf{x}_t$, i.e., $\mathbf{x}_{t+1}$ given $\mathbf{u}_t$, $\mathbf{x}_t$ is independant of the history $\mathbf{x}_{0:t-1}, \mathbf{z}_{0:t-1}, \mathbf{u}_{0:t-1}$
- The observation $\mathbf{z}_t$ is only dependant on the state $\mathbf{x}_t$

2) Prediction Step: Given a prior belief for the state $\mathbf{x}_t$ as $p_{t|t}$ and control input $\mathbf{u}_t$, we can use the probabilistic motion model p_f to compute the predicted pdf $p_{t+1|t}$ of x_{t+1}:

$$p_{t+1|t}(\mathbf{x}) = \int p_f(\mathbf{x}|\mathbf{s}, \mathbf{u}_t) p_{t|t}(\mathbf{s}) d\mathbf{s} \quad (4)$$

3) Update Step: Given a predicted pdf $p_{t+1|t}$ of $\mathbf{x}_{t+1}$ and measurement $\mathbf{z}_{t+1}$, we can use the probabilistic observation model p_h to obtain the updated pdf $p_{t+1|t+1}$ of x_{t+1}:

$$p_{t+1|t+1}(\mathbf{x}) = \frac{p_h(\mathbf{z}_{t+1}|\mathbf{x}) p_{t+1|t}(\mathbf{x})}{\int p_h(\mathbf{z}_{t+1}|\mathbf{s}) p_{t+1|t}(\mathbf{s})} \quad (5)$$

B. Particle Filter

The Particle Filter is a special case of the Bayes Filter where the distributions $p_{t+1|t}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t})$ and $p_{t+1|t+1}(\mathbf{x}_{t+1}) = p(\mathbf{x}_{t+1}|\mathbf{z}_{0:t+1}, \mathbf{u}_{0:t})$ are discrete distributions with N particles. Since the pdf is discrete (Probabilty Mass Function (pmf)) and we have N particles, we can use the dirac delta function to convert it to a continuous pdf.

Let's say we have N particles with probabilities $\alpha[1], \ldots, \alpha[N]$ for each particle that have values $\mu[1], \ldots, \mu[N]$. We can define the pdf as:

$$p(\mathbf{x}) = \sum_{k=1}^{N} \alpha[k] \delta(\mathbf{x} - \mu[k]) \quad (6)$$

where δ is the Dirac delta function:

$$\delta(x) = \left\{ \begin{array}{ll} \infty, & x = 0 \\ 0, & x \neq 0 \end{array} \right\} \quad (7)$$

$$\int_{-\infty}^{\infty} f(x)\delta(x) dx = f(0) \quad (8)$$

$$\int_{-\infty}^{\infty} \delta(x) dx = 1 \quad (9)$$

Using this we can define predicted and the updated pdfs. The prior pdf at time t is given by:

$$p_{t|t}(\mathbf{x}) = \sum_{k=1}^{N} \alpha_{t|t}[k] \delta(\mathbf{x}_t - \mu_{t|t}[k]) \quad (10)$$

1) Prediction Step: Assuming the probabilistic motion model as p_f and plugging this and the prior into bayes filter predict step, we have:

$$\textbf{Predicted pdf : } p_{t+1|t}(\mathbf{x}_{t+1}) = \sum_{k=1}^{n} \alpha_{t|t}[k] p_f(\mathbf{x}_{t+1}|\mu_{t|t}, \mathbf{u}_t) \quad (11)$$

This is then approximated by sampling the new mean from the motion model and then again representation it as a weighted sum of Dirac delta functions. Therefore we get:

$$p_{t+1|t}(x_{t+1}) \approx \sum_{k=1}^{N} \alpha_{t|t}[k] \delta(\mathbf{x}_{t+1} - \mu_{t+1|t}[k]) \quad (12)$$

where:

$$\mu_{t+1|t}[k] \sim p_f(\circ|\mu_{t|t}[k], \mathbf{u}_t) \quad (13)$$

So therefore only the particle mean is updated in this step.

2) Update Step: With the probabilistic observation model p_h and the above pdf, putting these in the Bayes filer update step, we get:

$$p_{t+1|t+1}(\mathbf{x}_{t+1}) = \frac{p_h(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}) \sum_{k=1}^{N} \alpha_{t+1|t}[k] \delta(\mathbf{x}_{t+1} - \mu_{t+1|t}[k])}{\int p_h(\mathbf{z}_{t+1}|\mathbf{s}) \sum_{j=1}^{N} \alpha_{t|t}[j] \delta(\mathbf{s} - \mu_{t+1|t}[j]) d\mathbf{s}}$$

$$= \sum_{k=1}^{N} \left[\frac{\alpha_{t+1|t}[k] p_h(\mathbf{z}_{t+1}|\mu_{t+1|t}[k])}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] p_h(\mathbf{z}_{t+1}|\mu_{t+1|t}[j]} \right] \delta(\mathbf{x} - \mu_{t+1|t}[k]) \quad (14)$$

So therefore, only the particle weights are updated in this step as:

$$\alpha_{t+1|t+1}[k] = \frac{\alpha_{t+1|t}[k] p_h(\mathbf{z}_{t+1}|\mu_{t+1|t}[k])}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] p_h(\mathbf{z}_{t+1}|\mu_{t+1|t}[j])} \quad (15)$$

C. State

The State maintained by the particle filter is:

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \quad (16)$$

where x_t, y_t are the X and Y coordinates of the robot and θ_t is the heading of the robot in the 2-D $X - Y$ plane.

D. Motion Model

As the robot is a differential drive robot, the non probabilistic motion model is defined as follows:

$$\mathbf{x}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = f_d(\mathbf{x}_t, \mathbf{u}_t, \tau_t)$$

$$= \mathbf{x}_t + \tau_t \begin{bmatrix} v_t sinc(\frac{\omega_t \tau_t}{2}) cos(\theta_t + \frac{\omega_t \tau_t}{2}) \\ v_t sinc(\frac{\omega_t \tau_t}{2}) sin(\theta_t + \frac{\omega_t \tau_t}{2}) \\ \omega_t \end{bmatrix} \quad (17)$$

The probabilistic version of this is as follows:

$$f(\mathbf{x}_t, \mathbf{u}_t, \tau_t, w_t) = f_d(\mathbf{x}_t, \mathbf{u}_t, \tau_t) + w_t \quad (18)$$

where:

$$w_t \sim \mathcal{N}(0, W) \quad (19)$$

where W represents the covariance of the we want to add.

E. Probabilistic Occupancy Grid Mapping

The map representation we use is a occupancy grid. We assume each cell is independant of each other consitioned on robot trajectory or:

$$p(\mathbf{m}|\mathbf{z}_{0:t}, \mathbf{x}_{0:t}) = \prod_{i=1}^{n} p(\mathbf{m}_i|\mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \tag{20}$$

Each map cell $\mathbf{m}_i$ is treated as a independant bernoulli random variable:

$$\delta(x) = \left\{ \begin{array}{ll} +1, & \text{with prob. } \gamma_{i,t} = p(\mathbf{m}_i = 1|\mathbf{z}_{0:t}, \mathbf{x}_{0:t}) \\ -1, & \text{with prob. } 1 - \gamma_{i,t} \end{array} \right\} \tag{21}$$

F. Observation Model

We have 2-D Laser scans as the observations. These are in polar coordinates and therefore have to converted to cartesian and then map coordinates. The LiDAR sweeps have the range $[-135°, 135°]$ with a constant increment leading to 1081 points per scan.

A scan at time t can be converted to cartesian coordinates as follows:

$$x = rcos(\phi), y = rsin(\phi) \tag{22}$$

where ϕ is the angle of that particular ray, and r is the range value for that ray. Assuming the state of the robot at the timestep t is $\mathbf{x}_t = \begin{bmatrix} x_t, y_t, \theta_t \end{bmatrix}^T$, We can write the transformation:

$$_{\{W\}}T_{\{R\}} = \begin{bmatrix} cos(\theta_t) & -sin(\theta_t) & 0 & x_t \\ sin(\theta_t) & cos(\theta_t) & 0 & y_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

This transformation can be used to convert the cartesian coordinates of the laserscan from robot to world frame. One they are in the world frame, we can map them to a grid cell in the map $\mathbf{m}$ as follows:

$$i = \frac{-y}{resolution} + origin_i \tag{24}$$

$$j = \frac{x}{resolution} + origin_j \tag{25}$$

Where $origin$ is the cell mapped to $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$. Using Bresenham's algorithm, all grid cells that lie between the current robot position and the above LiDAR point are found out. All these cells are marked as 0 and the cells with LiDAR points are marked as 1. This gives us the observation $\mathbf{z}_t = h(\mathbf{x}_t, l_t)$ where l_t is the LiDAR scan.

G. Map Correlation

Once the above observation $\mathbf{z}_t$ is created. The correlation between the $\mathbf{z}_t$ and $\mathbf{m}$ is defined as:

$$corr(\mathbf{z}_t, \mathbf{m}) = \sum_{i,j} \mathbb{1}(\mathbf{z}_t = \mathbf{m} = 1) \tag{26}$$

This correlation is calculated for every particle and then used in the update step.

We can also add pertubations to the locations of the particles while calculating the correlation. This led to a lot of compute time and did not improve my map that much. The results reported are calculated without the pertubations.

H. Particle Filter Update Step

For all N particles, the correlation is calculated. We then update the αs as follows:

$$\alpha_{t+1|t+1}[k] = \frac{\alpha_{t+1|t}[k] corr(h(\mu_{\mathbf{t+1|t}}[\mathbf{k}], \mathbf{m}))}{\sum_{j=1}^{N} \alpha_{t+1|t}[j] corr(h(\mu_{\mathbf{t+1|t}}[\mathbf{j}], \mathbf{m}))} \tag{27}$$

I. Map Update

After the Particle Filter Update step, we update the current map $\mathbf{m}$ by using the mode of all particles(particle with the highest probability), represented by μ_m. Let λ_t denote the log-odds at time t. We create the observation $\mathbf{z}_m$ using the observation model($h(\mu_m, l_t)$). Let's say that $\mathbf{z}_m$ indicates that the cell (i, j) is occupied, then we update the log-odds as:

$$\Delta\lambda_t^{i,j} = \left\{ \begin{array}{ll} +log4, & \text{if } \mathbf{z}_t \text{ indicates } \mathbf{m}_{i,j} \text{ is occupied} \\ -log4, & \text{if } \mathbf{z}_t \text{ indicates } \mathbf{m}_{i,j} \text{ is free} \end{array} \right\} \tag{28}$$

Then the map can be shown as:

$$\mathbf{m}_{i,j} = \left\{ \begin{array}{ll} +1, & \text{if } \lambda_{i,j} > thresh \\ -1, & \text{if } \lambda_{i,j} \leq thresh \end{array} \right\} \tag{29}$$

J. Particle Resampling

If the effective number of particles defined as:

$$N_{eff} = \frac{1}{\sum_k (\alpha[k])^2} \tag{30}$$

falls below a certain threshold, the particles should be resampled. The resampling is done by sampling N particles as follows:

$$\mu \sim \sum_{k=1}^{N} \alpha[k] \delta(\mathbf{x} - \mu[k]) \tag{31}$$

Each $\alpha[k]$ is reset as:

$$\alpha[k] = \frac{1}{N} \tag{32}$$

The final algorithm can be summarized as follows:

Algorithm 1 Particle Filter SLAM

```
Given Encoder readings ${e_t}$, IMU readings ${\omega_t}$, Laser Scans ${L_t}$
$pf \leftarrow ParticleFilter(N)$
$\tau \leftarrow 0$
for $l_t$ in ${L_t}$ do
    $pf.predict({e_t}, {\omega_t}, t - \tau)$
    $pf.update(pf.h(pf.state, l_t), pf.map)$
    $pf.update\_map()$
    $pf.resample()$
end for
```

K. Texture Mapping

At any time t, the position can be calculated as:

$$\mathbb{E}(\mathbf{x}) = \sum_{k=1}^{N} \alpha_{t|t}\mu_{t|t} = \mathbf{x}_t \tag{33}$$

With $\mathbf{x}_t = \left[x_t, y_t, \theta_t\right]^T$ with this we get the transform:

$$_{\{W\}}T_{\{R\}} = \begin{bmatrix} cos(\theta_t) & -sin(\theta_t) & 0 & x_t \\ sin(\theta_t) & cos(\theta_t) & 0 & y_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

We know the K matrix for the camera generating the RGB-D images. For any (i, j) pixel in the RGB image, let the depth be $d_{i,j}$ and. We know that:

$$\begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} = K^{-1} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} * d_{i,j} \tag{35}$$

where $\begin{bmatrix} X_o & Y_o & Z_o \end{bmatrix}^T$ is the 3-D coordinate in the optical frame. Then we have optical to robot frame as(Using the robot configuration file):

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} = \begin{bmatrix} cos(18°) & 0 & -sin(18°) & 0.33276 \\ 0 & 1 & 0 & 0 \\ sin(18°) & 0 & cos(18°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_T \\ Y_T \\ Z_T \\ 1 \end{bmatrix} \tag{36}$$

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix} \tag{37}$$

Finally we get:

$$\begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = {}_{\{W\}}T_{\{R\}} \begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} \tag{38}$$

These can then be filtered according to the Z coordinate and then converted to grid indices using Eq 24.

IV. Results

A. Dead Reckoning

The dead reckoning maps(log-odds) and trajectories can be seen in Fig 1 and Fig 2.

B. Particle Filter

The Particle Filter maps(log-odds) and trajectories can be seen in Fig 3 and Fig 4.

C. Texture Map

The Particle Filter Texture Maps can be seen in Fig 5 and Fig 6.

D. Overtime Results

For the trajectory and Occupancy grid, see Fig 7 to Fig 15. For the Texture map, see Fig 16 to Fig 24.

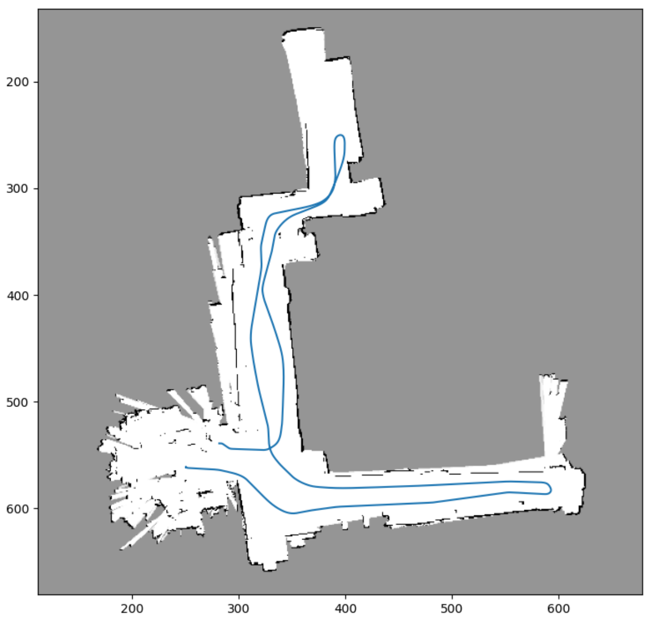

Fig. 1. Dataset 20 - Dead Reckoning Trajectory and Map

E. Discussion

1) Pertubation: Adding pertubations in the map correlation did not improve performance and therefore all results shown are without pertubations. The pertubations tried were 1 cell back and forth for x and y and -0.02 to 0.02 radians.

2) N_{eff} for resampling: Resampling after every update performed better than most resampling thresholds. The results reported resample after every update.

3) N: The higher the number of particles, the better the map was. 500 was the maximum I was able to test in the limited timeframe.

4) Resolution: The final resolution that showed a good balance between execution time and accuracy was $0.07 \times 0.07m^2$ per cell.

5) Variance: The final variance that worked best was

$$\begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \tag{39}$$

The value is a little high as the resampling frequency is also high, and so the resampling frequency and variance seem to be proportional to each other for achieving optimal results.

F. Imporvements

I believe that with higher resolution and a parameter sweep of the pertubations, variances and N_{eff}, a much better result can be obtained but due to a lack of time, this experimentation could not be performed.

On the horizontal hallway, the map is far from perfect. The map diverges when the U-turn is being made at this hallway. This should be solvable with parameter tuning.

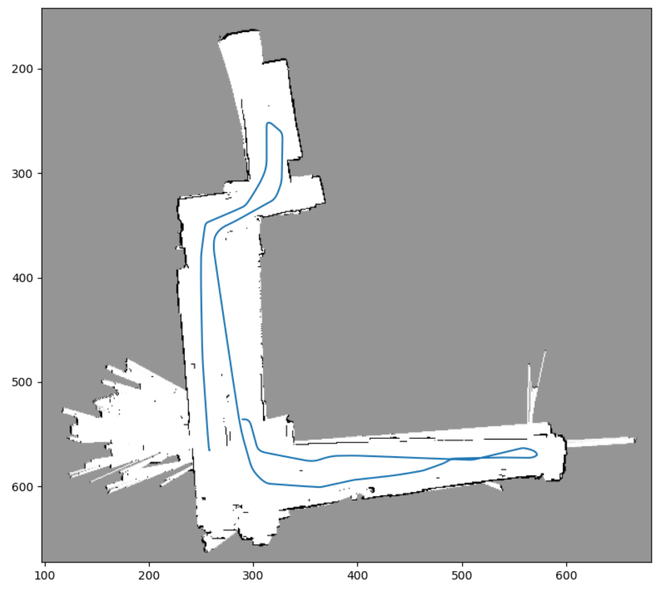

Fig. 2. Dataset 21 - Dead Reckoning Trajectory and Map

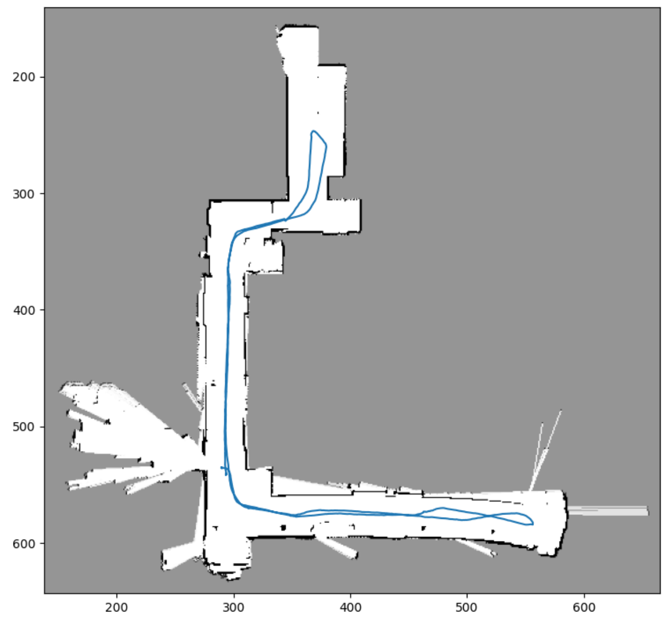

Fig. 4. Dataset 21 - Particle Filter Trajectory and Map

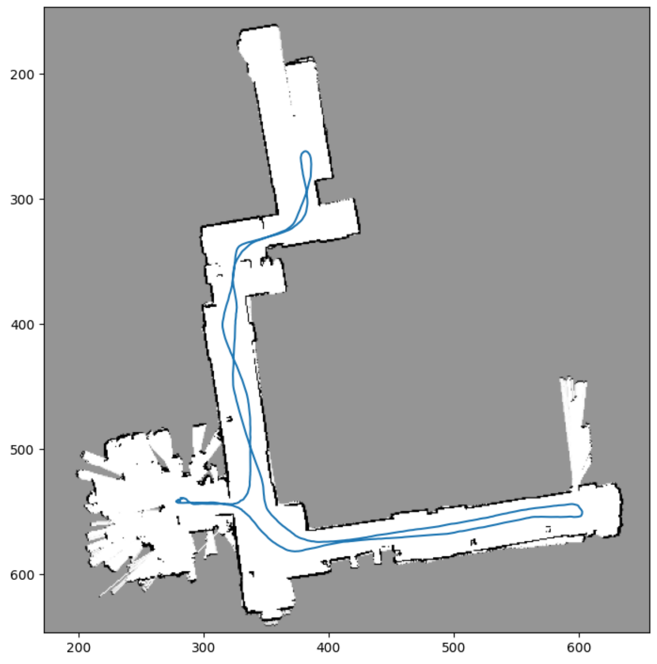

Fig. 3. Dataset 20 - Particle Filter Trajectory and Map

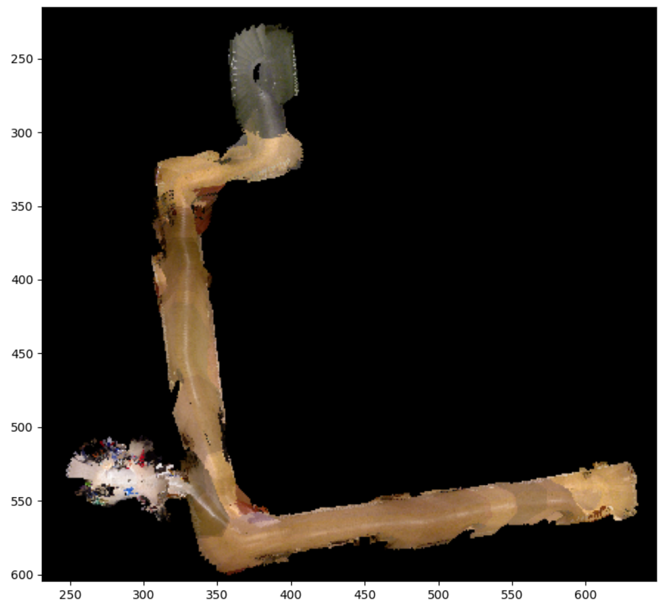

Fig. 5. Dataset 20 - Texture Map

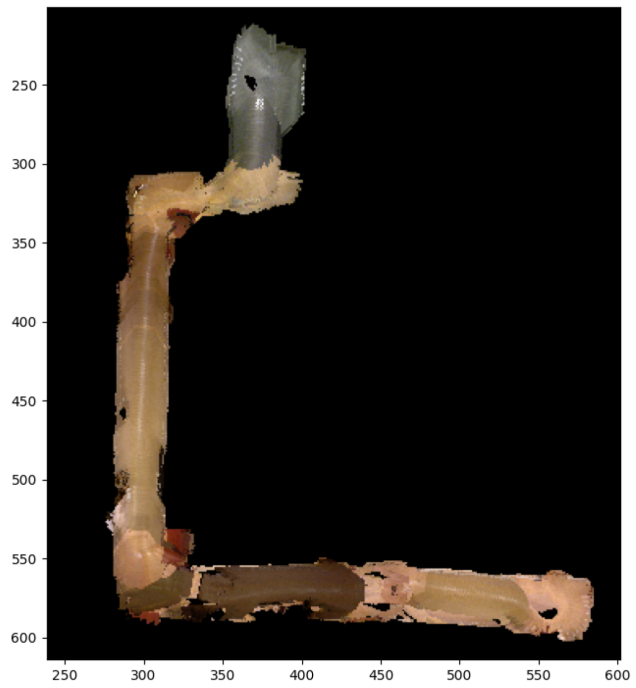
Fig. 6. Dataset 21 - Texture Map

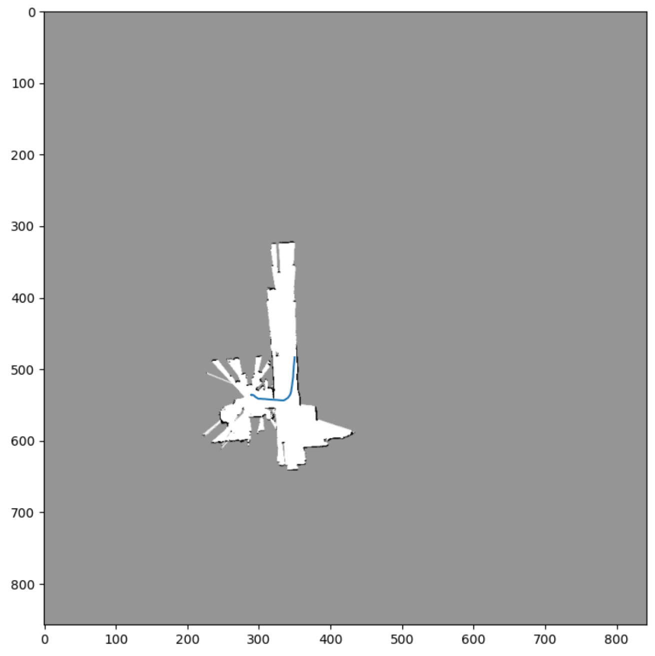
Fig. 8. Dataset 20 - Timestep 1000

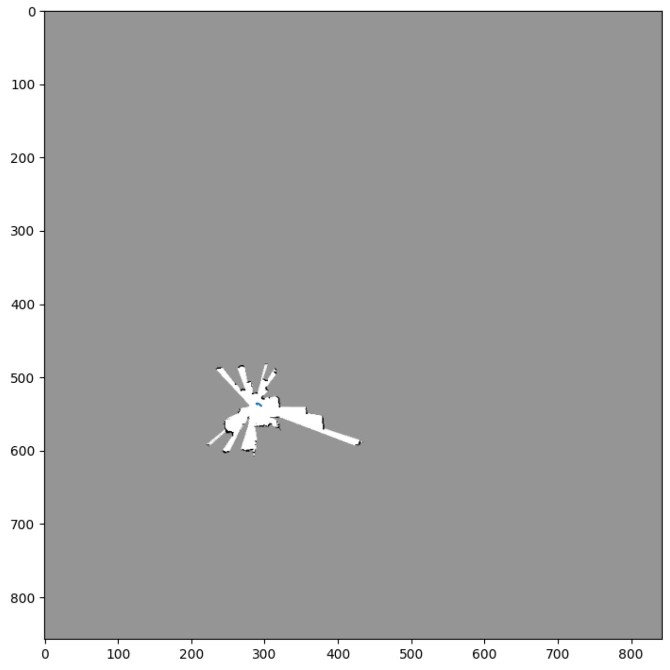
Fig. 7. Dataset 20 - Timestep 500

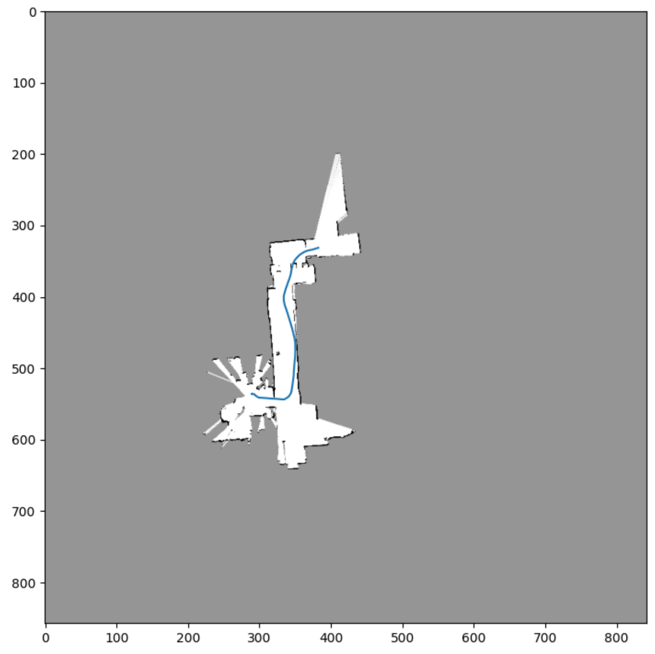
Fig. 9. Dataset 20 - Timestep 1500

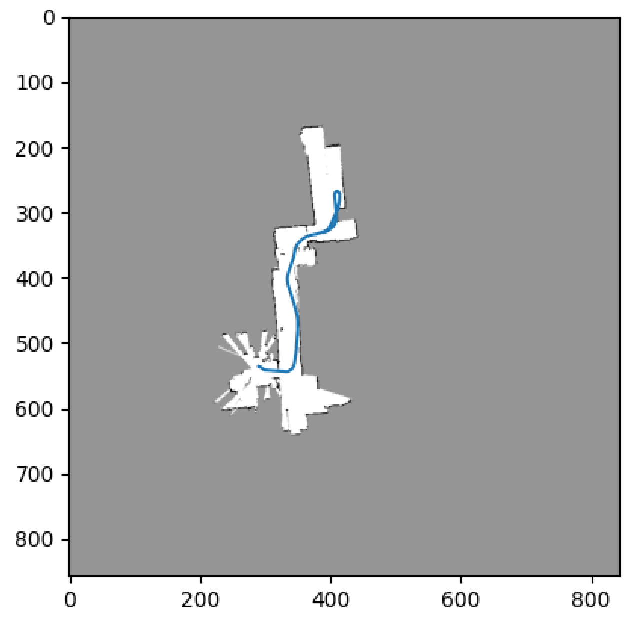
Fig. 10. Dataset 20 - Timestep 2000

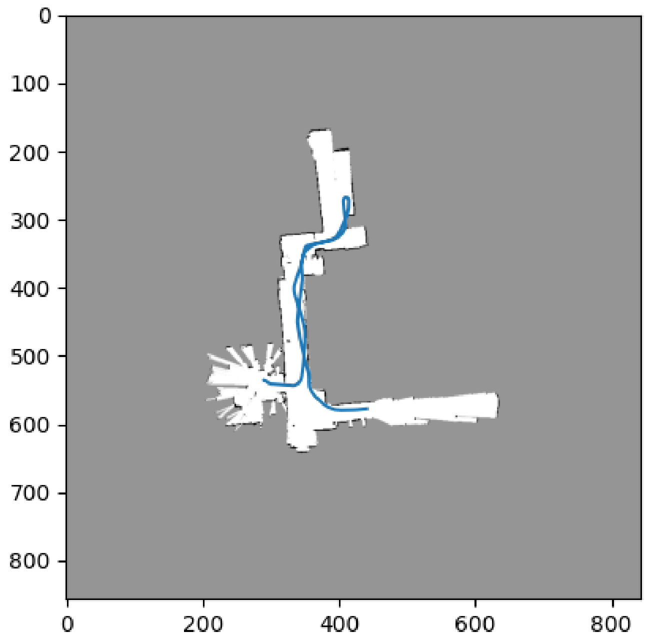
Fig. 12. Dataset 20 - Timestep 3000

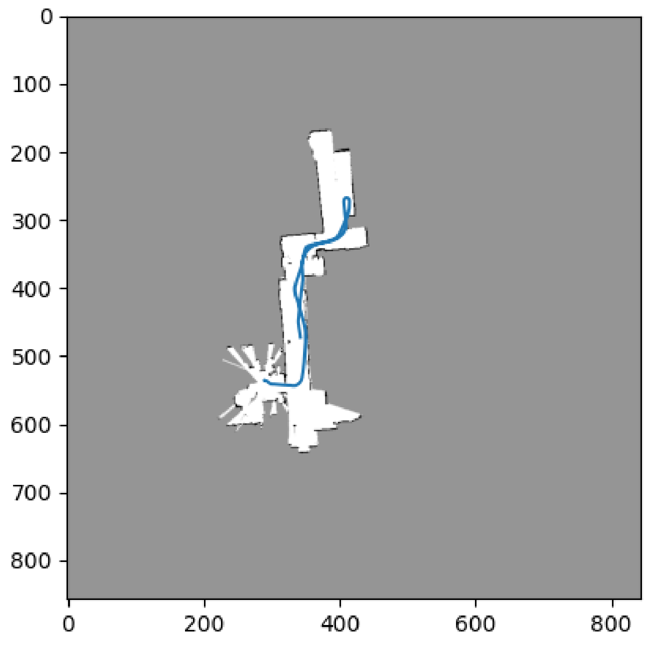
Fig. 11. Dataset 20 - Timestep 2500

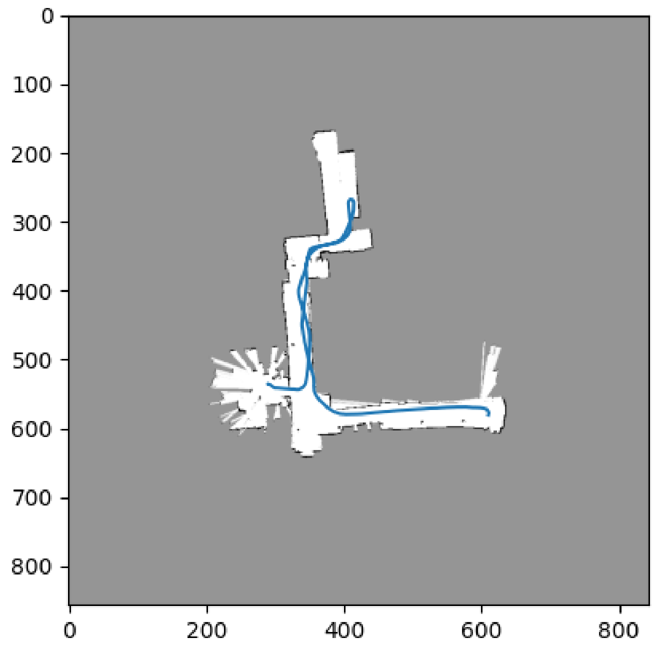
Fig. 13. Dataset 20 - Timestep 3500

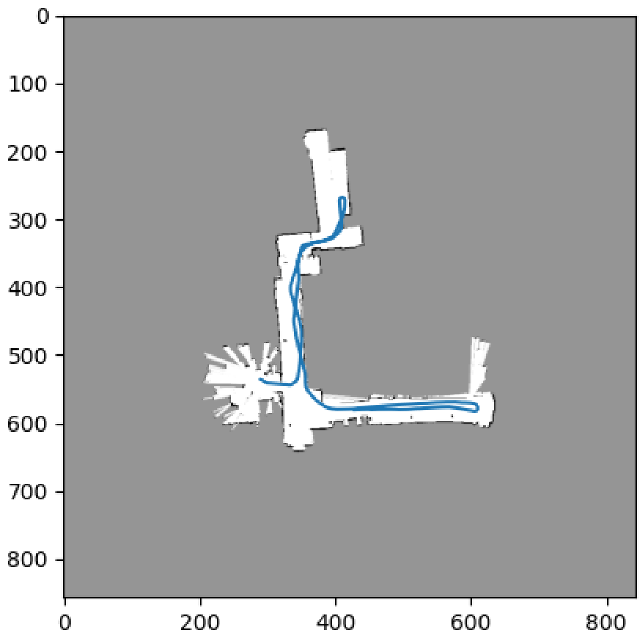

Fig. 14. Dataset 20 - Timestep 4000

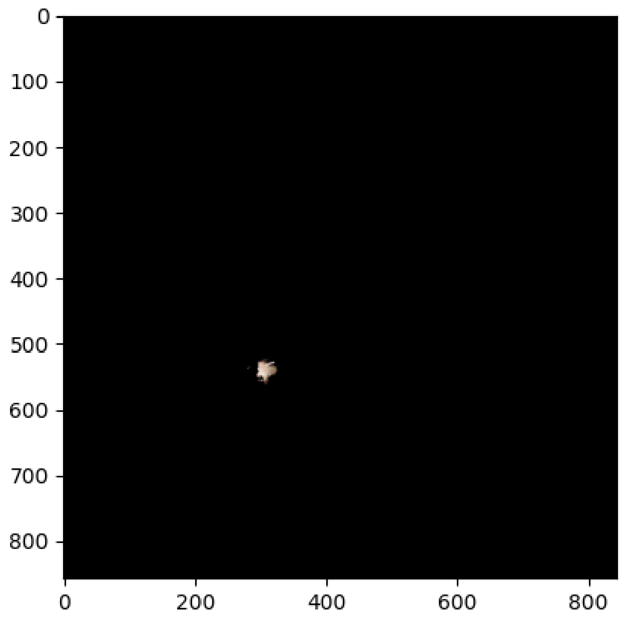

Fig. 16. Dataset 20 - Timestep 500

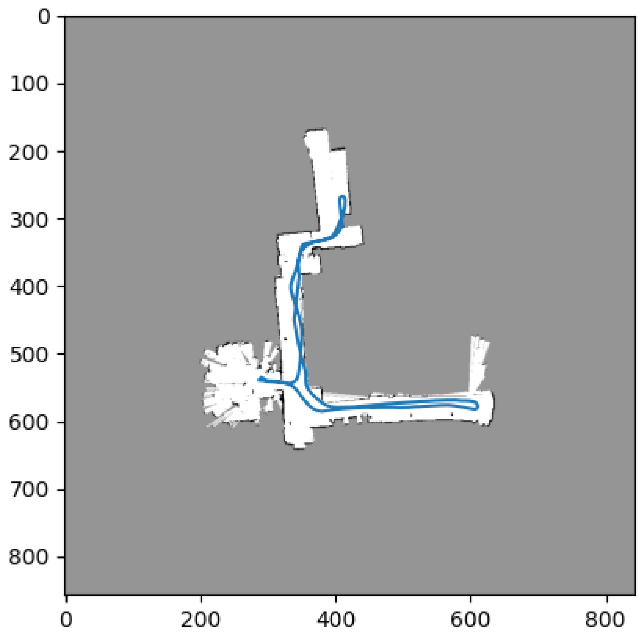

Fig. 15. Dataset 20 - Timestep 4500

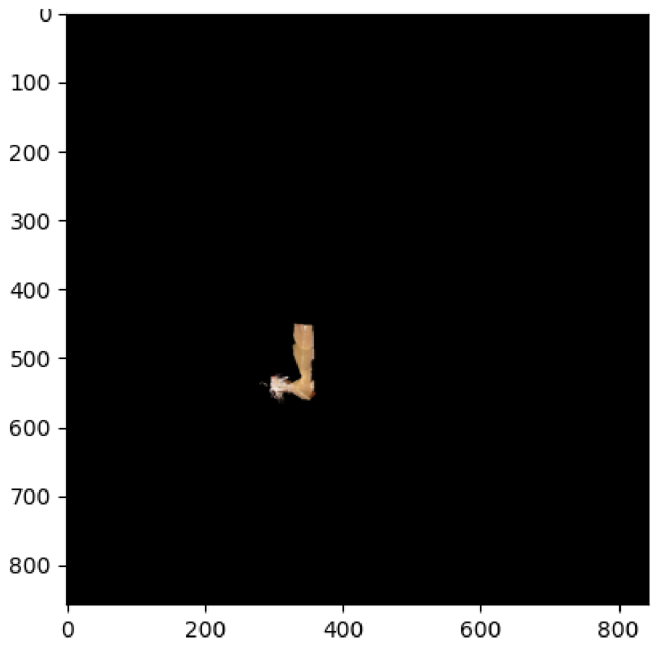

Fig. 17. Dataset 20 - Timestep 1000

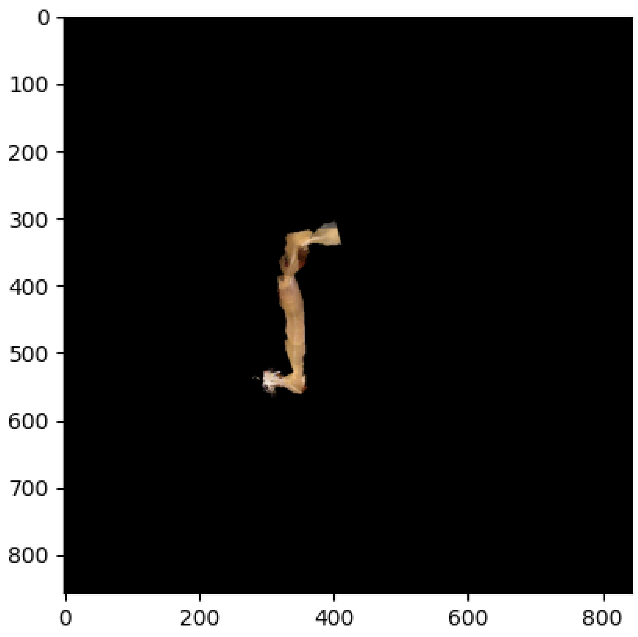

Fig. 18. Dataset 20 - Timestep 1500

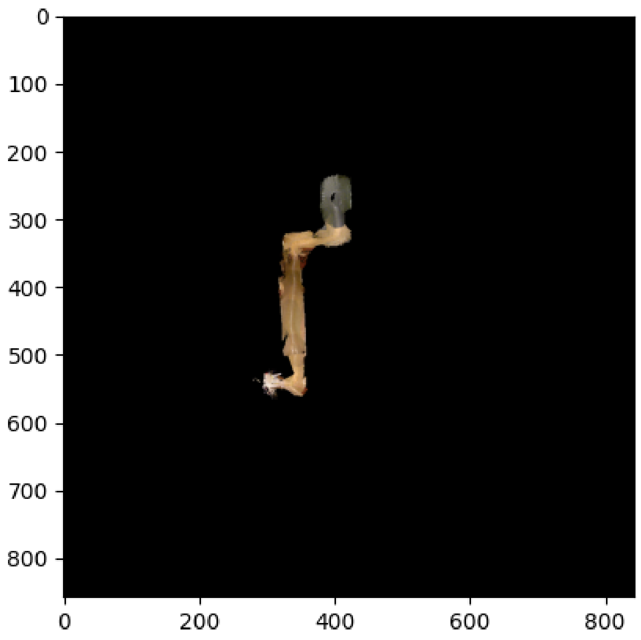

Fig. 20. Dataset 20 - Timestep 2500

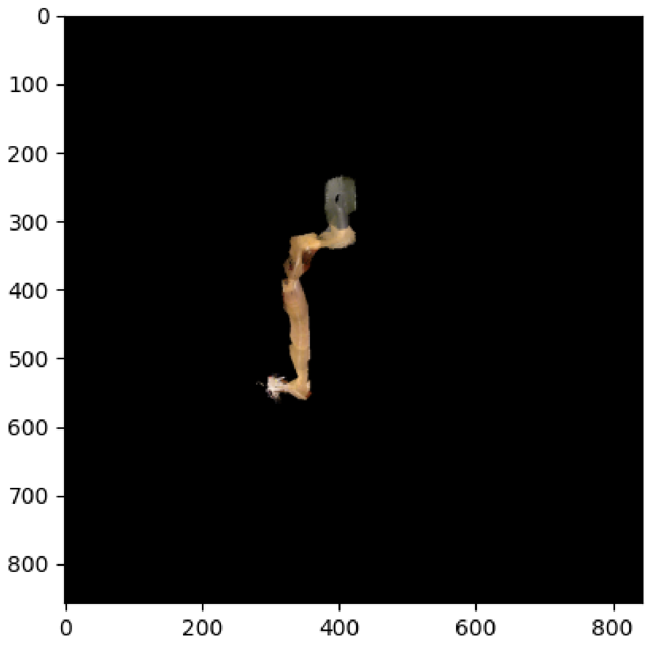

Fig. 19. Dataset 20 - Timestep 2000

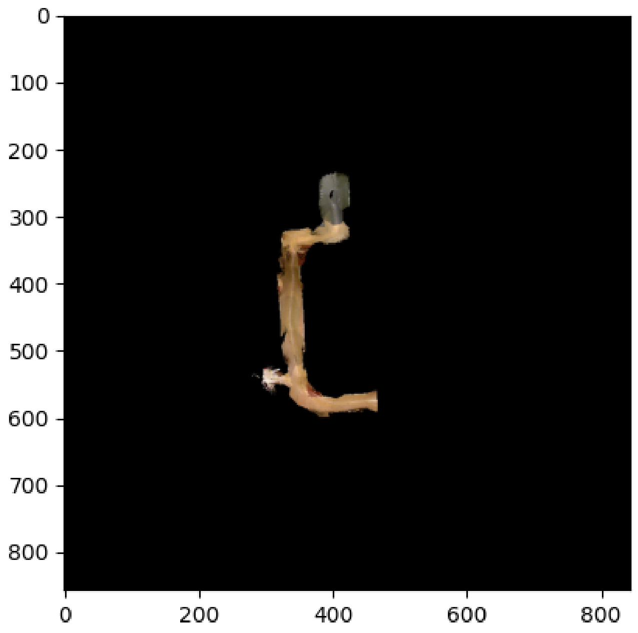

Fig. 21. Dataset 20 - Timestep 3000

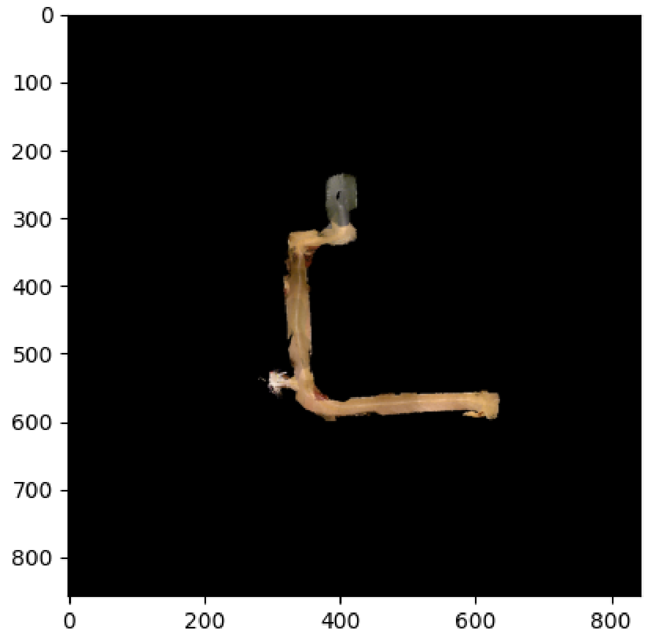

Fig. 22. Dataset 20 - Timestep 3500

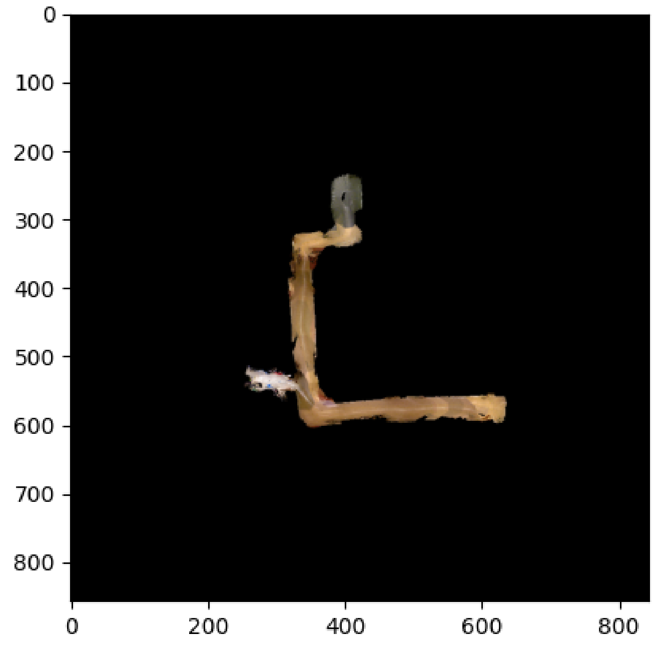

Fig. 24. Dataset 20 - Timestep 4500

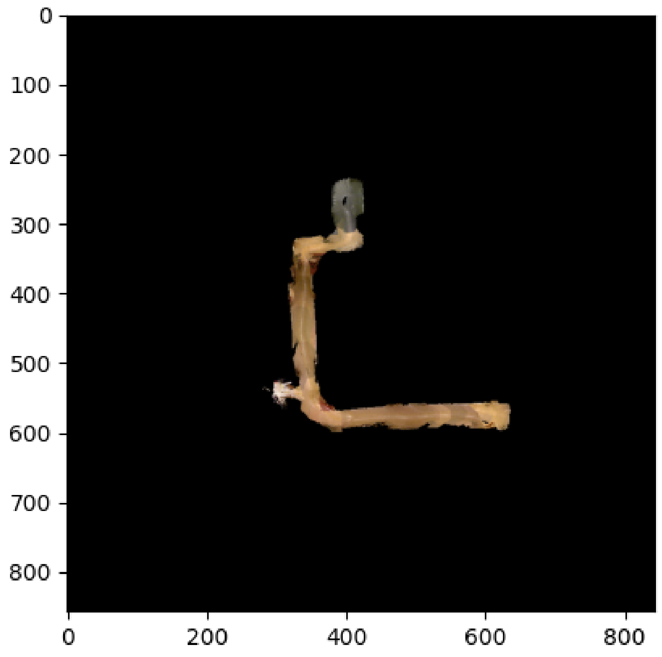

Fig. 23. Dataset 20 - Timestep 4000