LiDAR-Based SLAM for 2D Robotic Navigation

Zejia Wu
ECE Department
UC San Diego
zew024@ucsd.edu

***Abstract*—This project investigates the implementation of Simultaneous Localization and Mapping (SLAM) using LiDAR, encoders, IMU, and RGBD data to estimate a robot's position and construct a 2D occupancy grid map. The methodology integrates odometry-based motion estimation, Iterative Closest Point (ICP) for LiDAR scan matching, and loop closure detection using GTSAM for pose graph optimization. Additionally, RGBD data enhances the mapping by overlaying textures onto the generated occupancy grid. The results demonstrate an efficient SLAM framework, yielding accurate localization and robust mapping for autonomous robotic applications. This report provides a detailed analysis of each component, covering mathematical formulations, implementation details, and experimental results. The link to the video for dataset for dataset 20 is https://drive.google.com/file/d/1zR5J0KLBS5Vf0SiM2ORDguIoeM8UxiR6/view?usp=drive_link, while the link to the video 21 is https://drive.google.com/file/d/1Bmtt0MPmSmKz6nIyXlk225BDjlAeYRaZ/view?usp=drive_link.**

***Index Terms*—LiDAR SLAM, ICP, Pose Graph Optimization, Loop Closure Detection, Texture Mapping.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a critical technology in robotics that enables autonomous systems to navigate and explore unknown environments while constructing detailed maps. The fundamental challenge in SLAM is to estimate a robot's pose accurately while concurrently building a representation of the environment. This problem becomes even more complex due to sensor noise, uncertainty in motion, and environmental variations. To overcome these challenges, SLAM relies on multiple sensing modalities and advanced estimation techniques.

In this project, we employ a multi-sensor fusion approach, integrating encoders, an Inertial Measurement Unit (IMU), and RGBD sensors to mitigate the limitations of single-sensor reliance. The data obtained from these sensors undergo processing and alignment with each other, which is then incorporated into a factor graph-based optimization framework to enhance localization and mapping accuracy. The ability to combine different sensor modalities ensures robustness, reducing cumulative drift and improving real-time adaptability in dynamic environments.

A differential-drive motion model is used to process IMU and encoder data to generate initial pose estimates. However, odometry alone accumulates errors over time due to wheel slip and sensor inaccuracies. To refine these estimates, we apply the Iterative Closest Point (ICP) algorithm, which aligns LiDAR scans to correct localization drift. Furthermore, to construct a reliable and detailed environmental representation, we utilize a log-odds-based probabilistic occupancy grid mapping technique, allowing continuous map updates based on new sensor observations. The resulting occupancy grid is further enriched using ground point cloud data captured by RGBD sensors, adding texture and improving visualization.

To ensure long-term mapping accuracy and robustness, we implement loop closure detection to identify previously visited locations. When a loop closure is detected, the entire trajectory is optimized using graph-based SLAM techniques, specifically through the GTSAM library, which significantly reduces accumulated errors and enhances global consistency. This process is crucial for large-scale environments where localization drift can otherwise degrade the quality of the generated map.

Beyond the current implementation, the SLAM system is designed to be flexible and scalable, capable of integrating more advanced sensor modalities. This multi-sensor integration strategy not only enhances adaptability and robustness but also lays a foundation for future advancements in robotic navigation and mapping. The potential applications of this research extend to autonomous driving, industrial automation, and remote exploration in unstructured environments, making it a vital component in modern robotics.

This report details the SLAM pipeline developed in this project, including mathematical formulations, algorithmic implementation, and experimental validation. By systematically addressing the challenges of localization, mapping, and global consistency, we provide an in-depth analysis of the effectiveness of our approach and its potential for further enhancement.

II. Problem Formulation

The objective is to estimate the robot's trajectory over time and construct a 2D occupancy grid that represents the environment, using LiDAR measurements, encoder, and IMU readings. The SLAM problem can be formulated as

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

where $\mathbf{x}_t = (x_t, y_t, \theta_t)^{\mathrm{T}}$ is the robot pose at time t, $\mathbf{m}$ is the map to be built. $\mathbf{z}_t$ is the sensor measurement at time t, which refers to the point clouds obtained from LiDAR in this project. h and f are the observation model and motion model,

which are the LiDAR observation model and differential-driven motion model (or odometry obtained from LiDAR scan matching) in this project, respectively.

Specifically, in our project, this problem can be split into challenges such as motion estimation, point cloud registration, occupancy mapping, texture mapping, loop closure detection, and pose graph optimization, which are formulated as the following problems.

A. Motion Estimation

We use the encoder counts ($[FR, FL, RR, RL]$) and IMU readings (yaw angle velocity ω_t) to predict the robot's pose at each time step, denoted by (x_t, y_t, θ_t), using differential-drive motion model.

B. Point Cloud Registration

Given two sets of points $\{m_i \in \mathbb{R}^{\mathrm{d}}\}$ and $\{z_j\} \in \mathbb{R}^{\mathrm{d}}$ from consecutive LiDAR scans, find an optimal transformation $T \in SE(\mathrm{d})$ and a data association $\Delta = \{(i,j) \mid m_i \text{ corresponds to } z_j\}$ aligning them:

$$\min_{R \in SO(\mathrm{d}), p \in \mathbb{R}^{\mathrm{d}}, \Delta} \sum_{(i,j)\in\Delta} w_{ij} \left\| R z_j + p - m_i \right\|_2^2 .$$

In this project, we formulate the LiDAR scan matching problem between two LiDAR measurements as a point cloud registration problem, by solving which we can get the relative pose of the robot between two time steps.

C. Occupancy Mapping

Constructing a grid m based on LiDAR scans $P = \{p_i \mid p_i \in \mathbb{R}^{\mathrm{d}}\}$, where each cell m_i is classified as occupied $(+1)$ or free (-1). The probabilistic formulation follows:

$$p(m|Z_{0:t}, X_{0:t}) = \prod_{i=1}^{n} p(m_i|Z_{0:t}, X_{0:t}).$$

The map is updated over time while the robot moves to different areas and receives more scans. We decrease the probability of a grid being occupied if the LiDAR scan passes through it and increase the occupancy probability of a grid if the LiDAR scan stops there.

D. Texture Mapping

At each time step, we are given an RGB image (R, G, B) and depth values d for all pixels, which can be calculated using disparity images. The goal is to recover the point cloud $\{(x_i, y_i, z_i)\}$ in the world frame and its corresponding RGB color (r_i, g_i, b_i) first. By identifying the points with $z_i = 0$ and their colors, we can obtain the color texture of the floor and project their colors onto the occupancy map we generated, enhancing the visualization.

E. Loop Closure Detection

Identifying previously visited locations is crucial in SLAM to mitigate accumulated localization drift. Loop closure detection ensures that when a robot revisits a previously mapped area, the system can recognize the event and make necessary corrections. Given two robot poses $p_i = (x_i, y_i, \theta_i)$ and $p_j = (x_j, y_j, \theta_j)$ observed at different timestamps, a loop closure event is detected if the distance between these poses satisfies

$$D(p_i, p_j) < \text{threshold},$$

where D is a pre-defined distance metric between poses. The detection is performed periodically. Once detected, the loop closure constraint is incorporated into the pose graph to refine the estimated trajectory, improving global map consistency and reducing long-term drift.

F. Pose Graph Optimization

Given a set of poses in the world frame $\{T_i \mid T_i \in SE(\mathrm{d}),\ i = 1, 2, \cdots n\}$ (initialized using the encoder counts and IMU readings) and relative pose measurements $\{T_{ij} \mid T_{ij} \in SE(\mathrm{d}), (i,j) \in \mathcal{E}\}$ (obtained by LiDAR scan matching), where $\mathcal{E}$ is the set of edges in the pose graph, the optimization can be formulated as

$$\min_{\{T_i\}_{i=1}^{n}} \sum_{(i,j)\in\mathcal{E}} \left\| W_{ij} \log(T_{ij}^{-1} T_i^{-1} T_j) \right\|^2 ,$$

where W_{ij} is the information matrix, $\log(\cdot)$ is the matrix logarithm, and $\|\cdot\|$ denotes the Frobenius norm.

Each of these challenges is addressed systematically in the following sections, detailing their implementation, optimization, and integration into the SLAM pipeline.

III. Technical Approach

In this section, we will give detailed algorithms and implementations to deal with the problems formulated above.

A. Motion Estimation

We started by estimating the pose of the robot at each time point using the encoder counts and IMU readings. We can obtain the velocity of the left wheel and that of the right wheel by processing the encoder counts $[FR, FL, RR, RL]$:

$$(v_l, v_r) = (\frac{FL + RL}{2}, \frac{FR + RR}{2}) \times 0.0022.$$

The robot's movement is governed by a differential-drive kinematic model, which provides a mathematical framework for estimating the robot's position based on its wheel velocities. Given the left and right wheel velocities, v_l and v_r, the robot's linear velocity can be expressed as:

$$v = \frac{v_l + v_r}{2}.$$

For the yaw rate we use ω provided by the IMU. The robot's pose (x_t, y_t, θ_t) at time t is updated using the equations:

$$\begin{aligned} x_{t+1} &= x_t + \tau v \cos\theta_t \\ y_{t+1} &= y_t + \tau v \sin\theta_t \,, \\ \theta_{t+1} &= \theta_t + \tau\omega \end{aligned}$$

where τ represents the time step. This model serves as the foundation for motion estimation before incorporating sensor corrections.

B. Iterative Closest Point (ICP) for Scan Matching

To refine the pose estimates and reduce drift, we apply the ICP algorithm to LiDAR scans at different time steps to obtain the relative pose of the robot.

The ICP algorithm iteratively minimizes the difference between corresponding points in two point clouds. The steps involved are:

- Initial Alignment: Provide an initial guess of the transformation, denoted by p_0, R_0.
- Nearest Neighbor Search: Find corresponding points between the source and target scans. For a point m_i in the source scan, the corresponding index of the point in the target scan should be
$$\arg\min_j \|m_i - (R_k z_j + p_k)\|,$$
where p_k, R_k are the estimation of the transform obtained from the k-th iteration.
- Transformation Estimation: Given the correspondence Δ obtained from the last step, compute the optimal rigid transformation p_{k+1}, R_{k+1} using the Kabsch algorithm.
- Apply Transformation: Update the source scan based on the estimated transformation.
- Convergence Check: Repeat until the transformation error is below a predefined threshold.

This iterative process refines pose estimates and ensures accurate localization.

The key point of the ICP algorithm is the Kabsch algorithm inside the iteration loops, which estimates the relative pose between two sets of point clouds $\{m_i\}$ and $\{z_j\}$ with known point associations Δ.

$$\min_{R\in SO(\mathrm{d}),p\in\mathbb{R}^{\mathrm{d}}} \sum_{(i,j)\in\Delta} w_{ij} \|Rz_j + p - m_i\|_2^2.$$

We know that by taking the gradient and matrix transforms, we can convert the above optimization to Wahba's problem:

$$R^* = \arg\max_{R\in SO(\mathrm{d})} \mathrm{tr}(Q^{\mathrm{T}}R),\ p^* = \bar{m} - R^*\bar{z},$$

where

$$\bar{m} = \frac{\sum_i w_i m_i}{\sum_i w_i},\ \bar{z} = \frac{\sum_i w_i z_i}{\sum_i w_i},$$
$$\delta m_i = m_i - \bar{m},\ \delta z_i = z_i - \bar{z},$$

and

$$Q = \sum_i w_i \delta m_i \delta z_i^{\mathrm{T}}.$$

The solution can be obtained by applying Singular Value Decomposition (SVD) to Q:

$$Q = U\Sigma V^{\mathrm{T}}.$$

As a result, the optimal rotation matrix is

$$R = UWV^{\mathrm{T}},$$

where $W = \mathrm{diag}(1, 1, \det(UV^{\mathrm{T}}))$.

C. Occupancy Mapping

Probabilistic Occupancy Grid Mapping is a widely used technique in robotic mapping that enables a robot to construct an environment representation using LiDAR data. The environment is represented as a grid-based map, where each cell maintains a probability estimate of being occupied or free.

We define the occupancy grid as a vector $m \in \mathbb{R}^n$, where each grid cell m_i represents the occupancy state. The occupancy of each cell is modeled as a Bernoulli random variable, taking values $+1$ (occupied) or -1 (free). The probability mass function of the map given the robot's trajectory $X_{0:t}$ and sensor observations $Z_{0:t}$ is expressed as:

$$p(m|Z_{0:t}, X_{0:t}) = \prod_{i=1}^{n} p(m_i|Z_{0:t}, X_{0:t}).$$

Since direct probability updates can be computationally complex, we utilize the log-odds representation to simplify calculations. The log-odds ratio $\lambda_{i,t}$ for a grid cell i at time t is given by:

$$\lambda_{i,t} = \log(\frac{p(m_i = 1|Z_{0:t}, X_{0:t})}{p(m_i = -1|Z_{0:t}, X_{0:t})}).$$

The log-odds representation enables efficient Bayesian updates. The update rule for $\lambda_{i,t}$ at each time step is:

$$\lambda_{i,t+1} = \lambda_{i,t} + \log(\frac{p(Z_t|m_i = 1, X_t)}{p(Z_t|m_i = -1, X_t)}) - \log(\frac{p(m_i = 1)}{p(m_i = -1)}),$$

where $p(Z_t|m_i, X_t)$ is the inverse sensor model, representing the likelihood of sensor measurements given the occupancy state of a cell, and $\frac{p(m_i=1)}{p(m_i=-1)}$ represents the prior odds of the environment.

To prevent overconfident estimates, the log-odds values are constrained within a predefined range $[\lambda_{\min}, \lambda_{\max}]$. Additionally, a decay term can be introduced to allow the system to adapt to dynamic environments where previously mapped areas may change over time.

The final occupancy probability $\gamma_{i,t}$ for a grid cell is computed from the log odds using the sigmoid function:

$$\gamma_{i,t} = \frac{1}{1 + \exp(-\lambda_{i,t})}.$$

This probability provides a continuous measure between 0 and 1, indicating the confidence that a cell is occupied. By continuously updating the occupancy grid as new LiDAR scans are acquired, this method allows the robot to construct an accurate and dynamically evolving map. Probabilistic mapping effectively accounts for sensor uncertainty and environmental changes, leading to more robust and reliable autonomous navigation.

D. Texture Mapping

To enhance the occupancy grid with visual texture, we utilize RGBD images to assign color information to the map. The process involves transforming the pixels to point clouds using the depth information, aligning these coordinates

with the robot's global reference frame, and mapping the corresponding RGB values onto the occupancy grid.

Image pixels to point cloud. Given the values $d_{i,j}$ at pixel (i, j) of the disparity image, the depth can be calculated by

$$\text{depth} = \frac{1.03}{-0.00304 \times d + 3.31}.$$

Using this depth value, we compute the 3D position (X, Y, Z) in the robot frame by applying the camera's intrinsic calibration matrix K

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \cdot \text{depth} \cdot K^{-1} \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} + T,$$

where R is the rotation matrix aligning the camera's frame to the robot's frame, while T is the translation vector representing the displacement between the camera and the robot's base. Then the estimated robot pose in the world frame is incorporated to transform points into the world coordinate system. The corresponding RGB color of the point (i, j) can be obtained by applying

$$\text{rgbi} = \frac{526.37i + 19276 - 7877.07 \times dd}{585.051}$$

$$\text{rgbj} = \frac{526.37j + 16662}{585.051}$$

$$\text{rgb} = \text{RGB\_Image}[\text{int}(\text{rgbi}), \text{int}(\text{rgbj})].$$

Projection onto the Occupancy Grid. For texture mapping, we focus on a horizontal ground plane where $Z \approx 0$, meaning points near this plane are considered floor points. These projected points are mapped onto the occupancy grid, and each grid cell is assigned an RGB color based on the closest projected point falling within it.

E. Loop Closure Detection

Loop closure detection is essential in SLAM to mitigate the accumulation of localization drift and ensure long-term consistency in mapping. Our approach employs two complementary methods: fixed-interval loop closure detection and proximity-based loop closure detection. These methods help detect previously visited locations and incorporate constraints into the SLAM framework for improved trajectory estimation.

Fixed-interval loop closure detection. This method operates by evaluating loop closures at fixed intervals. After recording a predefined number of N of robot poses, we check the relative pose between the current pose p_i and the previous pose p_{i-N}. A loop closure is considered if the relative pose obtained by ICP can align two LiDAR scans well, i.e. the average distance between the aligned points is less than a predefined threshold:

$$\min_{R,t} \|Rp_i + t - p_j\|_2^2 < \epsilon.$$

A constraint is added to the factor graph for optimization if this loop closure is confirmed.

Proximity-based loop closure detection. Unlike the fixed-interval approach, proximity-based loop closure continuously (after a fixed small time interval) monitors the spatial relationship between the current robot pose and all previously recorded poses. The time interval between loop detections is subject to the computation time constraints. If the robot's current pose p_i is within a predefined proximity threshold of an earlier pose p_j, we compare the associated LiDAR scans to obtain their relative poses. Similar to the fixed-interval approach, if the ICP alignment error is smaller than a threshold ϵ, a loop constraint is added to the factor graph.

F. Pose Graph Optimization

Once a loop closure is detected, it is incorporated into the SLAM framework using factor graph optimization. The pose graph consists of nodes representing robot poses and edges encoding motion constraints. Loop closure constraints introduce additional edges, enforcing consistency across different parts of the trajectory and reducing accumulated drift. The optimization problem is formulated as follows:

$$\min_x \sum_{(i,j)\in\mathcal{E}} \phi_{i,j}(e(x_i, x_j)),$$

where $e(x_i, x_j)$ is the error and ϕ_{ij} is the cost function. The Levenberg-Marquardt algorithm is commonly used to solve this optimization problem. In this project, we use GTSAM for the graph optimization. By incorporating loop closure constraints into the pose graph and optimizing the trajectory, the SLAM system significantly improves global localization accuracy, enabling consistent mapping over extended trajectories.

IV. Results

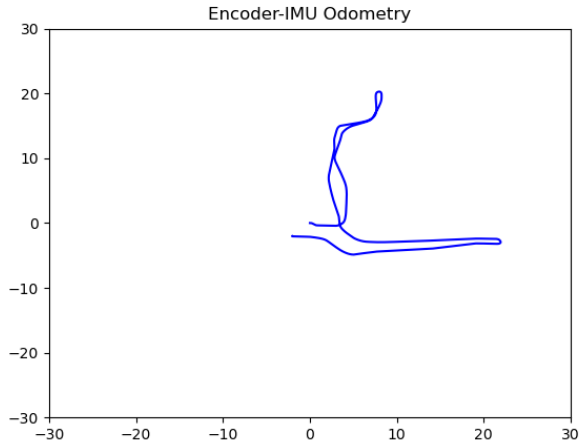
In the sections below, we will use results from training sets 20 and 21 as illustrative examples. Full results are in the "./figures" subfolder of the submitted code. Please see "README.md" for a more detailed explanation.

A. Motion Model

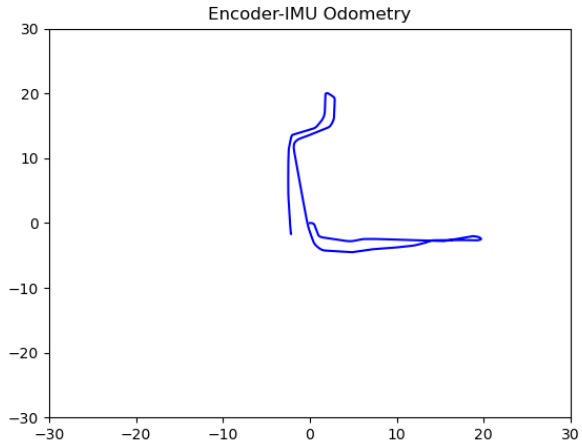
In the initial experiments, we evaluated the robot's pose estimation using only the motion model, relying solely on data from the Inertial Measurement Unit (IMU) and wheel encoders, combined with a differential drive kinematic model. As depicted in Fig 1, the estimated trajectory appears smooth and continuous, leading to an occupancy map that provides a rough representation of the environment's structure.

However, despite its ability to capture the general layout of the surroundings, the constructed map exhibits noticeable artifacts, including ghosting effects. These distortions primarily arise due to the absence of loop closure detection and global pose optimization, causing cumulative drift in the estimated trajectory. Without these correction mechanisms, inconsistencies accumulate over time, leading to a misalignment between overlapping map sections.

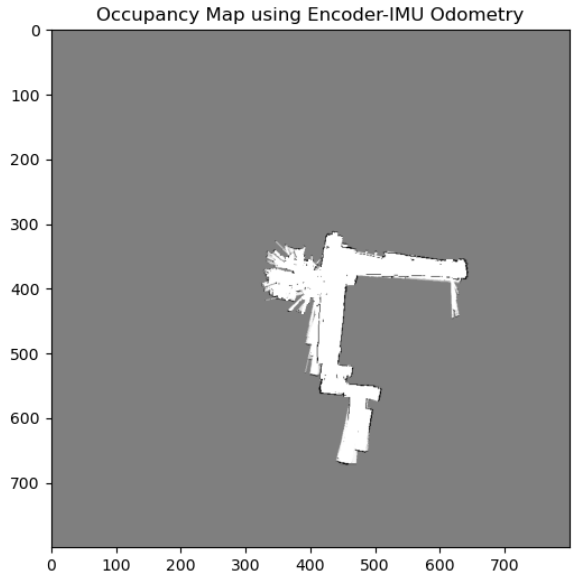
In subsequent sections, we address these limitations by incorporating loop closure detection and global optimization techniques to enhance map accuracy and consistency.



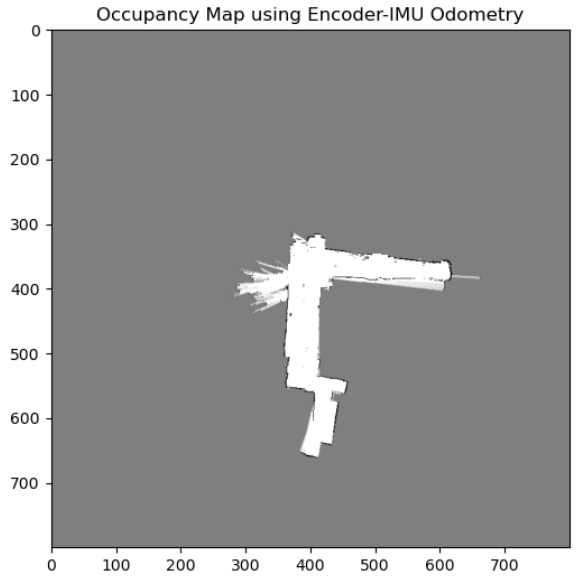
(a) Robot Trajectory of Dataset 20



(b) Robot Trajectory of Dataset 20



(c) Occupancy Map of Dataset 20



(d) Occupancy Map of Dataset 21

Fig. 1: The robot pose trajectories and occupancy maps estimated purely relied on the encoder data and the IMU data.

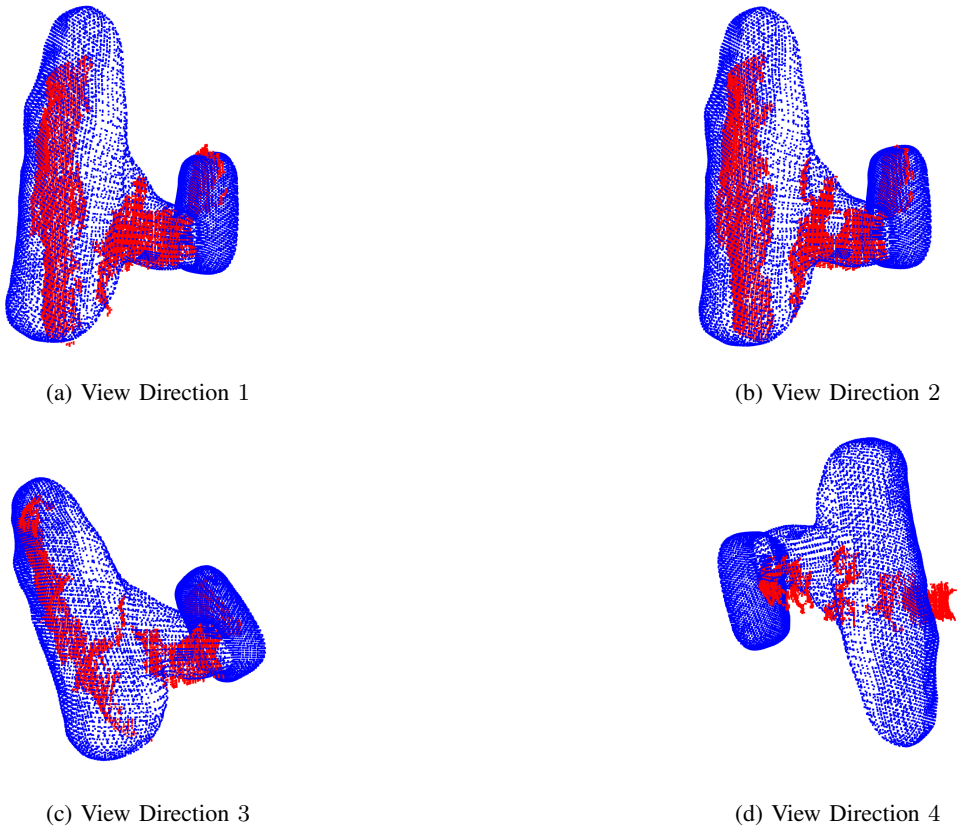
B. Evaluation of the Iterative Closest Point (ICP) Algorithm

To assess the performance of the ICP algorithm, we conducted experiments using two sets of 3D point clouds. We randomly sampled 100 initial poses for each run and chose the results with the lowest mean distance between the target point cloud and the aligned source point cloud.

As illustrated in Fig 2, the results make sense for the first 3 viewpoints, while in the last one, the points association is not correct. The last viewpoint is pretty challenging because the source points are much less than the target ones, making it difficult to find the correct association. Fig 3 showed similar patterns, which means that although the ICP implementation is correct, there are some certain limitations. One major drawback is its tendency to converge to local minima, particularly in cases where the initial alignment is poor. Further analysis revealed that in certain cases, one point cloud represented a full object while the other captured only a partial view, violating the closest-point assumption fundamental to ICP. This discrepancy underscores the need for highly accurate initial pose estimates to achieve globally optimal alignment and prevent misalignment due to local minimum convergence.

C. LiDAR Scan Matching

In this project, we applied the Iterative Closest Point (ICP) algorithm for scan matching using 2D LiDAR data. The



(a) View Direction 1

(b) View Direction 2

(c) View Direction 3

(d) View Direction 4

Fig. 2: The point cloud alignment of the object drill. In some cases, the points are not aligned well because the ICP algorithm is sensitive to pose initialization and it's hard to find a proper one.

evaluation was conducted in two stages: (1) single-frame scan matching without an initial pose estimate, and (2) single-frame scan matching incorporating pose estimates derived from the motion model.

As illustrated in Fig 4, the results on dataset 20 indicate that while ICP successfully captured the motion trend predicted by the motion model, significant drift accumulation resulted in highly inaccurate trajectories and a distorted occupancy map. Introducing initial pose estimation substantially reduced drift, though minor deviations persisted compared with the motion model.

For dataset 21, as shown in Fig 5, a similar pattern was observed; however, the extent of drift was more pronounced since the dataset is more challenging. While initial pose estimation helped mitigate trajectory deviation, the overall precision remained lower than that achieved using the motion model. A key observation was that drift was most prominent during the early stages of mapping.

D. Loop Closure and Pose Graph Optimization

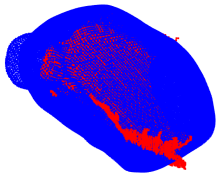
In this phase of the project, we added loop closure and pose graph optimization to optimize the poses globally. We explored two kinds of loop closure strategies, fixed interval loop closure and proximity-based loop closure, as mentioned above.

As depicted in Fig 6 and Fig 7, adding loop closure significantly improved the performance of LiDAR scan matching-based pose estimation and mapping. While fixed interval loop closure results are still not as good as those from the motion model, in the second case where proximity-based loop closure is used, loop closure detection successfully identified that the starting and ending points of the robot's trajectory were in close proximity. Utilizing ICP, a relative transformation between these poses was estimated and incorporated into the pose graph. The optimization significantly improved the consistency of the generated map, effectively eliminating duplicate structures and ghosting artifacts. This refinement resulted in an almost flawless reconstruction of the environment, demonstrating the effectiveness of loop closure constraints in correcting accumulated drift.

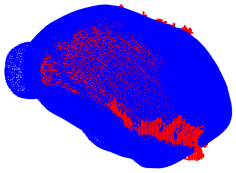
Overall, integrating loop closure detection with global pose graph optimization significantly enhanced both the accuracy of the reconstructed map and the reliability of the estimated trajectory. These results highlight the importance of incorporating global optimization techniques in SLAM to ensure consistent and drift-free localization over extended operations.

E. Texture Mapping

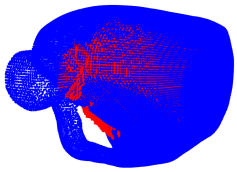
As a final step in our mapping process, we applied texture coloring to the occupancy grid to enhance its visual repre-



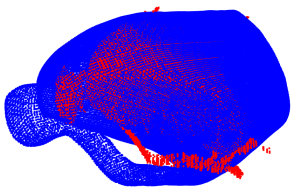
(a) View Direction 1



(b) View Direction 2



(c) View Direction 3



(d) View Direction 4

Fig. 3: The point cloud alignment of the object liq container. In some cases, the points are not aligned well because the ICP algorithm is sensitive to pose initialization and it's hard to find a proper one.

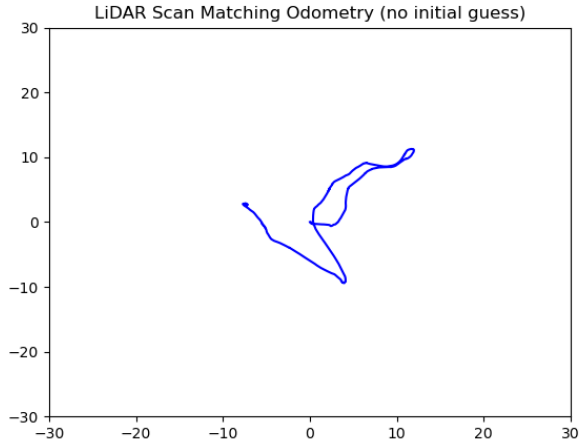
sentation, as illustrated in Fig 8. This approach resulted in a visually enriched representation of the environment.

Beyond aesthetic enhancement, this coloring process provides additional contextual information that improves environmental recognition and situational awareness. By incorporating RGB data into the occupancy grid, the map becomes more informative for robotic navigation, enabling better decision-making and path planning in complex environments.

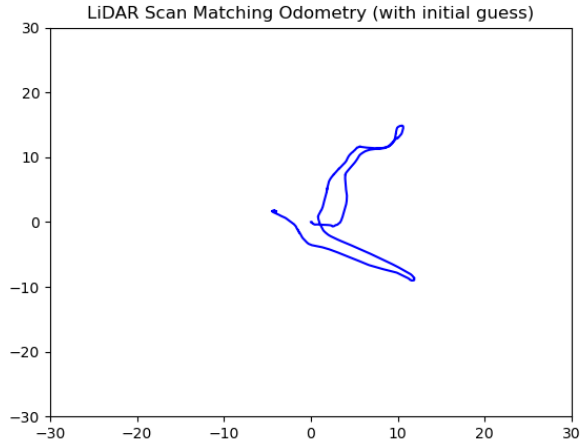
V. Conclusion

In this project, we successfully developed and implemented a LiDAR-based SLAM system for 2D robotic navigation, integrating motion estimation, scan matching, loop closure detection, and global pose optimization. By combining odometry data with the Iterative Closest Point (ICP) algorithm and enhancing localization through loop closure constraints, we significantly improved trajectory accuracy and mapping consistency. Additionally, texture mapping using RGBD data provided a visually enriched representation of the environment, further enhancing interpretability for robotic applications.

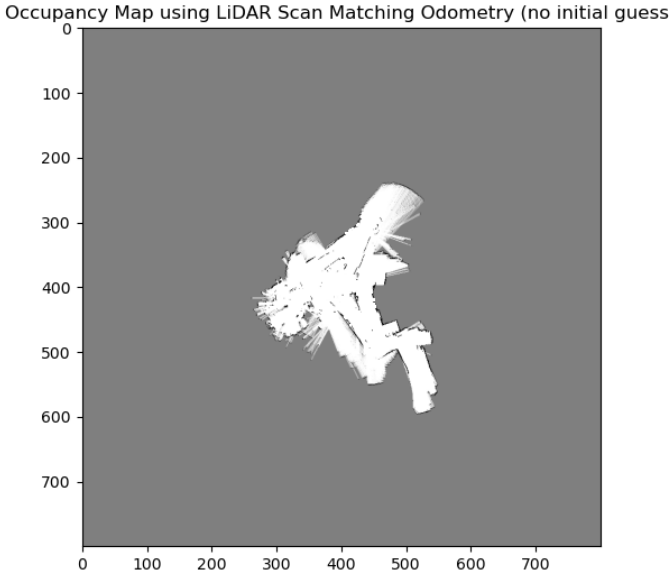
Experimental results demonstrated that while ICP and loop closure detection effectively mitigate trajectory drift, certain challenges remain in feature-deprived environments where reliable loop detection is difficult. Future work will focus on refining scan matching algorithms, improving feature recognition for loop closure, and integrating additional sensor modalities such as visual odometry to further enhance SLAM performance. Overall, this study highlights the effectiveness of multi-sensor fusion and graph-based optimization in achieving robust and accurate SLAM for autonomous robotic systems.



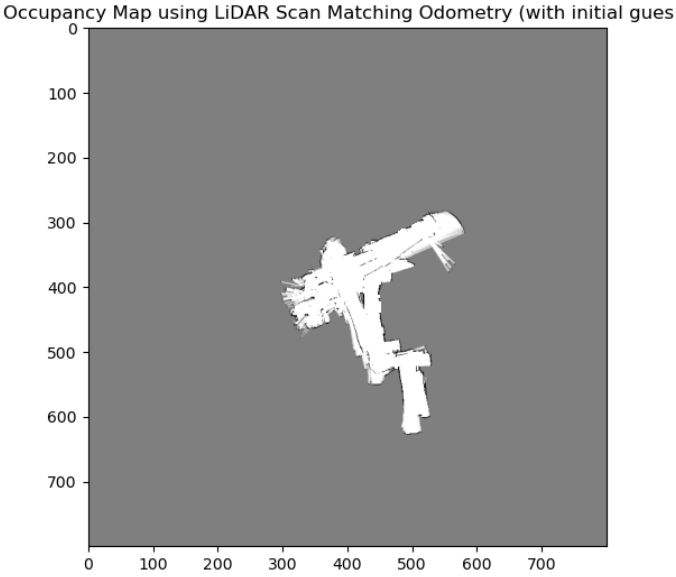
(a) Trajectory of Dataset 20 (no initial guess)



(b) Trajectory of Dataset 20 (with initial guess)

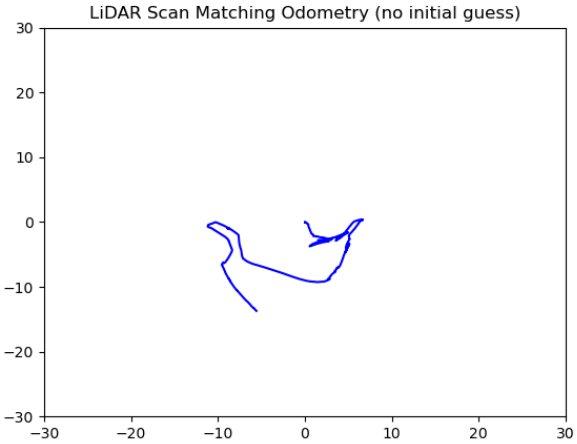


(c) Map of Dataset 20 (no initial guess)

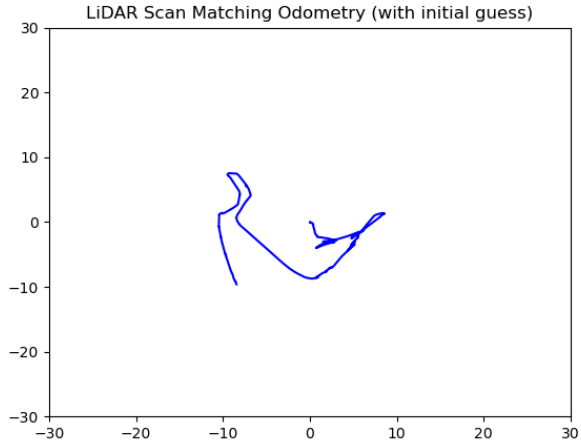


(d) Map of Dataset 20 (with initial guess)

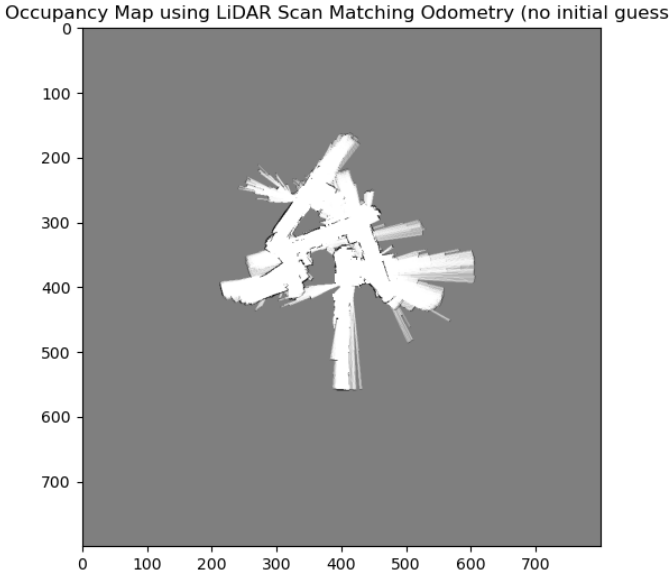
Fig. 4: The robot pose trajectories and occupancy maps for dataset 20 obtained by LiDAR scan matching. The left column shows the results using an identical matrix as an initial guess of poses, while the second column shows the results using the pose obtained from the motion model as the initial guess.



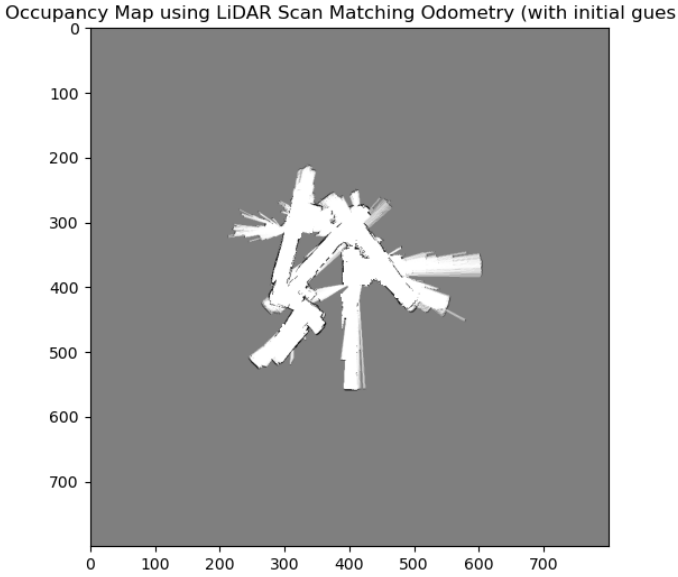
(a) Trajectory of Dataset 21 (no initial guess)



(b) Trajectory of Dataset 21 (with initial guess)

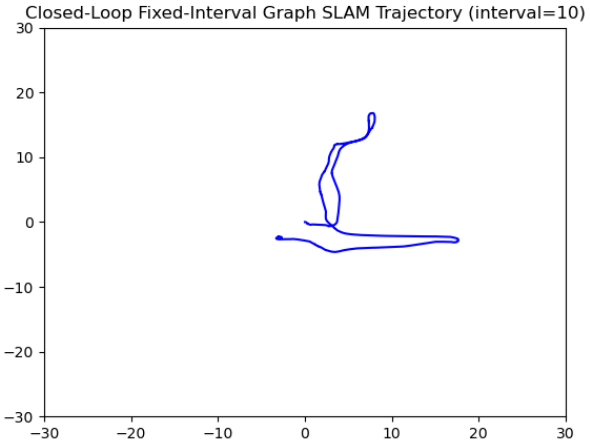


(c) Map of Dataset 21 (no initial guess)

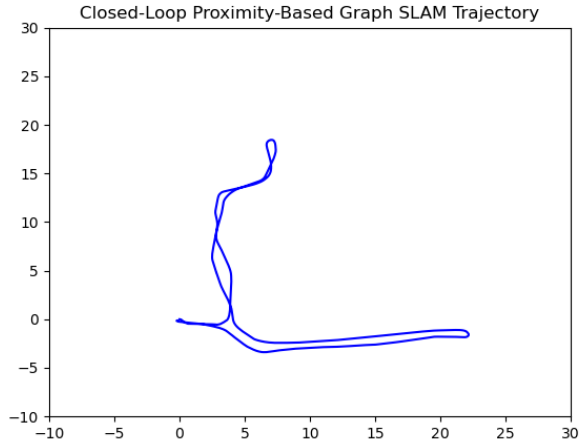


(d) Map of Dataset 21 (with initial guess)

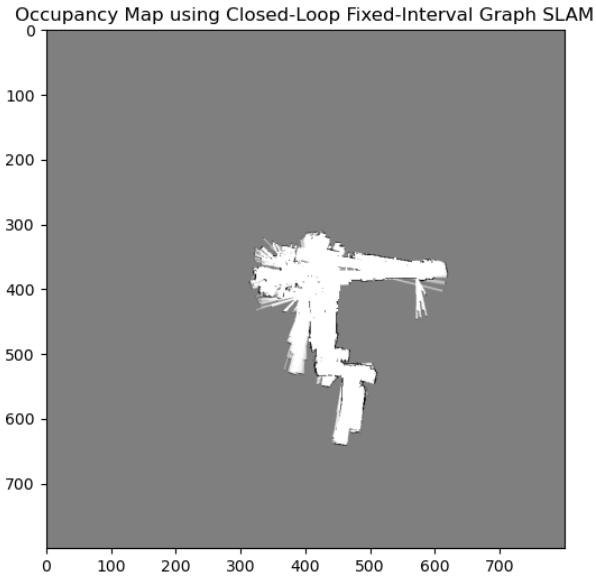
Fig. 5: The robot pose trajectories and occupancy maps for dataset 21 obtained by LiDAR scan matching. The left column shows the results using an identical matrix as an initial guess of poses, while the right column shows the results using the pose obtained from the motion model as the initial guess.



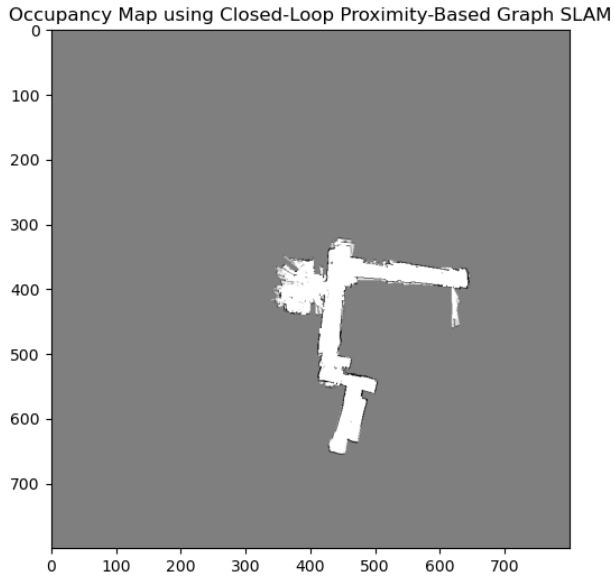
(a) Trajectory of Dataset 20 (fixed)



(b) Trajectory of Dataset 20 (proximity)

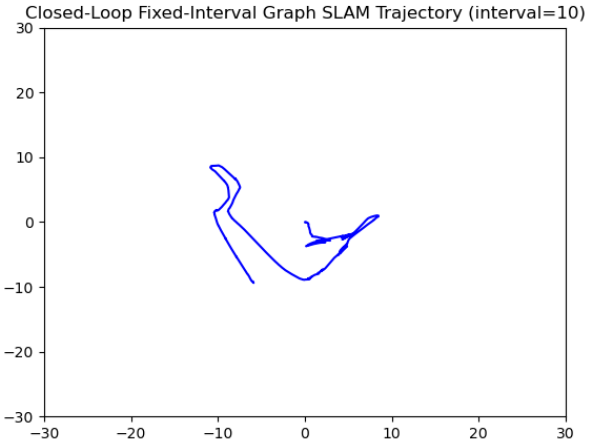


(c) Map of Dataset 20 (fixed)

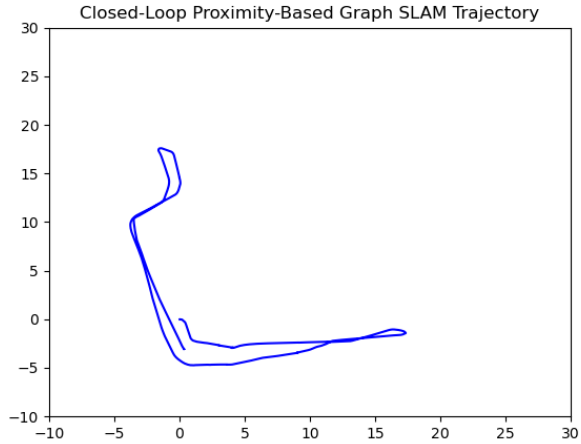


(d) Map of Dataset 20 (proximity)

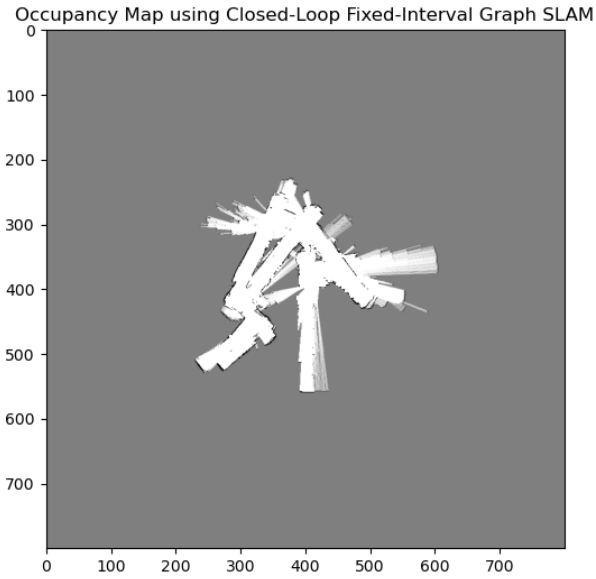
Fig. 6: The robot pose trajectories and occupancy maps for dataset 20 obtained by graph optimization and loop closure. The left column shows the results obtained using fixed interval loop closure, while the right column shows the results obtained using proximity-based loop closure.



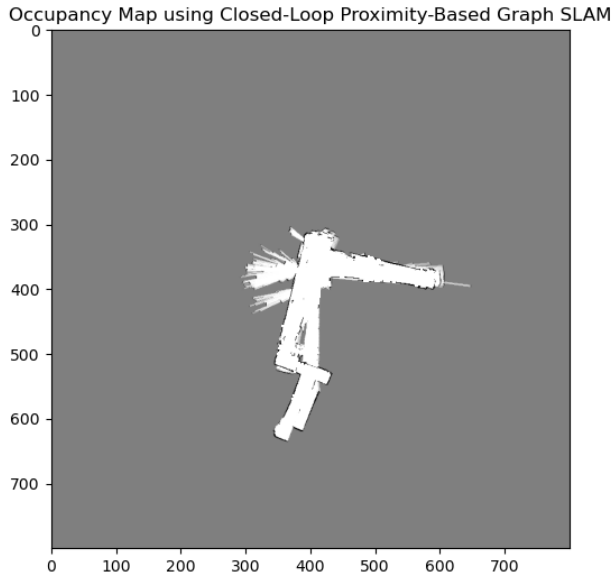
(a) Trajectory of Dataset 21 (fixed)



(b) Trajectory of Dataset 21 (proximity)

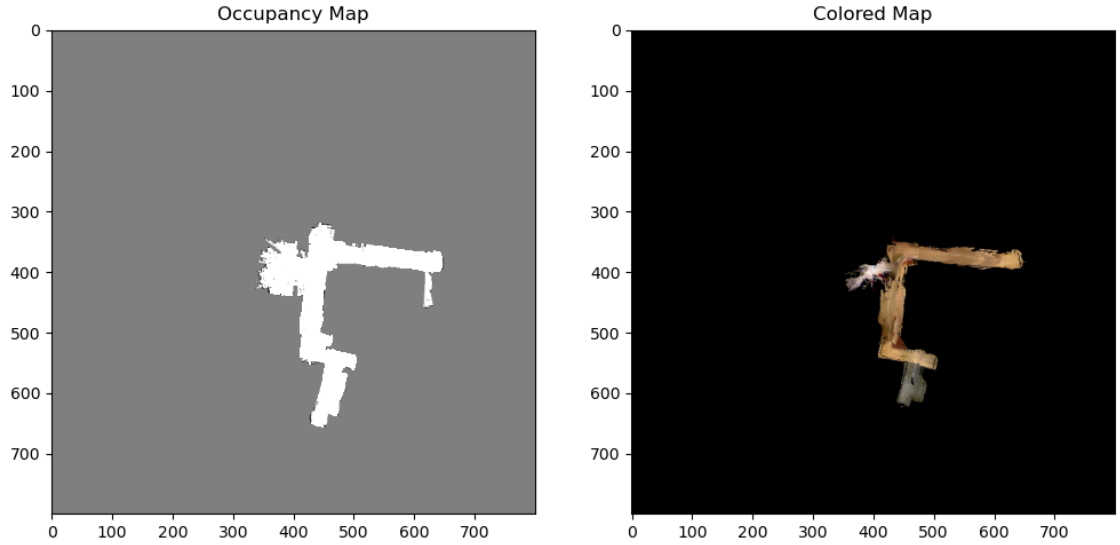


(c) Map of Dataset 21 (fixed)

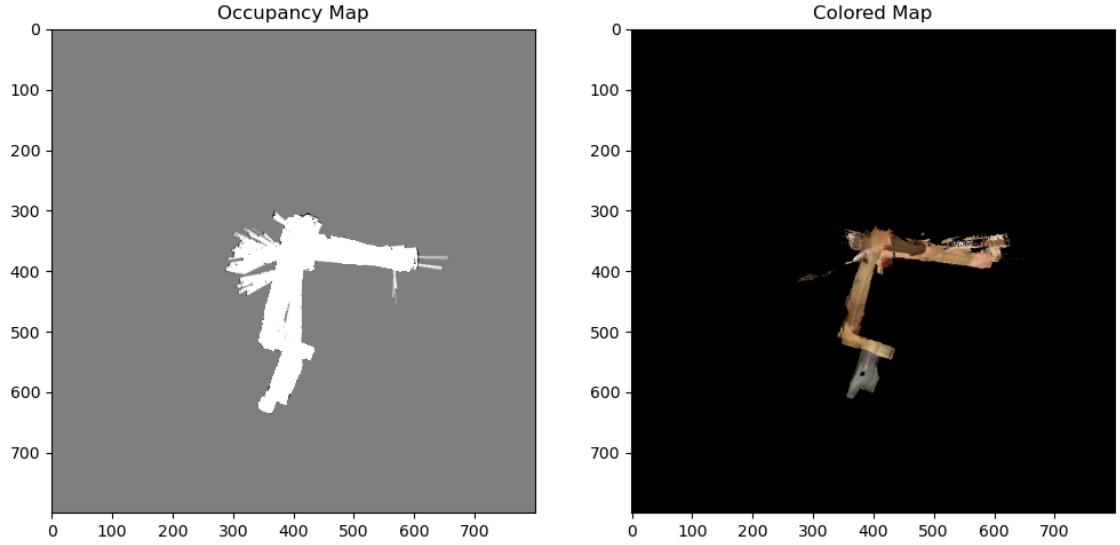


(d) Map of Dataset 21 (proximity)

Fig. 7: The robot pose trajectories and occupancy maps for dataset 21 obtained by graph optimization and loop closure. The left column shows the results obtained using fixed interval loop closure, while the right column shows the results obtained using proximity-based loop closure.



(a) Colored Map of Dataset 20



(b) Colored Map of Dataset 21

Fig. 8: The occupancy maps with texture mapping. The map is generated using proximity-based loop closure and graph optimization.