Project 2: LiDAR-Based SLAM

Mazeyu Ji, PID:A59023027

***Abstract*—In this project, we explore the challenges of implementing simultaneous localization and mapping (SLAM) for a robot using encoders, IMU, 2-D LiDAR scans, and RGBD measurements. Our goal is to accurately estimate the robot's position and orientation within an unknown environment while constructing a detailed map of the surroundings. We adopted a comprehensive approach, starting with initial position estimation using encoder and IMU data, followed by precise alignment of LiDAR point clouds using the Iterative Closest Point (ICP) algorithm to enhance localization accuracy. Furthermore, we applied loop closure detection and used the GTSAM library for pose graph optimization, improving the accuracy and robustness of the SLAM system. Finally, we added textures to the constructed map using RGBD images for a richer environmental representation. This work not only demonstrates an efficient implementation of SLAM but also provides valuable insights for future research.**

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

I. Introduction

In this project, we delve into Simultaneous Localization and Mapping (SLAM), a pivotal technology in robotics, enabling autonomous navigation and exploration in completely unknown environments. The core challenge of SLAM is accurately estimating the robot's position while concurrently mapping the environment. Our approach integrates multiple sensors, such as encoders, Inertial Measurement Units (IMU), and RGBD sensors, to overcome the limitations and inaccuracies of single-sensor reliance. These sensors' data, after meticulous processing, are transformed into key factors for the graph optimization process, aiding in precise localization and mapping.

Utilizing a differential drive model to process data from IMUs and encoders is crucial for generating initial pose estimates. These estimates are then fed into the Iterative Closest Point (ICP) algorithm as inputs to refine and optimize the robot's final position. For mapping, we employ a log-odds-based probabilistic model to update and maintain the occupancy grid map, enriching the map's detail and visualization by coloring it with ground point clouds captured by RGBD sensors.

To enhance the map's accuracy and reliability further, we systematically detect loop closure events and correct the entire map through graph optimization techniques. This step is key to ensuring accuracy in long-duration navigation and large-scale environmental mapping.

Looking ahead, our SLAM system's design is flexible, expected to integrate more advanced sensor technologies like visual odometry and Real-Time Kinematic (RTK), offering robust technical support for solving more complex navigation and mapping tasks. Additionally, this multi-sensor integration strategy not only enhances the system's adaptability and robustness but also opens new possibilities for future robotics technology development, especially in applications like autonomous driving and remote exploration.

II. Problem Formulation

In this project, our primary goal is to address the challenge of accurately determining the location and orientation of a robot within a specific environment over time, utilizing data from a LiDAR sensor, odometry information, and Inertial Measurement Unit (IMU) readings. Additionally, we aim to use these estimates to generate a comprehensive 2-D occupancy grid map that represents the environment. The mathematical formulation of this problem is articulated through a sequence of exact equations, which detail the process of integrating sensor inputs to achieve precise localization and mapping outcomes.

Motion Estimation: This research aims to estimate the robot's pose given input data. The input consists of the velocities of the left and right wheels at each moment (v_L, v_R) and the turning yaw rate (ω_t) provided by the IMU. The output objective is the robot's pose at each moment, represented by x_t, y_t, θ_t, where x_t and y_t denote the robot's position in the plane, and θ_t represents the robot's rotational angle relative to the starting orientation.

Point Cloud Registration: Given two sets of points in $\mathbb{R}^d$ space, $\{m_j\}$ and $\{z_j\}$, the task is to find a transformation $p \in \mathbb{R}^d$ and a rotation $R \in SO(d)$, as well as a data association Δ, that aligns the two point sets:

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

Occupancy Mapping: We have a sequence of lidar measurements $P = \{p_i | p_i \in \mathbb{R}^3\}$ obtained over time, with each p_i denoting a point in space measured by the lidar sensor. The objective is to use these measurements to estimate an occupancy grid $m \in \mathbb{R}^n$, which is a discretized representation of the environment. Each element of the grid, m_i, indicates the occupancy status of the corresponding cell, with the values of -1 for free and +1 for occupied.

The construction of the occupancy grid involves processing the lidar data along with the robot's pose to update the probability $\gamma_{i,t}$ for each cell, which reflects the likelihood of the cell being occupied at a given time t. This probability is conditioned on both the current and past sensor measurements $Z_{0:t}$ and the robot's trajectory $X_{0:t}$, assuming independence between cells given the robot's trajectory. The overall probability of the map is the product of probabilities of individual cells:

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

The goal is to continuously refine the occupancy grid as new measurements are made, to achieve a detailed and accurate map of the environment.

Texture Mapping: Given a set of RGBD images, where each pixel provides a color value (R, G, B) and a depth reading d, we aim to compute a mapping function f that projects these pixels onto a 2-D grid corresponding to the floor plane. The function f is defined to transform the depth reading d at each pixel coordinate (i, j) into a 3-D point in the robot's world frame, and then to extract the relevant color information for points that lie on the floor plane. Formally, the transformation can be expressed as:

$$f(d, i, j) \rightarrow (X_{\text{world}}, Y_{\text{world}}, RGB)$$

Here, the depth d is converted to a real-world z coordinate using a depth-to-distance conversion formula derived from the camera's intrinsic parameters. This real-world point is then mapped onto a horizontal plane (e.g., $Z = 0$) to identify floor points. The output of function f is the set of world coordinates $(X_{\text{world}}, Y_{\text{world}})$ that correspond to the floor, along with their associated RGB color values, effectively creating a textured representation of the environment's floor in the robot's world frame.

Loop Closure Detection: In Loop Closure Detection within SLAM systems, the goal is to identify when a robot revisits a previously encountered location by recognizing a loop closure event between poses p_i and p_j, where $p_i = (x_i, y_i, \theta_i)$ and $p_j = (x_j, y_j, \theta_j)$ represent the robot's position and orientation at times i and j respectively. The detection criteria can be succinctly formulated as finding i, j such that:

$$D(p_i, p_j) < \epsilon$$

where $D(p_i, p_j)$ is a distance metric (e.g., Euclidean distance) between the poses, and ϵ is a predefined threshold indicating sufficient proximity for a loop closure.

Pose graph Optimization: Given a set of robot poses $\{T_1, T_2, ..., T_n\}$ and a set of relative pose measurements $\{(R_{ij}, p_{ij}) | (i, j) \in \mathcal{E}\}$, where $\mathcal{E}$ is the set of edges connecting the poses in the pose graph, the problem of Pose Graph Optimization can be formulated as:

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

Where:

- $T_i \in SE(3)$ is the transformation matrix representing the i-th pose.
- T_{ij} is the relative transformation from pose i to pose j.
- W_{ij} is the information matrix for the edge (i, j).
- $\log(\cdot)$ is the matrix logarithm mapping $SE(3)$ to $\mathfrak{se}(3)$.
- $\| \cdot \|^2$ denotes the squared Frobenius norm.

III. Technical Approach

A. Differential-Drive Kinematic Model

We use the Differential-Drive Kinematic Model to compute the robot's pose from the input data. First, we calculate the average of the left and right wheel velocities v_L, v_R to obtain the linear velocity $v = \frac{v_R + v_L}{2}$, and then we use the yaw rate ω provided by the IMU. Pose updating is performed using the discrete-time differential-drive model, which employs the Euler discretization method to update the robot's state at each time interval τ_t. The formulas for pose updating are:

$$x_{t+1} = x_t + \tau_t v \cos(\theta_t)$$

$$y_{t+1} = y_t + \tau_t v \sin(\theta_t)$$

$$\theta_{t+1} = \theta_t + \tau_t \omega$$

B. Iterative Closest Point (ICP)

Kabsch Algorithm for Known Data Association:

The objective is to find the transformation $p \in \mathbb{R}^d$ and rotation $R \in SO(d)$ between the sets $\{m_j\}$ and $\{z_j\}$ of associated points:

$$\min_{R\in SO(d), p\in\mathbb{R}^d} f(R, p) := \sum_i w_i \left\| (Rz_i + p) - m_i \right\|_2^2$$

The optimal translation is obtained by setting the gradient of $f(R, p)$ with respect to p to zero:

$$0 = \nabla_p f(R, p) = 2 \sum_i w_i ((Rz_i + p) - m_i)$$

Let the point cloud centroids be:

$$\bar{m} := \frac{\sum_i w_i m_i}{\sum_i w_i}, \quad \bar{z} := \frac{\sum_i w_i z_i}{\sum_i w_i}$$

Solving $\nabla_p f(R, p) = 0$ for p leads to:

$$p = \bar{m} - R\bar{z}$$

To find the optimal rotation, we substitute p with $\bar{m} - R\bar{z}$ in $f(R, p)$, redefine the centered point clouds as $\delta m_i := m_i - \bar{m}$ and $\delta z_i := z_i - \bar{z}$, and minimize:

$$\min_{R\in SO(d)} \sum_i w_i \left\| R\delta z_i - \delta m_i \right\|_2^2$$

The objective function simplifies to:

$$\sum_i w_i \left\| R\delta z_i - \delta m_i \right\| * 2^2 =$$

$$\sum_i w_i \left((\delta z_i)^T R^T R \delta z_i - 2\delta m_i^T R \delta z_i + \delta m_i^T \delta m_i \right)$$

This leads to Wahba's problem, which is solved by maximizing:

$$\max_{R\in SO(d)} \text{tr}(Q^T R)$$

where $Q := \sum_i w_i \delta m_i \delta z_i^T$. The solution to Wahba's problem is obtained using Singular Value Decomposition (SVD) of Q:

$$Q = U \Sigma V^T$$

We then determine R as:

$$R = UWV^T$$

where W is a diagonal matrix that is chosen to ensure that $R \in SO(d)$, specifically ensuring that $\det(R) = 1$ to avoid improper rotations.

Iterative Closest Point (ICP) for Unknown Data Association:

For the transformation p, R between sets $\{m_j\}$ and $\{z_j\}$ with an unknown data association Δ, the ICP algorithm iterates between finding associations Δ based on the closest points and applying the Kabsch algorithm to determine p, R:

1. Initialize with p_0, R_0 (sensitive to initial guess) and iterate 2 and 3.

2. Given p_k, R_k, find correspondences $(i, j) \in \Delta$ based on the closest points:

$$i \leftrightarrow \arg\min_j \|m_i - (R_k z_j + p_k)\|_2$$

3. Given correspondences $(i, j) \in \Delta$, find p_{k+1}, R_{k+1} via the Kabsch algorithm.

The ICP algorithm continues these steps until convergence, typically when the changes in p and R between iterations fall below a predetermined threshold.

C. Occupancy mapping

In the field of robotic mapping, the method of Probabilistic Occupancy Grid Mapping using LiDAR data serves as a useful technique to represent an environment. The approach utilizes a vector $m \in \mathbb{R}^n$, where each element m_i corresponds to a grid cell and holds a value indicating the occupancy state: +1 for occupied and -1 for free. The probability mass function $p(m|Z_{0:t}, X_{0:t})$ is maintained over time to reflect the belief about the occupancy of each cell given the robot's trajectory $X_{0:t}$ and sensor observations $Z_{0:t}$.

To update the occupancy belief, each cell is modeled as an independent Bernoulli random variable. The log-odds form $\lambda_{i,t}$ is used to simplify the calculation, where the log-odds of a cell being occupied at time t is updated based on the sensor measurements Z_t and the map's prior at time $t-1$. The log-odds ratio is calculated as follows:

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

The update rule for the log-odds is:

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

This equation incorporates both the inverse sensor model $p(Z_t|m_i, X_t)$ and the prior odds $p(m_i = 1)/p(m_i = -1)$. The inverse sensor model captures the likelihood of sensor readings given the state of the cell, and the prior odds encapsulate prior knowledge about the environment.

In order to avoid overconfident estimations, the updated log-odds values are constrained within a range $[\lambda_{\text{MIN}}, \lambda_{\text{MAX}}]$. Furthermore, to account for changing environments, a decay term can be introduced to the log-odds over time.

Finally, the occupancy probability $\gamma_{i,t}$ for each cell is recovered from the log-odds value using the logistic function, providing a value between 0 and 1 that represents the probability of occupancy:

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

This process iteratively updates the occupancy grid as new LiDAR scans are obtained and processed, making it an essential technique for constructing detailed and dynamic maps for autonomous robots. The utilization of such probabilistic mapping methods allows for the handling of uncertainties and inaccuracies inherent in sensor measurements, leading to a more robust and reliable mapping solution.

D. Texture Mapping

To texture-map an occupancy grid using RGBD images, we apply a series of transformations to project color data onto the grid. For a depth pixel at coordinates (i, j), with depth d, we compute the corresponding real-world depth z using the formula (the parameters are given by the instructions):

$$z = \frac{1.03}{-0.00304 \cdot d + 3.31}$$

Next, we obtain the real-world coordinates (X, Y, Z) from the image coordinates (i, j), utilizing the depth z and the camera's intrinsic parameters K, as follows:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = z \cdot K^{-1} \cdot \begin{pmatrix} i \\ j \\ 1 \end{pmatrix}$$

These coordinates are then transformed from the camera frame to the world frame, taking into account the rotation R and translation T from the camera to the robot's coordinate system, as well as the robot's pose:

$$\begin{pmatrix} X_{\text{world}} \\ Y_{\text{world}} \\ Z_{\text{world}} \end{pmatrix} = R \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T$$

For the texture mapping, we focus on a horizontal plane at $Z = 0$ (the floor plane). Points with Z values near zero, within a certain threshold, are projected onto this plane. The grid map is populated with the RGB values of these points, where each grid cell's color corresponds to the color of the projected point falling within it. This process results in a textured grid map that visually represents the floor texture in addition to the occupancy data.

E. Loop Closure Detection

For the method of loop closure detection within SLAM, we combined two different methods to identify and verify the loop closures:

Fixed-interval loop closure is applied after a predetermined number of robot poses, N, have been recorded. For every N^{th} pose p_i, we search for a previous pose p_j such that the condition $D(p_i, p_j) < \epsilon$ is satisfied. Here, D is a distance

function measuring the similarity between the two poses, often computed using the ICP (Iterative Closest Point) algorithm:

$$D_{\text{ICP}}(p_i, p_j) = \min_{R,t} \|Rp_i + t - p_j\|$$

where R and t are rotation and translation applied to the pose p_i to align it with pose p_j. If $D_{\text{ICP}}(p_i, p_j) < \epsilon$, a loop closure is detected and added to the factor graph as a constraint.

Proximity-based loop closure involves continuously monitoring the proximity of the robot's current pose to all previous poses. For the current pose p_i and a previous pose p_j, the LiDAR scans associated with these poses are compared. If the scans match closely, it suggests a loop closure. The matching function can be represented as:

$$S(p_i, p_j) > \tau$$

where S is a scan-matching score function, and τ is a threshold for determining a good match. When a high score is achieved indicating a match, a loop closure constraint is added to the factor graph. In ou project, S is just the distance error from the ICP algorithm as the fixed-interval loop closure detection.

These loop closure constraints are then incorporated into the factor graph and processed through optimization algorithms to reconcile the loop closures with the current map, reducing cumulative error and improving the trajectory estimate.

F. Pose Graph Optimization

Factor Graph Optimization is a powerful method used in the context of SLAM for refining the estimates of the variables $x = [x_1^T \ ... \ x_n^T]^T$, which represent robot poses or landmark positions. The optimization is framed as a minimization problem over a graph where each factor represents a measurement or a constraint.

In this optimization, the cost function to be minimized is a sum of error terms associated with each constraint in the graph:

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

where $e(x_i, x_j)$ is the error between the estimated state x_i and the measured state x_j, and $\phi_{ij}(e)$ is a cost function, often chosen as a quadratic function of the error to represent the Mahalanobis distance of the error term weighted by the information matrix W_{ij}.

The optimization begins with an initial guess $x^{(0)}$, which is typically obtained from odometry and initial landmark positions. The descent method iteratively updates the variable estimates $x^{(k+1)}$ by moving in the direction of the negative gradient of the cost function, scaled by a step size $\alpha^{(k)}$:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \delta x^{(k)}$$

The Levenberg-Marquardt algorithm, a robust variant of the Gauss-Newton algorithm, is commonly used for this optimization. It adjusts the variables by solving the following linear system:

$$(\sum_{ij} J_{ij}^T W_{ij} J_{ij} + \lambda D)\delta x^{(k)} = -\sum_{ij} J_{ij}^T W_{ij} e(x_i^{(k)}, x_j^{(k)})$$

where J_{ij} is the Jacobian of the error function with respect to the variables x, evaluated at $x = x^{(k)}$, and λ is a damping factor that is adjusted dynamically during the optimization process.

The use of GTSAM [1] is notable in this context as it provides an efficient and flexible framework for solving Factor Graph Optimization problems. GTSAM leverages these principles and provides implementations of various solvers, including the Levenberg-Marquardt algorithm, to facilitate the development of SLAM and other nonlinear optimization applications.

IV. Results

In this chapter, we will present the results of all processes involved in the SLAM procedure, including motion model, scan match using the ICP algorithm, loop closure detection and global pose graph optimization, and the construction of occupancy maps and map coloring. We meticulously document the entire process from initial position estimation to the final map construction, demonstrating how the system precisely locates itself in an unknown environment and builds a detailed environmental map. Furthermore, dynamic demonstrations of the map construction are provided, with the results accessible via provided links dataset 20 results [2] and dataset 21 results [3]. These results not only showcase the efficacy of our system but also validate the effectiveness and accuracy of the technologies employed.

A. Motion Model

We initially conducted experiments using only the motion model to calculate the robot's pose, exclusively utilizing the Inertial Measurement Unit (IMU) and encoders in combination with a differential drive model for pose estimation. As shown in the Fig. 1, the poses provided by the motion model exhibit smooth and fluid characteristics, and the occupancy map constructed based on these poses roughly reflects the overall appearance of the terrain. However, the constructed map is not perfect, presenting certain ghosting phenomena. This is primarily due to the absence of loop closure detection and global pose optimization, resulting in a lack of consistency in the map. In the subsequent sections, we will attempt to address this issue by introducing loop closure detection and global optimization.

B. ICP Test

In our project, we first conducted tests on the Iterative Closest Point (ICP) algorithm with two sets of 3D point clouds. We began by decentralizing the point clouds and downsampling them to equal quantities. Then, we applied ICP matching in two ways: directly without any initial pose guesses, and by generating nine rough initial pose guesses around the z-axis at intervals of 40 degrees, selecting the

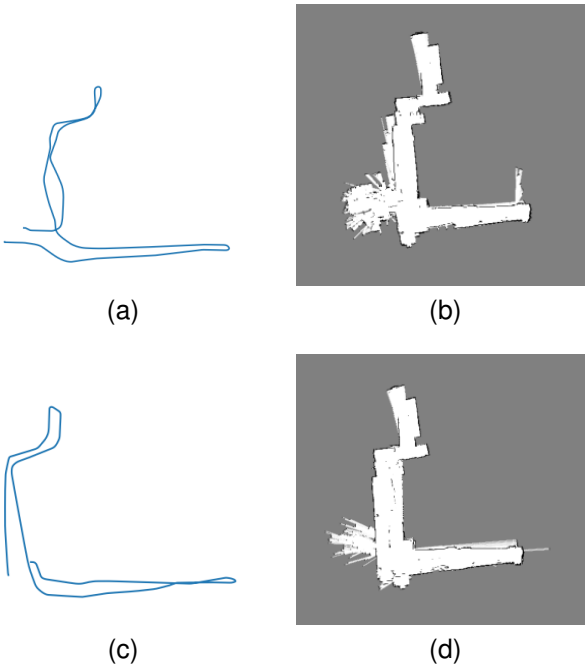

Fig. 1. The left two images display the trajectories obtained using the motion model on datasets 20 and 21, while the right two images correspondingly show the occupancy maps constructed based on these trajectories.

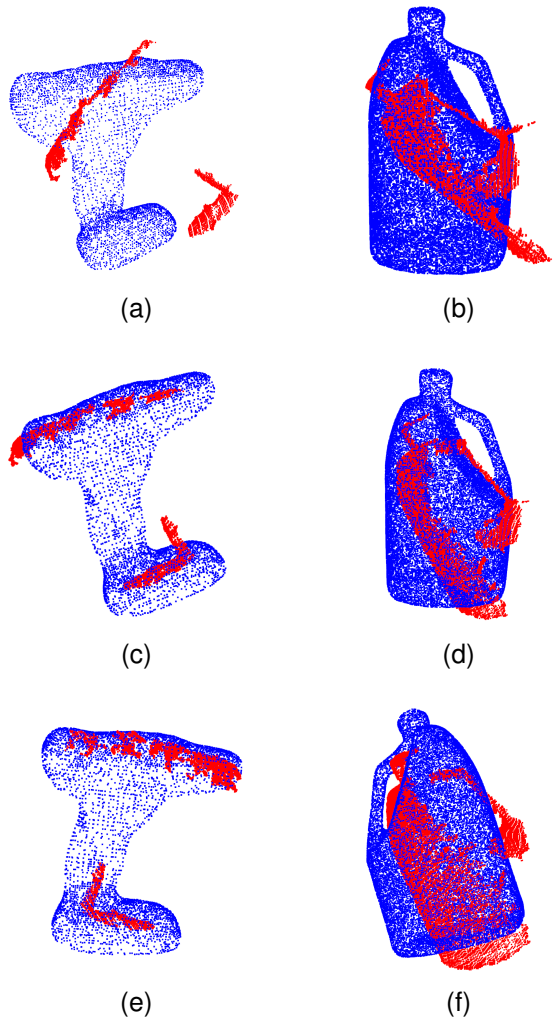

Fig. 2. The left three images show the original point cloud position relationships of the drill, the relationship obtained by the Iterative Closest Point (ICP) algorithm without using initial position estimation, and the results using the ICP with an initial guess, respectively. The right three images display the corresponding situations for the container point cloud.

match with the smallest average distance error as the result. As is shown in the Fig. 2, the outcomes demonstrated that initial pose guesses significantly impact the accuracy of ICP matching. For instance, the average distance error for the drill point cloud reduced from 0.017 to 0.015, and for the container dataset from 0.032 to 0.028, indicating a significant improvement in matching precision. However, we also observed that the ICP algorithm is not perfect, as it tends to fall into local minima. Especially in the second approach, despite a reduction in average distance error, the actual match was completely incorrect. An observation revealed that one point cloud represented a complete object while the other only a part, making the closest point assumption fundamentally flawed. Achieving a globally optimal match requires highly accurate initial pose estimation.

C. LiDAR Odometry

In this project, we applied the Iterative Closest Point (ICP) algorithm to the process of scan matching with 2D LiDAR. The experiment was divided into three main parts: firstly, single-frame scan matching without initial pose estimation; secondly, single-frame scan matching with pose estimation provided by the motion model; and lastly, the addition of fixed interval loop closures, which can be considered a form of loop closure detection. We only incorporated these as factors in subsequent optimizations if the matches met predefined threshold criteria.

First let us analyze the results of dataset 20 as is shown in the Fig. 3. Testing on dataset 20 revealed that while ICP provided a motion trend similar to that of the motion model, significant drift led to highly inaccurate trajectories and poor map quality. Introducing initial pose estimation essentially mitigated drift, although minor drifting still occurred. Further incorporating fixed interval loop closures brought the precision of ICP-based odometry on par with the motion model.

For dataset 21 as is shown in the Fig. 4, the trend was similar, but drift was more severe. Although initial pose estimation and fixed interval loop closures improved drift, the precision was still lower compared to the motion model. Therefore, we plan to adjust the credibility of factors for better results. Additionally, drift was observed mainly at the beginning of mapping, especially in a straight corridor area, where extreme environmental degradation rendered ICP unable to provide accurate estimates.

D. Global Pose Graph Optimization

In this part of the project, we incorporated Iterative Closest Point (ICP) factors obtained from LiDAR, along with motion model factors derived from Inertial Measurement Units (IMU)

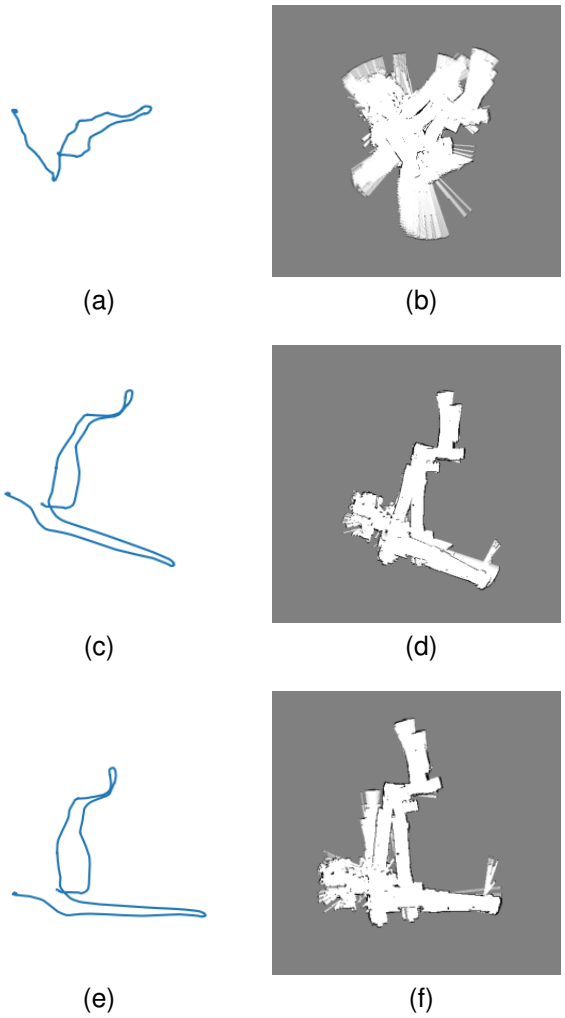

Fig. 3. The left three images respectively show the estimated trajectories of the dataset 20 by the ICP algorithm under three different conditions. The first image displays the trajectory obtained without using initial pose estimation for single-frame ICP. The second image shows the trajectory obtained with initial pose estimation for single-frame ICP. The third image further includes fixed loop closure in the ICP estimation. The right three images present the occupancy maps corresponding to the trajectories on the left, from top to bottom, according to different ICP estimation conditions.

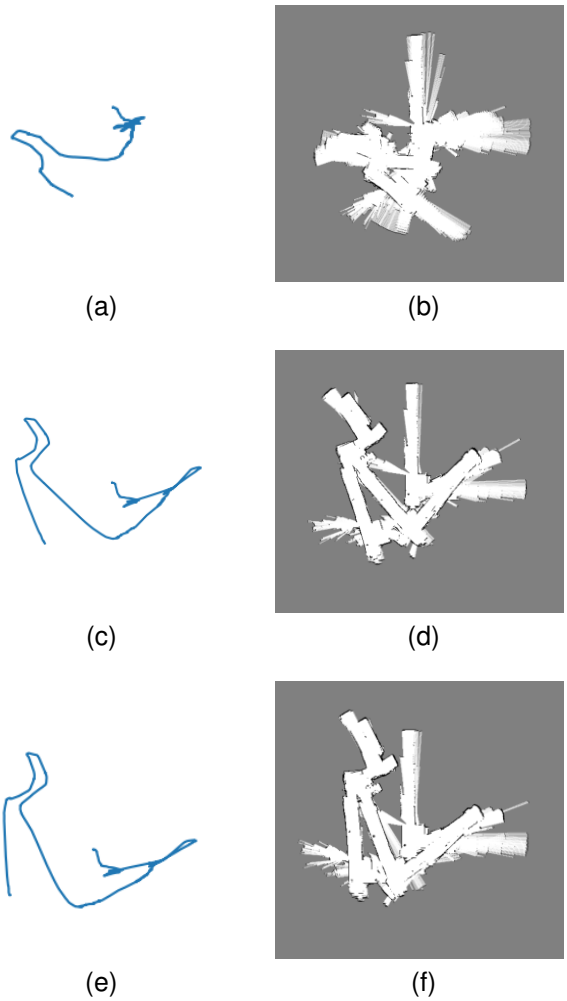

Fig. 4. The left three images respectively show the estimated trajectories of the dataset 21 by the ICP algorithm under three different conditions. The first image displays the trajectory obtained without using initial pose estimation for single-frame ICP. The second image shows the trajectory obtained with initial pose estimation for single-frame ICP. The third image further includes fixed loop closure in the ICP estimation. The right three images present the occupancy maps corresponding to the trajectories on the left, from top to bottom, according to different ICP estimation conditions.

and encoders, into the global pose graph. Additionally, loop closure detection was introduced, followed by global pose optimization. As is shown in the Fig. 5, for dataset 20, loop closure detection identified the starting and ending points of the robot's path as being very close. Through ICP, a relative transformation was computed and integrated into the pose graph for optimization, significantly improving map consistency and eliminating duplicate shadows, achieving an almost perfect mapping result. For dataset 21, a loop near the end and midway was detected, optimizing the trajectory effectively. However, initial loops were not detected, leading to some duplication at the start. Given the indistinct features at the beginning, effective loop detection is challenging and prone to errors. To address degenerate environments, optimizing the frontend scan matching algorithm is essential. Overall, this work proved invaluable, enhancing both the accuracy of mapping and trajectory.

E. Texture Mapping Results

Ultimately, we applied coloring to the occupancy map. Initially, by utilizing RGBD images, we calculated the spatial position of point clouds and filtered out ground points based on the Z-axis coordinate, as illustrated in the Fig. 6. Subsequently, we transformed the point cloud to the occupancy map's context. During the mapping process, we colored the free spaces with RGB colors. Through this method, we eventually obtained a beautifully textured map, as depictedin the Fig. 7. This coloring process not only enhances the map's visual appeal but also improves environmental recognition, offering more information for robotic navigation.

V. Conclusion

In this project, we have showcased the entire process and results of a Simultaneous Localization and Mapping (SLAM) system implemented using encoders, Inertial Measurement

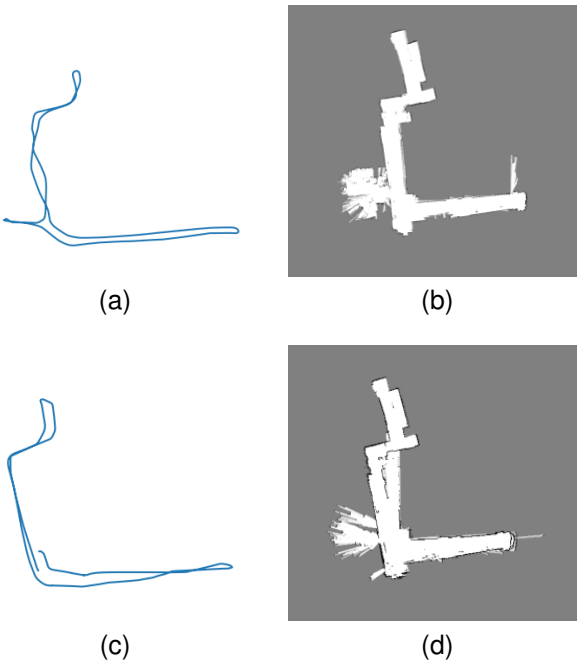

Fig. 5. The left two images display the trajectories obtained on datasets 20 and 21, while the right two images correspondingly show the occupancy maps constructed based on these trajectories.

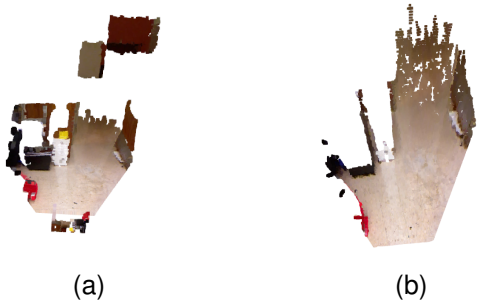

Fig. 6. The left image shows the point cloud generated from RGBD images while the right image displays the processed point cloud data with only the ground points remaining.

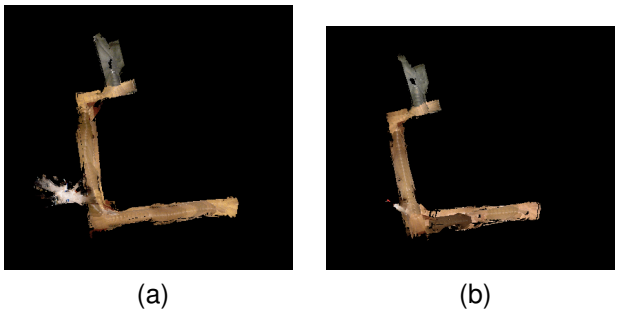

Fig. 7. The left colored occupancy map is generated from the dataset 20 and the right one if from the dataset 21.

Units (IMU), 2D LiDAR scans, and RGBD measurements. Through the application of key technologies such as motion models, scan matching with the ICP algorithm, loop closure detection, global pose graph optimization, occupancy map construction, and map coloring, we meticulously documented the entire journey from initial position estimation to final map construction. Dynamic demonstrations and result links further validate the efficacy of our system, demonstrating the effectiveness and accuracy of the technologies used. This work not only demonstrates the efficiency of SLAM implementation but also provides valuable insights for future research.

REFERENCES

[1] Frank Dellaert and GTSAM Contributors. *GTSAM: Georgia Tech Smoothing and Mapping*. May 2022. Version 4.2a8. Georgia Tech Borg Lab. DOI: 10.5281/zenodo.5794541. https://github.com/borglab/gtsam.
[2] Mazeyu Ji. Results of the dataset 20 [Video]. https://github.com/jimazeyu/img_lib/blob/main/276pr2_ds20.gif.
[3] Mazeyu Ji. Results of the dataset 21 [Video]. https://github.com/jimazeyu/img_lib/blob/main/276pr2_ds21.gif.