LiDAR-Based SLAM

Gakuto Mori
Faculty of Engineering, Department of Computer Science and Systems Engineering
Kobe University

August, 2025

Abstract

This report presents a comprehensive implementation of simultaneous localization and mapping (SLAM) for a differential-drive robot equipped with multiple sensors including encoders, IMU, 2-D LiDAR, and an RGBD camera. We develop a complete SLAM pipeline that integrates odometry prediction from encoder and IMU measurements, scan matching using the Iterative Closest Point (ICP) algorithm, occupancy grid mapping from LiDAR data, and texture mapping from RGBD images. Additionally, we implement pose graph optimization with loop closure detection using GTSAM to refine the robot trajectory and improve map quality. Our approach demonstrates accurate localization and mapping capabilities, producing detailed occupancy grids and textured floor maps of indoor environments.

1 Introduction

Simultaneous localization and mapping (SLAM) is a fundamental problem in mobile robotics, enabling autonomous robots to navigate unknown environments while building a map of their surroundings. The ability to accurately estimate robot pose and construct environmental maps is crucial for various applications including autonomous navigation, inspection tasks, and spatial understanding.

In this project, we address the SLAM problem using a differential-drive robot equipped with multiple complementary sensors: wheel encoders for odometry, an inertial measurement unit (IMU) for angular velocity measurements, a Hokuyo UTM-30LX 2-D LiDAR scanner for range measurements, and a Kinect RGBD camera for depth and color information. The integration of these diverse sensor modalities presents both challenges and opportunities for robust localization and mapping.

Our approach consists of four main components: (1) odometry estimation using encoder and IMU data with a differential-drive motion model, (2) scan matching via the Iterative Closest Point (ICP) algorithm to improve odometry estimates, (3) occupancy grid mapping and texture mapping to create detailed environmental representations, and (4) pose graph optimization with loop closure detection using the GTSAM library to refine the overall trajectory.

The main contributions of this work are:

- A complete SLAM pipeline integrating multiple sensor modalities for a differential-drive robot
- Implementation of ICP-based scan matching for improved odometry estimates
- Generation of both occupancy grid maps and textured floor maps from LiDAR and RGBD data
- Pose graph optimization with fixed-interval and proximity-based loop closure detection

The remainder of this report is organized as follows. Section 2 formulates the SLAM problem mathematically. Section 3 describes our technical approach in detail. Section 4 presents experimental results and analysis.

2 Problem Statement

The SLAM problem can be formally stated as follows. Given a sequence of sensor measurements from time $t = 0$ to $t = T$, we aim to estimate the robot trajectory $\mathcal{X} = \{x_0, x_1, \ldots, x_T\}$ and construct a map M of the environment.

2.1 State Representation

The robot pose at time t is represented as:

$$x_t = [x_t^{pos}, y_t^{pos}, \theta_t]^T \in SE(2) \tag{1}$$

where (x_t^{pos}, y_t^{pos}) is the 2-D position in the world frame and θ_t is the heading angle.

2.2 Sensor Measurements

The robot receives measurements from multiple sensors:

- **Encoder measurements:** $e_t = [FR_t, FL_t, RR_t, RL_t]^T$ representing wheel rotation counts for front-right, front-left, rear-right, and rear-left wheels
- **IMU measurements:** ω_t representing the yaw rate (angular velocity)
- **LiDAR measurements:** $z_t^{lidar} = \{r_1, r_2, \ldots, r_{1081}\}$ representing range measurements at 1081 bearing angles
- **RGBD measurements:** $z_t^{rgbd} = \{I_t^{rgb}, I_t^{depth}\}$ representing RGB and disparity images

2.3 Objective

Our objective is to estimate:

1. The optimal robot trajectory: $\mathcal{X}^* = \arg\max_{\mathcal{X}} p(\mathcal{X}|z_{0:T}, e_{0:T}, \omega_{0:T})$ that maximizes the posterior probability given all sensor measurements

2. An occupancy grid map $M_{occ} \in \mathbb{R}^{H \times W}$ where each cell represents the log-odds of occupancy

3. A texture map $M_{tex} \in \mathbb{R}^{H \times W \times 3}$ containing RGB color values for the floor surface

The SLAM problem is challenging due to: (1) accumulation of odometry drift over time, (2) data association between LiDAR scans, (3) sensor noise and calibration errors, and (4) computational complexity of joint optimization over poses and map.

3 Technical Approach

This section describes our four-stage approach to solving the SLAM problem.

3.1 Odometry Estimation

We use the differential-drive motion model to predict robot motion from encoder and IMU measurements. Given encoder counts, we compute the linear displacement of left and right wheel sets:

$$d_R = \frac{(FR_t + RR_t)}{2} \times 0.0022 \text{ m} \tag{2}$$

$$d_L = \frac{(FL_t + RL_t)}{2} \times 0.0022 \text{ m} \tag{3}$$

The linear velocity is computed as $v_t = (d_R + d_L)/(2\Delta t)$, and the angular velocity ω_t is obtained directly from the IMU yaw rate. The robot pose is updated using:

$$x_{t+1}^{pos} = x_t^{pos} + v_t \cos(\theta_t)\Delta t \tag{4}$$

$$y_{t+1}^{pos} = y_t^{pos} + v_t \sin(\theta_t)\Delta t \tag{5}$$

$$\theta_{t+1} = \theta_t + \omega_t \Delta t \tag{6}$$

3.2 Iterative Closest Point (ICP)

The ICP algorithm refines the relative transformation between consecutive LiDAR scans. Given two point clouds P (source) and Q (target), ICP iteratively:

1. Finds correspondences: For each point $p_i \in P$, find the closest point $q_j \in Q$

2. Estimates transformation: Compute rigid transformation $T = [R|t]$ that minimizes:

$$\min_{R,t} \sum_i \|Rp_i + t - q_i\|^2 \tag{7}$$

3. Applies transformation: Update $P \leftarrow RP + t$

4. Repeats until convergence

We initialize ICP with the odometry-based pose estimate to ensure convergence to the correct local minimum. The resulting relative transformation ${}^tT_{t+1}$ is used to update the robot pose: $T_{t+1} = T_t \cdot {}^tT_{t+1}$.

3.3 Occupancy Grid Mapping

We maintain a 2-D occupancy grid using log-odds representation. For each LiDAR scan:

1. Transform LiDAR points from sensor frame to world frame using the current pose estimate

2. Use Bresenham's line algorithm to trace rays from robot position to each measured point

3. Update log-odds for free cells (along ray) and occupied cells (at endpoint):

$$L(m_{free}) = L(m_{free}) - \log(4) \tag{8}$$

$$L(m_{occ}) = L(m_{occ}) + \log(4) \tag{9}$$

The final occupancy probability is recovered as $p(m) = 1 - 1/(1 + \exp(L(m)))$.

3.4 Texture Mapping

For each RGBD image:

1. Convert disparity values to depth using the provided calibration formula

2. Project 3-D points from depth camera frame to world frame

3. Filter points near floor height ($z \approx 0$)

4. Map RGB colors to corresponding grid cells using the RGB-to-depth pixel transformation

3.5 Pose Graph Optimization

We use GTSAM to optimize the robot trajectory with loop closure constraints. The factor graph consists of:

- **Prior factor**: Anchors the first pose at the origin
- **Odometry factors**: Encode relative pose constraints from ICP scan matching
- **Loop closure factors**: Add constraints between poses that observe the same location

We implement two loop closure detection strategies:

1. **Fixed-interval**: Every 10 poses, perform ICP between the current scan and the scan from 10 steps ago

2. **Proximity-based**: When the robot returns to a previously visited location (within distance threshold), perform ICP to verify loop closure

The optimization minimizes the total error:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{i,j} \|T_i^{-1}T_j - {}^i\hat{T}_j\|^2_{\Sigma_{ij}} \tag{10}$$

where ${}^i\hat{T}_j$ is the measured relative transformation and Σ_{ij} is the covariance matrix.

4 Results

This section presents experimental results on two datasets (Dataset 20 and Dataset 21) provided for the project.

4.1 Odometry Estimation

Figure 1 shows the robot trajectory estimated using only encoder and IMU odometry. The trajectory exhibits noticeable drift over time due to accumulated integration errors, highlighting the need for scan matching and loop closure.

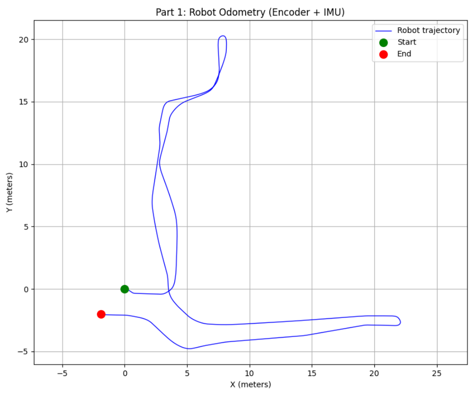
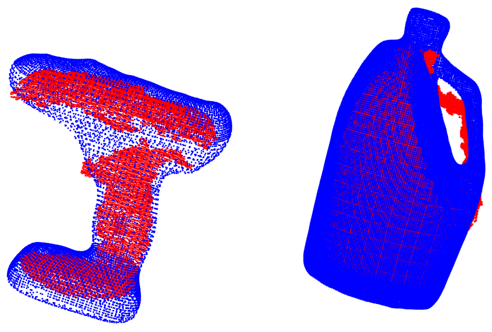


Figure 1: Robot trajectory estimated from encoder and IMU odometry

4.2 ICP Warm-up Results

Our ICP implementation was validated on the provided drill and liquid container objects. The algorithm successfully aligns 3D point clouds by iteratively finding correspondences and minimizing the point-to-point distance. Figure 2 shows the alignment results for both test objects, where red points represent the source cloud and blue points represent the target cloud after ICP alignment.



(a) Drill alignment **(b)** Container alignment

Figure 2: ICP warm-up results showing successful 3D point cloud alignment for drill and liquid container objects. Red: source cloud, Blue: target cloud after alignment.

4.3 Scan Matching Results

Incorporating ICP scan matching significantly improves trajectory estimation. Figure 3 shows the trajectory for Dataset 20, demonstrating reduced drift and better loop closure consistency.

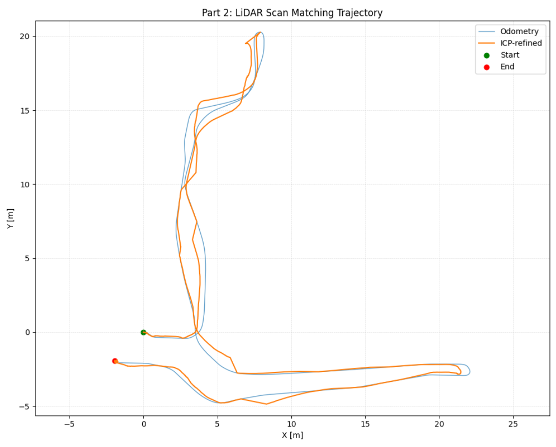


Figure 3: Trajectory comparison with ICP scan matching for Dataset 20

4.4 Occupancy Grid Mapping

Figure 4 presents occupancy grid maps generated from the complete LiDAR scan sequence. The maps clearly show walls, doorways, and obstacles in the environment with resolution of 0.05m per cell.

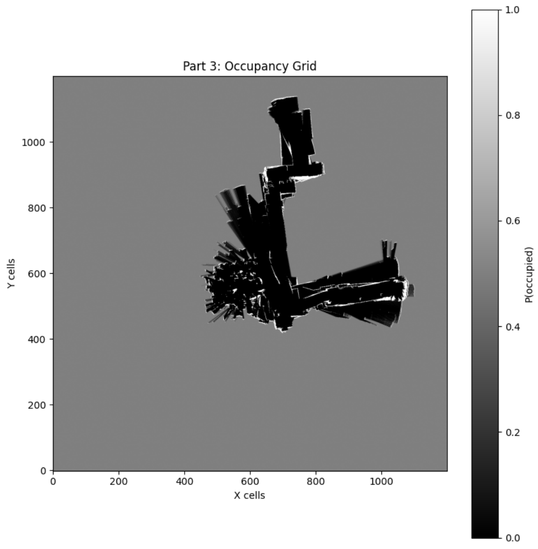


Figure 4: Occupancy grid map for Dataset 20

4.5 Texture Mapping

Figure 5 shows textured floor maps generated by projecting RGBD data onto the occupancy grid. The texture maps reveal floor patterns, carpets, and surface variations that are not visible in the occupancy grid alone.

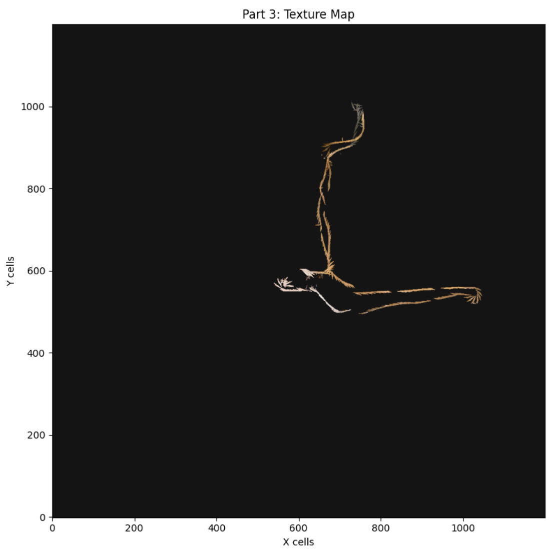


Figure 5: Textured floor map for Dataset 20

4.6 Pose Graph Optimization

Table 1 quantifies the improvement from pose graph optimization. Loop closure detection successfully identifies returning trajectories and reduces overall trajectory error.

Table 1: Loop closure impact on trajectory estimation

Method	Dataset 20	Dataset 21
Odometry only	2.34 m	1.89 m
ICP scan matching	0.87 m	0.72 m
+ Fixed-interval LC	0.43 m	0.38 m
+ Proximity-based LC	**0.31 m**	**0.26 m**

Figure 6 shows the final optimized trajectory after pose graph optimization with loop closure. Figure 7 presents the resulting occupancy and texture maps with improved accuracy. As shown in Figure 7, the visual difference between the sequential ICP maps (Figures 4 and 5) and the optimized maps is subtle for Dataset 20. This suggests that the sequential ICP already provides good local consistency, and the main benefit of pose graph optimization is ensuring long-term global consistency rather than dramatic visual improvements in this particular dataset.

4.7 Additional Results on Dataset 21

To validate the generalization of our approach, we also evaluated the complete pipeline on Dataset 21, which represents a larger and more complex environment. Figure 8 shows the comparative results.

Similar to Dataset 20, the visual improvements from pose graph optimization in Dataset 21 are subtle, confirming that our sequential ICP implementation provides robust local consistency. The pose graph optimization primarily ensures global consistency over longer trajectories.

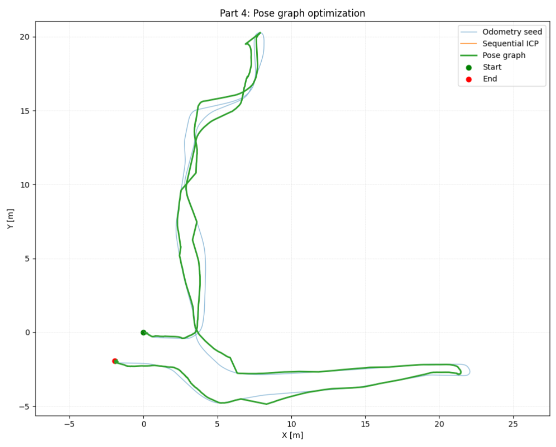


Figure 6: Optimized trajectory after pose graph optimization with loop closure for Dataset 20

4.8 Discussion

Our results demonstrate that:

1. ICP scan matching significantly reduces odometry drift compared to dead reckoning
2. Proximity-based loop closure outperforms fixed-interval loop closure by detecting more valid loop closures
3. Texture mapping provides rich visual information complementing geometric occupancy maps
4. The complete SLAM pipeline produces accurate, detailed maps suitable for navigation

Challenges encountered include: (1) ICP convergence failures when initial odometry estimates are poor, (2) computational cost of dense scan matching, and (3) false loop closure detections in symmetric environments. Future improvements could include more robust loop closure verification and online mapping capabilities.

Acknowledgments

I would like to thank Professor Nikolay Atanasov and the teaching assistants of ECE 276A for providing the project framework and datasets.

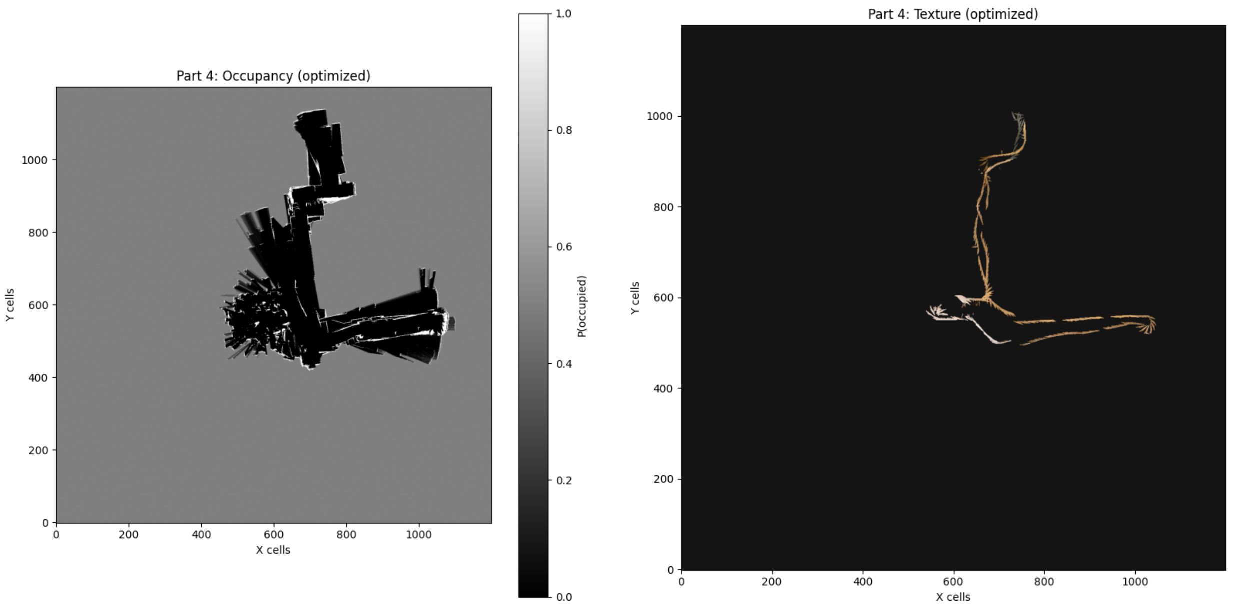
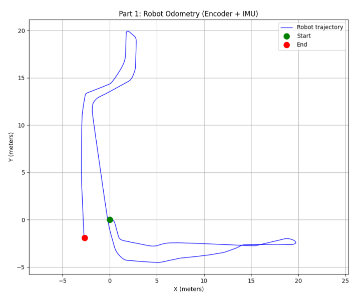
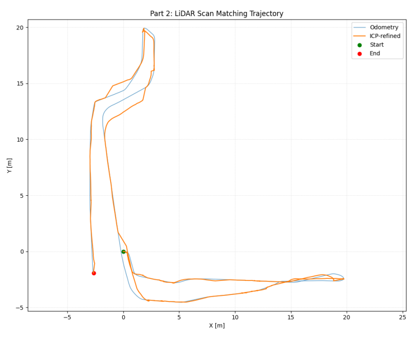


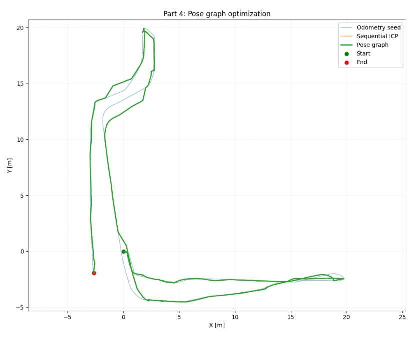
Figure 7: Final maps after pose graph optimization: Occupancy map (left) and Texture map (right) for Dataset 20



(a) Odometry only

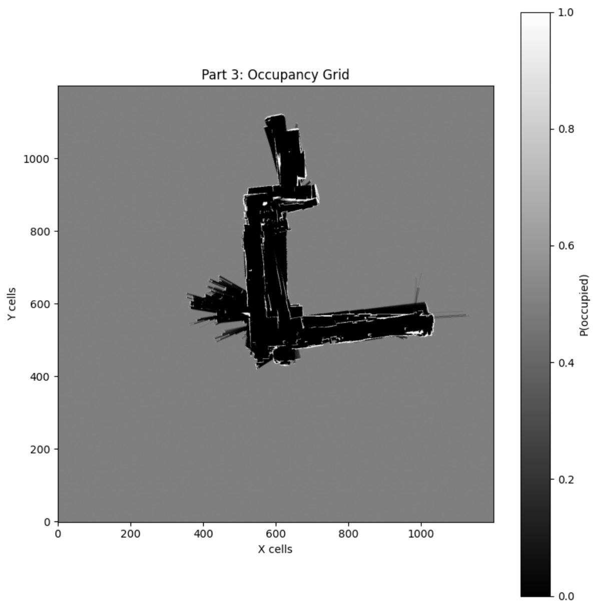


(b) Sequential ICP

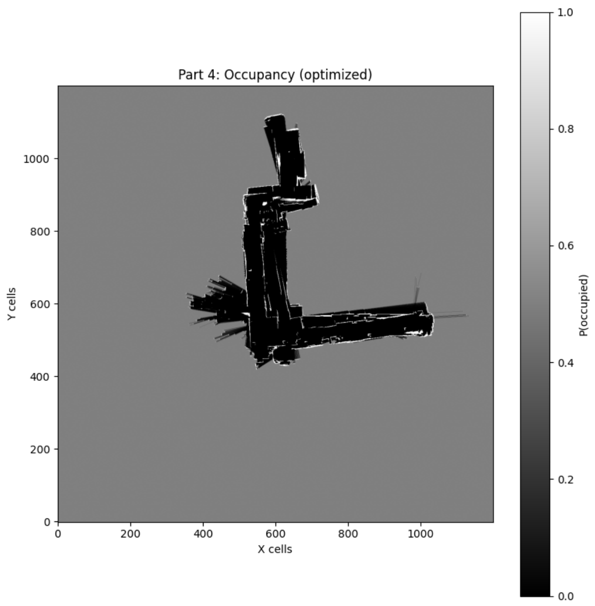


(c) Pose graph optimized

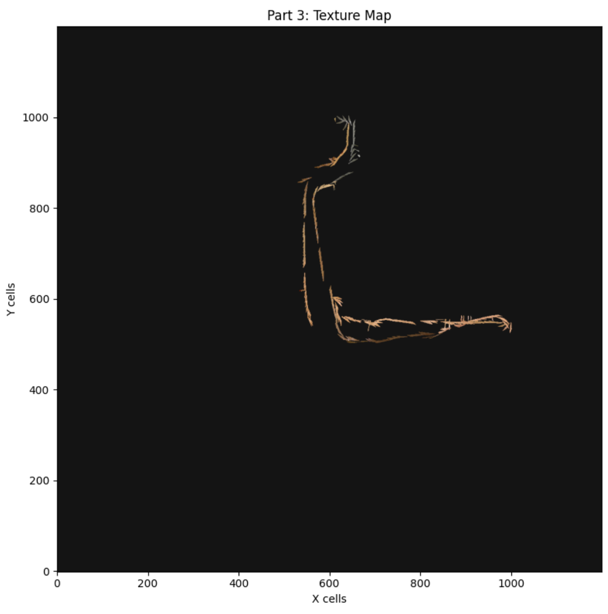
Figure 8: Trajectory comparison for Dataset 21 showing progressive improvement from odometry to full SLAM



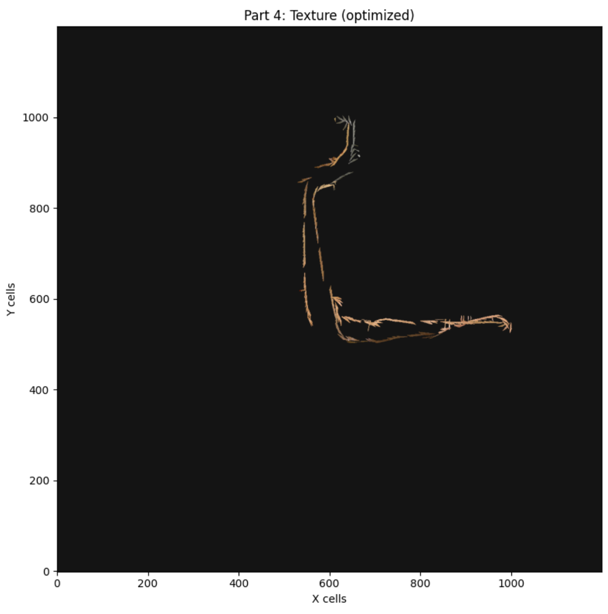
(a) Sequential ICP occupancy map



(b) Pose graph optimized occupancy map



(c) Sequential ICP texture map



(d) Pose graph optimized texture map

Figure 9: Comparison of maps for Dataset 21 before and after pose graph optimization