Factor Graph Optimization for 2D SLAM and RGB-D Texture Mapping

Siddharth Rama Sushil
Dept. of Electrical and Computer Engineering
University of California, San Diego
San Diego, USA

***Abstract*—This report presents a comprehensive pipeline for Simultaneous Localization and Mapping (SLAM) and texture mapping using a differential-drive robot equipped with wheel encoders, an IMU, a 2-D LiDAR, and an RGBD camera. We formulate the problem by estimating an initial dead-reckoning trajectory which is subsequently refined using LiDAR scan matching via the Iterative Closest Point (ICP) algorithm. Using these optimized poses, we construct a 2-D occupancy grid and project RGBD point clouds to map floor textures. To bound long-term drift, we implement a pose graph optimization framework using GTSAM, incorporating loop closure constraints based on spatial proximity, ICP fitness, and an overlap-ratio gate. Our results demonstrate that integrating local scan matching with global pose graph optimization reduces the factor graph error significantly, yielding sharp, consistent occupancy maps and photorealistic floor texture maps across the tested datasets.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in mobile robotics, enabling autonomous navigation in unknown environments without reliance on external positioning systems. This project addresses the challenge of building a coherent 2-D occupancy grid and a textured floor map using a differential-drive robot equipped with wheel encoders, an Inertial Measurement Unit (IMU), a 2-D LiDAR scanner (Hokuyo), and an RGBD Kinect camera.

Our approach proceeds in four stages. First, encoder and IMU data are fused via a differential-drive kinematic model to produce an initial dead-reckoning trajectory. Second, this trajectory is refined by sequential LiDAR scan matching using ICP, both preceded by a 3-D ICP warm-up exercise on canonical objects. Third, the refined trajectory seeds a log-odds occupancy grid and an RGBD texture projection pipeline to produce 2-D maps of the environment. Finally, a factor graph is constructed using the GTSAM library, with loop closure edges added based on spatial proximity and verified by an ICP overlap-ratio criterion; global Levenberg-Marquardt optimization then distributes and cancels accumulated drift.

II. Problem Formulation

The objective is to jointly estimate the robot's trajectory and construct a coherent environment representation. The problem is formulated mathematically across the four main project components.

A. Encoder and IMU Odometry

We are provided with control inputs $\mathbf{u}_t = [v_t, \omega_t]^T$, where v_t is the linear velocity derived from wheel encoder ticks and ω_t is the angular yaw rate measured by the IMU. The differential-drive kinematic model propagates the robot state $\mathbf{x}_t = [x_t, y_t, \theta_t]^T \in SE(2)$ over a time step τ:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

The goal is to compute the dead-reckoning trajectory $\mathcal{X}_{odom} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$.

B. Point-Cloud Registration via ICP

Given a source point cloud $\mathcal{P} = \{\mathbf{p}_i\}$ and a target $\mathcal{Q} = \{\mathbf{q}_i\}$, we seek the optimal transformation T^* minimizing alignment error:

$$T^* = \arg\min_{R,\mathbf{t}} \sum_i \|R\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2^2 \tag{2}$$

For the 3-D warm-up, point clouds derive from depth images and $T \in SE(3)$. For 2-D scan matching, consecutive LiDAR scans estimate relative transforms ${}_tT_{t+1} \in SE(2)$.

C. Occupancy and Texture Mapping

Using the estimated trajectory $\mathcal{X}$, 2-D LiDAR scans $\mathbf{z}_{1:T}$, and RGBD images $\mathcal{I}_{1:T}$ with camera intrinsics K, we map the environment. The log-odds occupancy update for cell m_i is:

$$l_i^{(t)} = l_i^{(t-1)} + \log \frac{P(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)}{1 - P(m_i = 1 \mid \mathbf{z}_t, \mathbf{x}_t)} \tag{3}$$

yielding a map $\mathcal{M} : \mathbb{R}^2 \to \mathbb{R}$. A 3-D camera-frame point $\mathbf{x}^c$ is computed from pixel disparity and projected to the world frame, generating a texture map $\mathcal{C} : \mathbb{R}^2 \to \mathbb{R}^3$.

D. Pose Graph Optimization and Loop Closure

The SLAM problem is modeled as a factor graph where nodes represent poses $\mathbf{x}_t \in SE(2)$, sequential edges encode ICP odometry constraints, and loop closure edges connect spatially proximate revisited poses. We minimize:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \left(\sum_t \|h(\mathbf{x}_t, \mathbf{x}_{t+1}) - \mathbf{z}_{t,t+1}\|^2_\Sigma + \sum_{(i,j)\in\mathcal{L}} \|h(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{z}_{i,j}\|^2_\Sigma \right) \tag{4}$$

where $h(\cdot)$ computes the expected relative pose and $\mathcal{L}$ is the set of accepted loop closures.

III. Technical Approach

A. Encoder and IMU Odometry (Part 1)

1) Velocity Computation and Sensor Synchronization: The wheel encoders report cumulative tick counts at approximately 40 Hz. The instantaneous velocity at step i is computed using the finite difference of timestamps $\Delta t_i = t_{i+1} - t_i$ and the hardware parameter of 0.0022 m/tick. Averaging the front and rear wheels on each side:

$$\begin{aligned} v_R[i] &= \frac{FR[i] + RR[i]}{2} \cdot \frac{0.0022}{\Delta t_i}, \\ v_L[i] &= \frac{FL[i] + RL[i]}{2} \cdot \frac{0.0022}{\Delta t_i} \end{aligned} \tag{5}$$

The linear velocity is $v_i = (v_R[i] + v_L[i])/2$. Because the IMU and encoders operate on independent hardware clocks, nearest-neighbor temporal matching is used: for each encoder timestamp t_i, the yaw rate ω_i is taken from the IMU at index $\arg\min |t_{IMU} - t_i|$.

2) Kinematic Update Model: With the synchronized (v_i, ω_i) pair, the state $\mathbf{x} = [x, y, \theta]^T$ is propagated using the standard differential-drive model:

$$x_{i+1} = x_i + v_i \cos(\theta_i)\, \Delta t_i, \tag{6}$$

$$y_{i+1} = y_i + v_i \sin(\theta_i)\, \Delta t_i, \tag{7}$$

$$\theta_{i+1} = \theta_i + \omega_i\, \Delta t_i. \tag{8}$$

The robot track width is $d = 0.3937$ m (hardware specification).

3) Sensor Modality Analysis: IMU vs. Encoders for Yaw: We evaluated two sources for ω: the direct IMU gyroscope and an encoder-derived estimate $\omega_{enc} = (v_R - v_L)/d$. As shown in Fig. 3, the IMU signal is smooth (± 0.6 rad/s), while the encoder-derived rate exhibits severe high-frequency noise with spikes reaching ± 1.2 rad/s from tick-count quantization. Integrating this noisy signal corrupts θ and irreversibly misprojects linear velocity onto the global axes.

B. 3D Point-Cloud Registration via ICP (Part 2a)

1) Algorithm Outline and SVD Optimization: The 3-D ICP loop aligns a source cloud $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^N$ to a target $\mathcal{T} = \{\mathbf{t}_i\}_{i=1}^M$:

1) **Correspondence:** KD-tree nearest-neighbor search for each $\mathbf{s}_i \in \mathcal{S}$.
2) **Optimal Transform:** SVD solves $\min_R \sum_i \|\tilde{\mathbf{t}}_i - R\tilde{\mathbf{s}}_i\|^2$. Given cross-covariance $H = \tilde{S}^\top \tilde{T} = U\Sigma V^\top$, the optimal rotation is $R^* = VU^\top$, with $\det(R^*) > 0$ enforced by negating the last column of V if needed.
3) **Update:** Apply R^*, $\mathbf{t}^* = \boldsymbol{\mu}_T - R^*\boldsymbol{\mu}_S$, and accumulate.
4) **Convergence:** Terminate when $\Delta\text{MSE} < \epsilon = 10^{-4}$ or 50 iterations elapsed.

2) Initialization via Yaw Discretization: ICP [3] is sensitive to large initial misalignments. For tabletop objects sharing a gravity-aligned z-axis, we try $K = 36$ uniformly-spaced yaw angles in $[0°, 350°]$ and select the run with minimum final MSE as the globally optimal alignment.

C. 2D LiDAR Scan Matching via ICP (Part 2b)

1) Data Processing and Initialization: Each Hokuyo scan provides 1081 range measurements. Points are projected to 2-D Cartesian coordinates and transformed to the robot body frame using the known extrinsic offset ($x = +0.13323$ m, $y = 0.0$ m, $\theta = 0.0$ rad). ICP is initialized with the relative transform predicted by odometry, providing a warm start to avoid local minima.

2) Dynamic ICP Algorithm:

- **Nearest-Neighbor Search:** KDTree query in $\mathcal{O}(N \log N)$.
- **Dynamic Outlier Rejection:** Correspondences with distance $> 3 \times$ median are discarded, robustly filtering moving obstacles without hard-coded thresholds.
- **SVD Alignment:** Optimal R and $\mathbf{t}$ computed from the inlier set, with $\det(R) > 0$ enforced.
- **Convergence:** Loop terminates when $\Delta\text{MSE} < 10^{-6}$. If either scan has fewer than 20 valid points, the odometry guess is used as a safe fallback.

TABLE I
2D ICP Scan Matching Parameters

Parameter	Value
LiDAR-to-body offset (x, y, θ)	0.13323 m, 0.0 m, 0.0 rad
Downsampling factor N	1 (all points)
Outlier rejection threshold	$> 3.0\times$ median distance
Convergence tolerance (ΔMSE)	10^{-6}
Degenerate scan minimum points	20

D. Occupancy and Texture Mapping (Part 3)

1) Occupancy Grid Construction: The construction of the 2-D occupancy grid is executed through a highly vectorized, four-step pipeline designed to process thousands of sequential LiDAR scans efficiently:

Dynamic Initialization: To minimize memory overhead, the map dimensions are not hard-coded. Instead, the bounding box of the entire estimated ICP trajectory $\mathcal{X}$ is computed. Given a safety padding $p = 10.0$ m and a resolution $r =$

0.05 m/cell, the grid dimensions (N_x, N_y) are dynamically calculated as:

$$N_x = \left\lceil \frac{\max(\mathbf{x}) - \min(\mathbf{x}) + 2p}{r} \right\rceil,$$
$$N_y = \left\lceil \frac{\max(\mathbf{y}) - \min(\mathbf{y}) + 2p}{r} \right\rceil$$

This yields a tightly bounded zero-initialized grid array.

Filtering and Transformation: For each scan, the temporally nearest ICP-optimized pose $\mathbf{x}_t = [x_t, y_t, \theta_t]^T$ is retrieved. Raw LiDAR ranges d_k at angles ϕ_k are filtered to remove invalid returns and aggressively clamped to a maximum effective range of 8.0 m. This capping is crucial to suppress "starburst" artefacts caused by long, unobstructed rays. The valid points are converted from polar to Cartesian coordinates and transformed into the global frame using the sensor extrinsics T_S^B and the $SE(2)$ robot pose T_B^W:

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = T_B^W T_S^B \begin{bmatrix} d_k \cos\phi_k \\ d_k \sin\phi_k \\ 1 \end{bmatrix}$$

Vectorized Rasterization: The continuous global coordinates (x_w, y_w) are discretized into grid cell indices (u, v) mapping to the array:

$$u = \left\lfloor \frac{x_w - x_{min}}{r} \right\rfloor, \quad v = \left\lfloor \frac{y_w - y_{min}}{r} \right\rfloor$$

Initially, we implemented Bresenham's line algorithm to strictly trace individual rays from the robot's optical center to each LiDAR endpoint. However, iterating through up to 1081 rays per scan in Python proved prohibitively slow due to the massive redundancy of multiple rays repeatedly updating the same proximal free-space cells. To resolve this computational bottleneck, we transitioned to a polygon-filling approach. The robot's cell index and the valid LiDAR hit indices are stacked to form the vertices of a 2-D scanner wedge. Using OpenCV's highly optimized `fillPoly` function, this entire wedge is rasterized into a binary mask in a single C++ pass.

Log-Odds Update: The map is updated using a log-odds probabilistic model. For all cells m_i within the generated OpenCV mask (representing free space), the log-odds value l_i is decremented by $l_{miss} = -0.35$. Because `fillPoly` marks the entire polygon including the boundary, the LiDAR endpoint cells inherently receive this miss penalty. To correct this, we execute a targeted array addition, incrementing only the specific endpoint hit cells by the difference:

$$l_i \leftarrow l_i + (l_{hit} - l_{miss})$$

where $l_{hit} = +0.85$. This elegantly achieves the intended net update of $l_i \leftarrow l_i + l_{hit}$ for obstacles while keeping the operation fully vectorized. Finally, the entire map is clamped to $[-10.0, +10.0]$ to prevent over-confidence and saturation.

2) Texture Mapping via Kinect RGBD Projection: The floor texture pipeline processes each Kinect disparity frame through an 8-step projection chain. To ensure the pipeline could process thousands of high-resolution frames tractably, the entire sequence was implemented using vectorized matrix operations and boolean masking, avoiding computationally prohibitive pixel-wise loops.

Time synchronization: The Kinect sensors capture RGB and disparity images asynchronously. For each disparity timestamp, the temporally nearest RGB frame and the nearest ICP-optimized robot pose are efficiently identified using vectorized nearest-neighbor array lookups.

Disparity to depth: The raw integer disparity d is converted to metric depth using the provided factory calibration polynomials:

$$dd = -0.00304\, d + 3.31, \quad \text{depth} = \frac{1.03}{dd} \tag{9}$$

To suppress sensor noise and remove distant unresolvable features, a boolean mask retains only pixels satisfying $0 < \text{depth} < 5.0$ m. The corresponding coordinates in the RGB image are computed simultaneously via the co-registration model:

$$r_{rgb} = \frac{526.37\, r_d + 19276 - 7877.07\, dd}{585.051},$$
$$c_{rgb} = \frac{526.37\, c_d + 16662}{585.051} \tag{10}$$

A secondary mask filters out any mapped (r_{rgb}, c_{rgb}) coordinates that fall outside the valid RGB image bounds.

Back-projection to camera frame: Valid pixels are structured into a $3 \times N$ homogeneous coordinate matrix. A single batch matrix multiplication is performed using the Kinect intrinsic matrix K ($f_x = f_y = 585.05$, $c_x = 242.94$, $c_y = 315.84$):

$$\mathbf{P}_{cam} = \text{depth} \cdot K^{-1} \begin{bmatrix} \mathbf{c}_d \\ \mathbf{r}_d \\ \mathbf{1} \end{bmatrix} \tag{11}$$

Camera to body frame: Points are transformed to the robot's kinematic frame via a 4×4 rigid body transformation $T_{K \to B}$. Crucially, an intermediate base rotation matrix R_{base} is applied to rotate the camera's optical frame (Z forward, Y down, X right) into the robot's standard frame (X forward, Y left, Z up). The extrinsic parameters—position $(0.18, 0.005, 0.36)$ m and orientation (pitch=0.36 rad, yaw=0.021 rad)—are then applied via an $SE(3)$ batch multiplication.

Floor filter: To map only the floor texture and avoid projecting walls onto the ground plane, a strict physical height filter is applied. Only points satisfying $|z_{body}| < 0.5$ m in the kinematic frame are retained. This masking step aggressively reduces the point cloud size before the final global transformation, saving significant computation time.

Body to world frame: The filtered floor points are projected into the 2-D global map frame using the current estimated robot state $[x_r, y_r, \theta]^T$:

$$\begin{bmatrix} X_w \\ Y_w \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_b \\ y_b \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{12}$$

Discretize and paint: The continuous world coordinates (X_w, Y_w) are divided by the occupancy grid resolution (0.05 m/cell) to yield integer map indices. Advanced array indexing is then used to assign the corresponding RGB color from the synchronized image directly into the global texture map array in a single operation.

E. Pose Graph Optimization with Loop Closure (Part 4)

1) Factor Graph Construction: The SLAM back-end is modeled as a GTSAM `NonlinearFactorGraph`, representing the trajectory as a set of pose nodes $\mathcal{X} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T\}$ where each $\mathbf{x}_t \in SE(2)$. A tight prior factor anchors the origin ($\sigma = 0.01$ m, 0.01 rad). The sequential relative transforms computed by the local ICP scan matcher form the odometry edges (factors). These are encoded as `BetweenFactorPose2` constraints with a noise model $\boldsymbol{\sigma}_{odom} = [0.10 \text{ m}, \ 0.10 \text{ m}, \ 0.05 \text{ rad}]$. The angular noise is bounded tighter to reflect the superior accuracy of the underlying IMU gyroscope prior.

2) Loop Closure Detection and Gating Logic: Evaluating ICP between all possible pose pairs yields an intractable $\mathcal{O}(N^2)$ search space. To balance computational speed, recall, and precision, loop closures are identified using a strict, mathematically defined three-gate acceptance criterion.

Gate 0 – Kinematic and Spatial Proximity: For a candidate pair of poses $\mathbf{x}_i$ and $\mathbf{x}_j$, the baseline spatial requirement is a Euclidean distance threshold:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 < 2.0 \text{ m}$$

Initially, we paired this with a simple temporal gate, requiring an index separation of $|i - j| \geq 50$ poses. However, early optimizations using only these two gates resulted in severe trajectory distortions. By building a visual debugging tool to inspect the accepted pairs (overlaying the first-visit scan in cyan and the re-visit scan in fuchsia, as shown in Fig. 16), we diagnosed a critical failure mode: the robot frequently remained entirely static for extended periods. During these stationary phases, the index gap $|i - j| \geq 50$ was easily exceeded while the robot occupied the exact same physical space. This flooded the factor graph with hundreds of redundant, zero-information constraints that pinned the graph locally and caused the optimizer to hallucinate structural deformations.

To solve this, we formulated a mandatory cumulative travel-distance gate. We require the path integral of the odometry between the two poses to exceed a threshold $S_{min} = 10.0$ m:

$$\sum_{k=i}^{j-1} \|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2 \geq S_{min}$$

This elegantly guarantees that the robot physically navigated away from the area and actively explored before returning, entirely eliminating the stationary false-positives. To further constrain the search, candidates are capped to the 3 spatially nearest poses per stride (evaluated every 20th pose).

Gate 1 – ICP Convergence: The candidate scan pair is passed to the 2D ICP engine (up to 200 iterations). The alignment is rejected if the final Mean Squared Error (MSE) fails to converge below a strict tolerance $\epsilon_{mse} = 0.001$:

$$\frac{1}{N} \sum_{k=1}^{N} \|R^* \mathbf{p}_k + \mathbf{t}^* - \mathbf{q}_k\|_2^2 < \epsilon_{mse}$$

Gate 2 – Overlap Ratio (The Geometric Check): Achieving a low ICP MSE does not guarantee a correct global alignment if the dynamic outlier filter discards too many points. To prevent these specific hallucinations, the aligned source scan $\mathcal{S}_{aligned}$ is projected into the target frame $\mathcal{T}$. We define an indicator function $\mathbb{I}$ that equals 1 if a source point finds a nearest neighbor in the target scan within a radius $r_{inlier} = 0.3$ m. The overlap ratio must exceed $\tau_{overlap} = 0.50$:

$$\frac{1}{|\mathcal{S}_{aligned}|} \sum_{s \in \mathcal{S}_{aligned}} \mathbb{I}\left(\min_{t \in \mathcal{T}} \|s - t\|_2 < r_{inlier} \right) > \tau_{overlap}$$

This final critical gate ensures that at least 50% of the raw, unfiltered geometry matches, reliably eliminating degenerate corridor alignments. Accepted loop closure edges are injected into the graph with a robust noise model $\boldsymbol{\sigma}_{lc} = [0.05 \text{ m}, \ 0.05 \text{ m}, \ 0.05 \text{ rad}]$.

3) Graph Optimization via Levenberg-Marquardt: With the factor graph fully populated with odometry and loop closure constraints, the objective is to find the maximum a posteriori (MAP) estimate of the trajectory $\mathcal{X}$. This is equivalent to minimizing the nonlinear least-squares cost function:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_k \|h_k(\mathcal{X}) - \mathbf{z}_k\|_{\Sigma_k}^2$$

where $h_k(\mathcal{X})$ is the measurement model predicting the relative pose, $\mathbf{z}_k$ is the actual ICP measurement, and Σ_k is the covariance matrix.

Because the $SE(2)$ measurement model is nonlinear, we utilize the Levenberg-Marquardt (LM) algorithm to iteratively solve for the optimal state update $\Delta\mathcal{X}$. At each iteration, the system solves the augmented normal equations:

$$\left(J^\top \Sigma^{-1} J + \lambda I\right) \Delta\mathcal{X} = -J^\top \Sigma^{-1} \mathbf{r}$$

where J is the Jacobian matrix of the measurement models evaluated at the current state, $\mathbf{r}$ is the residual error vector, and λ is a damping parameter.

The inclusion of the damping factor λ is crucial for the stability of this SLAM back-end. When the accumulated trajectory drift is severe (causing high residuals), a large λ forces the optimizer to behave like gradient descent, ensuring stable convergence. As the graph error drops, λ decreases, and the algorithm smoothly transitions into Gauss-Newton optimization for rapid quadratic convergence near the minimum. Following optimization, a `map_range_max` of 10.0 m is

applied when building the final coherent occupancy map from the GTSAM-optimized poses to prevent ray artefacts.

TABLE II
GTSAM POSE GRAPH PARAMETERS

Parameter	Value
Prior factor $\boldsymbol{\sigma}$	$[0.01, 0.01, 0.01]$
Odometry factor $\boldsymbol{\sigma}$	$[0.10, 0.10, 0.05]$
Loop closure factor $\boldsymbol{\sigma}$	$[0.05, 0.05, 0.05]$
Spatial search radius	2.0 m
Min. index separation	50 poses
Min. travel distance	10.0 m
ICP MSE threshold	0.001
Overlap-ratio threshold	0.50
Overlap inlier radius	0.30 m
ICP max iterations (LC)	200
Pose stride (sampling)	20
Map range max (GTSAM map)	10.0 m

IV. RESULTS

A. Odometry Performance: IMU vs. Encoders

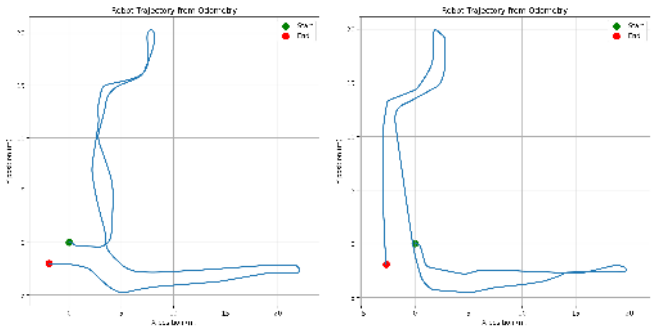


Fig. 1. Dead-reckoning trajectories using encoder velocities and IMU yaw rates. The path is locally consistent and nearly closes the loop. Left: Dataset 20. Right: Dataset 21.

Using the IMU's z-axis gyroscope for ω_t produces a highly accurate, locally consistent baseline trajectory (Fig. 1). Despite the lack of external corrections, the IMU-integrated path nearly closes the loop on its own.

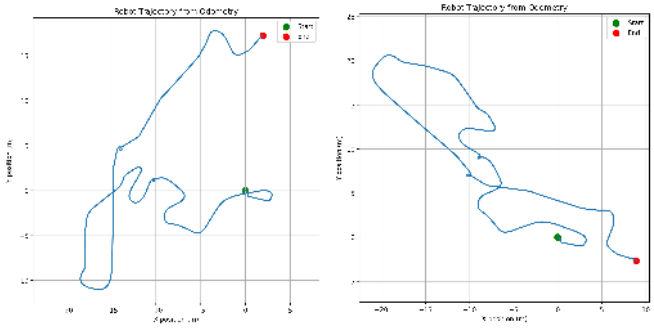


Fig. 2. Dead-reckoning trajectories using encoder velocities and encoder-derived yaw rates for Dataset 20. The two plots illustrate the extreme sensitivity to kinematic calibration, showing how a 10 mm variation in the assumed track width causes drastically different, but equally catastrophic, trajectory drift.

In contrast, using the encoder-derived yaw rate caused catastrophic global drift (Fig. 2). As established in Section III-A, this failure is primarily driven by tick-count quantization noise being continuously integrated (visualized in Fig. 3). Furthermore, our analysis reveals that deriving yaw from differential wheel encoders is acutely sensitive to the kinematic calibration—specifically the track width parameter d. Figure 2 demonstrates this vulnerability: altering the modeled distance between the wheel centers by as little as 10 mm drastically warps the estimated trajectory, pushing the terminal endpoint to approximately $x = -20$ m, $y = 17$ m. This severe parameter sensitivity, compounded by high-frequency quantization noise, irreversibly corrupts the heading θ and subsequently mis-projects all linear velocities. Consequently, encoder-derived yaw was deemed entirely unsuitable for the mapping pipeline.

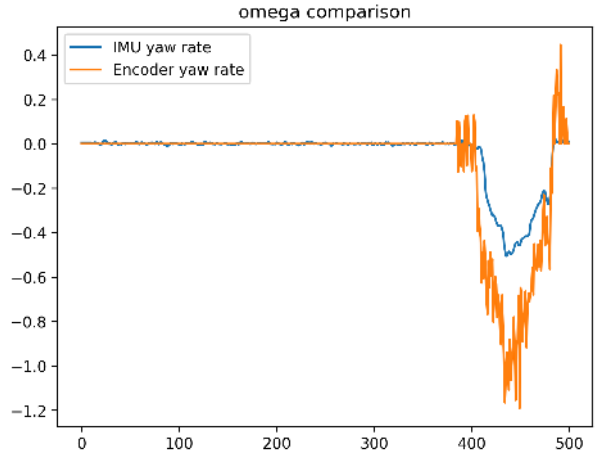


Fig. 3. Yaw rate comparison for dataset 20. The encoder-derived rate (orange) suffers from high-frequency quantization noise; the IMU rate (blue) provides a smooth, reliable measurement.

B. 3D ICP Warm-Up Performance

The 3-D ICP implementation was validated on a drill and a liquid container across four viewing angles each. Table III summarises the convergence achieved across all eight object–view pairs using the 36-angle yaw sweep strategy.

TABLE III
3D ICP WARM-UP CONVERGENCE RESULTS (36-ANGLE YAW SWEEP)

Object	View	Init MSE	Final MSE	Best Yaw	Iters
Drill	0	0.3046	0.000202	130°	8
Drill	1	0.3106	0.000173	110°	6
Drill	2	0.3052	0.000867	290°	8
Drill	3	0.2451	0.001079	300°	6
Liq. Cont.	0	0.2739	0.001286	300°	5
Liq. Cont.	1	0.3844	0.001085	310°	6
Liq. Cont.	2	0.3792	0.001715	310°	6
Liq. Cont.	3	0.3395	0.001133	300°	7

Drill: The 36-angle sweep prevented local minima entrapment, reducing MSE by up to $4\times$ compared to a fixed $0°$ ini-

tialization (e.g., View 0: $0.00078 \rightarrow 0.000202$). Convergence dynamics were rapid, dropping from initial MSEs > 0.30 to near-zero within 3 iterations, never exceeding 8 iterations (Fig. 4).

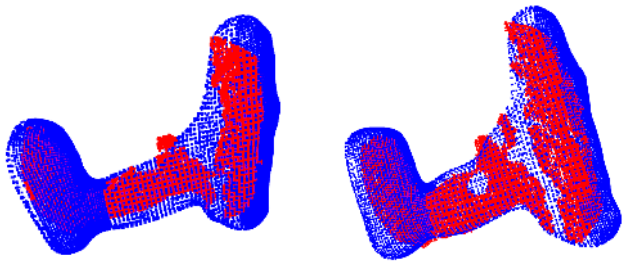


Fig. 4. 3-D ICP alignment of the Drill object. Left: View 0. Right: View 1. The source (red) aligns cleanly to the target (blue) across multiple viewing angles.

Liquid Container: All four views converged to a higher MSE floor (~ 0.001–0.0017) with low sensitivity to the initial yaw (Fig. 5). This is attributable to the container's near-cylindrical rotational symmetry: the ICP cost landscape is nearly flat in the yaw direction, so no initialization angle provides a decisive advantage, and residual MSE remains elevated due to partial surface visibility.

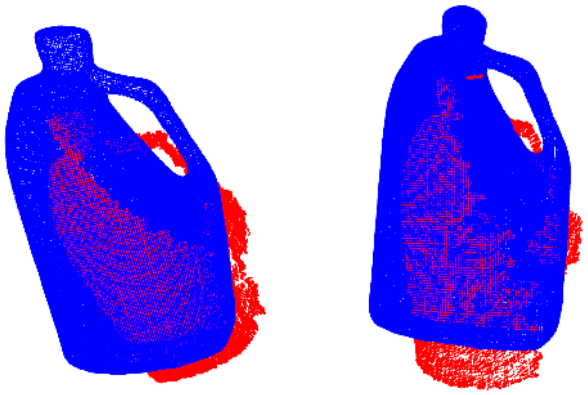


Fig. 5. 3-D ICP alignment of the rotationally symmetric Liquid Container. The higher residual MSE reflects the flat yaw optimization landscape of the cylindrical geometry.

C. 2D Scan Matching Performance and Parameter Analysis

1) ICP Trajectory vs. Odometry: Across both datasets, the IMU-driven odometry trajectory proved remarkably stable, while the standalone ICP trajectory accumulated incremental drift (Fig. 6). This counter-intuitive result arises because frame-to-frame ICP errors compound continuously, particularly in geometrically degenerate corridors where sparse point clouds introduce alignment ambiguity. The IMU, being inertial, is immune to spatial degeneracy. Consequently, the final GTSAM-optimized trajectory aligned far more closely with the IMU odometry prior than with the raw ICP output, with GTSAM's loop closures correcting the cumulative ICP drift.

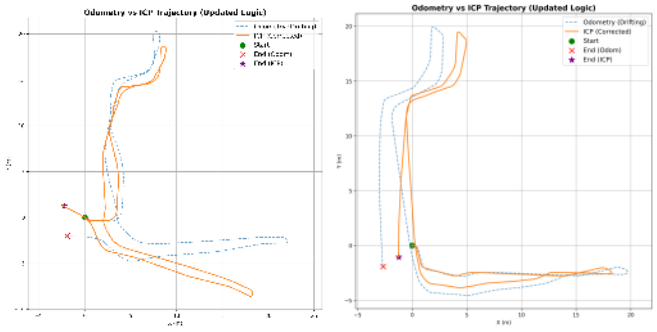


Fig. 6. Odometry vs. ICP scan matching trajectories. Left: Dataset 20. Right: Dataset 21.

2) Downsampling Factor Sweep: Increasing the downsampling factor N (keeping every N-th LiDAR point) reduces computation time by $\sim 2.5\times$ at $N = 10$ vs. $N = 1$ (Fig. 7), but causes progressively greater trajectory drift (Fig. 8). Discarding points removes geometric constraints from the cross-covariance matrix, exacerbating alignment ambiguity in corridors. We therefore use $N = 1$ (all points) for all final results.

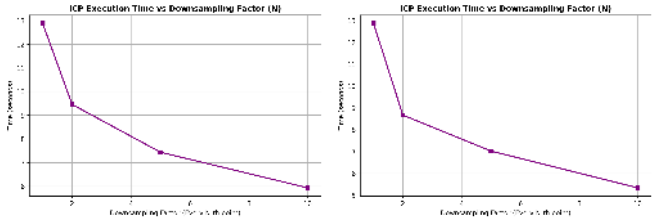


Fig. 7. Execution time vs. downsampling factor N. Processing time drops significantly as fewer points are used. Left: Dataset 20. Right: Dataset 21.

3) Convergence Tolerance Sweep: Looser tolerances (10^{-4}, 10^{-5}) terminate the local optimization prematurely, allowing micro-errors to persist and compound into significant global drift (Fig. 9). A tolerance of 10^{-6} provides the best balance: it forces each scan alignment to fully converge, yielding a substantially more stable global trajectory with marginal added computation.

D. Occupancy and Texture Mapping Results (Pre-Optimization)

1) Final Fused SLAM Output and Artifact Analysis: While the parameter tuning experiments below were conducted exclusively on Dataset 20 to isolate variables, the final fused maps for this stage of the pipeline are presented for both Dataset 20 and Dataset 21 in Fig. 10. These maps combine the log-odds occupancy grid with the projected Kinect floor textures, relying entirely on the raw ICP scan-matching trajectory.

While the maps successfully capture the general topological layout of the environments, they clearly exhibit the limitations of relying solely on local, sequential scan matching. Because ICP accumulates slight translational and rotational errors at

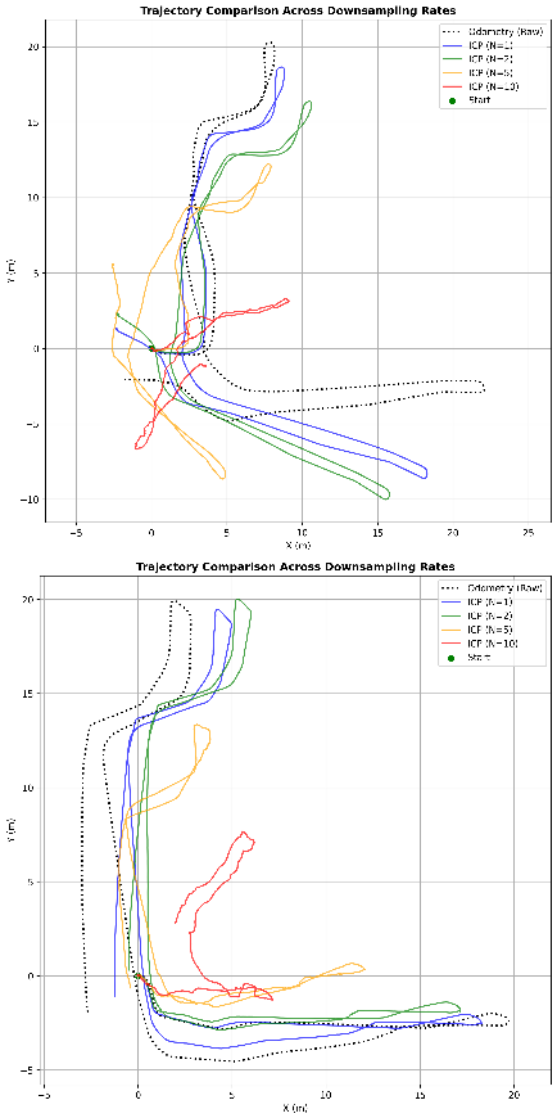


Fig. 8. Trajectory quality vs. downsampling factor. Higher N causes progressively greater global drift. Left: Dataset 20. Right: Dataset 21.

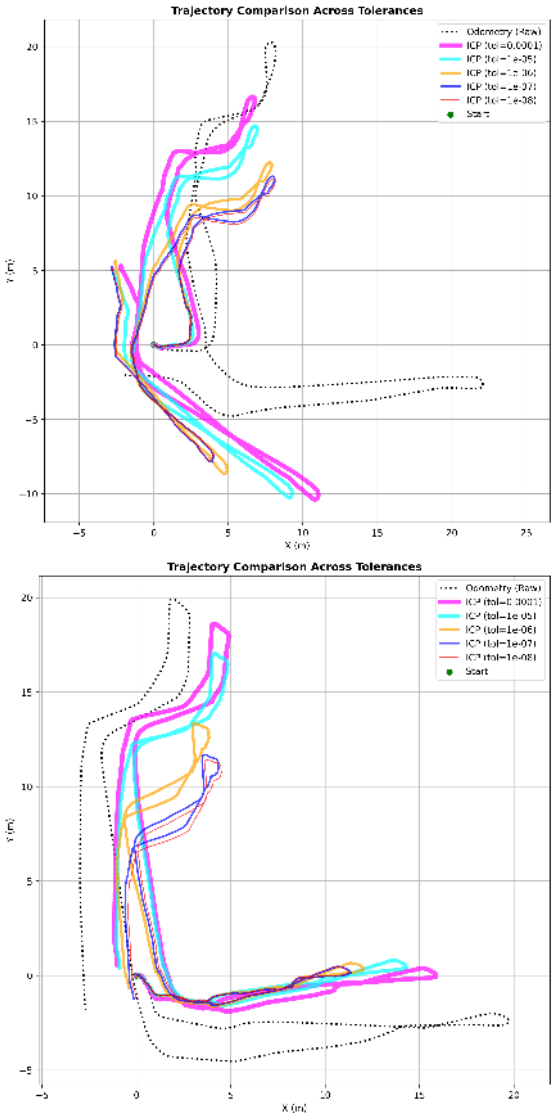


Fig. 9. Impact of convergence tolerance on global trajectory drift. A tight tolerance of 10^{-6} is required to prevent long-term error accumulation. Left: Dataset 20. Right: Dataset 21.

each step, the global trajectories suffer from severe long-term drift. Visually, this manifests as "ghost hallways" and double-walled corridors, where the robot re-visits an area but plots the newly observed walls at an offset from the original ones. Furthermore, early iterations of the map suffered from severe "starburst" artifacts—long, false free-space rays projecting in the lobby—which necessitated capping the maximum LiDAR ray length to 8.0 m. Similarly, the RGB texture mapping suffers from smudging and duplication where overlapping passes fail to align perfectly. These artifacts underscore the necessity of the global pose graph optimization.

To extract the best possible map quality from this drifting trajectory, we conducted isolated experiments to tune the log-odds update rules and the grid resolution on Dataset 20.

2) Experiment 1: Log-Odds Parameter Sweep: Three values of the hit probability log-odds, l_{hit}, were evaluated while holding the miss penalty $l_{miss} = -0.35$ fixed (Fig. 11). A conservative value of $l_{hit} = 0.4$ produces a sparse, low-confidence map with faint wall boundaries, making it highly susceptible to the blurring effects of trajectory drift. Conversely, an aggressive value of $l_{hit} = 2.0$ saturates cells too rapidly, causing wall boundaries to bleed outward and producing artificially thick, bloated obstacle regions. The selected intermediate value of $l_{hit} = 0.85$ achieved the best balance.

3) Experiment 2: Map Resolution Sweep: Four grid resolutions were evaluated to find the optimal balance between geometric fidelity and computational efficiency (Fig. 12). At a highly granular 0.025 m/cell, fine structural details are preserved, but computation time increases quadratically and LiDAR sensor noise becomes visually pronounced as scattered speckles. At coarser resolutions of 0.10 m/cell and 0.20 m/cell, the grid becomes too blocky; corridors lose their geometric

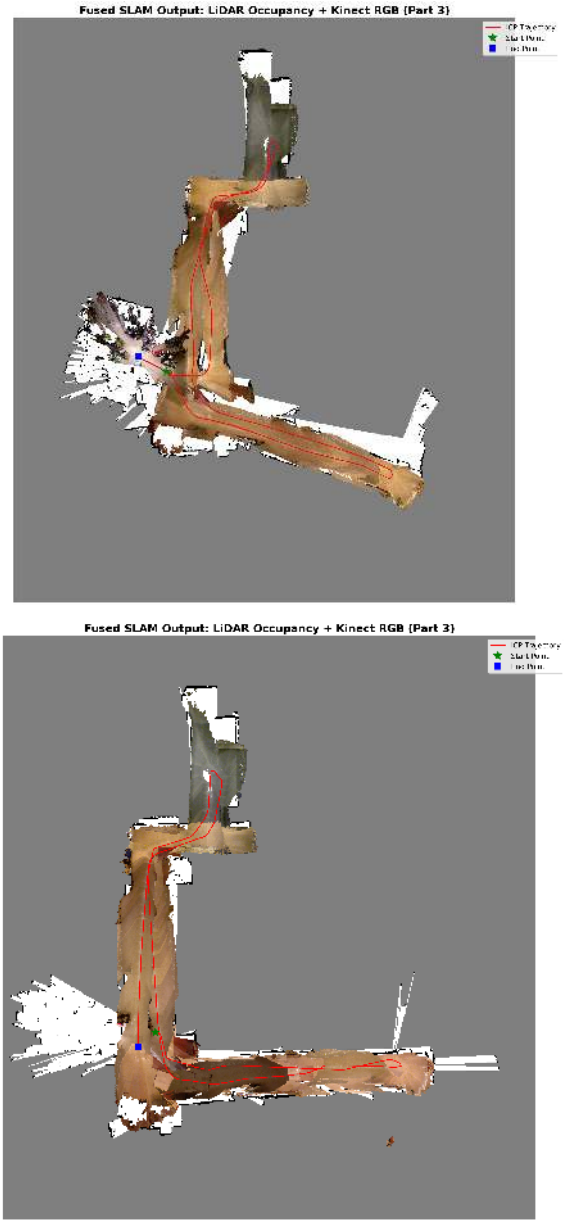


Fig. 10. Final fused SLAM output (Part 3) prior to pose graph optimization. Top: Dataset 20. Bottom: Dataset 21. While the general structure is visible, accumulated ICP drift causes obvious "ghost hallways," blurred wall boundaries, and smudged floor textures. Green star: start. Blue square: end. Red line: ICP trajectory.

definition. The chosen resolution of 0.05 m/cell provides the optimal trade-off.

4) Experiment 3: Progressive Map Construction: To visualize the temporal evolution of the environment representation, we captured sequential snapshots of the occupancy grid at 10%, 25%, 50%, 75%, and 100% of the dataset duration (Fig. 13). As the differential-drive robot navigates the space, the sequential LiDAR scan-matching process iteratively updates the log-odds probabilities. The visualization clearly demonstrates how the scanner wedge carves out confirmed free space (white) and registers structural boundaries (black) against the unmapped prior (gray). This progression highlights how areas with high traversal density quickly converge to sharp boundaries, whereas distal regions remain uncertain until the robot physically approaches them.

E. Pose Graph Optimization and Loop Closure

1) Quantitative Results and Trajectory Correction: Table IV summarizes the pose graph optimization outcome for Dataset 20. The Levenberg-Marquardt optimizer successfully reduced the total graph error by over 99.98%, plunging from an initial value of 9827.41 to a final optimized cost of 1.75.

TABLE IV
GTSAM POSE GRAPH OPTIMIZATION RESULTS (DATASET 20)

Metric	**Value**
Total poses	4962
Loop closures added	111
Closure/pose ratio	0.022
Initial graph error	9827.41
Final graph error	1.75
Error reduction	$> 99.98\%$

This dramatic reduction in mathematical error corresponds directly to the physical trajectory correction shown in Fig. 14. By distributing the accumulated error across the graph via loop closure constraints, the GTSAM-optimized trajectory snaps the endpoint back to within $\sim$1.1 m of the true origin for Dataset 20, with comparable global corrections observed in Dataset 21.

2) Loop Closure Analysis and Debugging: To debug and validate the loop closure engine, candidate pairs were visualized by overlaying the stored point cloud from the first visit (cyan) onto the re-visit scan (fuchsia). Initial visualizations revealed a critical failure mode: the system was registering hundreds of loop closures when the robot remained entirely static. Because consecutive frames trivially matched, the algorithm flooded the factor graph with redundant, zero-information constraints.

To resolve this, we engineered a strict multi-gate acceptance criterion, introducing a mandatory cumulative travel-distance gate ($\geq$ 10.0 m). Fig. 15 proves the efficacy of these gates: the index gap histogram shows no closures occurring beneath a gap of $\sim$450 poses, confirming that stationary false-positives were eliminated. Furthermore, Fig. 16 showcases one of the top-ranked loop closures. The overlap ratio of 79% easily clears our 50% geometric consistency gate.

3) Occupancy Map Quality: The impact of this trajectory correction on the global environment model is striking. Fig. 17 compares the occupancy grids generated before and after GTSAM optimization. The raw ICP map exhibits blurry corridors caused by accumulated heading drift, whereas the GTSAM-optimized map snaps these misalignments together.

Fig. 18 presents the final, isolated GTSAM-optimized occupancy maps for both datasets with the optimized trajectory overlaid, demonstrating clean, highly navigable representations of the space.

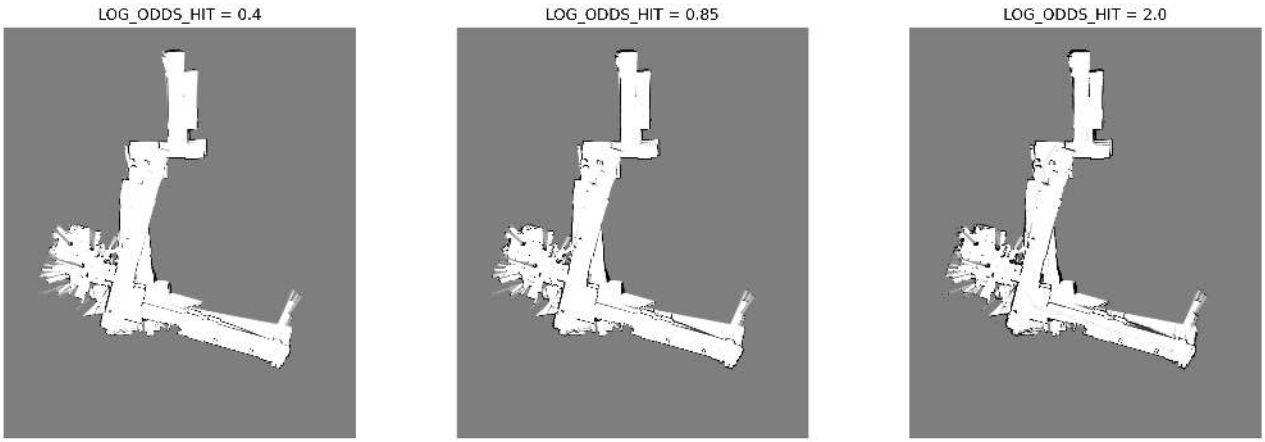


Fig. 11. Log-odds parameter sweep. From left to right: $l_{hit} \in \{0.4, 0.85, 2.0\}$. The middle value (0.85) produces the sharpest wall boundaries without oversaturating the obstacle thickness.

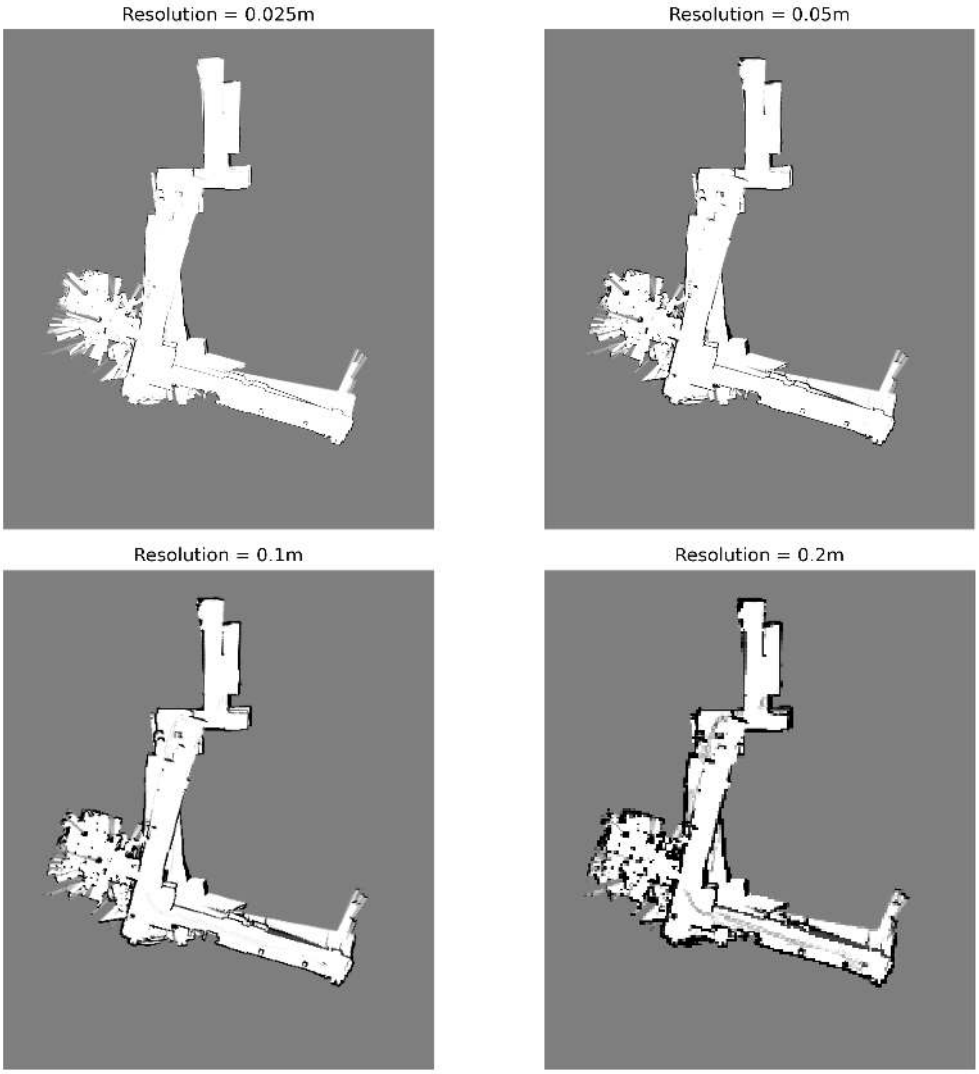


Fig. 12. Map resolution sweep over $\{0.025, 0.05, 0.10, 0.20\}$ m/cell. The 0.05 m/cell resolution (top-right) provides the best balance of structural fidelity and processing efficiency.

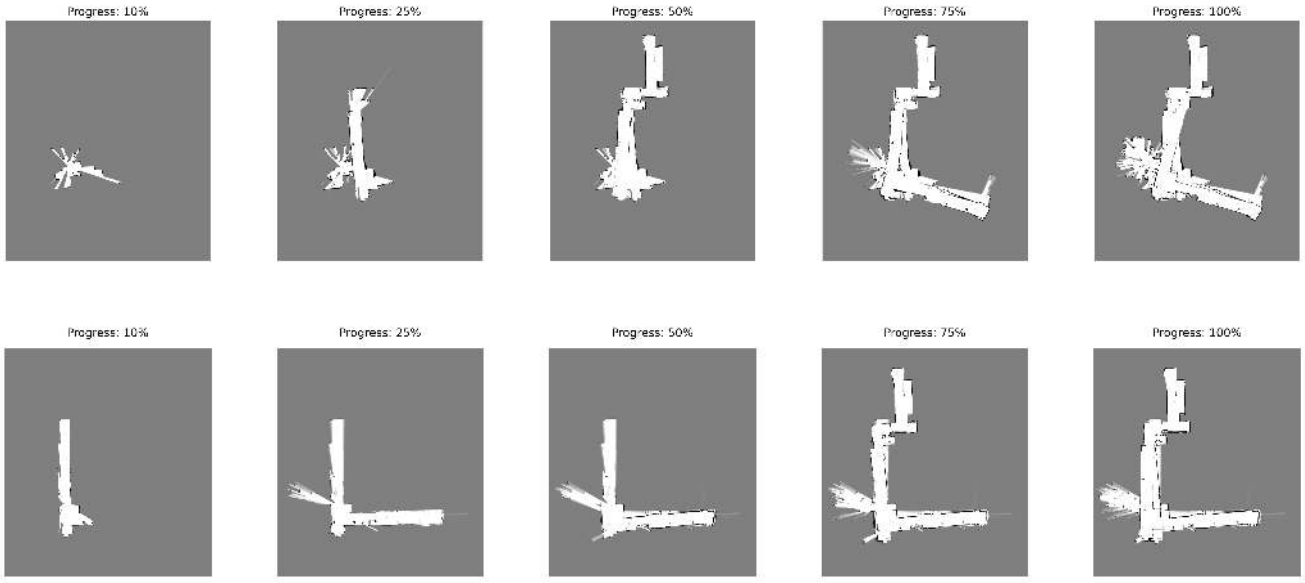


Fig. 13. Progressive construction of the log-odds occupancy grid over time (10%, 25%, 50%, 75%, and 100% dataset completion). Top: Dataset 20. Bottom: Dataset 21. The sequential accumulation of LiDAR scans incrementally resolves the free space and structural boundaries of the environment.

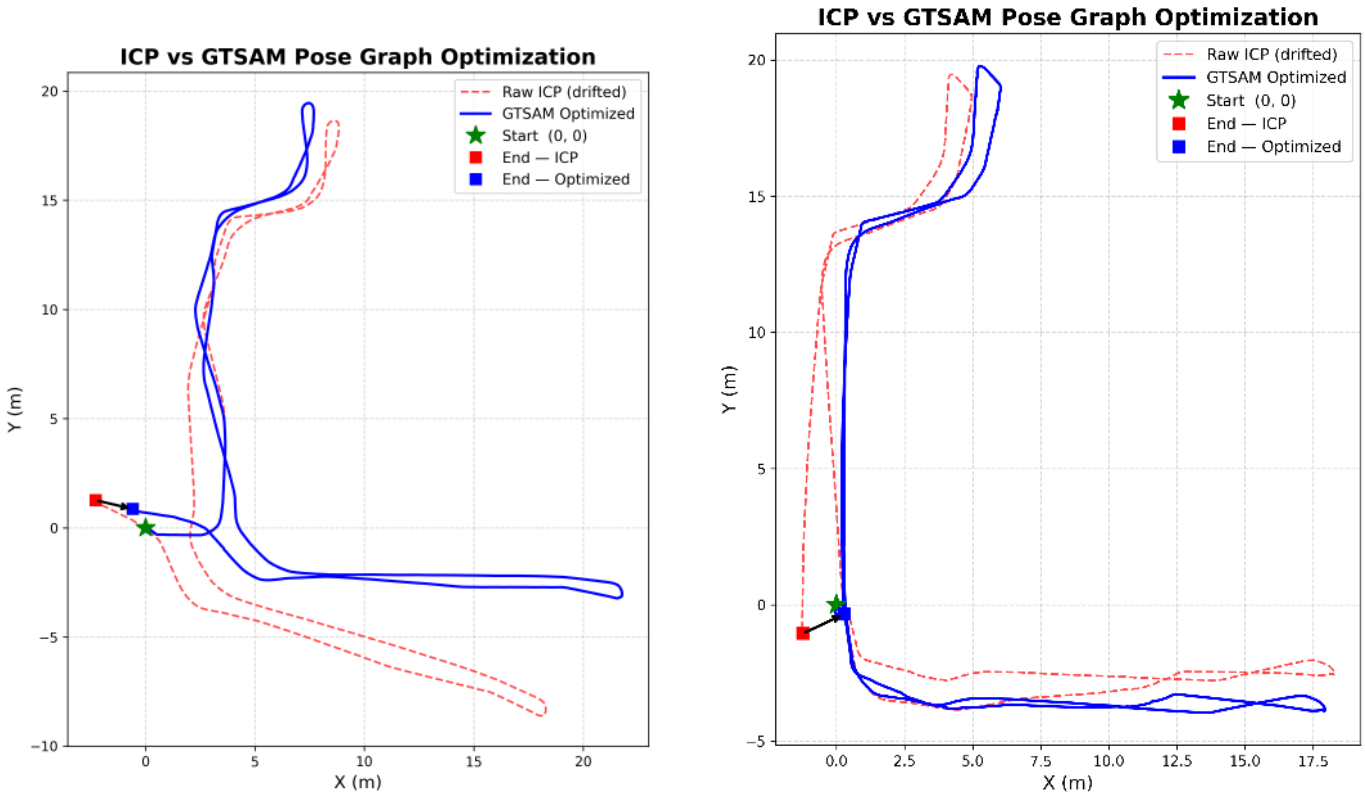


Fig. 14. Trajectory before (red dashed) and after (blue solid) GTSAM optimization. Left: Dataset 20. Right: Dataset 21. Loop closures pull the drifting ICP path back, bringing the endpoint (blue square) close to the origin (green star) and restoring global consistency.

4) Parameter Sensitivity and Trajectory Hallucination: Balancing the loop closure detection parameters required navigating a strict trade-off between recall (identifying genuine revisits) and precision (rejecting false positives). During early parameter tuning, looser spatial search radii and relaxed index gaps permitted incorrect candidate pairs to be evaluated by the ICP engine. This exposed a critical vulnerability in the pipeline that led to severe trajectory "hallucinations"—catastrophic

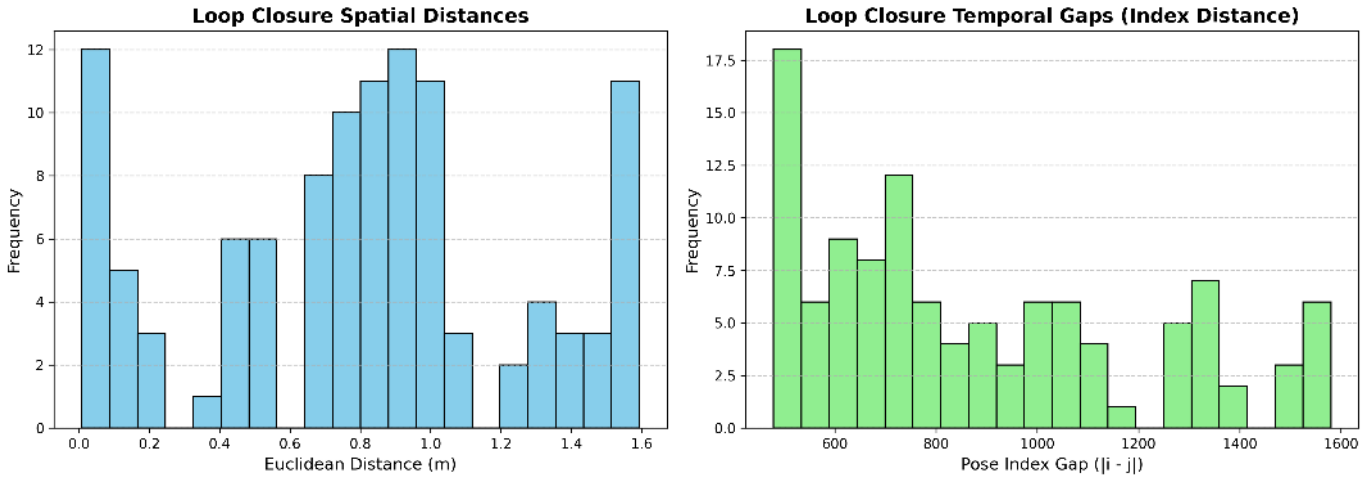


Fig. 15. Distribution of the 111 accepted loop closures. The Euclidean distance (left) shows closures mostly occurring within 0–0.2 m and 0.8–1.6 m. The pose index gap (right) demonstrates that the temporal/distance gating successfully prevented stationary false-positives.

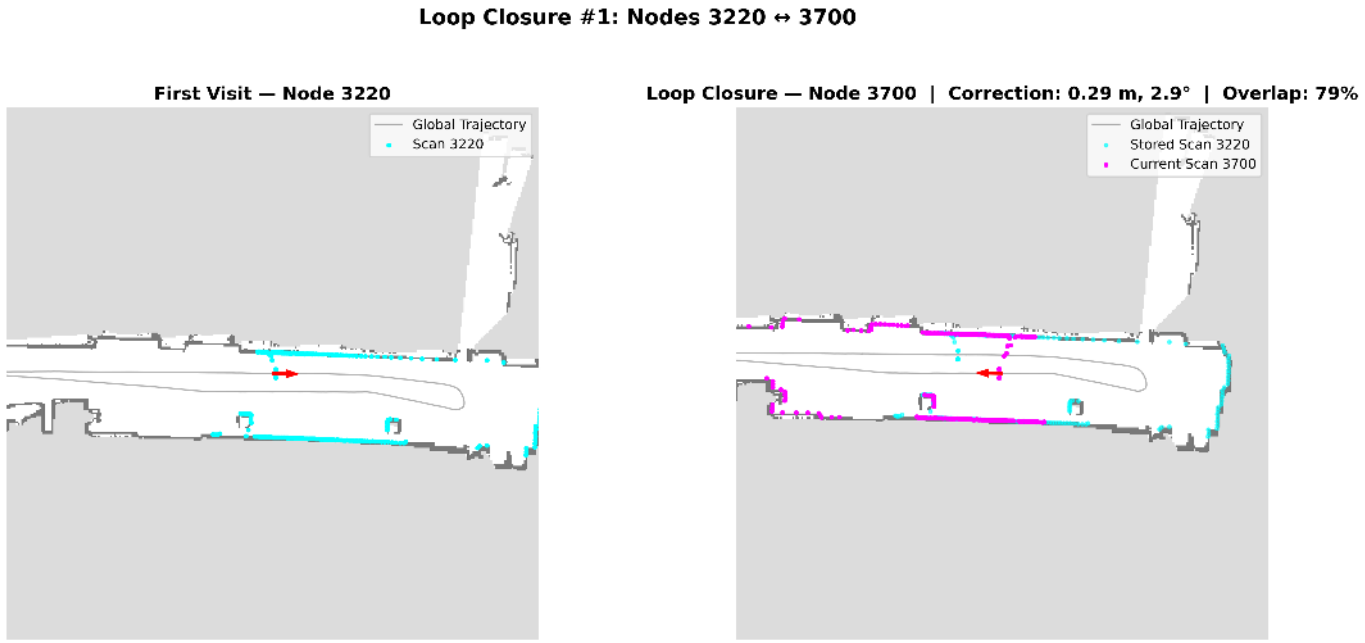


Fig. 16. Top-ranked loop closure pair (Nodes 3220 ↔ 3700). Left: Stored LiDAR scan from the first visit (cyan). Right: Current re-visit scan (fuchsia). The 79% geometric overlap confirms a genuine closure.

deformations of the global factor graph where the optimized map folded in on itself, yielding graph errors in excess of 46,000.

We traced the root cause of these hallucinations to an unintended interaction with the dynamic outlier rejection logic implemented in the 2D ICP algorithm (Section III-C). Originally designed to robustly filter moving obstacles during sequential scan matching, this dynamic threshold became a significant liability during loop closure detection. When provided with a geometrically ambiguous candidate pair (e.g., two parallel but physically distinct corridors), the dynamic filter would aggressively classify up to 80% of the source point cloud as outliers. The ICP algorithm would then flawlessly align the remaining 20% of the points—typically a single, featureless wall segment—yielding a deceptively low Mean Squared Error (MSE). Because this MSE easily cleared the initial acceptance threshold, a completely false $SE(2)$ constraint was injected into the GTSAM factor graph, warping the global optimization.

To combat this, we introduced the percentage overlap check (Gate 2). This geometric consistency check effectively neutralized the false positives generated by the dynamic outlier rejection, immediately stabilizing the optimizer and eliminating all trajectory hallucinations.

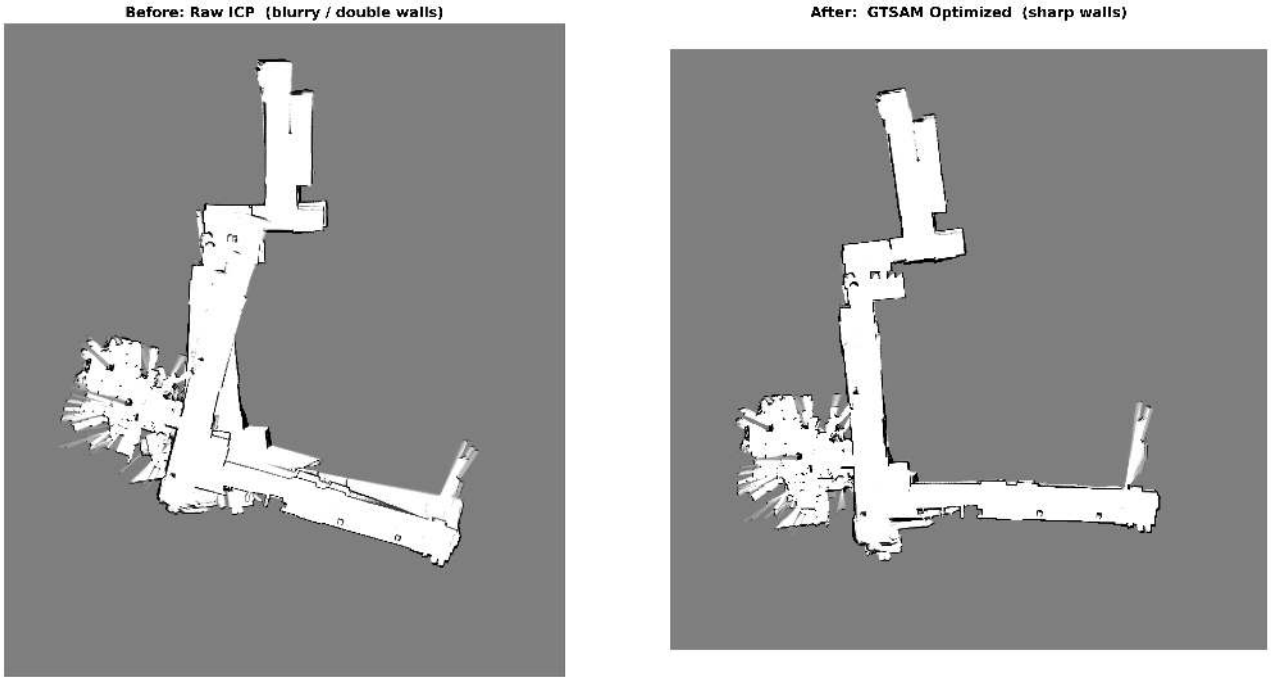


Fig. 17. Occupancy map before GTSAM (left, showing blurry/double walls from drift) vs. after GTSAM optimization (right, showing sharp single walls). The global pose graph correctly registers the corridors into a single, cohesive floor plan.

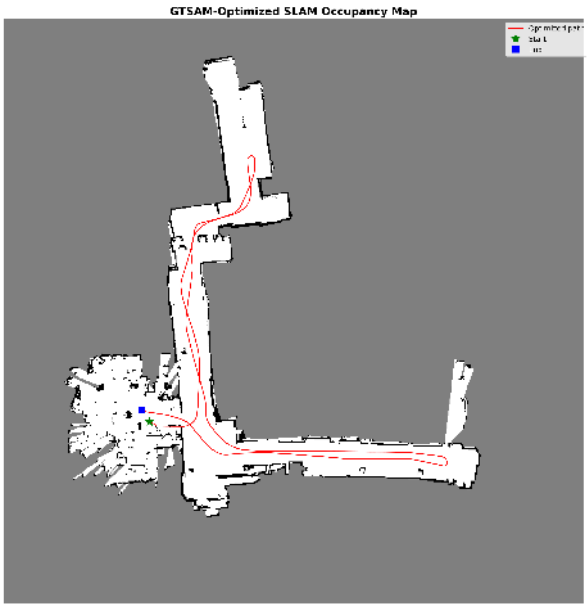




Fig. 18. Final GTSAM-optimized occupancy maps. Top: Dataset 20. Bottom: Dataset 21. Black cells: occupied walls/obstacles. White cells: confirmed free space. Red line: optimized trajectory. All structural redundancies have been resolved.



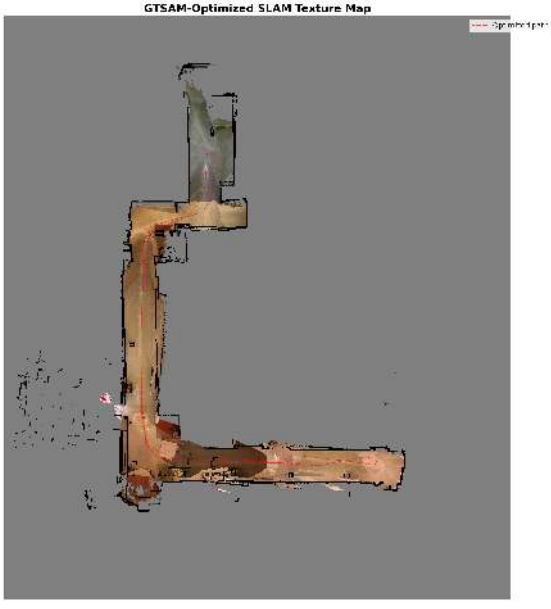


Fig. 19. Floor texture maps constructed using GTSAM-optimized poses. Top: Dataset 20. Bottom: Dataset 21. Kinect RGB colors are accurately projected onto the global floor plane, eliminating the smudging and duplicate textures seen in the pre-optimized map.

ACKNOWLEDGMENT

I express sincere gratitude to Prof. Atanasov for designing this project, which provided an invaluable learning experience in SLAM, ICP, and factor graph optimization. Special thanks to the Teaching Assistants for their guidance and support.

REFERENCES AND ATTRIBUTION

The mathematical formulation for the SLAM framework was derived from the lecture notes "Sensing and Estimation in Robotics" by Prof. Nikolay Atanasov (UCSD, 2026). The Georgia Tech Smoothing and Mapping (GTSAM) library [1] was used for pose graph optimization. The Bresenham [2] ray-tracing algorithm and OpenCV were used for occupancy map updates.

REFERENCES

[1] F. Dellaert and GTSAM Contributors, "GTSAM," Georgia Tech, 2012. [Online]. Available: `https://gtsam.org`

[2] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[3] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.