ECE276A Project2 LiDAR-Based SLAM Report

Cheng Han Yu
University of California, San Diego
La Jolla, California
chy084@ucsd.edu

I. Introduction

Accurate Simultaneous Localization and Mapping (SLAM) is essential for autonomous navigation in GPS-denied indoor environments, where small drift can quickly degrade map quality and localization reliability. This project aims to build a robust pipeline that reduces drift while preserving fine-scale geometric structure. We present a LiDAR-based SLAM pipeline that estimates the robot trajectory and builds a consistent occupancy map with texture. The physical setup consists of a differential-drive robot equipped with wheel encoders, an Inertial Measurement Unit (IMU), a 2D Hokuyo LiDAR scanner, and a Kinect RGBD camera. Our method combines motion-model odometry, scan matching, and loop-closure optimization, followed by occupancy grid and texture mapping.

II. Problem Formulation

The objective of this project is to solve the SLAM problem by recursively estimating the robot's trajectory while concurrently mapping its surrounding environment. To achieve a geometrically consistent map, we decompose the problem into four interconnected components:

A. Motion Model

From IMU and encoder data, we compute the global pose $(x_{t+1}, y_{t+1}, \theta_{t+1})$ using the exact discrete-time motion model of a differential drive:

$$\begin{aligned} x_{t+1} &= x_t + \tau_t v_t \,\mathrm{sinc}\Big(\frac{\omega_t \tau_t}{2}\Big)\cos\Big(\theta_t + \frac{\omega_t \tau_t}{2}\Big), \\ y_{t+1} &= y_t + \tau_t v_t \,\mathrm{sinc}\Big(\frac{\omega_t \tau_t}{2}\Big)\sin\Big(\theta_t + \frac{\omega_t \tau_t}{2}\Big), \\ \theta_{t+1} &= \theta_t + \tau_t\, \omega_t. \end{aligned} \tag{1}$$

Here τ_t is the time step and ω_t is the yaw rate.

B. Iterative Closest Point (ICP)

We estimate the robot pose by aligning LiDAR point clouds collected at different times. Let $m_i \in \mathbb{R}^3$ and $z_i = R^\top(m_i - \mathbf{p}) \in \mathbb{R}^3$ for $i = 1, \dots, n$, where $R \in SO(3)$ and $\mathbf{p} \in \mathbb{R}^3$ are the rotation and translation. Given two sets of points $\{m_i\}$ and $\{z_j\}$, the data association $\Delta = \{(i,j) : m_i \text{ corresponds to } z_j\}$ is unknown. The weighted ICP objective is

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

After fixing Δ, we solve

$$\min_{R\in SO(d),\, \mathbf{p}\in\mathbb{R}^d} f(R, \mathbf{p}) := \sum_i w_i \left\| (Rz_i + \mathbf{p}) - m_i \right\|_2^2. \tag{3}$$

Let the weighted centroids be

$$\bar{\mathbf{m}} := \frac{\sum_i w_i \mathbf{m}_i}{\sum_i w_i}, \qquad \bar{\mathbf{z}} := \frac{\sum_i w_i \mathbf{z}_i}{\sum_i w_i}. \tag{4}$$

Solving $\nabla_{\mathbf{p}} f(R, \mathbf{p}) = 0$ yields

$$\mathbf{p} = \bar{\mathbf{m}} - R\bar{\mathbf{z}}. \tag{5}$$

Define centered point clouds $\delta\mathbf{m}_i := \mathbf{m}_i - \bar{\mathbf{m}}$ and $\delta\mathbf{z}_i := \mathbf{z}_i - \bar{\mathbf{z}}$. Then

$$\min_{R\in SO(d)} \sum_i w_i \left\| R\delta\mathbf{z}_i - \delta\mathbf{m}_i \right\|_2^2 \iff \max_{R\in SO(d)} \mathrm{tr}(Q^\top R), \tag{6}$$

where $Q := \sum_i w_i \delta\mathbf{m}_i\, \delta\mathbf{z}_i^\top$. This is the Kabsch/Wahba problem. With $Q = U\Sigma V^\top$, the optimal rotation is

$$R^\star = U \begin{bmatrix} 1 & & \\ & \ddots & \\ & & \det(UV^\top) \end{bmatrix} V^\top. \tag{7}$$

C. Occupancy Mapping and Texture Mapping

We model each cell m_i as a Bernoulli variable and update it with a log-odds filter:

$$\lambda_{i,t} = \log g_h(z_t \mid m_i, x_t) + \lambda_{i,t-1}, \tag{8}$$

where $g_h(z_t \mid m_i, x_t)$ is the observation model odds. The occupancy probability is recovered by

$$\gamma_{i,t} = p(m_i = 1 \mid z_{0:t}, x_{0:t}) = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})}. \tag{9}$$

From RGBD data, we back-project each depth pixel into the depth camera frame using the camera intrinsics, then transform points to the robot and world frames using the estimated poses. We map RGB colors to the 3D points and threshold by height to extract the floor texture.

D. Factor Graph Optimization

To optimize trajectory estimation, we construct a factor graph that connects variables (robot states x_t and landmarks m_j) with factors derived from inputs u_t and observations z_t. Motion factors encode errors between consecutive poses and the motion model,

$$e_f(x_{t+1}, x_t) = x_{t+1} \ominus f(x_t, u_t), \tag{10}$$

and observation factors encode errors between measurements and the observation model,

$$e_h(x_t, m_j) = z_{t,j} \ominus h(x_t, m_j). \tag{11}$$

The operator $\ominus$ denotes a geometry-aware difference on the state manifold. After constructing the factor graph, the back-end solves

$$\min_{\{x_i\}} \sum_{(i,j)\in\mathcal{E}} \phi_{ij}\big(e(x_i, x_j)\big), \tag{12}$$

where $\phi_{ij}(\cdot)$ is a distance function (e.g., $\phi_{ij}(e) = e^\top \Omega_{ij} e$ with positive-definite Ω_{ij}) and $\mathcal{E}$ is the set of graph edges.

III. Technical Approach

A. Preprocessing

We calibrate IMU, encoder, Hokuyo LiDAR, and Kinect sensors, then synchronize all measurements to the closest Kinect RGB timestamp. LiDAR range scans are downsampled by a factor of 3 to reduce computation, and we build a 500×500 grid map for mapping.

B. Odometry

Given encoder counts $[FR, FL, RR, RL]$ for the front-right, front-left, rear-right, and rear-left wheels, the right and left travel distances are

$$d_R = \frac{FR + RR}{2} \cdot 0.0022 \text{ m}, \qquad d_L = \frac{FL + RL}{2} \cdot 0.0022 \text{ m}. \tag{13}$$

The forward velocity is computed as

$$v = \frac{v_R + v_L}{2}. \tag{14}$$

The forward speed v_t is fully derived from the left and right wheel encoders, while the angular velocity ω_t is taken directly from the IMU Z-axis (yaw rate) because the IMU provides more accurate rotation measurements and is less susceptible to wheel slip. From IMU and encoder data, we compute the global pose $(x_{t+1}, y_{t+1}, \theta_{t+1})$ using (1)

C. Point Cloud Registration (ICP)

To ensure computational efficiency, the LiDAR range scans are downsampled by a factor of 3. We also apply a distance mask, filtering out points closer than 0.3 m or farther than 20 m to remove sensor noise and physical robot occlusions. Scan-to-scan registration uses the Iterative Closest Point (ICP) algorithm. First, we construct a KD-tree from the target point cloud to efficiently establish point-to-point correspondences. The kinematic displacement from the encoders and IMU is used to initialize the transformation. The Kabsch algorithm—utilizing Singular Value Decomposition (SVD)—is then applied to compute the optimal rotation matrix R and translation vector $\mathbf{p}$ that minimize the Root Mean Square Error (RMSE) of the matched pairs. This iterative alignment provides a robust local odometry estimate for the mapping framework.

We use a drill and a liquid container for practice. Although they are 3D objects, we assume they rotate only about the yaw axis (planar rotation on the table). Therefore, we fix roll and pitch and use the standard z-axis rotation matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where θ is the yaw angle. See the results in the ICP warm-up section.

For 2D mapping, we match point clouds $\{m_{t+1}\}$ and $\{m_t\}$ using ICP and obtain a 2×2 rotation matrix R and translation $\mathbf{p}$ to construct $T_{t\to t+1}$ in the LiDAR frame. To achieve robust scan-to-scan registration, the kinematic displacement estimated from the wheel encoders and IMU is used as the initial guess for the Iterative Closest Point (ICP) algorithm. After ICP, we update the LiDAR pose via

$$T_{\text{lidar},t+1} = T_{\text{lidar},t}\, T_{t\to t+1}, \tag{15}$$

To transform the 2D LiDAR scans from the sensor frame to the robot center frame, we apply a static extrinsic translation of 0.13323 m along the x-axis. The transformation matrix $T_{\text{robot}\leftarrow\text{lidar}} \in SE(2)$ is defined as:

$$T_{\text{robot}\leftarrow\text{lidar}} = \begin{bmatrix} 1 & 0 & 0.13323 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

To improve ICP robustness, we compute the yaw rate estimated by ICP; if it exceeds 90 degrees, we use the pose estimated by the IMU and encoders. If the number of inliers between two point clouds is below 100, we also use the pose estimated by the IMU and encoders. After that we can obtain the robot pose with the fixed extrinsic transform

$$T_{\text{robot},t+1} = T_{\text{robot}\leftarrow\text{lidar}}\, T_{\text{lidar},t+1}. \tag{17}$$

After collecting ICP-refined robot pose overtime, we get robot trajectory.

D. Occupancy and Texture Mapping

The occupancy grid map is built at a resolution of 0.05 m per cell. We model each cell m_i of a 2D grid as a Bernoulli variable representing its occupancy status. The map is updated using a log-odds filter:

$$\lambda_{i,t} = \log g_h(z_t \mid m_i, x_t) + \lambda_{i,t-1}.$$

Assuming an 80% correct sensor with a uniform map prior (occupied and free space are equally likely), the inverse observation model odds yield a constant update factor of $\pm \log 4$. This allows us to simply increase or decrease the log-odds of a cell based on whether the LiDAR ray observed it as occupied or free. We also clip the grid values to ± 10 to prevent overconfidence. The final map probability mass function (pmf) can then be recovered from the log-odds using the logistic sigmoid function:

$$\gamma_{i,t} = p(m_i = 1 \mid z_{0:t}, x_{0:t}) = \sigma(\lambda_{i,t}) = \frac{\exp(\lambda_{i,t})}{1 + \exp(\lambda_{i,t})}.$$

To determine observation mapping, We transform LiDAR points from the sensor frame into the global frame using the estimated poses. We implement Bresenham's 2D ray-tracing algorithm to determine the cells traversed by each laser beam. Cells along the ray decrease in log-odds (free space), while the terminal cell increases in log-odds (occupied space).

For texture mapping, RGB images and disparity maps are synchronized. Because the depth and RGB cameras are physically offset, we first transform depth pixels into RGB coordinates before texturing. For a disparity pixel (i, j) with disparity d, we compute

$$dd = -0.00304d + 3.31, \qquad depth = \frac{1.03}{dd}. \tag{18}$$

The corresponding RGB pixel is

$$\begin{aligned} \mathrm{rgb}_i &= \frac{526.37\, i + 19276 - 7877.07\, dd}{585.051}, \\ \mathrm{rgb}_j &= \frac{526.37\, j + 16662}{585.051}. \end{aligned} \tag{19}$$

We then map the RGB value at $(\mathrm{rgb}_i, \mathrm{rgb}_j)$ to the 3D point from the depth image. Next, we transform the 3D points from the camera optical frame (z-forward, x-right, y-down) to the standard robot frame (x-forward, y-left, z-up). This requires an optical-to-robot axis permutation followed by the extrinsic rotation (pitch = 0.36 rad, yaw = 0.021 rad) and translation $\mathbf{t} = [0.18, 0.005, 0.36]^\top$ m. The full 3D rotation $R_{\text{depth\_to\_robot}} \in SO(3)$ is computed as

$$R_{\text{depth\_to\_robot}} = R_z(0.021) R_y(0.36) R_x(0) \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{20}$$

The resulting 3D points in the robot frame are then calculated as $\mathbf{p}_{\text{robot}} = R_{\text{depth\_to\_robot}} \mathbf{p}_{\text{optical}} + \mathbf{t}$. Finally, we extract the floor texture by applying a strict height threshold ($z < 0.1$ m) and map the RGB colors to the corresponding 2D grid cells. By integrating the robot pose over time, we obtain the final texture map.

E. Pose Graph Optimization and Loop Closure

1) Graph Initialization and Odometry Constraints: To reduce global drift and enforce geometric consistency across the mapped environment, the trajectory estimation is formulated as a factor graph optimization problem using the GTSAM (Georgia Tech Smoothing and Mapping) library. The robot's trajectory is modeled as a sequence of discrete 2D poses (x, y, θ) representing the nodes of the graph, while sensor measurements and scan registrations act as constraint edges (factors) connecting these nodes.

We combine local constraints and global constraints. Local constraints add an extra odometry edge every 5 frames to stabilize short-term trajectory smoothness. Global constraints search historical nodes within a 1.0 m KD-tree radius and require at least 10 timesteps of separation to avoid redundant adjacent edges.

2) Graph Initialization and Odometry Constraints: The factor graph is anchored to a global coordinate frame by applying a `PriorFactorPose2` to the initial node, defining the starting pose as $(0, 0, 0)$. This prior uses a highly confident diagonal noise model with standard deviations $\sigma = [10^{-5}\ \text{m}, 10^{-5}\ \text{m}, 10^{-6}\ \text{rad}]$. At each timestep, the globally accumulated pose estimated by ICP is inserted into the graph as the node's initial value. Concurrently, edges are added between consecutive nodes to represent relative motion. Both the raw IMU/encoder dead reckoning and the ICP-refined relative transformations are added to the factor graph as sequential odometry constraints.

3) Loop Closure Detection: To correct the accumulation of global drift inherent to dead reckoning and sequential ICP, we implement two mechanisms for loop-closure detection. Temporal heuristic: a localized loop closure is attempted at fixed intervals (every 5 timesteps) to bind recent trajectory segments tightly together and prevent short-term drift. Spatial proximity: a global loop closure is triggered whenever the robot's current estimated position falls within a 1.0 m search radius of a historical node. To prevent redundant and trivial matches with immediately preceding scans, only consider 10 timesteps away.

4) Constraint Validation and Graph Optimization: When a loop closure candidate is identified, ICP is executed between the current scan and the historical scan to determine the relative transformation. To prevent false positives from corrupting the graph geometry, the loop closure constraint is only accepted if the ICP Root Mean Square Error (RMSE) is strictly less than 0.1. Finally, the constructed factor graph is optimized using the Levenberg–Marquardt (LM) algorithm. The LM optimizer iteratively solves the nonlinear least-squares problem, successfully minimizing global drift and yielding a geometrically consistent robot trajectory.

As summarized in Table I, the performance of the factor graph heavily relies on a carefully tuned trust hierarchy. The initial node is strictly anchored to the origin with negligible variance. Due to inherent wheel slip and sensor noise, the raw dead-reckoning odometry is assigned a high uncertainty ($\sigma_{x,y} = 0.2$ m). Conversely, the sequential ICP alignments are highly trusted ($\sigma_{x,y} = 0.01$ m) to preserve the local geometric structure. Finally, to effectively eliminate accumulated global drift, validated loop closures are assigned the highest confidence ($\sigma_{x,y} = 0.001$ m). This aggressive loop-closure noise model forces the optimizer to prioritize snapping the trajectory back into alignment when revisiting known areas.

TABLE I
GTSAM Factor Graph Noise Models (Standard Deviations)

Factor Type	σ_x **(m)**	σ_y **(m)**	σ_θ **(rad)**
Prior Pose	10^{-5}	10^{-5}	10^{-6}
Raw Odometry	0.2	0.2	0.08
ICP Registration	0.01	0.01	0.01
Loop Closure	0.001	0.001	0.001

IV. Results

A. ICP Warm up

Because we only optimize the yaw angle, the ICP fit is poor when roll or pitch is present, so the warm-up results are less accurate for out-of-plane rotations.

B. Trajectory Estimation and the Impact of Optimization

The impact of factor graph optimization and loop closure detection is most evident when comparing the estimated trajectories across different stages of the pipeline. As shown in the trajectory comparisons for Dataset 20 and Dataset 21 (included in the supplementary videos `Trajectory comparison of dataset 20.mp4` and `Trajectory comparison of dataset 21.mp4`), we evaluated three distinct trajectory estimates:

1) **Dead Reckoning (Red):** Relying purely on wheel encoders and the IMU resulted in severe error accumulation. The trajectory quickly diverged from the true path due to wheel slip, minor integration biases, and the inability to correct heading drift.
2) **Odometry + ICP (Blue):** Integrating ICP scan matching significantly improved local consistency. The trajectory maintained the structural shape of the environment for longer periods. However, over extended distances, sequential ICP still accumulated global yaw drift, causing straight corridors to artificially bend and preventing loops from closing.
3) **GTSAM Optimized (Green):** The introduction of pose graph optimization with loop closure detection successfully eliminated the accumulated drift. By identifying spatial proximity loop closures and assigning them a high-confidence noise model ($\sigma = 0.001$), the Levenberg-Marquardt optimizer forced the trajectory to snap back into geometric alignment. The optimized path yields perfectly closed loops and straight corridors, demonstrating the critical necessity of back-end optimization for long-term SLAM.

What Worked: The hierarchical trust system in the GTSAM factor graph proved highly effective. By heavily trusting the ICP relative transformations locally, but allowing loop closures to aggressively override the accumulated drift, the system struck a perfect balance between local smoothness and global consistency. Furthermore, using the IMU's Z-axis for the yaw rate instead of the wheel encoders drastically improved the initial guess for the ICP algorithm, preventing local minima during scan matching.

What Did Not Work (and Why): First, the ICP algorithm occasionally struggled during fast rotational maneuvers or in long, featureless corridors. In these scenarios, LiDAR scans lack the longitudinal geometric constraints necessary for precise point-to-point matching, leading to slight longitudinal slip. We mitigated this by enforcing an inlier threshold; if ICP failed, the system fell back to the IMU/Encoder odometry.

Second, the system assumes a strictly 2D planar environment ($SE(2)$ kinematics). Minor out-of-plane dynamics —such as the robot traversing uneven flooring, causing subtle roll or pitch—cannot be modeled by the 2D trajectory graph. This limitation manifests as misaligned seams in the texture map, as the fixed height threshold ($z < 0.1$ m) occasionally captures the lower edges of walls or misses dips in the floor when the robot tilts.

Future improvements would involve upgrading the back-end to a full 3D pose graph ($SE(3)$) to absorb pitch and roll variance, and incorporating visual features from the RGB camera to assist ICP in featureless corridors.

References

[1] N. Atanasov, "Ece 276a: Sensing & estimation in robotics project 2: Project 2: Lidar-based slam," Course handout / website, 2026, accessed: 2026-02-02.

[2] G. Developers, "Gtsam intro tutorial," Online documentation, 2026, accessed: 2026-02-28. [Online]. Available: https://gtsam.org/tutorials/intro.html

[3] N. Atanasov, "Ece 276a: Motion and observation models," Course handout / slides, 2026, accessed: 2026-02-28.

[4] ——, "Ece 276a: Factor graph slam," Course handout / slides, 2026, accessed: 2026-02-28.

[5] ——, "Ece 276a: Localization and odometry," Course handout / slides, 2026, accessed: 2026-02-28.

[6] ——, "Ece 276a: Bayes filter," Course handout / slides, 2026, accessed: 2026-02-28.

[7] ——, "Ece 276a: Particle filter," Course handout / slides, 2026, accessed: 2026-02-28.

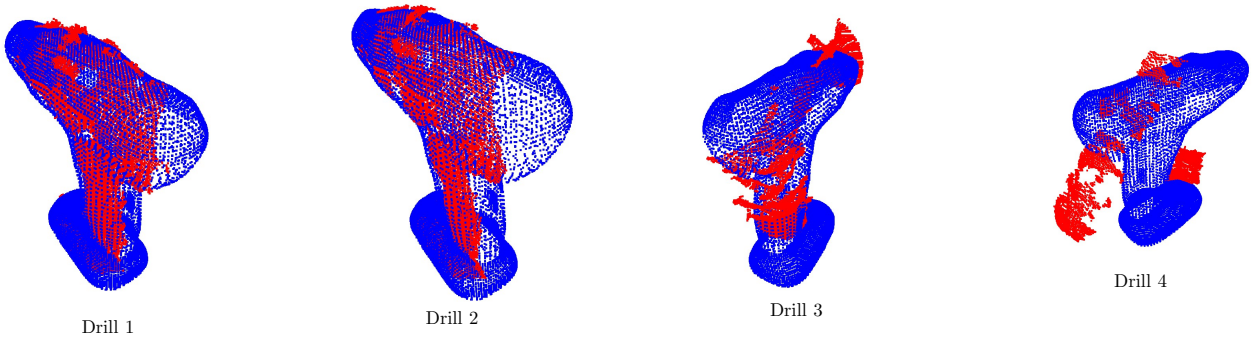
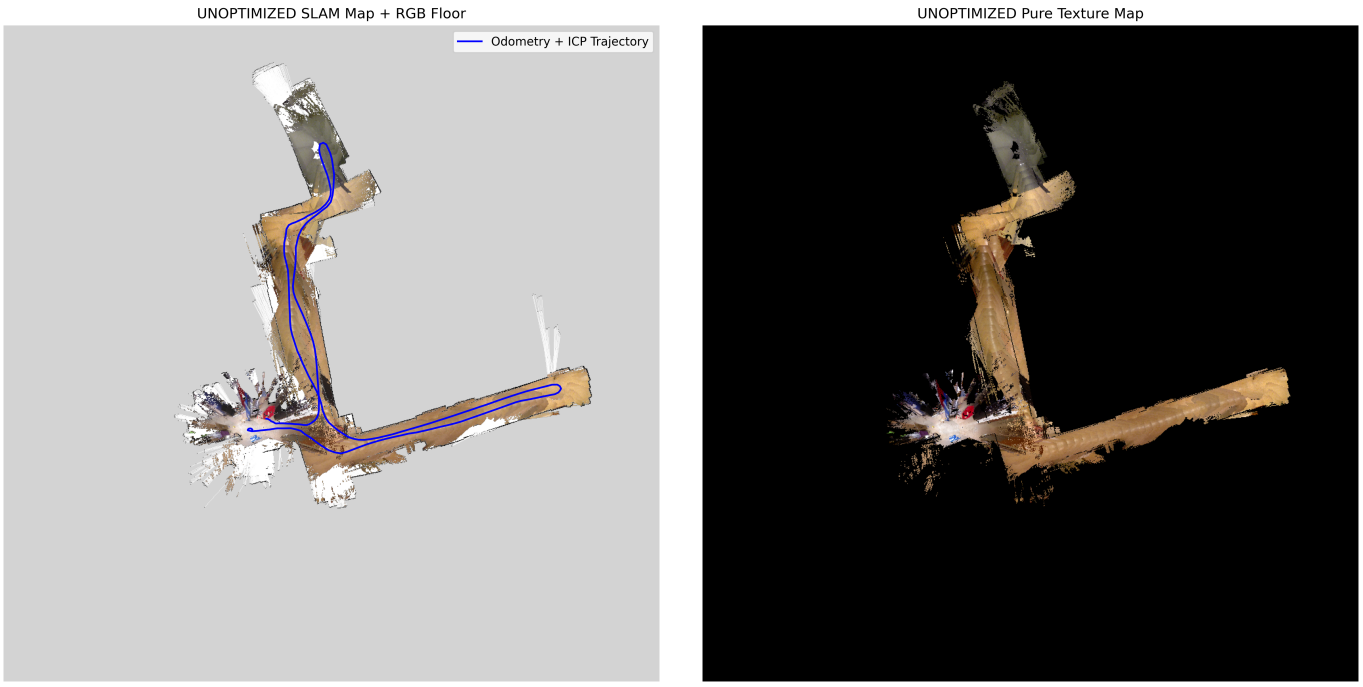


Fig. 1. ICP warm-up objects (drill and liquid container, sets 1–4).

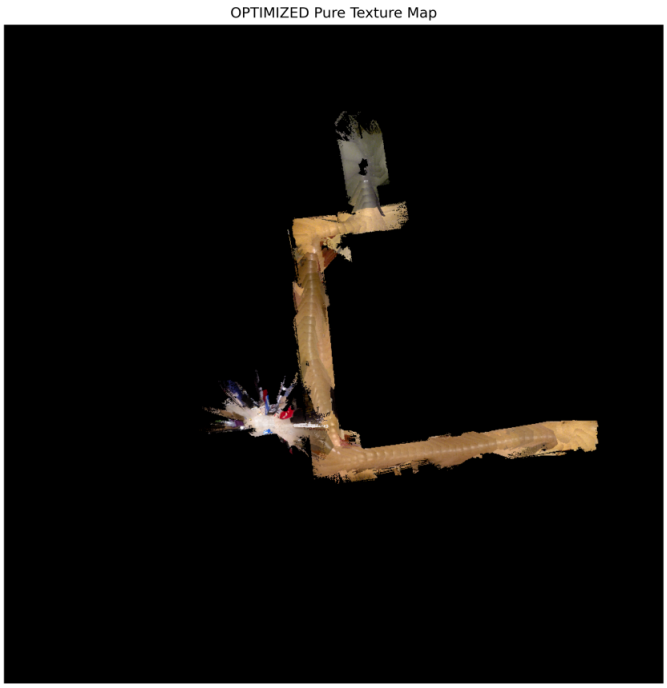


Dataset 20 unoptimized SLAM map (overall)

Dataset 20 unoptimized texture map (overall)

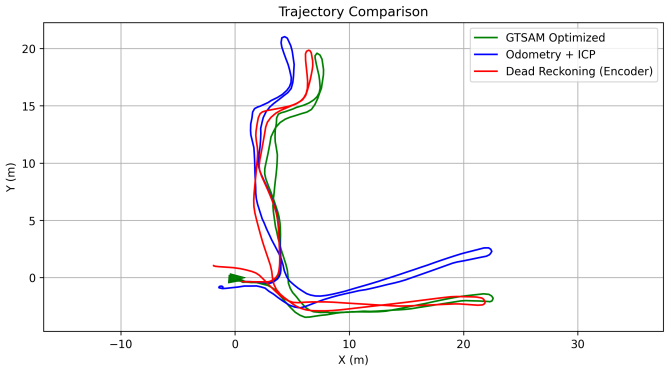
Fig. 2. Dataset 20 unoptimized SLAM and texture maps (overall).

Dataset 20 optimized SLAM map (overall)

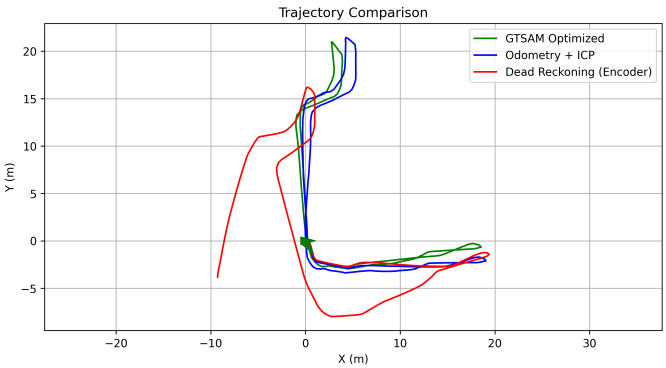


Dataset 20 optimized texture map (overall)

Fig. 3. Dataset 20 optimized SLAM and texture maps (overall).



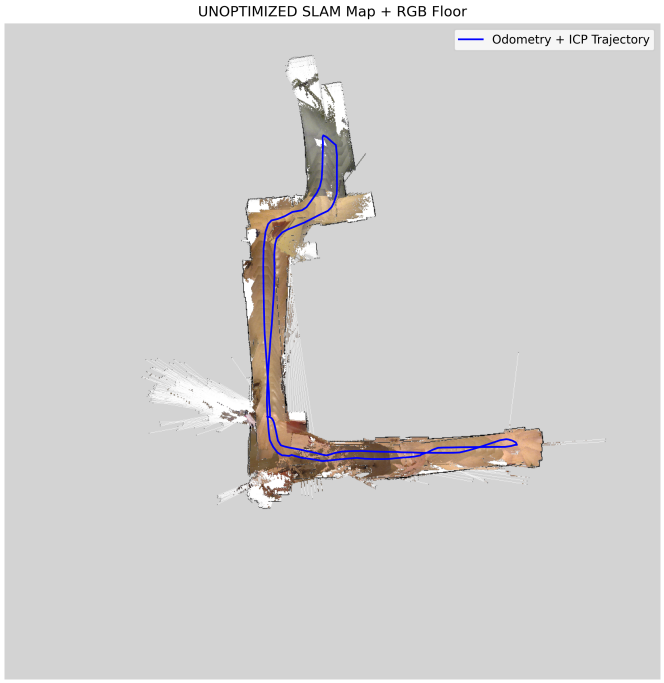
Dataset 20 trajectory comparison (overall)



Dataset 21 trajectory comparison

Fig. 4. Trajectory comparisons for Datasets 20 and 21.

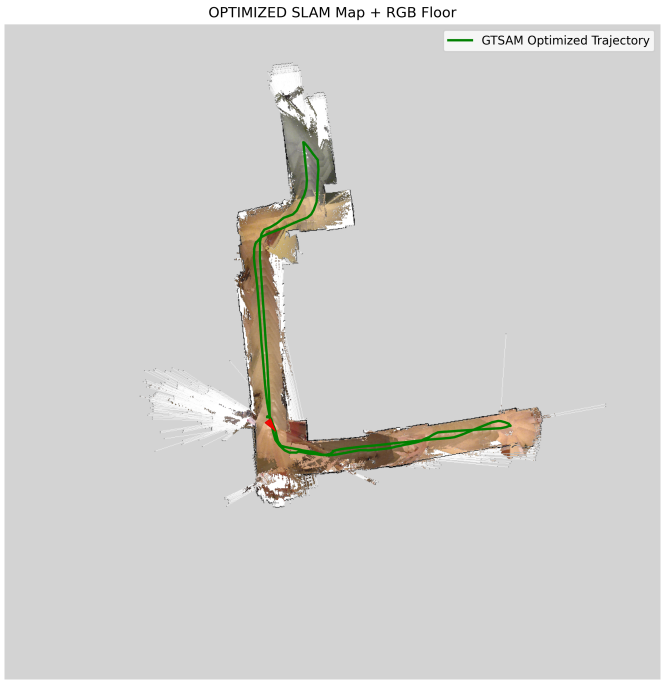
Dataset 21 ICP texture map



Dataset 21 SLAM + texture map (ICP)

Fig. 5. Dataset 21 results (ICP).

Dataset 21 optimized texture map



Dataset 21 SLAM + texture map (optimized)

Fig. 6. Dataset 21 results (optimized).