LiDAR-Based SLAM with Pose Graph Optimization

Anderson Compalas
Department of Electrical and Computer Engineering
University of California San Diego
La Jolla, California
acompalas@ucsd.edu

***Abstract*—This project implements a complete Simultaneous Localization and Mapping (SLAM) pipeline for a differential-drive robot using encoder and IMU odometry, 2D LiDAR scan matching, and RGBD texture mapping. Odometry from wheel encoders and IMU provides an initial trajectory estimate. The ICP algorithm refines this estimate by aligning consecutive LiDAR scans. A probabilistic log-odds occupancy grid is built from the refined trajectory and LiDAR measurements, while Kinect RGBD images are used to colorize the floor as a 2D texture map. Finally, pose graph optimization via GTSAM with loop closure constraints further improves trajectory accuracy and map quality. Results are presented for two datasets collected in an indoor environment.**

***Index Terms*—SLAM, ICP, occupancy mapping, pose graph optimization, GTSAM, loop closure, differential drive**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics: a robot must build a map of an unknown environment while simultaneously estimating its own pose within that map. This is challenging because accurate mapping requires knowing where the robot is, and accurate localization requires knowing the map. SLAM is essential for autonomous navigation in environments where GPS is unavailable, such as indoor buildings, underground facilities, and planetary surfaces.

This project implements a LiDAR-based SLAM pipeline on a differential-drive robot equipped with wheel encoders, an IMU, a Hokuyo UTM-30LX LiDAR, and a Kinect RGBD camera. Our approach proceeds in four stages. First, encoder and IMU measurements are fused using the differential-drive motion model to produce an initial odometry trajectory. Second, 2D LiDAR scan matching via the Iterative Closest Point (ICP) algorithm refines consecutive pose estimates. Third, the refined trajectory is used to build a probabilistic 2D occupancy grid map and a 2D texture map of the floor using RGBD data. Fourth, GTSAM pose graph optimization with loop closure detection corrects accumulated drift and further improves map quality.

II. Problem Formulation

A. Robot Pose and Trajectory

We represent the robot pose at time t in two equivalent forms: (i) a minimal state vector $\mathbf{x}_t := [x_t,\ y_t,\ \theta_t]^\top \in \mathbb{R}^3$, and (ii) a homogeneous transform $T_t \in SE(2)$:

$$T_t = \begin{bmatrix} R(\theta_t) & \mathbf{p}_t \\ \mathbf{0}^\top & 1 \end{bmatrix}, \qquad \mathbf{p}_t = [x_t,\ y_t]^\top. \tag{1}$$

The goal is to estimate the trajectory $\{T_0, \dots, T_T\}$ and a map $\mathcal{M}$ of the environment from asynchronous encoder/IMU, LiDAR, and RGBD measurements.

B. Transform Convention

We use T_B^A to denote the rigid transform that maps coordinates expressed in frame B to frame A (homogeneous coordinates):

$$\mathbf{p}^A = T_B^A\, \mathbf{p}^B. \tag{2}$$

Let w denote the world frame, b the robot body frame, ℓ the LiDAR frame, and d the Kinect depth camera frame. The robot pose is $T_b^w(t)$, and the LiDAR extrinsic is T_ℓ^b, so

$$T_\ell^w(t) = T_b^w(t)\, T_\ell^b. \tag{3}$$

C. Differential-Drive Motion Model

Given linear velocity v_t from encoders and yaw rate ω_t from the IMU, the (continuous-time) differential-drive model is:

$$\dot{x} = v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{\theta} = \omega. \tag{4}$$

Let $\Delta t_t := t_{t+1} - t_t$ be the elapsed time between consecutive encoder timestamps. We propagate pose forward using the exact discrete-time integration over Δt_t (Section III-A).

D. LiDAR Scan Matching

Given two consecutive LiDAR scans $\mathcal{Z}_t = \{\mathbf{z}_t^i\}$ and $\mathcal{Z}_{t+1} = \{\mathbf{z}_{t+1}^j\}$ expressed in the LiDAR frame (2D points in the scan plane), we estimate the relative 2D transform ${}^{\ell_t}T_{\ell_{t+1}} \in SE(2)$ that best aligns the scans:

$${}_{\ell_t}T^*_{\ell_{t+1}} = \arg\min_{T\in SE(2)} \sum_{(i,j)\in\mathcal{C}} \left\| \mathbf{z}_t^j - T\,\mathbf{z}_{t+1}^i \right\|^2, \tag{5}$$

where $\mathcal{C}$ is the set of nearest-neighbor correspondences. Given an estimate of $T_b^w(t)$, we chain poses using the estimated relative motion (Section III-C).

E. Probabilistic Occupancy Grid Map

The environment is represented as a 2D occupancy grid with n cells. Each cell $m_i \in \{-1, +1\}$ is a Bernoulli random variable ($+1$ occupied, -1 free). Under the conditional independence assumption, it suffices to maintain the log-odds for each cell:

$$\lambda_{i,t} := \log \frac{p(m_i = +1 \mid z_{0:t}, x_{0:t})}{p(m_i = -1 \mid z_{0:t}, x_{0:t})}. \tag{6}$$

Using an inverse sensor model, the update is additive:

$$\lambda_{i,t} = \lambda_{i,t-1} + \Delta\lambda_{i,t}, \tag{7}$$

where $\Delta\lambda_{i,t}$ is positive for occupied endpoints and negative for free cells along LiDAR rays. Occupancy probabilities for visualization are recovered by $p(m_i = +1 \mid \cdot) = \sigma(\lambda_{i,t}) = \frac{e^{\lambda_{i,t}}}{1+e^{\lambda_{i,t}}}$. (We use tuned asymmetric increments in Section III-D.)

F. Pose Graph Optimization

The robot trajectory is modeled as a factor graph with nodes $\{x_0, \ldots, x_M\}$ (Pose2 variables) and factors encoding:

- A prior factor fixing x_0 at the origin.
- Odometry (between) factors encoding relative poses between consecutive ICP poses.
- Loop-closure factors encoding relative poses between non-consecutive poses that revisit the same location.

GTSAM optimizes the graph (Levenberg–Marquardt) to produce refined poses $\{x_0^*, \ldots, x_M^*\}$, which are used to rebuild occupancy and texture maps.

III. Technical Approach

A. Encoder and IMU Odometry

Linear velocity is estimated from encoder counts. The project spec confirms encoder counters are reset after each reading, so raw counts are per-step increments. Given tick counts [FR, FL, RR, RL] (front-right, front-left, rear-right, rear-left), the right/left wheel travel distances are:

$$d_R = \frac{FR + RR}{2} \cdot s, \qquad d_L = \frac{FL + RL}{2} \cdot s, \tag{8}$$

where $s = 0.0022$ m/tick (wheel diameter 0.254 m, 360 ticks/rev). The forward displacement over the step is $\Delta s_t = (d_R + d_L)/2$, so

$$v_t = \frac{\Delta s_t}{\Delta t_t}. \tag{9}$$

The yaw rate ω_t is taken from the IMU z-axis and interpolated to encoder timestamps. A midpoint yaw rate $\omega_t \leftarrow (\omega_t + \omega_{t+1})/2$ is used for improved accuracy.

Exact integration. Let $\Delta\theta_t = \omega_t \Delta t_t$. Using the exact discrete-time integration for constant (v_t, ω_t) over Δt_t:

$$x_{t+1} = x_t + v_t \Delta t_t \,\mathrm{sinc}\left(\frac{\Delta\theta_t}{2}\right) \cos\left(\theta_t + \frac{\Delta\theta_t}{2}\right), \tag{10}$$

$$y_{t+1} = y_t + v_t \Delta t_t \,\mathrm{sinc}\left(\frac{\Delta\theta_t}{2}\right) \sin\left(\theta_t + \frac{\Delta\theta_t}{2}\right), \tag{11}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t, \tag{12}$$

where $\mathrm{sinc}(x) = \frac{\sin x}{x}$ with $\mathrm{sinc}(0) = 1$. For numerical stability we also handle the straight-line case $|\omega_t| < 10^{-9}$ using $\Delta x = v_t \Delta t_t \cos\theta_t$, $\Delta y = v_t \Delta t_t \sin\theta_t$.

B. ICP Warm-Up: 3D Point Cloud Registration

Before applying ICP to 2D LiDAR scans, we implemented and validated a 3D ICP algorithm on depth images of a drill and liquid container, following the warm-up instructions.

Algorithm. Given source point cloud $\{s_i\} \subset \mathbb{R}^3$ and target $\{m_j\} \subset \mathbb{R}^3$, ICP iterates:

1) **Correspondences:** For each s_i (after applying current estimate T), find nearest neighbor $m_{j(i)}$ in target via KD-tree.
2) **Outlier rejection:** Discard pairs with distance $> d_{\max}$.
3) **Kabsch/SVD step:** Compute centroids $\bar{s}, \bar{m}$, centered clouds $\tilde{S} = S - \bar{s}$, $\tilde{M} = M - \bar{m}$, cross-covariance $H = \tilde{S}^\top \tilde{M}$, and SVD $H = U\Sigma V^\top$. The optimal rotation is:
$$R^* = V \begin{bmatrix} 1 & & \\ & 1 & \\ & & \det(VU^\top) \end{bmatrix} U^\top, \tag{13}$$
and translation $t^* = \bar{m} - R^* \bar{s}$.
4) **Update:** Apply T_{step} and accumulate $T_{\text{total}} = T_{\text{step}}\, T_{\text{total}}$.

The Kabsch algorithm uses the *original* source points $\{s_i\}$ (not repeatedly transformed points), ensuring the accumulated transform T_{total} maps from the initial canonical pose to the observed pose.

Initialization. Since ICP converges to local optima, we initialize by discretizing yaw into 36 angles (0 to 2π) and selecting the result with lowest MSE, as specified. Translation is initialized by aligning centroids.

C. 2D LiDAR Scan Matching

For the main SLAM pipeline, we use 2D ICP (SE(2)) between consecutive Hokuyo scans.

Scan preprocessing. Raw ranges are filtered to $[0.1\text{ m}, r_{\max}]$: points closer than 0.1 m cause ICP jumps, and NaN/∞ values are excluded. For occupancy mapping, rays reaching $r_{\max}$ (no-return beams) are retained separately with a hit mask. These rays still carve free space but do not mark occupied endpoints.

Odometry initialization. ICP is initialized using the odometry-estimated relative motion between consecutive body poses. Let $T_b^w(t)$ and $T_b^w(t+1)$ be the odometry poses at the two LiDAR timestamps. The relative body transform is:

$${}_{b_t}T_{b_{t+1}} = (T_b^w(t))^{-1} T_b^w(t+1). \tag{14}$$

We convert this into the LiDAR frame using the extrinsic T_ℓ^b:

$${}_{\ell_t}T_{\ell_{t+1}} = (T_\ell^b)^{-1}\, {}_{b_t}T_{b_{t+1}}\, T_\ell^b. \tag{15}$$

Source/target convention. ICP finds T such that $T \cdot \text{source} \approx \text{target}$. To obtain a relative transform consistent with pose chaining, we pass the *newer* scan as source and the *older* scan as target: `icp_2d(scan_t+1, scan_t)`. This yields $T \approx {}^{\ell_t}T_{\ell_{t+1}}$.

MSE gating. After ICP converges, if the final MSE exceeds a threshold (0.05), the step is discarded and the odometry relative transform is used instead. This prevents badly converged ICP steps from corrupting the trajectory.

Pose chaining. Let the ICP estimate in the LiDAR frame be ${}^{\ell_t}T_{\ell_{t+1}}$. We update body poses by:

$$T_b^w(t+1) = T_b^w(t)\, T_\ell^b\, {}^{\ell_t}T_{\ell_{t+1}}\, (T_\ell^b)^{-1}. \tag{16}$$

D. Probabilistic Occupancy Mapping

We maintain a log-odds occupancy grid over the map plane. For each pose and LiDAR scan:

1) Transform scan points from LiDAR frame ℓ to world using $T_\ell^w(t) = T_b^w(t)\, T_\ell^b$.
2) For each ray, use Bresenham's line algorithm to enumerate all cells from the robot origin to the endpoint. Bresenham operates on grid indices, so world coordinates are converted to cell indices before the call.
3) For *real hit* beams: decrement log-odds of all cells along the ray except the endpoint by $|l_{\text{free}}| = 0.5$, and increment the endpoint cell by $l_{\text{occ}} = 2.0$.
4) For *no-return* beams: decrement log-odds of all cells along the ray *including* the endpoint, since no obstacle was detected.

We use asymmetric updates ($l_{\text{occ}} = 2.0$, $l_{\text{free}} = -0.5$; ratio 4:1) to reflect that a valid Hokuyo hit is highly reliable while a single free ray should not erase a wall. Log-odds are clipped to $[-10, 10]$ to avoid overconfident saturation. The final map is visualized as probabilities via $p = \sigma(\lambda) = \frac{1}{1+e^{-\lambda}}$.

E. Texture Mapping

The Kinect depth camera is located at $(0.18, 0.005, 0.36)$ m relative to robot center with pitch 0.36 rad, yaw 0.021 rad, and roll 0. The optical frame (Z-forward, Y-down, X-right) is first rotated to body-aligned axes, then the physical mount angles are applied:

$$R_d^b = R_z(\psi)\, R_y(\phi)\, R_x(\rho)\, R_{\text{opt}}^b, \tag{17}$$

where R_{opt}^b maps the optical frame to standard body convention ($x_b = z_{\text{opt}}$, $y_b = -x_{\text{opt}}$, $z_b = -y_{\text{opt}}$).

Disparity d at pixel (i, j) is converted to depth and the associated RGB pixel location per the spec formulas:

$$dd = -0.00304\, d + 3.31, \tag{18}$$

$$Z = 1.03/dd, \tag{19}$$

$$\text{rgb}_i = (526.37\, i + 19276 - 7877.07\, dd)/585.051, \tag{20}$$

$$\text{rgb}_j = (526.37\, j + 16662)/585.051, \tag{21}$$

where i is the *column* index and j is the *row* index. Disparity images are loaded with `cv2.IMREAD_ANYDEPTH` to preserve raw 16-bit values.

3D points in the depth camera frame are computed from the depth image using the intrinsic matrix K, then transformed to world frame via the robot pose and Kinect extrinsic. Floor points are identified by thresholding on world-frame height ($|z_{\text{world}}| < 0.1$ m) and their corresponding RGB colors are painted onto a grid at the same resolution as the occupancy map.

F. Pose Graph Optimization with GTSAM

We construct a GTSAM factor graph over ICP poses $\{x_0, \ldots, x_M\}$ (Pose2 variables).

Odometry factors. For each consecutive pair, a `BetweenFactorPose2` encodes the relative ICP pose with a diagonal noise model $\boldsymbol{\sigma}_{\text{odom}} = [0.1 \text{ m},\ 0.1 \text{ m},\ 0.05 \text{ rad}]$ (per TA recommendation).

Loop closure — fixed interval. Every 10 poses, ICP is run between poses k and $k + 10$. If the ICP MSE is below a threshold, a loop closure factor is added.

Loop closure — proximity-based. A KD-tree is built over the ICP pose positions. For each pair (i, j) with $j - i \geq N_{\text{gap}}$ and Euclidean distance below radius r, ICP is run. Accepted loop closures (MSE below threshold) are added as factors.

Consistency check. Both loop closure methods include an MSE-based consistency check before adding a factor, ensuring only geometrically consistent constraints enter the graph.

Noise model. Loop closure factors use a looser noise model $\boldsymbol{\sigma}_{\text{loop}} = [0.3 \text{ m},\ 0.3 \text{ m},\ 0.1 \text{ rad}]$ wrapped in a Huber robust kernel to prevent outlier loop closures from corrupting the solution.

Optimization. The graph is optimized using GTSAM's Levenberg–Marquardt optimizer. The resulting optimized poses are used to rebuild both the occupancy and texture maps.

IV. Results

A. ICP Warm-Up: 3D Point Cloud Registration

The 3D ICP algorithm was validated on depth images of two objects, a drill and a liquid container, each captured from four different viewing directions. The canonical model (source, shown in blue) was aligned to each observed point cloud (target, shown in red) using yaw-discretized initialization over 36 angles from 0 to 2π, with centroid-aligned translation as the initial guess. The result with the lowest final MSE across all initializations was selected.

Figs. 1 and 2 show the aligned point clouds in three orthogonal projections (XY, XZ, YZ) for all four viewpoints of each object. Table I summarizes the final MSE and point cloud sizes for each run.

TABLE I: ICP Warm-Up Final MSE Results

Object	PC	Source pts	Target pts	Final MSE
Drill	0	14,298	11,690	1.61×10^{-4}
	1	14,298	11,824	1.58×10^{-4}
	2	14,298	8,569	1.50×10^{-4}
	3	14,298	5,032	2.98×10^{-4}
Liq. container	0	46,563	30,954	1.08×10^{-3}
	1	46,563	22,897	9.36×10^{-4}
	2	46,563	21,661	1.51×10^{-3}
	3	46,563	19,544	9.86×10^{-4}

Drill. The drill's asymmetric geometry — comprising a distinct body, handle, and bit — provides strong rotational constraints across all projections. ICP converged to accurate alignments in all four cases, with the aligned source consistently overlapping the target in the XZ and YZ views (Fig. 1). Mean MSE across the four point clouds was 1.92×10^{-4} m^2.

PC 3 yielded the highest MSE (2.98×10^{-4}), which is attributable to its significantly smaller target point cloud (5,032 points vs. $\sim$11,000 for the others), reducing correspondence coverage and increasing sensitivity to noise.

Liquid container. The liquid container presents a more challenging registration problem due to its near-cylindrical geometry. Mean MSE across the four point clouds was 1.13×10^{-3} m^2, approximately $5.9\times$ higher than the drill. While the XY projection (top-down view) shows a circular boundary with good overlap in all cases, the XZ and YZ projections reveal a consistent residual tilt misalignment (Fig. 2). This is attributable to the rotational ambiguity inherent to cylindrically symmetric objects: the top-down circular projection provides no yaw gradient, so the objective landscape is nearly flat in the yaw direction, making it difficult for ICP to reliably recover the correct orientation from yaw initialization alone. Despite this, the overall shape boundary aligns plausibly in all four viewpoints.

B. Dataset 20

1) Part 1 — Encoder/IMU Odometry: Fig. 3a shows the odometry trajectory for Dataset 20, estimated using exact differential-drive integration with IMU yaw rate fused at encoder timestamps. The robot begins at the origin, makes a brief southward excursion to $y \approx -5$ m, then traverses a long northward corridor reaching $y \approx 20$ m, executes a U-turn, and finally sweeps east to $x \approx 21$ m before returning near the start. The trajectory covers an area of approximately 21×25 m, consistent with a multi-room indoor environment. The end point lands at approximately $(-7, -2)$ m relative to the start, indicating modest accumulated drift over the full run.

2) Part 2 — ICP Scan Matching: Fig. 3b shows the ICP-refined trajectory for Dataset 20. The overall topology is identical to odometry, the same southward dip, northward corridor, loop, and eastward sweep are all preserved, confirming that ICP converged correctly throughout the run. The ICP trajectory is initialized from odometry at the first LiDAR timestamp, so both trajectories share a common starting frame.

The overlay in Fig. 3c reveals two systematic differences between the two estimates. First, the ICP trajectory is more compact: the northward corridor reaches only $y \approx 16$ m (vs. 20 m for odometry) and the eastward sweep extends to $x \approx 13$ m (vs. 21 m). This compression reflects the removal of wheel slip that accumulates in odometry over long straight runs, ICP directly measures scan-to-scan displacement rather than integrating encoder counts, and is therefore less susceptible to this error source. Second, the ICP end point $(-5, 2)$ m is closer to the origin than the odometry end point $(-7, -2)$ m, indicating reduced overall drift. The residual discrepancy between the two trajectories motivates the pose graph optimization in Part 4.

3) Part 3 — Occupancy and Texture Mapping: First-scan sanity check. Fig. 4 shows the occupancy map constructed from the very first LiDAR scan at the robot's starting pose. The map displays several radiating free-space arms corresponding to the Hokuyo's 270° sweep, with shorter arms indicating nearby walls and longer arms indicating open corridors. The multi-directional pattern is consistent with the robot starting at a corridor intersection or lobby junction. This sanity check validates that the sensor-frame-to-world-frame transformation chain ($T_\ell^w(t) = T_b^w(t)\, T_\ell^b$) and Bresenham ray casting are functioning correctly on a single scan.

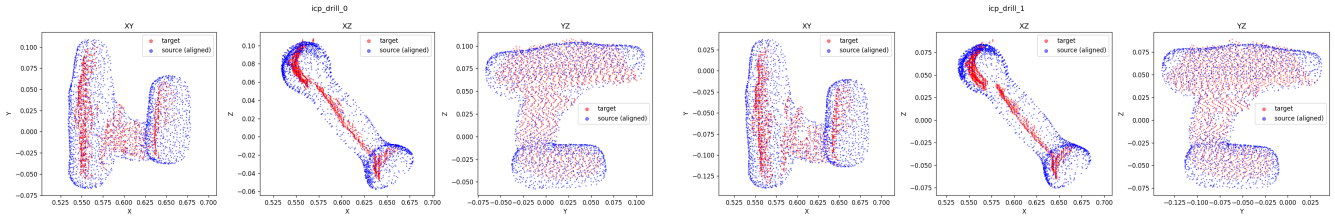
Occupancy map progression. Fig. 5 shows the ICP-based occupancy map at 24%, 50%, 74%, and 100% of the trajectory. By 24% (Fig. 5a), the primary building structure is already recognizable: a vertical corridor running north–south and a horizontal corridor extending east are clearly delineated, with crisp black walls and confirmed white free space. By 50% (Fig. 5b), the upper room ($y \approx$ 10–22 m, $x \approx$ 0–10 m) is fully resolved and the eastward hallway is completely carved out. The 74% snapshot (Fig. 5c) adds only the lower-left room cluster ($x \approx -12$ to -5 m). The final map at 100% (Fig. 5d) shows four distinct spatial regions: a large upper room, a central connecting corridor, a lower-left room, and an eastward hallway, covering approximately 27×27 m.

Final ICP occupancy and texture maps. The final ICP occupancy map (Fig. 6a) shows well-defined walls at $P(\text{occupied}) \approx 1.0$ (black) and confirmed free space at $P(\text{occupied}) \approx 0.0$ (white). Some residual clutter in the lower-left room is attributable to dynamic obstacles or occasional ICP fallback on geometrically degenerate corridor segments. The texture map (Fig. 6b) colorizes the floor using Kinect RGBD data at 5 cm resolution. The floor is predominantly warm tan/beige, consistent with wood/linoleum tile typical of the indoor environment. A gray patch in the upper room indicates either a different floor material or a shallower Kinect observation angle. A pink/blue cluster near the central junction corresponds to dynamic objects passing through the scene. The texture map footprint aligns closely with the occupancy map, confirming correct Kinect extrinsic calibration.

4) Part 4 — Pose Graph Optimization: The pose graph for Dataset 20 contained 4,961 odometry factors, 496 fixed-interval loop closure factors (every 10 poses), and 1 proximity-based loop closure (MSE = 0.0131, detected at $r < 0.01$ m with a minimum pose gap of 2,000). The optimizer converged in 4 iterations, reducing the graph error from 5.7240 to 2.0802, a 63.7% reduction (Table II). The proximity loop closure confirms the robot physically revisited a prior location, providing a global constraint that anchors the trajectory correction.

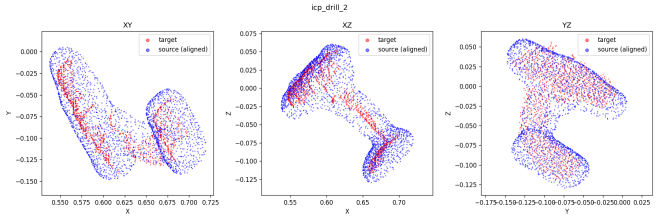
Fig. 7 shows the GTSAM-optimized trajectory (red) overlaid on the ICP trajectory (blue). The GTSAM corridor is shifted approximately 1 m to the right relative to ICP with more internally consistent walls, indicating that the optimizer redistributed residual heading drift across the corridor traversals. The eastward sweep extends further ($x \approx 15$ m vs. 13 m for ICP) and with less southward bias. The optimized end point at $(-4, 3)$ m is closer to the origin than the ICP end point at $(-5, 2)$ m. This global correction improves map consistency by bringing revisited structure (corridors and junctions) into better alignment.

Figs. 8 and 9 show the GTSAM occupancy and texture maps. Wall clarity improves relative to ICP, particularly in

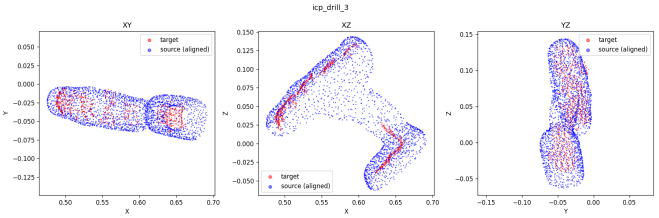


(a) PC 0

(b) PC 1

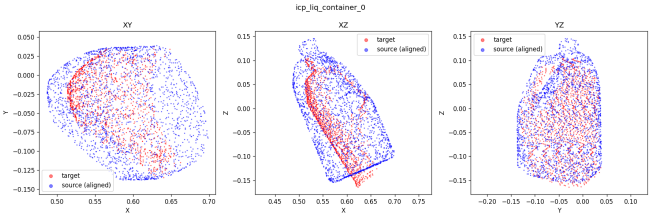


(c) PC 2

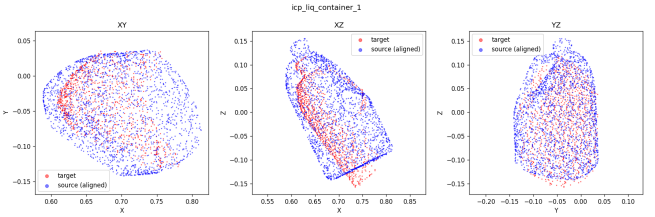


(d) PC 3

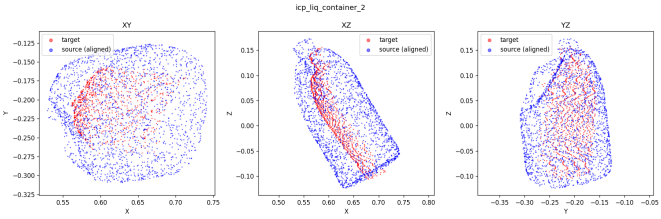
Fig. 1: 3D ICP warm-up results for the *drill* object across four point clouds. Each subfigure shows three orthogonal projections (XY, XZ, YZ). Red: target (observed). Blue: source (canonical model, aligned by ICP). The drill's asymmetric geometry enables accurate alignment across all viewpoints, with blue and red boundaries closely overlapping in all three projections.



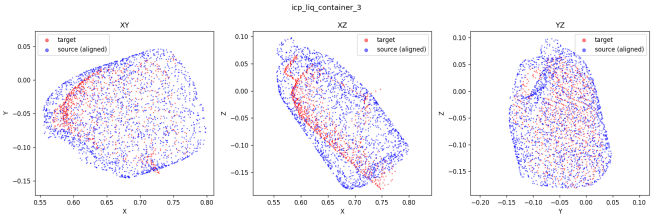
(a) PC 0



(b) PC 1



(c) PC 2



(d) PC 3

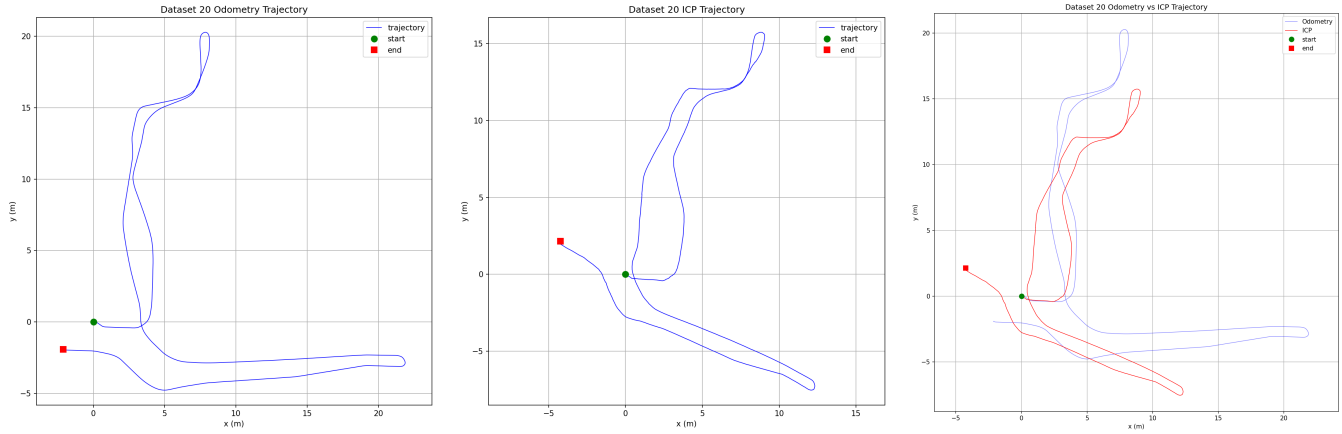
Fig. 2: 3D ICP warm-up results for the *liquid container* object across four point clouds. Red: target. Blue: source (aligned). The near-cylindrical shape introduces rotational ambiguity visible as a consistent residual tilt offset in the XZ and YZ projections, while the circular XY boundary aligns well across all cases. MSE is approximately $5.9\times$ higher than for the drill on average.

the main corridor and upper room where walls appear as sharper, more continuous black lines rather than the slightly blurred walls of the ICP map. The texture map shows modest improvement in floor registration: corridor strips are more continuous and the eastern hallway coverage extends slightly further, consistent with the corrected trajectory extent.

C. Dataset 21

1) Part 1 — Encoder/IMU Odometry: Fig. 10a shows the odometry trajectory for Dataset 21. The robot starts at the origin, dips briefly south to $y \approx -5$ m, then traverses a northward corridor reaching $y \approx 20$ m, executes a U-turn, and sweeps east to $x \approx 20$ m before returning near the start at $(-4, -1.5)$ m. This topology mirrors Dataset 20, confirming both runs were conducted in the same indoor environment. A notable difference is that the northward corridor in Dataset 21 is positioned further west ($x \approx -3$ to 2 m) compared to Dataset 20 ($x \approx 0$ to 4 m), reflecting a different starting heading.

2) Part 2 — ICP Scan Matching: The ICP trajectory (Fig. 10b) shows the same L-shaped topology as odometry, but



(a) Odometry trajectory (b) ICP trajectory (c) Odometry vs. ICP overlay

Fig. 3: Dataset 20 robot trajectories. (a) Odometry (encoder + IMU) trajectory spanning approximately 21×25 m. (b) ICP scan-matching trajectory initialized from odometry; same topology but reduced drift. (c) Direct overlay showing the ICP trajectory (red) is more compact than odometry (blue), consistent with removal of wheel-slip accumulation.

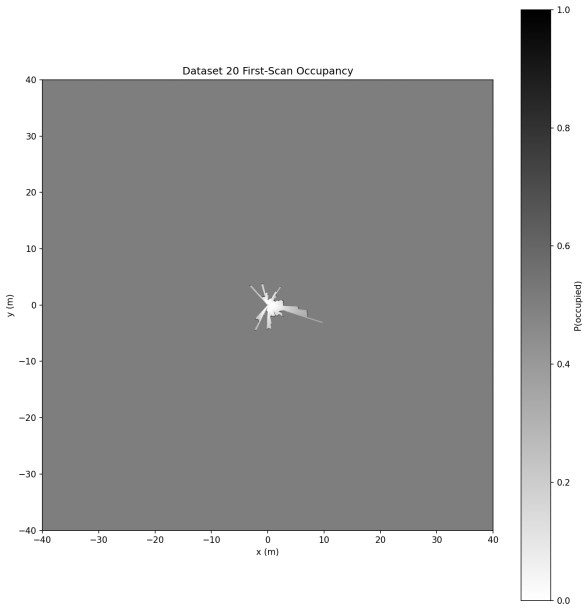


Fig. 4: Dataset 20 first-scan occupancy map. Radiating arms from the robot's starting position correspond to the Hokuyo's 270° sweep, confirming correct sensor-to-world transformation and ray casting on a single scan.

more compact: the north corridor peaks at $y \approx 14$ m (vs. 20 m for odometry, a 30% reduction) and the east sweep reaches only $x \approx 14$ m (vs. 20 m). The overlay in Fig. 10c reveals that ICP substantially corrects the leftward heading bias present in odometry, the odometry corridor drifts to $x \approx -3$ m while ICP stays near $x \approx 0$–2 m. Both estimates end near $(-5, -1)$ m, indicating that while ICP removes mid-run wheel slip, some residual end-point drift remains.

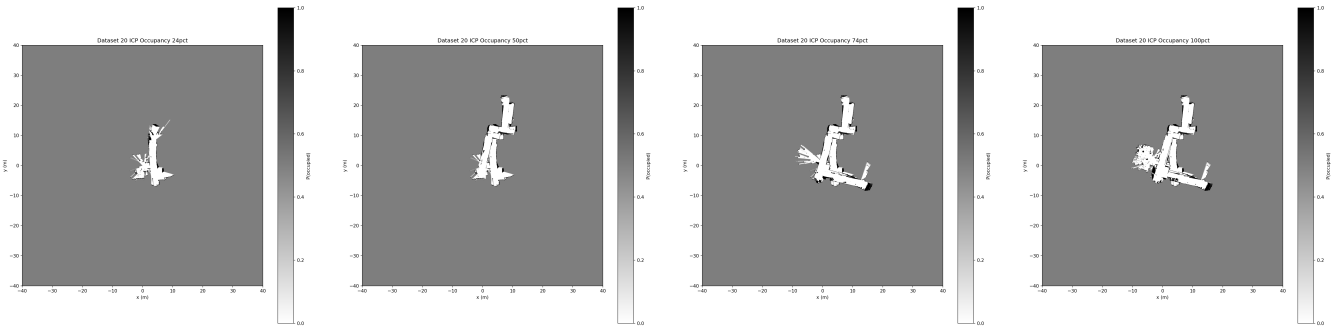
A notable artefact is doubled corridor walls visible in the north corridor from 24% onward (Figs. 12a–d). This indicates residual heading drift accumulating over the two traversals of the same corridor in opposite directions, causing outgoing and return scans to be projected with a slight angular offset.

3) Part 3 — Occupancy and Texture Mapping: First-scan sanity check. Fig. 11 shows the first-scan occupancy for Dataset 21. Compared to Dataset 20, fewer radiating arms are visible and they are concentrated predominantly northward, consistent with the robot starting inside a narrow corridor rather than at an intersection. The sanity check confirms correct sensor-to-world transform initialization.

Occupancy map progression. Fig. 12 shows the ICP occupancy map at 24%, 49%, 74%, and 100% of poses. The east hallway is already roughed in at 24% and fully formed with clean walls by 49%. The upper room ($y \approx 10$–21 m) appears at 74% as the robot enters it in the latter portion of the run. Throughout all snapshots, the north corridor exhibits doubled wall artefacts caused by the residual heading drift noted above. The lower-left lobby cluster ($x \approx -10$ m) is also noisier than in Dataset 20, suggesting more dynamic content in that region during this recording session.

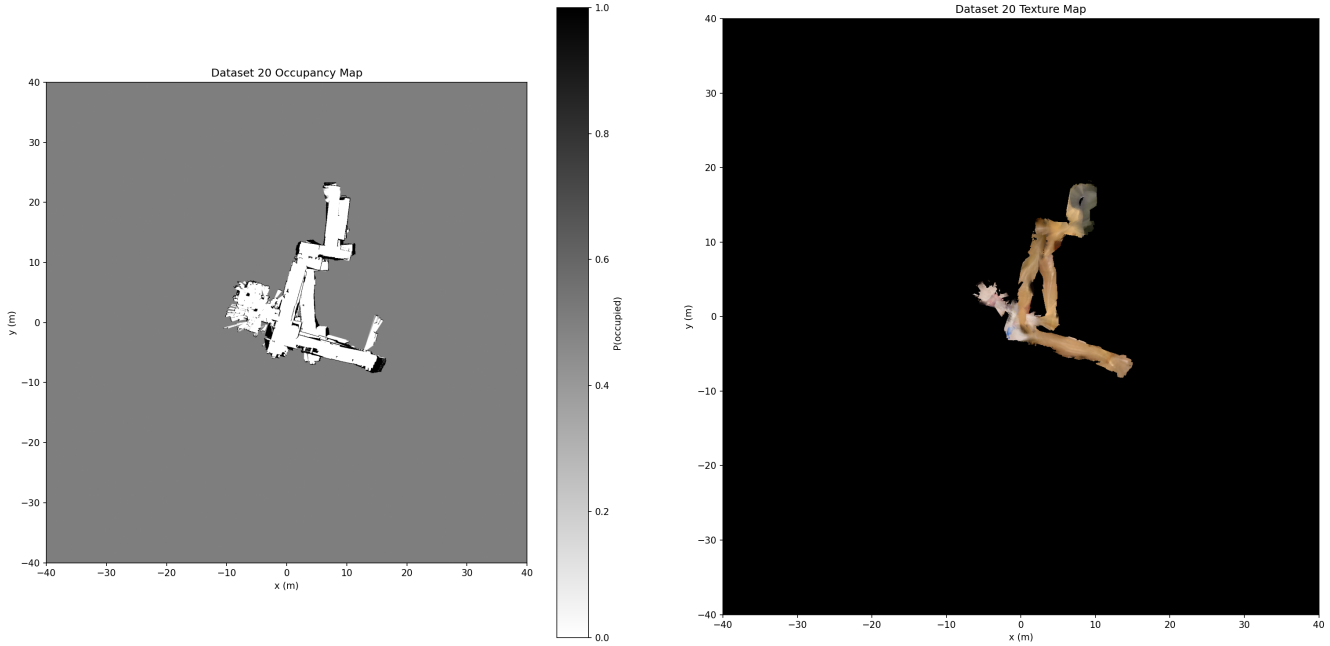
Final ICP occupancy and texture maps. The final maps are shown in Fig. 13. The floor plan forms a distinct L-shape: a northward corridor connecting to an upper room, with an eastward hallway branching from the junction. The texture map shows the same tan/beige linoleum as Dataset 20, with a gray patch in the upper room and dynamic-object clusters near the junction. The north corridor shows two parallel floor color strips — the texture-domain signature of the doubled wall artefact.

4) Part 4 — Pose Graph Optimization: Unlike Dataset 20, the proximity-based loop closure detector found **zero** candidate pairs for Dataset 21, the robot path never returned within 0.01 m of a prior pose with sufficient temporal separation. Consequently, the pose graph contains only 4,784



(a) 24% (b) 50% (c) 74% (d) 100%

Fig. 5: Dataset 20 ICP occupancy map progression at 24%, 50%, 74%, and 100% of poses processed. The floor plan is already coherent at 24% and remains stable through 100%, with new rooms added incrementally as the robot explores them.



(a) Final ICP occupancy map (b) ICP texture map

Fig. 6: Dataset 20 final maps from ICP trajectory. (a) Occupancy map showing four distinct spatial regions with crisp black walls ($P \approx 1.0$) and confirmed white free space ($P \approx 0.0$). (b) Texture map colorizing the floor with Kinect RGB data; warm tan/beige indicates linoleum tile, with dynamic-object clutter visible near the central junction.

odometry factors and 478 fixed-interval smoothing constraints. The optimizer still reduced the graph error from 9.1485 to 4.1764, a 54.3% reduction (Table II), but the correction is primarily smoothing rather than a globally anchoring loop constraint. Dataset 21's higher initial error (9.15 vs. 5.72) reflects greater accumulated drift, likely due to longer stretches of corridor motion with weaker geometric constraints for ICP.

Fig. 14 shows the GTSAM-optimized trajectory (red) overlaid on ICP (blue). The corridor is shifted rightward and elongated ($y \approx 17$ m vs. 14 m for ICP) as the smoother redistributes accumulated drift, and the end point moves from $(-5, -1)$ m to $(-2, 0)$ m — substantially closer to the origin.

Fig. 15 shows the GTSAM occupancy progression. At 24% and 49%, the maps are nearly identical to their ICP counterparts. By 74% and 100%, the corridor doubled-wall artefact is modestly reduced as fixed-interval smoothing redistributes heading corrections incrementally. The lower-left lobby cluster remains noisy with no loop closure to anchor that region.

The GTSAM texture map (Fig. 16) shows the most tangible benefit: the north corridor displays a single, more coherent floor strip rather than the two parallel strips of the ICP texture map. This confirms that the heading correction from smoothing, while insufficient to fully resolve occupancy map doubling, is sufficient to merge the two corridor passes in the

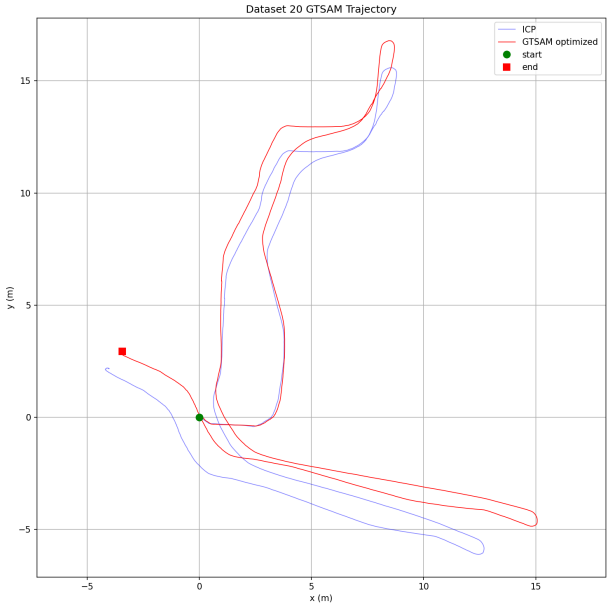


Fig. 7: Dataset 20 ICP (blue) vs. GTSAM-optimized (red) trajectory. Loop closure constraints shift the corridor rightward and extend the eastward sweep, redistributing accumulated ICP drift globally across the trajectory.

texture projection.

D. Cross-Dataset GTSAM Summary

Table II compares pose graph optimization results across both datasets. Dataset 20 benefits from one proximity-based loop closure in addition to fixed-interval smoothing, yielding a lower final error (2.08 vs. 4.18) and cleaner map walls. Dataset 21, lacking any proximity closure, relies entirely on smoothing and retains more residual drift artefacts. The fixed-interval MSE is identical for both datasets (mean 0.0009), confirming that consecutive-scan ICP quality is consistent across runs; the difference in outcome is attributable to whether a revisitation event was detected.

TABLE II: GTSAM Pose Graph Optimization Summary

Parameter	Dataset 20	Dataset 21
Total poses	4,962	4,785
Odometry factors	4,961	4,784
Fixed-interval closures	496	478
Proximity-based closures	1	0
Fixed-interval MSE (mean)	0.0009	0.0009
Initial graph error	5.7240	9.1485
Final graph error	2.0802	4.1764
Error reduction	63.7%	54.3%

V. Discussion

A. What Worked Well

The most impactful fix in this project was correcting the ICP source/target convention. ICP finds T such that $T \cdot \text{source} \approx \text{target}$; passing the newer scan as source and older as target yields a transform with the correct sign convention for pose chaining. Before this fix, every ICP step returned the inverse transform, producing a mirrored trajectory and arc-shaped occupancy map artifacts. After the fix, the ICP trajectory closely matched the odometry trajectory in shape and scale.

No-return beam handling significantly improved occupancy map clarity. Previously, beams that reached maximum range were discarded entirely, leaving large open areas as "unknown" grey. By carving free space along these rays (including the endpoint), corridors and rooms now appear properly free in the map.

MSE-gated ICP fallback prevented trajectory corruption from occasional poor ICP convergence, which commonly occurs in geometrically degenerate environments such as long featureless corridors.

B. Challenges

Point-to-point ICP is known to struggle with rotational estimation in corridors, where the dominant geometric feature (parallel walls) provides weak rotational constraints. This manifests as heading drift that accumulates over the trajectory. GTSAM pose graph optimization partially compensates by redistributing this error globally.

The texture map depends on accurate Kinect extrinsic calibration. The optical-to-body frame rotation convention required careful handling: the Kinect optical frame uses Z-forward, Y-down, X-right, which must be rotated to body convention before applying mount angles.

C. Lessons Learned

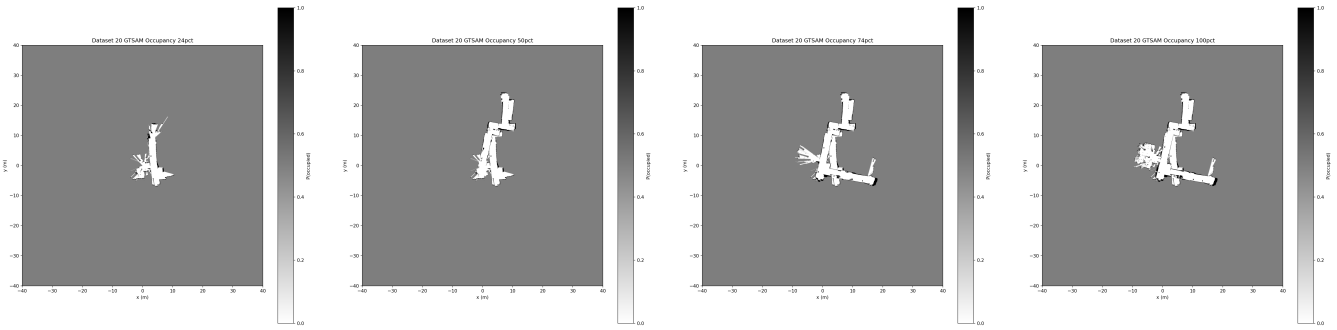
Coordinate frame conventions are a frequent source of subtle bugs. Explicit documentation of each frame (optical, body, LiDAR, world) and each transform direction (T_B^A maps from B to A) is essential. The source/target convention in ICP and the row/column convention for disparity pixel indices (i = column, j = row per Piazza) are two examples where unclear conventions led to incorrect results that were not immediately obvious from output inspection.

VI. Conclusion

This project implemented a complete LiDAR-based SLAM pipeline combining differential-drive odometry, ICP scan matching, probabilistic occupancy mapping, RGBD texture mapping, and GTSAM pose graph optimization. The pipeline successfully produced occupancy grid maps and texture maps of two indoor environments. Key contributions include correct ICP source/target convention for consistent pose chaining, no-return beam free-space carving for improved map clarity, MSE-gated ICP fallback for trajectory robustness, and GTSAM-based loop closure (when available) for global drift correction.

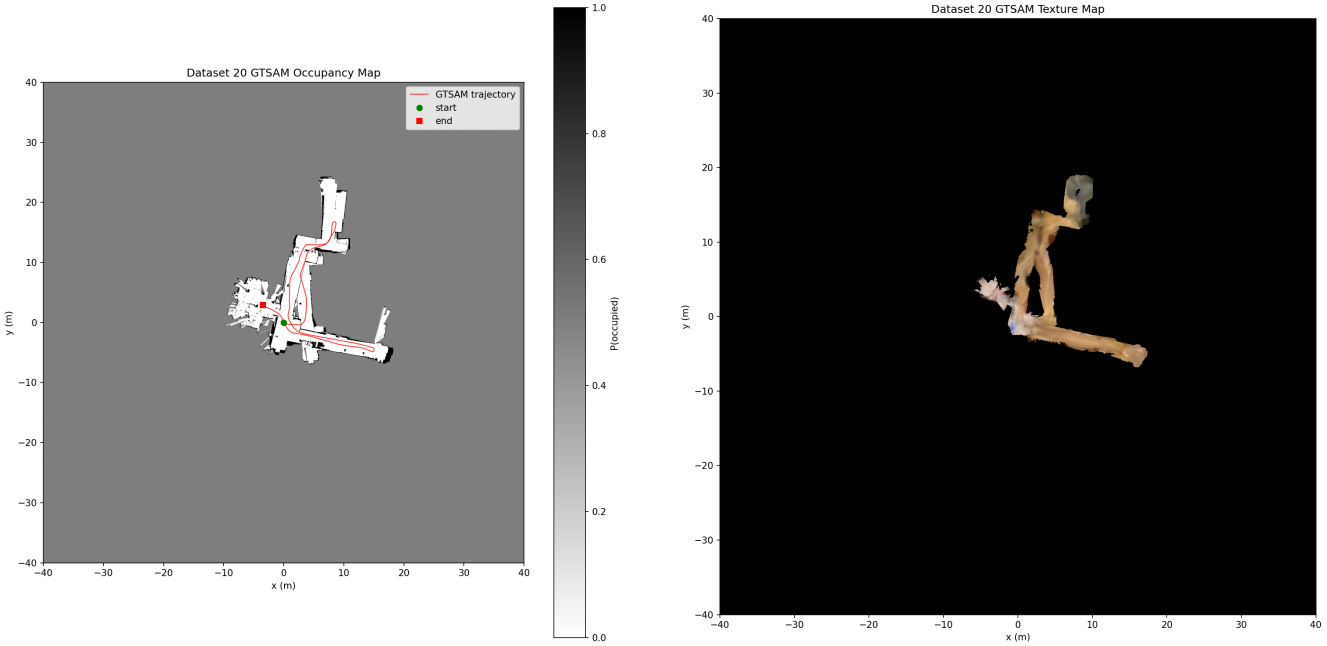
Acknowledgments

We thank Professor Nikolay Atanasov and the ECE 276A teaching assistants for providing the project framework, datasets, and Piazza feedback.



(a) 24% (b) 50% (c) 74% (d) 100%

Fig. 8: Dataset 20 GTSAM occupancy map progression at 24%, 50%, 74%, and 100%. Wall clarity improves relative to the ICP map, particularly in the main corridor where loop closure enforces geometric consistency.



(a) GTSAM occupancy map with trajectory (b) GTSAM texture map

Fig. 9: Dataset 20 final GTSAM maps. (a) Occupancy map with optimized trajectory overlay (red); the path runs cleanly through free-space regions and the end point is closer to the origin than the ICP estimate. (b) Texture map showing improved floor registration relative to ICP, with wider east corridor coverage consistent with the corrected trajectory.

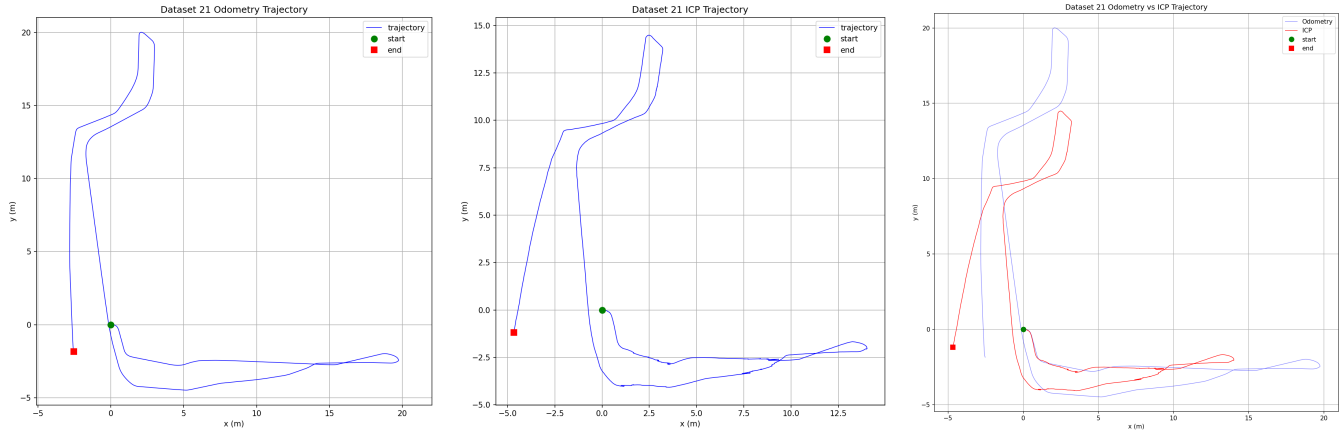
REFERENCES

[1] N. Atanasov, "ECE 276A: Sensing & Estimation in Robotics – Project 2," University of California San Diego, 2026.

APPENDIX

A. *Project Structure*

```
ECE276A_PR2/
  code/
    slam.py             # Main SLAM pipeline (Parts 1-4)
    pr2_utils.py        # Provided utilities (bresenham2D, etc.)
    icp_warm_up/
      icp.py            # 3D ICP warm-up implementation
      test_icp.py       # Provided warm-up test script
      utils.py          # Provided warm-up utilities
```



(a) Odometry trajectory (b) ICP trajectory (c) Odometry vs. ICP overlay

Fig. 10: Dataset 21 robot trajectories. (a) Odometry trajectory spanning approximately 20×25 m with a leftward corridor bias. (b) ICP trajectory — same L-shaped topology but 30% more compact and with the corridor shifted rightward. (c) Overlay showing ICP (red) correcting the leftward heading drift of odometry (blue).

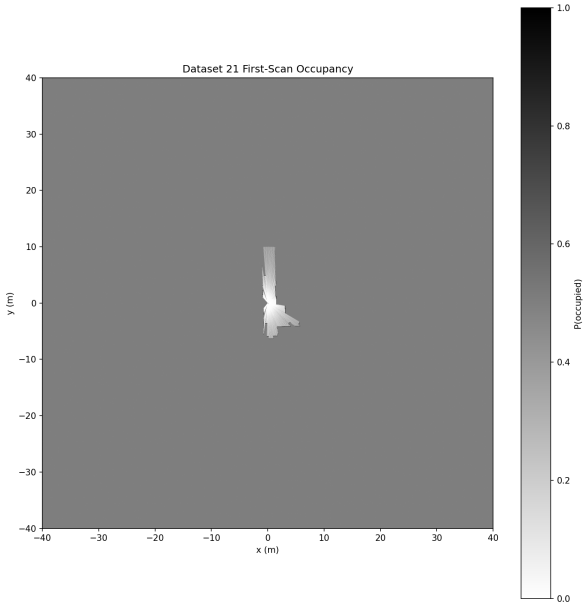


Fig. 11: Dataset 21 first-scan occupancy. Fewer and more northward-oriented arms compared to Dataset 20, consistent with starting inside a narrow corridor rather than at a multi-way intersection.

```
      data/              # Warm-up point cloud data
  data/
    Encoders20.npz      Encoders21.npz
    Imu20.npz           Imu21.npz
    Hokuyo20.npz        Hokuyo21.npz
    Kinect20.npz        Kinect21.npz
    dataRGBD/
      Disparity20/      RGB20/
      Disparity21/      RGB21/
  docs/
```

B. Dependencies

```
pip install numpy scipy matplotlib opencv-python
    gtsam
```

C. ICP Warm-Up (Part 2a)

```
cd ECE276A_PR2/code/icp_warm_up
python3 icp.py
```

Results are saved as `icp_drill_{0-3}.png` and `icp_liq_container_{0-3}.png` in the same directory.

D. Main SLAM Pipeline (Parts 1–4)

All parts are run from the `code/` directory. Output PNGs and cache files (`cache_ds{20,21}_part{1-4}.npz`) are saved there automatically.

Run all parts on both datasets:

```
cd ECE276A_PR2/code
python3 slam.py
```

Run individual parts:

```
python3 slam.py --parts 1           # odometry only
python3 slam.py --parts 2           # ICP (loads
    Part 1 cache)
python3 slam.py --parts 3           # occupancy +
    texture maps
python3 slam.py --parts 4           # GTSAM
    optimization
```

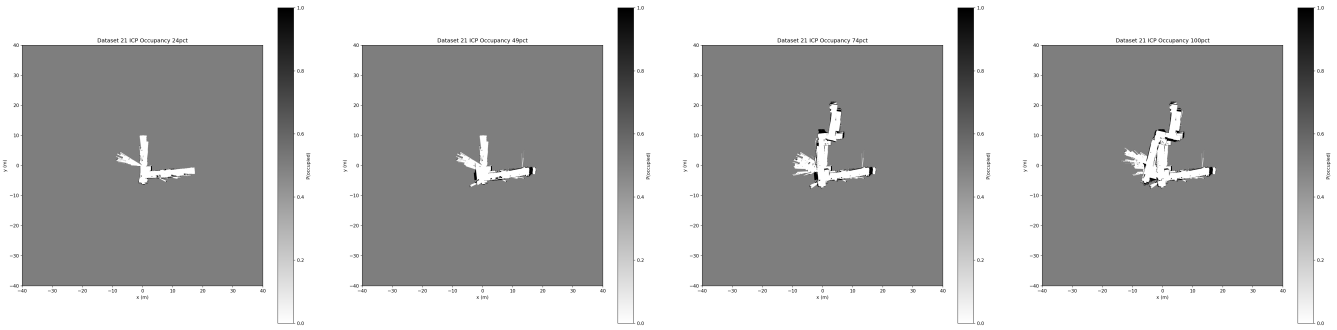
Run on a single dataset:

```
python3 slam.py --dataset 20
python3 slam.py --dataset 21
```

Force recompute (ignore cache):

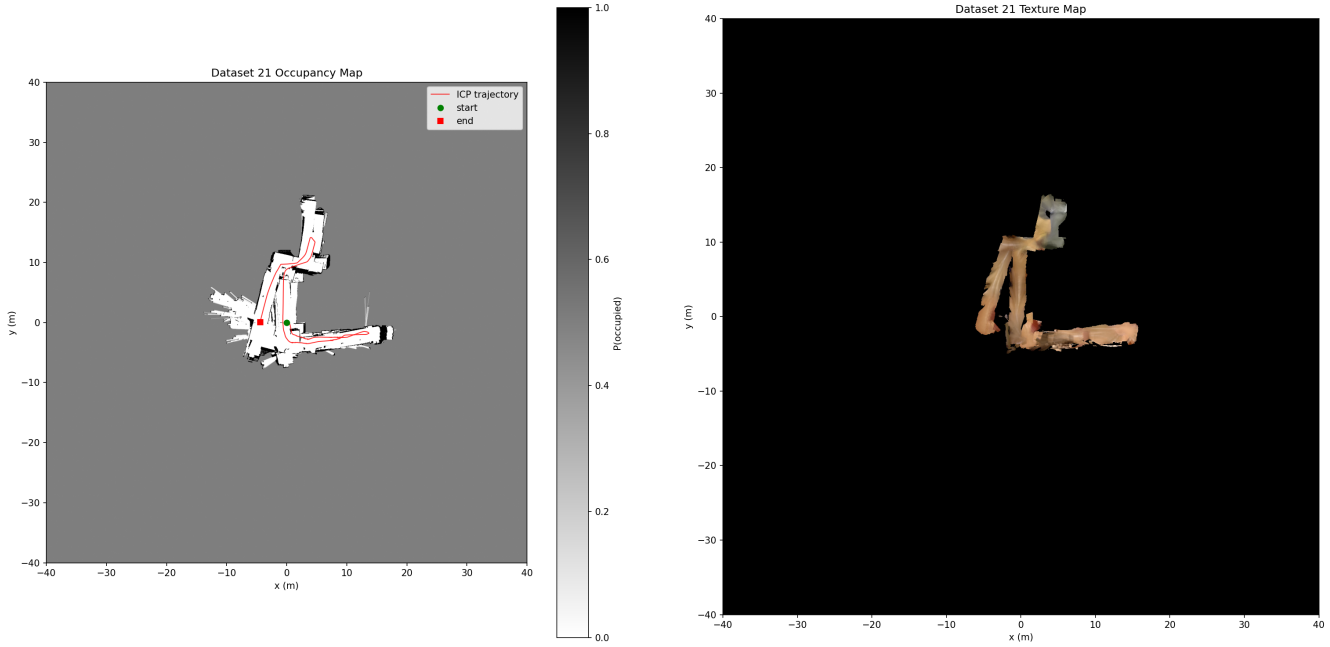
```
python3 slam.py --parts 2 --force
```

Key tunable parameters:



(a) 24% (b) 49% (c) 74% (d) 100%

Fig. 12: Dataset 21 ICP occupancy map progression at 24%, 49%, 74%, and 100% of poses processed. The east hallway fully forms by 49%, and the upper room appears at 74%. Doubled north corridor walls are visible throughout, indicating residual ICP heading drift between outgoing and return passes.



(a) Final ICP occupancy map with trajectory (b) ICP texture map

Fig. 13: Dataset 21 final ICP maps. (a) L-shaped floor plan with ICP trajectory overlay (red). (b) Texture map with tan/beige corridor floors, gray upper room, and two parallel corridor strips reflecting the heading drift artefact.

```
# ICP correspondence distance threshold (metres)
python3 slam.py --parts 2 --max-dist 0.2

# LiDAR max range cap for occupancy mapping (metres)
python3 slam.py --parts 3 --lidar-max-range 10

# GTSAM loop closure settings used for final results
python3 slam.py --parts 4 --dataset 20 \
    --loop-mse 0.05 --loop-radius 0.01 --min-pose-
    gap 2000
```

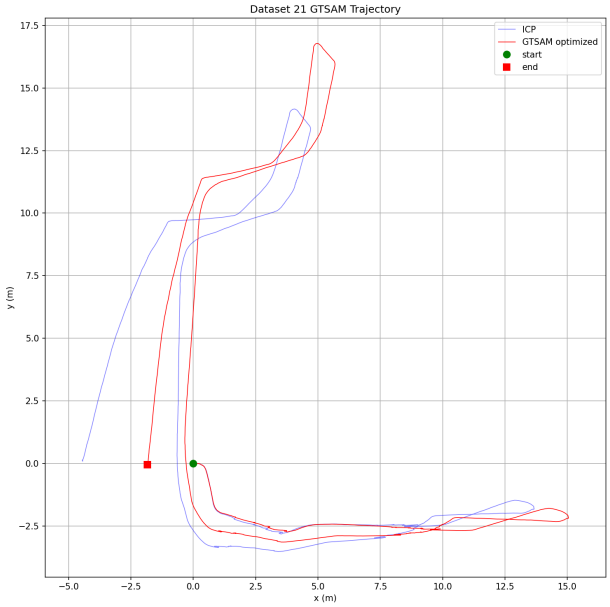
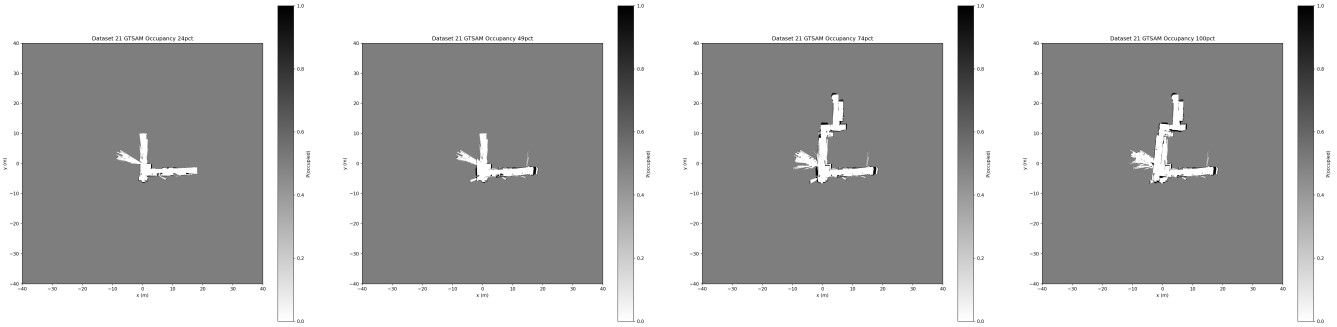
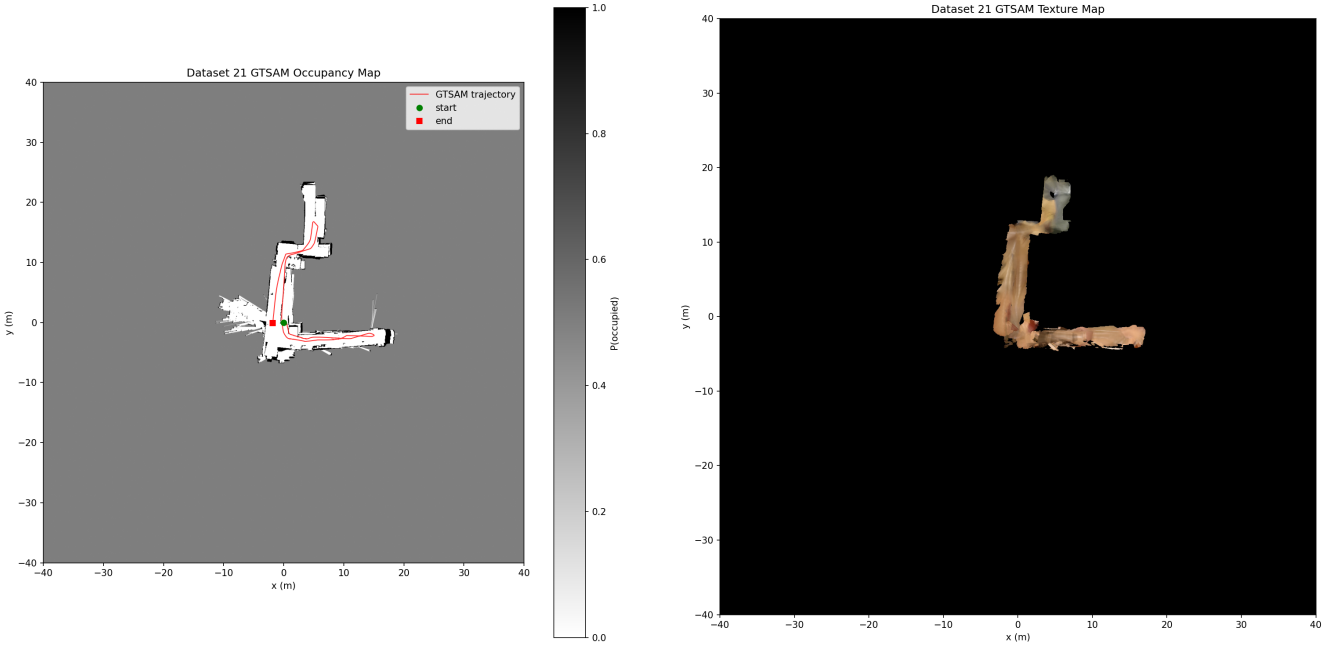


Fig. 14: Dataset 21 ICP (blue) vs. GTSAM-optimized (red) trajectory. With zero proximity-based loop closures, optimization is driven entirely by fixed-interval smoothing, which shifts the corridor rightward, elongates it, and moves the end point from $(-5, -1)$ m to $(-2, 0)$ m.



(a) 24% (b) 49% (c) 74% (d) 100%

Fig. 15: Dataset 21 GTSAM occupancy map progression at 24%, 49%, 74%, and 100%. Without a proximity loop closure, early snapshots are nearly identical to ICP. Corridor wall doubling is modestly reduced by 100% as fixed-interval smoothing redistributes heading corrections across all poses.



(a) GTSAM occupancy map with trajectory (b) GTSAM texture map

Fig. 16: Dataset 21 final GTSAM maps. (a) Occupancy map with optimized trajectory (red); end point is much closer to the origin than ICP. (b) Texture map showing a single, more coherent north corridor strip compared to the two-strip ICP texture, the most visible benefit of the smoothing correction.