ECE276A PR2 Report

1st Weixiao Zhan
weixiao-zhan[at]ucsd[dot]edu

I. Introduction

Simultaneous Localization and Mapping (SLAM) is pivotal in robotics, enabling autonomous systems to understand their surroundings and their locations with minimal human intervention. In this project, SLAM on a differential-drive robot equipped with a wheel encoder, an Inertial Measurement Unit (IMU), 2-D LiDAR, and an RGBD camera is implemented, which optimize the robot trajectory, and reconstructed detailed occupancy and floor texture map.

The process began with the construction of a motion model trajectory based on the differential-drive kinematics, utilizing data from the wheel encoder and IMU. Concurrently, an observation model trajectory was estimated using LiDAR scans and the Iterative Closest Points (ICP) method. Subsequently, I applied a Factor Graph and loop closure techniques to refine and optimize the trajectory. Lastly, I leveraged the trajectories to generate detailed maps, showcasing the potential.

II. Problem Formulation

The robot trajectory, discretized as a set of pose and time stamp tuples, is the core of computation. Denote the robot's pose at given time stamp t as P_t. The pose can be represented in both two-dimensional (2D) and three-dimensional (3D) spaces, with the z-axis value fixed at zero.

$$P_t = \begin{cases} \overset{2D}{=\!=} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} & \in \mathbb{R}^3 \\ \overset{3D}{=\!=} \begin{bmatrix} R_{yaw}(\theta) & \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix} & \in \mathbb{R}^{[4\times4]} \end{cases}$$

These representations imply the same intrinsics and will be used seamlessly in following 2D-3D-mixed analysis. The initial pose P_0 at time 0 is also defined as the world frame serving as the reference for all subsequent poses.

$$P_0 = \begin{cases} \vec{0}_3 \\ I_4 \end{cases}$$

A. Motion Model: IMU & Differential-drive Kinematics

Consider a small time interval τ between two consecutive time stamp $t-1$ and t. Suppose wheel encoder and IMU reports the robot has traveled δd (m) distance and and rotated $\delta\theta$ (rad) during this interval, the motion of robot can be approximate as an arc:

$${}_{t-1}O_t = \begin{bmatrix} \delta d \cdot \operatorname{sinc}\left(\frac{\delta\theta}{2}\right) \sin\left(\theta_t + \frac{\delta\theta}{2}\right) \\ \delta d \cdot \operatorname{sinc}\left(\frac{\delta\theta}{2}\right) \sin\left(\theta_t + \frac{\delta\theta}{2}\right) \\ \delta\theta \end{bmatrix}$$

$$P_t^{(2M)} = P_0^{(2M)} + \sum_{f=1}^{t} {}_{f-1}O_f$$

in which, the superscript $^{(2M)}$ means this is a 2D motion model trajectory,

B. Observation Model: LiDAR & Scan Matching

Scan matching can, given observations at different poses, estimate their elative transformations. Scan matching on LiDAR scan is used as observation model in this project.

2D-LiDAR scan reports a set of range and angle tuple $L_t = \{(r_i, \phi_i)\}$ at time stamp t, which can be converted to a set of 3D point cloud in body frame using following equations:

$$\mathrm{PC}_{\text{LiDAR frame}} = \left\{ \begin{bmatrix} r_i \sin\phi_i \\ r_i \cos\phi_i \\ 0 \end{bmatrix} \middle| \forall (r_i, \phi_i) \in L_t \right\}$$

$$\begin{bmatrix} \mathrm{PC}_t \\ 1 \end{bmatrix} =_{\text{body}} T_{\text{LiDAR}} \begin{bmatrix} \mathrm{PC}_{\text{LiDAR frame}} \\ 1 \end{bmatrix}$$

Since there is prior knowledge of data association between two LiDAR scans, ICP algorithm is used. Given two homogenous point clouds, source $S = \left\{ \begin{bmatrix} PC_s & 1 \end{bmatrix}^T \right\}$ at timestamp s, and target $T = \left\{ \begin{bmatrix} PC_t & 1 \end{bmatrix}^T \right\}$ at timestamp t, and an initial guess transformation T_0, ICP iterates following steps:

1) Apply initial guess: $S \leftarrow T_0 \cdot S$ and set $k \leftarrow 1$.
2) Find point association using nearest neighbor:

$$\Delta = \left\{ \left(s, \arg\min_{t \in T} \|s - t\| \right) \middle| \forall s \in S \right\}$$

3) Estimate relative transformation T_k using Kabsch algorithm:

$$T_k = \arg\min_{T \in SE(3)} \sum_{(s,t)\in\Delta} \|t - Ts\|$$

.

4) Update $S \leftarrow T_k \cdot S$, increase k by 1, and repeat from step 2 until no improvement.

5) Compose and return the total transformation and error

$$
\begin{aligned}
{}_tT_s &= \prod_k T_k \\
{}_t\epsilon_s &= \frac{1}{|\Delta|} \sum_{(s,t)\in\Delta} \|t - {}_tT_s \cdot s\|
\end{aligned}
$$

Using ICP to estimate relative transformation between consecutive LiDAR scans, observation model trajectory is composed as following:

$$
\begin{aligned}
{}_{t-1}T_t &= \text{ICP}(S = \text{PC}_t, T = \text{PC}_{t-1}) \\
P_t^{(3O)} &= P_0 \prod_{f=1}^{t} {}_{f-1}T_f = {}_0T_t
\end{aligned}
$$

in which, the superscript $^{(3O)}$ means this is a 3D observation trajectory.

C. Factor Graph and Loop Closure

Factor graph is a constrained optimization model for SLAM problem. Graph nodes represent robot states as random variables, while edges represent motion and observation constraints as conditional distributions. The optimizer of a factor graph is the random variables that maximize likelihood of all factors.

Loop Closure is a special type of constraint in factor graph. Typical constraints based on Markov assumption builds a single chain graph, in which error and noise accumulate. However, when some temporally distant robot states are spatially close, it is possible to estimate their relative transformations, thereby create loops in factor graph and reduce accumulated error.

A loop detection criteria based on max location difference d^*, max yaw difference θ^*, and min interval τ^* is used to find potential loop closure pose pairs lc:

$$
lc = \left\{ (i,j) \middle| \begin{array}{c} \|P_i[xy] - P_j[xy]\| < d^* \\ \|P_i[\theta] - P_j[\theta]\| < \theta^*, \forall i,j \\ |i-j| > \tau^* \end{array} \right\}
$$

The factor graph is construct as:

$$
\begin{aligned}
G &= (V, E) \\
V &= \{P_t,\ \forall t\} \\
E &= \left\{ \begin{array}{c}
\underbrace{P_t^{(2)} \ominus \left(P_{t-1}^{(2)} + {}_{t-1}O_t\right), \forall t}_{\text{motion constraints}} \\
\underbrace{P_t^{(3)} \ominus \left(P_{t-1}^{(3)} \cdot {}_{t-1}T_t\right), \forall t}_{\text{observation constraints}} \\
\underbrace{P_i^{(3)} \ominus \left(P_j^{(3)} \cdot \text{ICP}(S = \text{PC}_i, T = \text{PC}_j)\right) \\ \forall (i,j) \in lc}_{\text{loop closure constraints}}
\end{array} \right\}
\end{aligned}
$$

The three types of constraints are:

1) Motion constraints applied to two consecutive poses, derived differential drive kinematics.
2) Observation constraints on two consecutive poses, utilizing scan matching (ICP).
3) Loop closure constraints on poses that are spatially proximate yet temporally distant.

D. Mapping

Mapping is another critical component of SLAM, which reconstructs the surrounding environment using various sensors. 2D LiDAR was used to construct an occupancy map, while an RGB-D camera, which provides both RGB and disparity information, was employed for texture mapping of the environment.

1) Occupancy: LiDAR scan (r_i, ϕ_i) infer that the endpoints ($\{(r_i \cos\phi_i, r_i \sin\phi_i), \forall i\}$ in LiDAR frame) are occupied, while the space between the end-points and LiDAR sensor origin ($(0,0)$ in LiDAR frame) and is empty.

By transforming the endpoints and LiDAR sensor to world frame, discretizing the world floor into a grid of cells, using Bresenham ray tracing to obtain empty and occupied cell set, the probability of occupancy of each cells can be estimate by logistic regression over all scans.

2) RGB Texture: Consider values d at pixel (u, v) of the disparity image and its associated rgb value. Given the intrinsic matrix K of the D camera, the equations to convert disparity value d to the associated depth z, and the projection equation:

$$
\begin{aligned}
z &= g(d) \\
\begin{bmatrix} u \\ v \end{bmatrix} &= K\pi \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{opt} \right)
\end{aligned}
$$

One can solve the (u, v, d) associated 3d coordinate in camera frame:

$$
\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{camera} = {}_oR_c^{-1} \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{opt} = {}_oR_c^{-1} \cdot g(d) \cdot K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}
$$

The point cloud with RGB color is transformed to body frame using ${}_{\text{body}}T_{\text{camera}}$, then to world frame using robot poses P_t. By isolating floor points, whose $|z| < z^*$, and mapping rgb values to the cell corresponding to xy coordinate, a texture floor map can be generated.

III. Technical Approach

A. Data interpolation on timestamps

Robot sensors operate at varying frequencies and no synchronization is enforced. Thus, interpolation is used to infer data values at desired timestamps, allowing integrating data from multiple sensors. Specifically, following strategies are employed:

- Wheel encoder data is linearly interpolated at the IMU timestamps in motion models.
- Poses are linearly interpolated for various applications:
  - Motion poses are linearly interpolated at LiDAR timestamps when providing initial guesses for ICP.

  - Poses are linearly interpolated at LiDAR timestamps in occupancy mapping.
  - Poses are linearly interpolated at camera timestamps in texture mapping.
- RGB image is interpolated to nearest-in-time D image in texture mapping.

B. Body frame

The origin of the body frame is defined as the geometric center of four wheels, instead of the center of the rear axle suggested by robot configuration. This positioning aligns body frame origin more closely to IMU sensor, making IMU yaw rate a more accurately approximation of true yaw rate.

In this body frame definition, distance traveled at time t, denoted as d_t, and the distance difference δd used in motion model, are computed as:

$$\begin{aligned} d_{t-\text{wheel counter}} &= 0.0022 \times \sum_{i=0}^{t} \frac{FR_i + FL_i + RR_i + RL_i}{4}, \\ d_t &= \text{interp}_{t-\text{IMU}}(d_{t-\text{wheel counter}}) \\ \delta d &= d_t - d_{t-1}, \end{aligned}$$

in which FR, FL, RR, and RL represent the wheel count readings from the front right, front left, rear right, and rear left wheels, respectively.

C. Sensor frame to Body frame

Based on robot configuration and body frame definition, following transformations are used to convert sensor frame to body frame:

$${}_{\text{body}}T_{\text{LiDAR}} = \begin{bmatrix} I_3 & \begin{bmatrix} 0.13323 \\ 0 \\ 0.51435 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix}$$

$${}_{\text{body}}T_{\text{camera}} = \begin{bmatrix} R_{yaw}\left(0.021\right) R_{pitch}\left(0.36\right) R_{roll}\left(0\right) & \begin{bmatrix} 0.18 \\ 0.005 \\ 0.36 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix}$$

D. 2D-3D pose conversion

The 2D and 3D representation of poses often needs to be converted to each other. Following method is used to perform the conversion and handle ambiguity of additional degree of freedom in 3D representation:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \rightarrow \begin{bmatrix} R_{yaw}(\theta) & \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & 0 & x \\ c & d & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ \dfrac{\arctan(c,a) + \arctan(-b,d)}{2} \end{bmatrix}$$

E. Revise ICP in Scan Matching

In traditional ICP, the algorithm try to find data association for all source points. But in robot LiDAR applications, there may exists points in source set S, which don't have truly associated points in target set T. The false nearest neighbor association on these points would caused unreliable ICP results.

Thus I propose proportion ICP: given extra proportion parameter $\gamma \in (0, 1]$, only consider data association, whose nearest neighbor distance is in the smaller γ proportion of all associations.

Formally, denote the nearest neighbor of s as $nn(s, T) = \arg\min_{t \in T} \|s - t\|$ and their distance $nd(s, T) = \min_{t \in T} \|s - t\|$. The threshold distance corresponding to the proportion γ is d_γ s.t. $|\{s | nd(s, T) < d_\gamma, \forall s \in S\}| = \lfloor \gamma |S| \rfloor$

The proportion ICP use following revised data association and optimization objective:

$$\begin{aligned} \Delta_\gamma &= \{(s, nn(s, T)) \,|nd(s, T) \leq d_{thr}, \forall s \in S\} \\ {}_tT_s &= \arg\min_{T \in SE(3)} \sum_{(s,t) \in \Delta_\gamma} \|t - Ts\| \\ {}_t\epsilon_s &= \frac{1}{|\Delta_\gamma|} \sum_{(s,t) \in \Delta_\gamma} \|t - {}_tT_s \cdot s\| \end{aligned}$$

In this project practice, the proportion parameter γ is dynamically inferred from initial guess transformation. Suppose the initial transformation suggests a $\delta\theta$ difference in yaw angle, the proportion parameter is:

$$\gamma = 1 - \frac{0.5}{\pi}|\delta\theta| - 0.1$$

The equation first computation the overlap proportion of two scans, then subtract 10% for some safe margin and potential outliers.

F. Factor Graph

In this project practice, Two factor graphs are built, one without loop closure and one with loop closure. The potential loop closure criteria are $d^* = 0.1, \theta^* = \frac{3}{4}\pi, \tau^* = 30$ Motion constraints are set to have static diagonal variance $[0.0025, 0.0025, 0.0025]$ Observation and loop closure constraints are set to have diagonal variance $[{}_t\epsilon_s, {}_t\epsilon_s, {}_t\epsilon_s]$, which is proportional to the ICP error.

G. mapping

1) discretized map: The world is discretized grid map with resolution 10 cells per meter. All points are rounded to the cell center.

2) LiDAR to occupancy: For every scan, the empty cells decreased odds ratio by 2, where as end point cells increase odds ratio by 2. The occupancy probability is calculated by:

$$p\left(\text{occupancy}\right) = \frac{1}{1 + \exp\left(-\text{odds ratio}\right)}$$

3) RGBD to texture: Based on robot configuration, following intrinsic matrix and conversion function g is used:

$$K = \begin{bmatrix} 585.05 & 0 & 241.94 \\ 0 & 585.05 & 315.84 \\ 0 & 0 & 1 \end{bmatrix}$$
$$dd = -0.00304d + 3.31$$
$$z = g(d) = \frac{1.03}{dd}$$
$${}_cR_o^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$
$$z^* = 0.5$$

Meanwhile, the D camera and RGB camera are not in the same location. There is an x-axis offset between them. Following transformation is used to match disparity pixel coordinate (u, v) with value d to its RGB color pixel location $(rgbu, rgbv)$:

$$rgbu = (526.37u + 19276 - 7877.07dd)\,/585.051$$
$$rgbv = (526.37v + 16662)\,/585.051$$

In practice, points with $dd > 1088$ are discarded due to their close distant causing less reliable estimate. RGB image and D image are sampled with step 2 on both axis to reduce computation.

IV. Results

Figure 1 and Figure 2 show the results from dataset 20 and 21 respectively. Sub-figure 1a to 1d and 2a to 2d show the trajectory from motion model, observation model, factor graph without loop closure constraints, and factor graph with loop closure constraints. Sub-figure 1e to 1h and 2e to 2h show the occupancy map based on different trajectories. Sub-figure 1i to 1l and 2i to 2l show the floor texture map based on different trajectories.

Result discussion:

- On dataset 20, the motion model has produced solid result as a starting point (Fig. 1e). Observation model, factor graph, and loop closure optimize on the basis of the motion model and improve on details: boundaries are clearer in occupancy map 1g, the walls are straight and parallel in floor map 1l, etc.
- On dataset 21, the motion model produced some errors in the vertical hallway area. The observation model, however, stuck to some local minimal, and even made the trajectory worse. The loop closure factor graph was able to correct some errors in x coordinate, but messed up the orientation, causing less reliable mapping results.
- ICP algorithm is prone to trap in local minimals. Thus a good initial-guess translation is critical. If the robot is equipped with 3D-LiDAR or using the 3d point cloud from RGBD camera, the ICP estimation could be improved by extra dimension and points.

Big images are at `img/`.

Appendix

A. ICP warm up

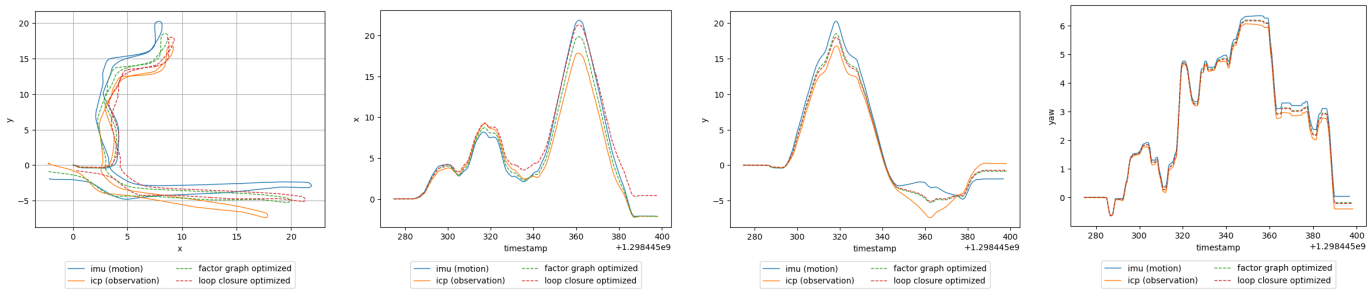
Figure 3 shows ICP warm up results. Because, the source point cloud is just one side of the object, source point cloud tends to be squashed into the target point cloud. In sub figure 3d, The source and target is aligned inverted on 1 axial but correctly on the other 2 axis. This is an example of ICP stuck on local minimal and could not improve over iteration.

B. proportion ICP improvement

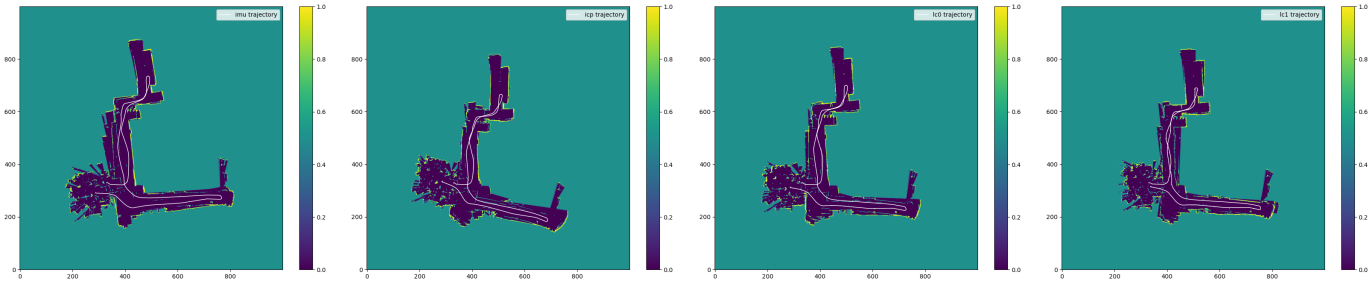
Figure 4 shows the improvement of revised proportion ICP on observation model trajectory.

C. 3D texture mapping

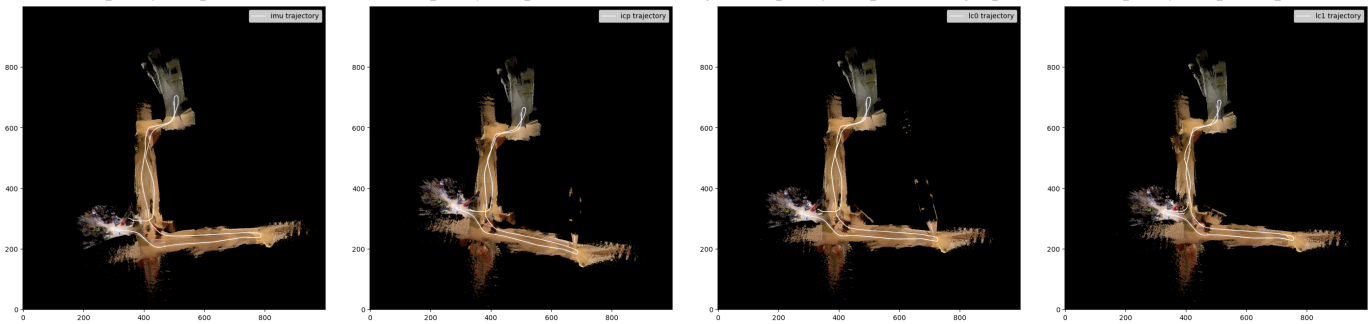
Figure 5 and video at https://youtu.be/yPtubAjLYnc shows the result of a 3D texture mapping.



(a) trajectory (b) trajectory x-t (c) trajectory y-t (d) trajectory yaw-t

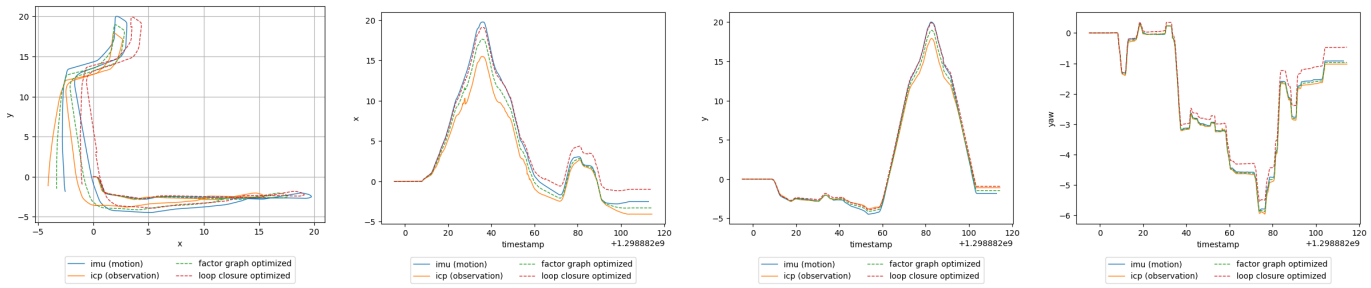


(e) occupancy map (motion) (f) occupancy map (observation) (g) occupancy map (factor graph) (h) occupancy map (loop closure)

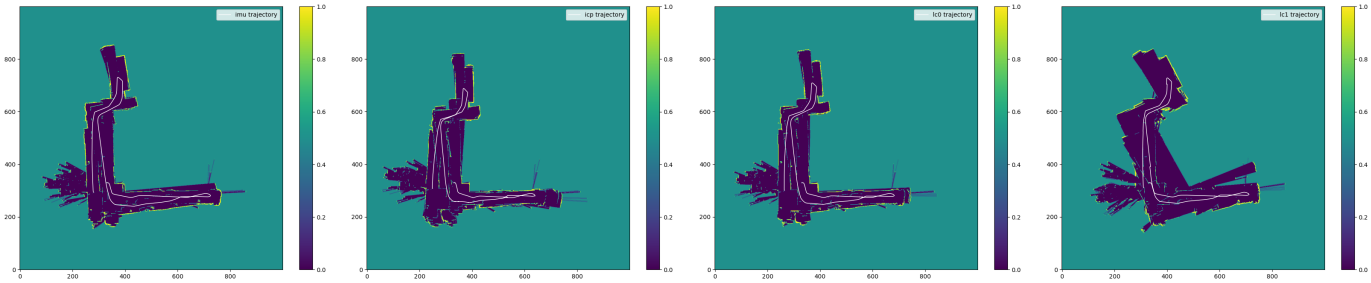


(i) floor map (motion) (j) floor map (observation) (k) floor map (factor graph) (l) floor map (loop closure)

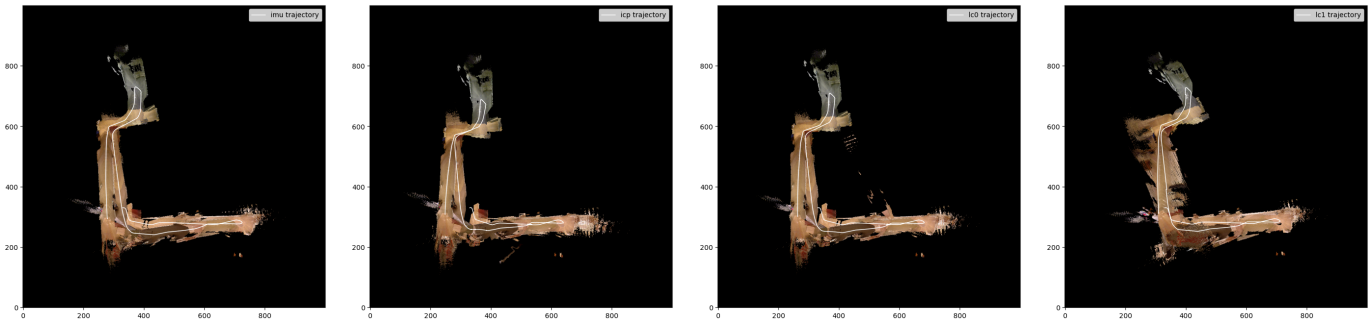
Fig. 1: dataset 20 results



(a) trajectory (b) trajectory x-t (c) trajectory y-t (d) trajectory yaw-t

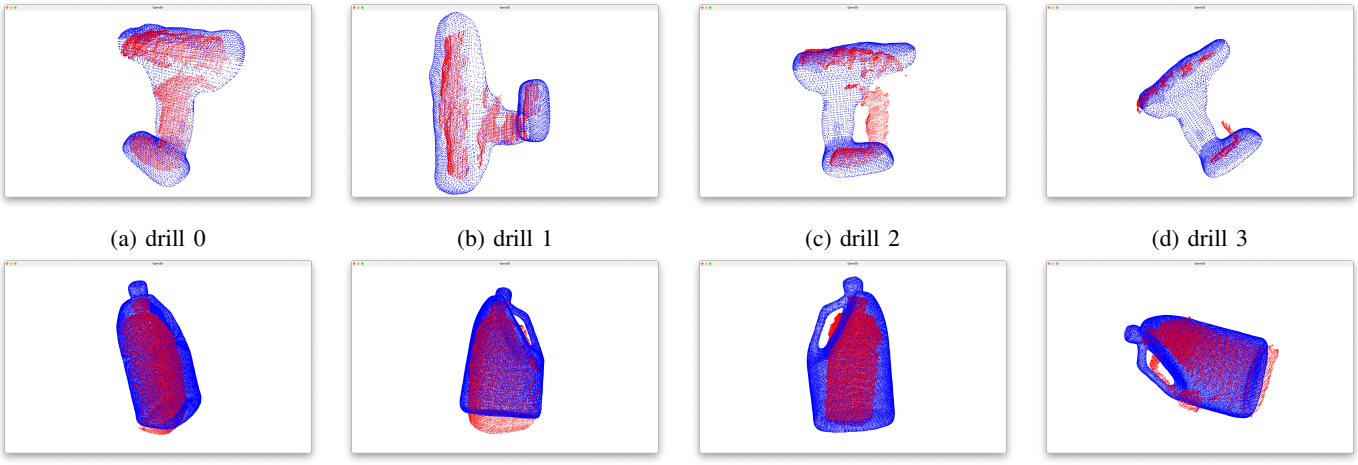


(e) occupancy map (motion) (f) occupancy map (observation) (g) occupancy map (factor graph) (h) occupancy map (loop closure)



(i) floor map (motion) (j) floor map (observation) (k) floor map (factor graph) (l) floor map (loop closure)

Fig. 2: dataset 21 results



(a) drill 0 (b) drill 1 (c) drill 2 (d) drill 3

(e) liq 0 (f) liq 1 (g) liq 2 (h) liq 3

Fig. 3: icp warm up results

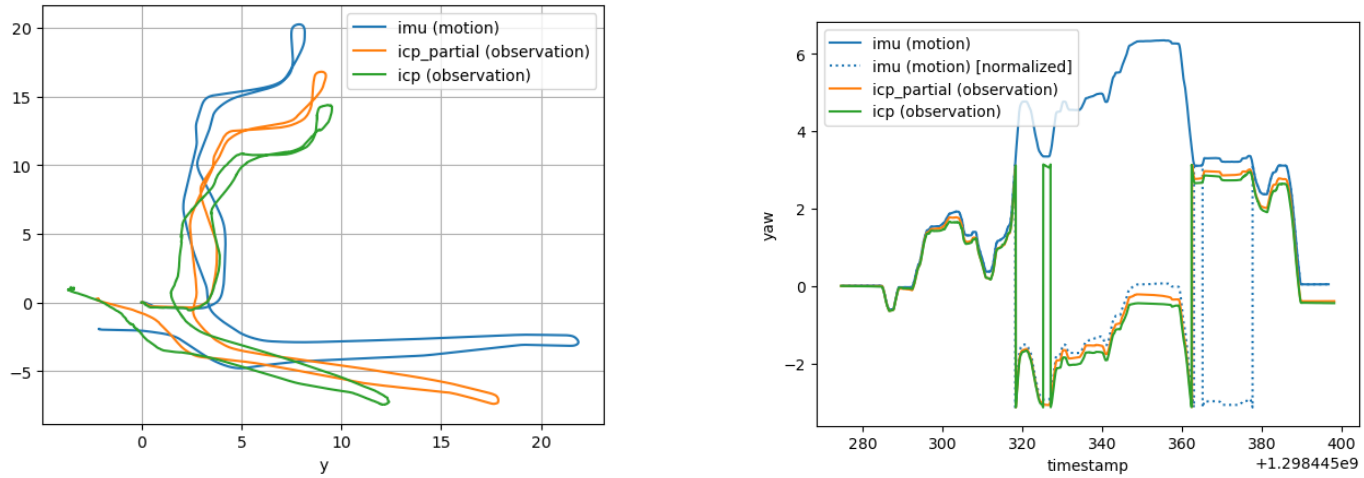


Fig. 4: ICP vs Propotion ICP

Fig. 5: 3D texture mapping