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 all 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.

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

This dual representation enables seamless transition between 2D and 3D spatial analyses. The initial pose P_0 at time 0 is defined as the origin of 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 and $t+1$. Suppose wheel encoder reported δd and IMU's yaw reading reproted $\delta\theta$ during this interval, the motion of robot can be approximate as an arc:

$$P_{t+1}^{(2M)} = P_t^{(2M)} + \begin{bmatrix} \delta d \cdot \operatorname{sinc}\left(\frac{\delta\theta}{2}\right) \sin\left(\theta_t + \frac{\delta\theta}{2}\right) \\ \delta d \cdot \operatorname{sinc}\left(\frac{\delta\theta}{2}\right) \sin\left(\theta_t + \frac{\delta\theta}{2}\right) \\ \delta\theta \end{bmatrix}$$

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

B. Observation Model: LiDAR & Scan Matching

Scan matching employed ICP algorithm, which can estimate the relative transformation between two point clouds, source S and target T, without prior knowledge of data association. It achieved this by iterates over these steps:

1) Find point association for all source points using closest distance:
$$\Delta = \left\{ \left(s, \arg\min_{t\in T} \|s-t\| \right) \middle| \forall s \in S \right\}$$
2) Estimate relative transformation ${}_tT_s$ using Kabsch algorithm:
$$T_k = \arg\min_T \sum_{(s,t)\in\Delta} \|t - Ts\|$$
assuming s and t are in homogenous coordinate.
3) Update $S \leftarrow T_k S$, increase k by 1, and repeat step 1 until no improvement.

Lastly, compose and report the total transformation ${}_tT_s = \prod_k T_k$ and error ${}_t\epsilon_s = \sum_{(s,t)\in\Delta} \|t - {}_tT_s \cdot s\| / |\Delta|$.

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

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

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

Leveraging ICP, I can estimate relative transformation between two LiDAR scan and compose observation model trajectory:

$${}_tT_{t+1}^{(O)} = \text{ICP}(S = \mathrm{PC}_{t+1}, T = \mathrm{PC}_t)$$

$$P_t^{(3O)} = P_0 \prod_{f=1}^{t} {}_{f-1}T_f^{(O)}$$

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

C. Factor Graph and Loop Closure

ToDo add slides reports A Factor Graph is a constrained optimization problem modeled over a directed graph. Nodes in the graph represent random variables, while edges signify dependencies between them, i.e., conditional distributions. Factor graph assumes random variables with not directly edges are conditionally independent. The optimizer of a factor graph is the maximize likelihood.

In this project, robot states are treated as graph nodes, with observations and the motion model represented as edges. The graph is defined as follows:

$$
\begin{aligned}
G &= (V, E) \\
V &= \{P_t,\ \forall t\} \\
E &= \left\{ \begin{array}{c}
\underbrace{(P_{t-1}^{(M)} \rightarrow P_t^{(M)}),\ \forall t}_{\text{motion constraints}} \\
\underbrace{(P_{t-1}^{(O)} \rightarrow P_t^{(O)}),\ \forall t}_{\text{observation constraints}} \\
\underbrace{(P_i \rightarrow P_j)}_{\text{loop closure constraints}}
\end{array} \right\}
\end{aligned}
$$

There are three types of constraints:

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.

The potential for loop closure arises when some poses are spatially close to permit the use of ICP for estimating relative transformations between temporally distant poses, thereby correct accumulated errors.

Let $lc(P_i)$ represent the set of potential loop closure pose of P_i. Following loop detection criteria based on super-parameter max location difference d^*, max yaw difference θ^*, and min interval τ^* is used to find potential poses:

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

The actual loop closure pairs L_n is sampled from all potential loop closure pairs:

$$
L_n \subset \{(P_i, P_j), P_j \in lc(P_i), \forall P_i\}
$$

in which, $n = |L_n|$ denote the sample size.

Finally, for all $(P_i, P_j) \in L_n$, proportion ICP is used to estimate relative transformations ${}_jT_i$ and error ${}_j\epsilon_i$. New edges $P_i \rightarrow P_j = {}_jT_i$ with variance matrix $diag({}_j\epsilon_i, {}_j\epsilon_i, {}_j\epsilon_i)$ is added to factor graph.

D. Mapping

Mapping is a critical component of SLAM, focused on reconstructing the surrounding environment using various sensors.

In this project, 2D LiDAR was used to construct an occupancy map, while an RGB-D camera, which provides both RGB and depth information, was employed for texture mapping of the environment.

1) Occupancy: LiDAR scan (r_i, ϕ_i) infer that the end-points ($\{(r_i \cos\phi_i, r_i \sin\phi_i), \forall i\}$ in LiDAR frame) are occupied, while the space between the LiDAR sensor origin ($(0, 0)$ in LiDAR frame) and end-points is empty. By transforming the endpoints and LiDAR sensor to world frame using robot poses, and discretized the world into a grid of cells, I can employ the Bresenham ray tracing algorithm to obtain empty cell sets of each scan, and use logistic regression over all scans to estimate the probability of occupancy of each cells.

2) Texture: Utilizing the supplementary depth information, an RGB-D image is transformed into a 3D point cloud within the camera frame, which is subsequently converted to the world frame leveraging the robot poses. This process creates a 3D texture map of the environment. By isolating and presenting points where $Z = 0$, a texture map specifically representing the floor is generated.

III. Technical Approach

A. Time Stamp Synchronization

Given that sensors operate at varying frequencies, and even with identical frequencies, data synchronization is not inherently precise. To address this challenge, data interpolation is utilized to synchronize datasets across different sensor timestamps. Specifically, the following strategies are employed:

- Wheel encoder data is linearly interpolated to match the IMU timestamps within motion models.
- Poses are linearly interpolated for various applications:
  - Motion poses are interpolated to LiDAR timestamps when providing initial guesses for ICP.
  - Poses are interpolated to LiDAR timestamps during occupancy mapping.
  - Poses are interpolated to camera timestamps during texture mapping.
- RGB image and D image are interpolated to nearest.

B. Body frame

The origin of the body frame is defined as the geometric center of the four wheels, instead of the center of the rear axle. This positioning aligns more closely with the IMU sensor, enabling the yaw rate measured by the IMU to be more accurately approximated as the true yaw rate.

The distance at time t, denoted as d_t, and the delta distance, δd used in motion model, are given by the following equations:

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

where 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, following transformations are used to convert from sensor frame to body frame:

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

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

D. 2D-3D pose conversion

The process of converting a 2D pose to a 3D pose is deterministic. However, converting a 3D pose to a 2D pose can be ambiguous due to the additional degree of freedom involved and floating point error. The conversion is detailed below:

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

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

E. Scan Matching

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

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

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_{thr} s.t. $|\{s|d(s, T) < d_{thr}, \forall s \in S\}| = \lfloor\gamma|S|\rfloor$

Thus the proportion ICP has following data association and optimization objective:

$$\Delta_\gamma = \{(s, nn(s, T))\,|nd(s, T) \le d_{thr}, \forall s \in S\}$$

$$_tT_s = \arg\min_T \sum_{(s,t)\in\Delta_\gamma} \|t - Ts\|$$

In ICP applications of this project, suppose the initial guess suggests a $\delta\theta$ difference in yaw angle, the proportion parameter is set to:

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

this equations first computation the expected overlap proportion of two scans, then subtracted 10% for potential outliers.

Lastly, my ICP implementation also reports the average error:

$$\frac{\sum_{(s,t)\in\Delta_\gamma} \|t - {}_tT_s \cdot s\|}{|\Delta_\gamma|}$$

which is later used in factor graph to estimate edge noise.

F. Factor Graph

ToDo variance of motion model edge variance of observation model

G. Loop closure

In practice, $d^* = 0.5$, $\theta^* = \frac{3}{4}\pi$

L_{615} and L_{1838} is sampled on dataset 20; L_{651} and L_{1959} is sampled on dataset 21.

H. 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: Consider values d at pixel (u, v) of the disparity image. 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:

$$\begin{cases} K = \begin{bmatrix}585.05 & 0 & 241.94\\0 & 585.05 & 315.84\\0 & 0 & 1\end{bmatrix} \\ dd = -0.00304d + 3.31 \\ z = \dfrac{1.03}{dd} \\ \begin{bmatrix}u\\v\end{bmatrix} = K\pi\left(\begin{bmatrix}x\\y\\z\end{bmatrix}_{opt}\right) \end{cases}$$

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

$$\begin{bmatrix}x\\y\\z\end{bmatrix}_{opt} = z \cdot K^{-1}\begin{bmatrix}u\\v\\1\end{bmatrix}$$

$$\begin{bmatrix}x\\y\\z\end{bmatrix}_{camera} = \begin{bmatrix}0 & 0 & 1\\-1 & 0 & 0\\0 & -1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}_{opt}$$

Meanwhile, the D camera and RGB camera are not in the same location. There is an x-axis offset between them and it is necessary to use a transformation to match the color to the depth. Given values d at pixel (u, v) of the disparity

image, the associated RGB color pixel location $(rgbu, rgbv)$ are calculated as follows:

$$\begin{aligned} dd &= -0.00304d + 3.31 \\ rgbu &= (526.37u + 19276 - 7877.07dd)/585.051 \\ rgbv &= (526.37v + 16662)/585.051 \end{aligned}$$

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 filtering the points whose $|z_{world}| < 0.2$ and assigning RGB values to discretized map, the texture floor map is obtained.

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

IV. Results

[15 pts] Results: Present your results, and discuss them – what worked, what did not, and why. Analyze the impact of loop closure detection and pose graph optimization on the accuracy of the robot trajectory estimate and the resulting map quality. Make sure your results include (a) images of the trajectory and occupancy grid map over time constructed by your SLAM algorithm and (b) textured maps over time. If you have videos, include them in your submission zip file and refer to them in your report.

A. *trajectory*

B. *occupancy map*

C. *floor map*