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 PC_s and target PC_t, and an initial guess transformation T_0, ICP iterates following steps:

1) Apply initial guess: $PC_s \leftarrow T_0 \cdot PC_s$ and set $k \leftarrow 1$.
2) Find point association using nearest neighbor:
$$\Delta = \left\{ \left(s, \arg\min_{t \in PC_T} \|s - t\| \right) \middle| \forall s \in PC_s \right\}$$
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 $PC_s \leftarrow T_k PC_s$, increase k by 1, and repeat from step 2 until no improvement.
5) Compose and report the total transformation ${}_tT_s = \prod_k T_k$ and error ${}_t\epsilon_s = \frac{1}{|\Delta|} \sum_{(s,t)\in\Delta} \|t - {}_tT_s \cdot s\|$.

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:

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

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 end-points ($\{(r_i \cos\phi_i, r_i \sin\phi_i), \forall i\}$ 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) Texture: By transforming disparity information to depth information, converting RGB-D images to 3D colored point cloud in world frame, isolating points whose $|z| < z^*$, a texture floor map can be 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:

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

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

D. 2D-3D pose conversion

The 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:

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

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

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

Finally, for all $(i,j) \in lc$, 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.

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. multi Factor Graph

I. 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 following:

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

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

Appendix

A. ICP warm up

B. 3D texture mapping

https://youtu.be/yPtubAjLYnc