ELEC 3210 Introduction to Mobile Robotics
Hong Kong University of Science and Technology
Fall 2023, Huan Yin

Project 1: ICP Odometry

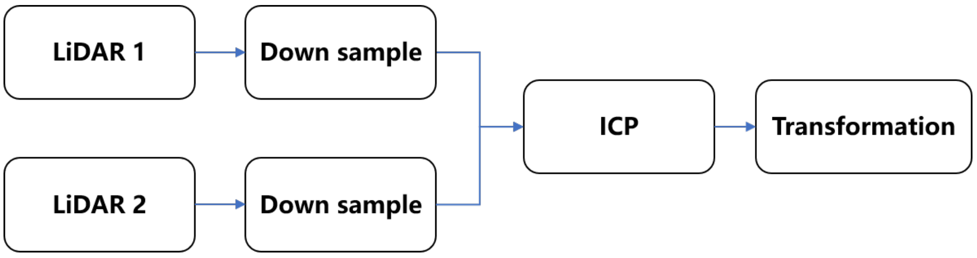
TA: Zhijian Qiao & Qiucan Huang

1 Project Work

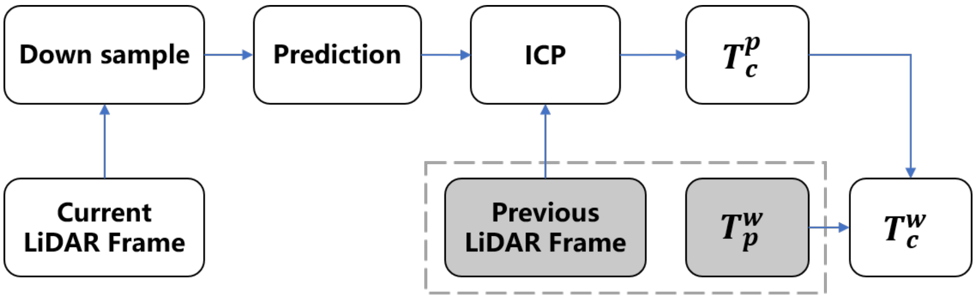
This assignment requires you to implement a LiDAR odometry based on the Iterative Closest Point (ICP) method. ICP odometry accepts a LiDAR sequence as input and outputs the pose of each LiDAR frame in the world coordinate system. Conventionally, the first frame is assigned as the world frame, characterized by an identity matrix rotation and a zero vector translation.

We offer several odometry pipelines, structured in ascending order of complexity but potentially offering greater accuracy.

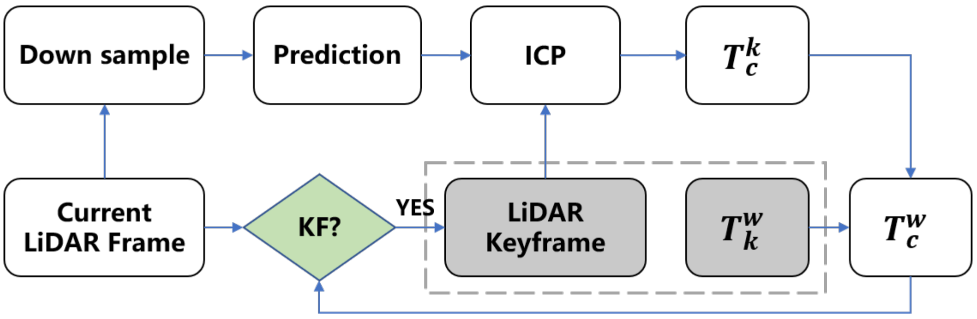
1. *Toy example.* In "examples/toy_example.cpp", you are encouraged to implement the ICP algorithm yourself. This toy example tests your understanding of class content. The true transformation is provided in "true_tf.txt".



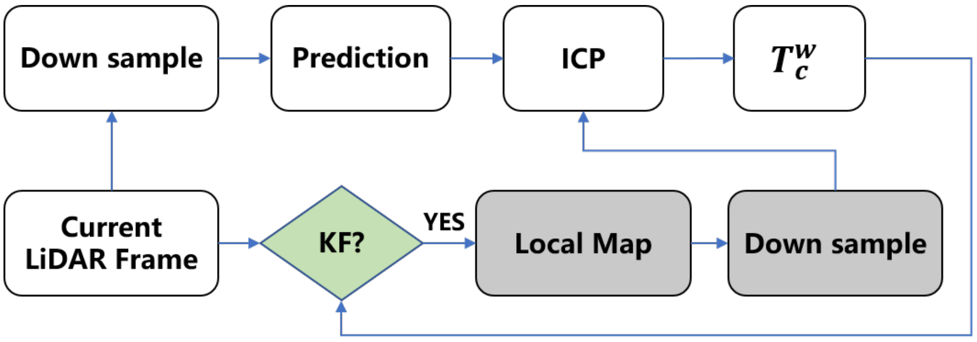
2. *Scan-to-scan odometry.* As demonstrated below, you can get the relative transformation T_c^p from the current LiDAR frame to the previous one. By accumulating these relative transformations with $T_p^w * T_c^p$, you can derive the pose of the current frame T_c^w. We recognize that ICP performance depends on finding the true corresponding point. For this reason, we use the iterative nearest point method, anticipating each iteration will improve the transformation and bring the nearest point closer to the true corresponding point. Can we provide a good transformation prediction in advance, such as using a constant motion model to predict T_c^p?



3. *Scan-to-keyframe odometry.* The third pipeline differs from the second in its use of a keyframe (KF). If we assume a constant transformation error from ICP, accumulating T_c^p will also accumulate this error with each frame, causing drift. However, if you opt to register the keyframe instead of the previous frame, the error will only accumulate when the keyframe is updated, thus limiting the drift. Furthermore, you can strategize when to update the keyframe if it becomes too outdated to provide an ICP reference. You might consider the timestamp, $(\mathrm{R}_c^k, \mathrm{t}_c^k)$, or the overlap of two point clouds.



4. *Scan-to-map odometry.* In this pipeline, a local map replaces the keyframe as the reference. The advantage is clear: the local map provides a more complete structure, enabling ICP to find a more accurate nearest point. However, the local map is larger, so you may need to use a passthrough filter or voxel down-sampling to limit the map size to an acceptable range.



2 Provided Materials

- Link
- Run-able but incomplete code.
- Groun truth of pipeline 1 and 2,3,4: see "true_tf.txt" and "true_trajectory.txt" respectively.
- Rosbag which records all measurements you need: see "turtlebot.bag".
- Config file "odometry.yaml", and its loader function "readParameters".
- Template evaluation code in "traj_eval.cpp".

3 Running Example

```
cd catkin_elec3210/src/
catkin_init_workspace
cd .. && catkin_make
source devel/setup.bash
# toy example
./devel/lib/icp_odometry/toy_example
# icp odometry. Remember to press space to start.
roslaunch icp_odometry icp.launch
```

4 Scoring Rules

Notes:

1. Please title your submission "PROJECT1-YOUR-NAMES". Your submission should include a maximum 2-page report and all necessary files to run your code, and a video in 1 minute, excluding the bag.
2. The report should contain at least a visualization of your final point cloud map, trajectory, and contribution of each group member. The code should save your best trajectory to a txt file, so we can compare it with your provided result in the report.
3. If your code is similar to others' or any open-source code on the internet, the score will be 0 unless the sources are properly referenced in your report. Refer to [1] for details.
4. Excessive white space, overly large images, or redundant descriptions in report will not contribute to a higher score. Simple but efficient code is preferred.
5. If any problem, please google first. If unsolved, create a discussion on Canvas.
6. If you have no experience of C++ coding, start this project as early as possible. Programming proficiency is not determined by the courses you've taken, but rather by ample practice and persistent debugging.
7. Submit your code and documents on Canvas before the deadline. Scores will be deducted for late submission.

Higher scores will be awarded if:

1. You implement one or more pipelines, with more complex pipelines being encouraged.
2. A concise and clear method description, along with ample ablation studies using the method of control variables. Potential control variates could include algorithm parameters, the KF update strategy and so on.
3. Coding: object-oriented programming logic, initializing each variable in the constructor, standardized variable and function naming, and a clear code structure.

[1] https://registry.hkust.edu.hk/resource-library/academic-integrity