# Laser scan matching through ICP

David FILLIAT – ENSTA Paris

September 9, 2024

## 1 Introduction

In this practical work, we will implement and test the laser scan correlation method named Iterated Closest Point (ICP) [1] and some of its variants. For this, we will use the python code available on the course Moodle. The provided code makes it possible to read laser scan datasets associated with the odometry of the robot during the acquisition and provide a code skeleton to implement the ICP method.

Upload your report as a pdf file that includes your answers to the questions and the code you wrote on the Moodle (Paste the code in your report, don't send the python scripts).

## 2 Provided code

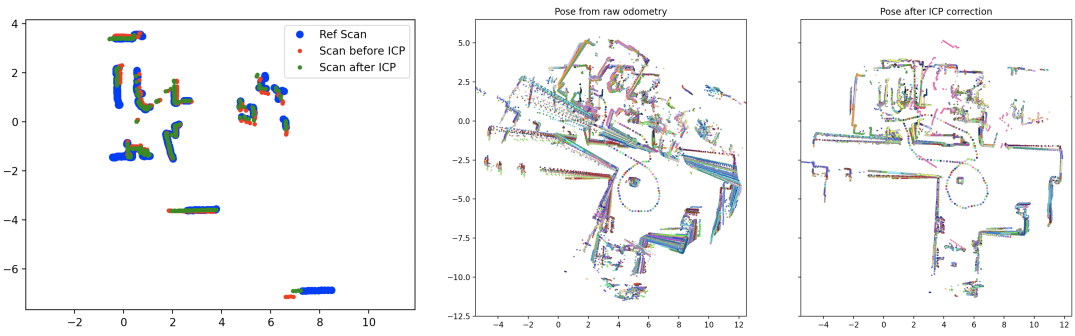


Figure 1: Sample of the `icpTest` and `icpLocalization` function display.

The provided code requires the installation of the `numpy`[1] and `matplotlib`[2] python packages. It contains several python scripts:

– `readDatasets.py` contains a set of functions to read and process the datasets provided in the `dataset` directory. Mainly the `read_u2is` and `read_fr079` functions read and return data as a list of dictionaries representing laser scans. The elements of each scan are:

- `ranges`: a numpy array of distances read by the laser scanner
- `angles`: a numpy array of angles corresponding to the distances above, it is different for each laser scanner
- `pose`: the absolute pose of the laser scanner computed from the robot odometry when the scan was recorded
- `x` et `y`: absolute position of each point of the laser scan computed from `pose`, `ranges` et `angles`

– `icp.py` contains a very basic implementation of the ICP method, without any pre-processing nor point association improvements. Matching is done with the nearest point using kdtrees[3] for a better computational efficiency. This code is designed for teaching purpose and readability, but is too slow for practical applications.

[1] https://numpy.org/
[2] https://matplotlib.org/
[3] https://en.wikipedia.org/wiki/K-d_tree

– `icpTest.py` is a script designed to evaluate the performance of the ICP method and the variants you will implement. It reads two scans whose true relative position is known, applies a random displacement to the second scan and applies the `icp` function to register the scan on the first one (Figure 1, left). It then measures the error between the computed position and the true position and reports mean translation and rotation error, their variance and computation time.

– `icpLocalization.py` reads a full dataset, then use ICP to register each scan on the previous one in order to correct the odometry pose error (Figure 1, right). It is able to correct a part of the Localization error, but will not provide a globally consistent view of the environment as it is not building any map.

# 3 Questions

You have to implement and evaluate the interest of different pre-processing seen during the course. You have to modify the `icp` function in the `icp.py` file. For each question, you should justify your answers with measures showing the improvements (or not) using the `testICP` function.

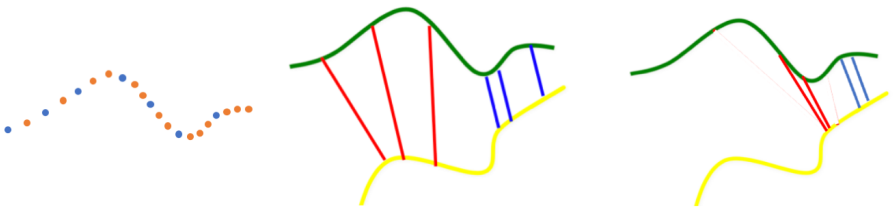

Figure 2: Illustration of the ICP optimisation tricks. Left: Keeping only evenly spaced points (blue). Center: Keeping only 50% of the best matching. Right: Keeping only matching pairs whose end-point distances are similar.

**Question 1:** Implement points filtering (removing points too close to each other in the 'scan_points' array, Figure 2, left) and show the consequences (on error, variance, computation time, ...) as a function of the parameter values.

**Question 2:** Implement match filtering by keeping a ratio $r < 1$ of the best matchings (Figure 2, center, illustrates $r = 0.5$) in the 'matched_scan_points' array. Try different values for $r$ and show the consequences (error, variance, computation time, ...).

**Question 3:** Use the best setting you found with the `icpLocalization` function. Analyse visually the quality of the result as a function of the `step` parameter and of the parameters of the previous filtering. Propose a reasonable compromise between the quality of the localization and the computation time.

**Question 4:** Implement filtering of the matchings by keeping only pairs of neighbor matchings whose endpoints are approximately at the same distance (Figure 2, right). Try different threshold values and show the consequences (error, variance, computation time, ...).

# References

[1] Yang Chen and Gerard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992. Range Image Understanding.