

# Assignment 1

---

January 20, 2023

## INSTRUCTIONS

This is the first coursework for course COMP0119 on Acquisition and Processing of 3D Geometry. The subject of the coursework is implementing iterative closest point (ICP) alignment. The coursework is worth *100 points*.

You are encouraged to use Python or C/C++ for this assignment, but can alternatively use Matlab. For efficient implementation, you will need a nearest-neighbor data structure. You may use a quad tree, or a BSP tree, or use a library such as Scikit-kdtree<sup>1</sup> or FLANN<sup>2</sup>.

Please **note**, that the coursework is *individual work*, we are expecting to see your solution (please do not attempt to look up answers from external sources or friends), including the code you hand in. For libraries you used, please reference them properly in your report (no need to include them in the zip). Please write your code, **do not** call a library's function, that does ICP for you.

**LATE POLICY** See Moodle for submission details. For late policy, please refer to departmental guidelines. If you are suspected of plagiarism, you might be requested to present your work in a one-to-one session (date to be decided) demonstrating what you have implemented.

## SUBMISSION GUIDELINES

- Please submit a report (PDF) on your work (to be submitted via Moodle). The report should start with a brief list of the questions you have attempted and to what extent you have achieved each goal.
- Next, write a short description of the methods you used for each part together with any conclusions you have drawn from your experiments.
- Augment your experiments (tasks 3, 4, and 5) and present plots or tables.
- Where appropriate visualize meshes from at least 2 different directions and provide screenshots.
- Color meshes differently for better visibility.

In summary, you are expected to submit (a) a full report (pdf) and (b) code (as a zipped file along with documentation and instructions for running the code). Please do *not* upload build directories, executables or 3D models you used as input.

---

<sup>1</sup><https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html>

<sup>2</sup><http://www.cs.ubc.ca/research/flann/>, <https://github.com/jlblancoc/nanoflann>

# 1 CORE SECTION

In this assignment we will explore different aspects of the ICP algorithm. Please refer to the lecture notes and also the original ICP papers (see lecture notes for relevant references). Start by downloading the bunny models from Moodle<sup>3</sup>. Select two models as  $M_1$  and  $M_2$  for the subsequent tests. Try to select models with sufficient overlap (you will need a viewer described in the first task to roughly judge this).

For all the tasks you are expected to write your own code, although looking up the original paper(s) is encouraged. Geometry processing is usually a computation-heavy task, hence it is important to write efficient code. Try to modularize your code into re-usable objects/files/modules.

1. The source data is in PLY format. The format is relatively simple and you can write your own parser, or use existing code (see the first tutorial<sup>4</sup> or libIgl's readPLY<sup>5</sup>). You may also convert the models to another format (e.g., OBJ, OFF, or xyz) using Meshlab.

Your first task is to make sure you have access to a simple mesh viewer (e.g., libIgl, or Meshlab) where you can visualize and inspect your intermediate results. You should be able to load meshes, and then interactively rotate, or translate them. It should be possible to load two (or more) meshes at the same time. Suppose, you load mesh  $M_1$  along with transformation  $T_1$ , and similarly  $M_2$  along with transformation  $T_2$ , then your viewer should show  $T_1(M_1)$  and  $T_2(M_2)$  in a common coordinate system. By default, you may assume  $T_1$  to be the trivial transform with the rotation component as the identity and translation being zero. You can use the example framework provided in our tutorial or similar (e.g., Meshlab) to start from. You will need this viewer throughout the course.

In this part, you will align model  $M_2$  to model  $M_1$  using only rigid transforms. Assume that the models are *nearly aligned* to start with and have a *large* (but unknown) *overlap*. Implement point-to-point ICP to align  $M_2 \rightarrow M_1$  and also mark out the non-overlapping regions (after alignment). You will need to have access to an eigen solver. You can try Numpy, Eigen or other libraries. You should use KDTree or similar to speed up nearest neighbor computations. (25 points)

In class, we discussed the solution to the optimization to minimize alignment error of the form  $E(\mathbf{R}, \mathbf{t}) = \sum_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2$  over unknown rotation  $\mathbf{R}$  and translation  $\mathbf{t}$ . Now, assume we have a confidence score for each measurement, *i.e.*, for each point  $\mathbf{p}_i$  we have an associated weight  $w_i \in [0, 1]$  with higher weight denoting higher confidence (or reliability). Refine the derivation done in class to solve the optimization  $E(\mathbf{R}, \mathbf{t}) = \sum_i w_i \|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|^2$  over unknown rotation  $\mathbf{R}$  and translation  $\mathbf{t}$ . Note that this part only requires a derivation – you are not required to implement this version. (10 points)

2. Assume that the model  $M_1$  is at the origin (*i.e.*, centroid of  $M_1$  is at the origin). Now assume  $M_2 = \mathbf{R}(M_1)$ , *i.e.*, a rotated version of model  $M_1$  about the origin. Progressively perturb the initial rotation of  $\mathbf{R}$  and evaluate the convergence behavior of ICP trying to align  $M_2 \rightarrow M_1$ .  $\mathbf{R}$  can be rotation about any of the coordinate axes for example. One way to simulate the effect of increasing misalignment (*i.e.*, initial alignment) is to rotate the object about z-axis over increasing rotation angles (say  $\pm 5, \pm 10, \pm 15, \dots$  degrees). (10 points)
3. Perturb model  $M_2$  to simulate a noisy model say  $M'_2$ . Evaluate how well ICP performs as you continue to add more noise. As for noise, add zero-mean Gaussian noise – you can simply perturb each vertex of the mesh  $M_2$  under this Gaussian model. Adjust the amount of noise based on the bounding box dimensions of  $M_2$ . (10 points)
4. Instead of directly aligning  $M_2$  to  $M_1$ , speed up your computation by estimating the aligning transform using subsampled versions of  $M_1$  and/or  $M_2$  as appropriate. Report accuracy with increasing subsampling rates. (10 points)
5. Now given multiple scans  $M_1, M_2, \dots, M_5$  align all of them to a common global coordinate frame. Make sure to think about the best strategy to do this. Describe the method you propose and discuss its pros/cons. (10 points)
6. Now assume that you have access to normals at each vertex of the meshes. Update your viewer to shade the models based on these normals. Now, use the normal information to improve the ICP performance. Like

---

<sup>3</sup>original data source :<http://graphics.stanford.edu/data/3Dscanrep/>

<sup>4</sup>[https://github.com/smartgeometry-ucl/COMP0119\\_20-21](https://github.com/smartgeometry-ucl/COMP0119_20-21)

<sup>5</sup><https://github.com/libigl/libigl/blob/master/include/igl/readPLY.h>

discussed in class, you are attempting to solve minimize the following objective function  $E(\mathbf{R}, \mathbf{t}) = \sum_i \|(\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot \mathbf{n}_i^q\|^2$  instead of minimizing  $E(\mathbf{R}, \mathbf{t}) = \sum_i \|(\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)\|^2$  as implemented in Q1. Note that points  $\mathbf{p}_i \in M_1$  and  $\mathbf{q}_i \in M_2$  with  $\mathbf{n}_i^q$  denoting normal at point  $\mathbf{q}_i$  to the mesh  $M_2$ . This method is commonly referred to point-to-plane ICP. (25 points)