Computer Vision 2 – Assignment 1
Iterative Closest Point (ICP)

Rick Groenendijk, Weijie Wei

Deadline: Friday 22-04-2022, 23:59:59 (Amsterdam time)

General guidelines

Students should work on the assignments in **groups of 3**. Students are supposed to work on this assignment for two weeks, during which some minor additions and changes to the assignment may happen. Students will be informed about these changes via Canvas announcements.
Please post your questions and discussion regarding the assignment content on Canvas Discussions. Let us know if you don't have access to it.
For this assignment, you can choose your preferred language for implementation, however support will be provided **only for Python**. Please provide a requirements.txt (or language equivalent) listing all external dependencies and a README.txt file with **clear descriptions on how to reproduce your results**.

Late submissions: Source code and report must be handed in together in a zip file (ID1_lastname1_ID2_lastname2_ID3_lastname3.zip) before the deadline by uploading through the Canvas submission system. For full credit, make sure your report follows these guidelines:

- Include an introduction and a conclusion to your report.
- The maximum number of pages is 10 (single-column, including tables and figures). Please be concise. The number of words does not necessarily correlate with how well you understand the concepts.
- Answer all given questions. Briefly describe what you implemented. Motivate your choices where applicable. Reports comprising of solely results will have the grades lowered (Avoid reports like, "Result 1 - check code outputs").
- Try to understand the problem as much as you can. When answering a question, give evidences (qualitative and/or quantitative results, references to papers, etc.) to support your arguments.
- Analyze your results and discuss them, *e.g.* why algorithm A works better than algorithm B in a certain problem.
- Tables and figures must be accompanied by a brief description. Do not forget to add a number, a title, and if applicable, name and unit of variables in a table, name and unit of axes and legends in a figure.

Late submissions are not allowed. Assignments that are submitted after the strict deadline will not be graded. In case of submission conflicts, TAs' system clock is taken as reference. We strongly recommend submitting well in advance, to avoid last minute system failure issues.

Plagiarism note: Keep in mind that plagiarism (submitted materials that are not your work) is a serious crime and any misconduct shall be punished according to the university regulations. For the assignment (and all subsequent future assignments), you are allowed to discuss on the respective Canvas Discusssion board. You are encouraged to discuss with your class mates regarding the problem. However, this should NOT include any code snippets.

Points: Maximum points are mentioned in the section headings. There are **no bonus points**. The total number of possible points in this assignment is **30 points**.

1 Introduction: Iterative Closest Point

The ICP algorithm was first introduced by Besl and McKay in 1992 (See [1] for details). Even now, as deep learning is often the standard to solve problems in computer vision, ICP is still used as a baseline because it is a powerful and robust technique. It also inspires contemporary research – *e.g.* [2, 3, 4, 5, 6].

Given two point-clouds A_1 (source) and A_2 (target) in a d-dimensional space, ICP attempts to find a spatial transformation that minimizes the distance (e.g. Root Mean Square (RMS)) between A_1 and A_2 which can be defined as

$$RMS(A_1, A_2, \psi) = \sqrt{\frac{1}{n}\sum_{a \in A_1} \|a - \psi(a)\|^2} \quad \text{where } \psi\colon A_1 \to A_2\,, \tag{1}$$

$$\min_{R \in SO(d), t \in \mathbb{R}^d} \sum_{a \in A_1} \|Ra + t - \psi(a)\|^2\,, \tag{2}$$

where R and t are the rotation matrix and the translation vector, respectively. For this assignment, assume a 3×3 rotation matrix and a corresponding translation column vector. Moreover, ψ is a one-to-one matching function that creates correspondences between the elements of A_1 and A_2. Finally, R and t, that minimize the above equation, are used to define the camera pose transformation between A_1 and A_2.

In this assignment, you will do analysis of the ICP algorithm and some of its variants. The practical aim is to automatically register many overlapping point clouds: Given a set of N overlapping point clouds $\mathbf{P} = \{A_n\}_{n=0}^{N}$ can you find all transformations R_n, t_n? Throughout the assignment you will be asked to implement (parts of) referenced papers. In these cases, you can use the default hyper parameters as discussed in the papers, although you are free to vary the values of the parameters to improve your results. A README.txt file is attached to this assignment; it contains the information about the provided data.

1.1 ICP Implementation (5 points)

First, the basic ICP algorithm will be implemented. To this end, implement each of the steps below:

1. Initialize $R = \mathbf{I}$ (identity matrix), $t = \mathbf{0}$.
2. Transform the current point cloud A_1 using the current values of R, t.
3. Find the closest point in the target cloud A_2 for each point in the source point set A_1 using a brute-force approach.
4. Solve for R and t using Singular Value Decomposition (Please check `https://igl.ethz.ch/projects/ARAP/svd_rot.pdf` for details). Note that the weights are set as 1 as default.
5. Go to step 2 unless RMS is unchanged or within a certain threshold.

There are different ways to improve the efficiency and the accuracy of ICP. Some of these techniques are discussed in [7]. In this assignment, you should do analysis of various aspects such as (a) accuracy, (b) speed, (c) stability and (d) tolerance to noise by changing the point selection technique.

Hint: You may find that A_1 and A_2 do not have the same number of points. For now, choose the most straight-forward approach to deal with this.

Hint: We recommend you use the provided dummy data (wave and bunny) to test the correctness of the estimated transformation for the questions in Section 1&2.

Hint: Finding all correspondences through brute force may take much computation time and/or memory. If you find yourself limited by your system hardware, you are allowed to downsample the point clouds. Take this into account in your analysis, especially in the following questions.

2 Improving Speed & Quality

As you may have noticed, the one-to-one point correspondence takes much computing time and/or memory each iteration since its complexity is quadratic in the number of points. Below, you will implement a number of strategies to improve the correspondence process.

2.1 Sampling (9 points)

The first improvement is changing how either point cloud is sampled. This also helps to ensure both point clouds consist of the same number of points. Use **(a)** all the points (implemented above); **(b)** uniform sub-sampling; **(c)** random sub-sampling in each iteration; **(d)** multi-resolution sub-sampling as described in [8]; and **(e)** sub-sampling from informative regions. You are expected to implement these variants and report findings in the report. For **(e)**, this is an open-ended question and many approaches are correct (*e.g.* corner points, curvature, point density, etc.). You only have to implement one, but be sure to clarify your method.

Hint: For uniform sampling you can sample once, whereas you have to sample again each iteration for random sampling.

2.2 kd-tree (2 points)

One of the most prevalent ways to reduce matching speed is the use of a *kd-tree*[9]. Please implement it, you are allowed to use external packages like *Scipy*, *sklearn*, *open3d*.

For your experiments, please combine the kd-tree with sampling methods. Can you replicate the results from [8]?

2.3 z-buffer (5 points)

Another approach to speed up matching is the use of a z-buffer, which has similarities to z-buffering in computer graphics. In this question you will implement the first part of [10] – *i.e.* the mono-z-buffer structure explained in detail in section 2.1 of the paper.

The z-buffer uses a single *z-buffer referential* to denote a reference for the two point clouds; it determines the direction of projection as the z-axis, and two orthogonal directions for the x,y-projection plane. For this exercise, let the z-buffer referential align with the origin of the target cloud A_2; that is, you can use the coordinate system of A_2 as is and you do **not** have to consider an arbitrary z-buffer referential. The mono-z-buffer uses two buffers for source and target clouds A_1, A_2 respectively. For the matching step of each iteration in your ICP, implement the following:

1. Take the union of A_1, A_2 and determine the minimum enclosing box on the x,y-plane.
2. Divide the minimum enclosing box into $H \times W$ rectangular cells.
3. Project points $a \in A_1$ orthogonally into the cells of the source buffer, keeping only the nearest one but saving all geometric (*i.e.* location) information. You thus keep a single point per cell.
4. Project points $a \in A_2$ orthogonally into the cells of the target buffer, keeping only the nearest one but saving all geometric (*i.e.* location) information.
5. For each cell at h, w in the source buffer, obtain the closest point in the target buffer by comparing it to the points in an $m \times m$ window centered at h, w.

The paper also implements an adaptive hierarchical strategy to change the $m \times m$ window throughout the iteration process and improving the point-to-point correspondence by adding normal estimation. You do not have to implement these improvements.

Think about the following questions for your report: What are the downsides of using a mono-z-buffer and how could it be improved? Does the z-buffer work well on the wave dummy data? Does it work well on the bunny dummy data? Is the choice of the z-buffer referential important to the quality of the registration? How would you implement a change in the z-buffer referential?

3 Global Registration

In this question, you are given a set of 100 overlapping point clouds in which a Kinect sensor has rotated around a student. All frames are located in the **Data** directory. Can you find all transformation R_n, t_n

between the frames? Using these camera pose estimate, is it possible to stitch together a full 3d model of the student?

Note that up to this point, you have been working with synthetic (idealized) data, whereas from now on you will work with real-world data. This data may be more noisy and contain artefacts that have to be cleaned.

3.1 Estimate pose every N frames & merge at the end (4 points)

Estimate the camera poses every N frames of given data where $N \in \{1, 2, 4, 10\}$. We recommend starting with the N=1 case. That is:

1. Given $\text{frame}_1, \text{frame}_2$ compute R_1, t_1.
2. Given $\text{frame}_2, \text{frame}_3$ compute R_2, t_2.
3. ... and so on.

Using all estimated camera poses R, t merge together the point clouds into a single cloud. Visualize and discuss the results. Does the merging produce sufficiently accurate results? Repeat the exercise for all N. Use the improvements to the ICP algorithm from Section 2 to increase speed and quality of the global registration.

Hint: Merging the full point cloud, especially for lower values of N, takes serious compute time. You do not have to perform an exhaustive grid search over all methods and parameters, but make use of your insights from Section 2.

Hint: The merged point cloud will contain many points, slowing down significantly your visualization. Make sure to use sub-sampling in displaying your results.

3.2 Estimate pose and merge every N frames (2 points)

You will now use the best set-up found in the previous question (Section 3.1). That is, whatever improvements you had found to work best for merging the point clouds will be used for this question as well. The difference is the merging strategy of the point clouds. Implement the following **only for** $N \in \{4, 10\}$, example given for N=4:

1. Given $\text{frame}_1, \text{frame}_5$ compute R_1, t_1. Merge $\text{frame}_1, \text{frame}_5$ into $\text{frame}_{1,5}$ using R_1, t_1.
2. Given $\text{frame}_{1,5}, \text{frame}_9$ compute $R_{1,5}, t_{1,5}$. Merge $\text{frame}_{1,5}, \text{frame}_9$ into $\text{frame}_{1,5,9}$ using $R_{1,5}, t_{1,5}$.
3. ... and so on.

Similarly for N=10. Do the estimated camera poses change in comparison with the previous estimates (Section 3.1)? Does this estimation produce better results?

4 Additional Questions (3 points)

Please answer the following questions during the discussion of your results:

1. What are the drawbacks of the ICP algorithm? Please provide proof to back up your claims.
2. How do you think the ICP algorithm can be improved, besides the techniques mentioned in [7], in terms of efficiency and accuracy?

5 Self-Evaluation

Please briefly describe the contributions of each team member to this assignment in an appendix to the report. Self-evaluation will be taken into consideration in cases where the contributions are significantly different.

References

[1] P. Besl and N. D. McKay, "A method for registration of 3-d shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 239–256, 1992.

[2] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3523–3532, 2019.

[3] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[4] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "Deepvcp: An end-to-end deep neural network for point cloud registration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12–21, 2019.

[5] S. Rusinkiewicz, "A symmetric objective function for ICP," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 38, July 2019.

[6] R. Huang, Y. Xu, W. Yao, L. Hoegner, and U. Stilla, "Robust global registration of point clouds by closed-form solution in the frequency domain," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 171, pp. 310–329, 2021.

[7] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, 2001.

[8] T. Jost and H. Hugli, "A multi-resolution scheme icp algorithm for fast shape registration," in *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pp. 540–543, IEEE, 2002.

[9] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[10] R. Benjemaa and F. Schmitt, "Fast global registration of 3d sampled surfaces using a multi-z-buffer technique," *Image and Vision Computing*, vol. 17, no. 2, pp. 113–123, 1999.