Computer Vision 2 - Assignment 1
Iterative Closest Point - ICP

Guido Visser
10199187
Guido-visser@hotmail.com

Bella Nicholson
12094870
nicholsonbella@gmail.com

Peter Heemskerk
11988797
peter.heemskerk@planet.nl

1 Introduction

Since its inception in 1992, the iterative closest point (ICP) algorithm has become the standard method for the geometric alignment of three-dimensional models (Besl and McKay [1992]). The increase in demand for more computationally efficient and robust ICP performance has lead researches, such as Rusinkiewicz and Levoy [2001] and Gelfand et al. [2003], to continuously improve and build upon the original algorithm throughout the past three decades. Within this report, we use the ICP to reconstruct a three-dimensional object from a series of images that were taken at different camera positions. In order to do so, we first developed our own implementation of ICP by iteratively testing and improving it until satisfactory results were obtained.

As such, this report is organized into two primary parts: (a) ICP development and testing, and (b) the 3D reconstruction of objects using ICP. All implementations within this report were executed using Matlab.

2 Iterative Closest Point - ICP

The iterative closest point (ICP) algorithm is a classic general-purpose, representation method for accurate 3D free-form curve and surface registrations (Besl and McKay [1992]). In essence, it finds the spatial transformation that minimizes the difference between two point clouds. Its most fundamental implementation is done as follows:

Step 1. Initialize rotation and transformation matrices R and t.

Step 2. Using brute force, find the closest data point from target point cloud for all data points present in the source point cloud.

Step 3. Refine matrices R and t using Singular Value Decomposition (SVD).

Step 4. Repeat Steps $1 - 3$ until the RMS ceases to change.

As the “basic" version of ICP requires processing the *entire* point cloud, a procedure which can become computationally draining very quickly, several different methods of point cloud sampling were developed and tested:

1. *Uniform Sampling*. Randomly and uniformly data points from the entire source cloud data.
2. *Uniform Sampling*. Rather than initially fixing the points selected for sampling, we can also randomly sample new data points at the start of each new iteration.

3. *Region-of-Interest Sampling.* The previous two approaches arbitrarily sample data-points; however, it may be wiser to sub-sample information-dense regions —which, would result in exclusion of outliers. Thus, in this particular sampling approach, we perform k-means clustering and only sample from the n most information-dense clusters.

Additionally, the following performance measures have been implemented: (a) accuracy, (b) speed, (c) stability, and (d) tolerance to noise. Performance testing was conducted by using the provided source.mat and target.mat files.

2.1 Accuracy, speed and stability

Twenty different trials were conducted per the experimental configurations detailed in Table 1. To keep all results comparable and ensure speedy computations, the maximum number of iterations and and the convergence criteria of RMS were set to was set at 300 and 0.0005 respectively as preliminary testing showed these values provide reasonably good RMS values within a manageable computational budget.

	sample size	**accuracy**	**std(RMS)**	**speed (nr iters)**	**speed (time(s) / iteration)**
Basic	6,400	0.16388	8.543e-17	300	13.380
Sample Once	20	0.19956	0.016281	300	0.071
	100	0.16206	0.0092731	300	0.277
	300	0.16104	0.0061539	300	0.647
	1,00	0.16308	0.0032119	300	1.940
Sample per Iter	20	0.15198	0.0057495	300	0.0715
	100	0.15014	0.0024913	300	0.255
	300	0.1539	0.008037	300	0.632
	1,000	0.16391	0.00098322	300	1.843
Informative Regions	20	0.21861	0.026924	300	0.067
	100	0.18594	0.022037	300	0.199
	300	0.17402	0.012522	300	0.643
	1,000	0.16889	0.01418	300	1.815

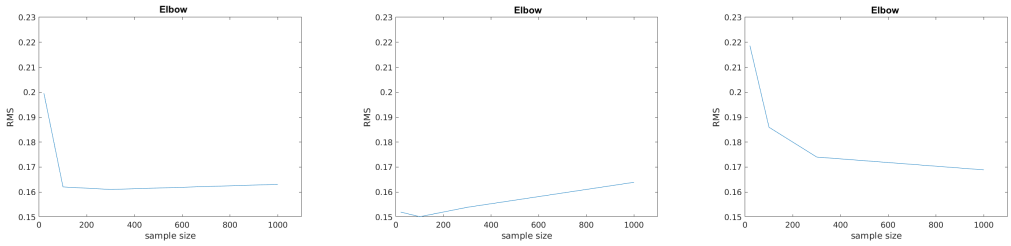
Table 1: Accuracy (RMS) and Speed performance on the ICP implementation for each method and considered sample size. Standard deviation for RMS on 20 tests. Although speed time per iteration is machine dependent, it can be concluded that time per iteration is consistent with sample size.

First, we can observe that the number of iterations is consistent across all sampling methods and sample sizes, suggesting a fundamental flaw with convergence criteria selected. For this set of experiments, convergence was defined as the case in which the RSM difference between the current and subsequent iteration fell below 0.0005. As it can be seen from the accuracy column of Table 1, this value was likely an 1-2 orders of magnitude too low. This, in turn, forces the algorithm to run until it hits its maximum number of iterations.

As expected, sampling the entire point cloud is the most computationally draining method; however, its pay-off is very low since certain instances of fixed random sampling (Method 2) and randomly sampling per iteration (Method 3) outperform Method 1. In terms of accuracy, all methods are more or less comparable, making further implementation of Method 1 an illogical choice.

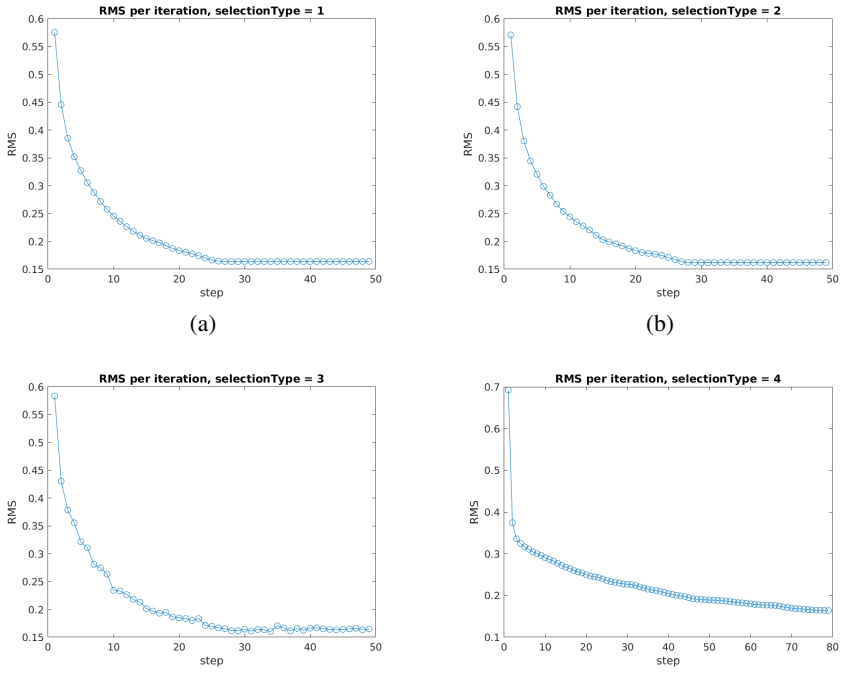
Interestingly enough, informative regions sampling generally performs *worse* than all instances of randomly sampling. Two plausible explanations are: (a) That in only sampling the most densely clustered areas of a point cloud, we lose the amount of variety needed to make a series suitable spatial transformations. (b) A lack of hyper-parameter tuning. The method of informative-region sampling introduces the hyperparameters of K predefined clusters and n most-dense clusters. If hyper-parameter tuning were to be conducted, the optimal K and n values would likely vary depending upon the sample size taken as a larger sample size requires finer partition than a smaller sample size would. However, doing so would become a time-consuming process which may not yield better results than Methods 2 or 3.

To make more informed decision about what method to implement, the data in Table 1 was plotted in order to infer the optimal number of samples with respects to each method (Figure 1) and to better understand each method's convergence behavior (Figure 2).



(a) (b) (c)

Figure 1: RMS for different sample sizes. **(a)** Method 2: Sample once. An "elbow" is observed at about 200 samples, indicating this sample size as optimal for Method 2. **(b)** Method 3: Sample every iteration. Interestingly, there appears to be a sharp increase in RMS after a sample size of 100 that appears to then plateau quickly. **(c)** Method 4: Sample informative points. This approach yields RMS values slightly higher than those of Method 1. The optimal sample size to appears to be ≈ 300.



(a) (b) (c) (d)

Figure 2: Method Stability. **(a)** Method 1: No sampling results in the smoothest stability curve. Convergences occurs around 28 iterations. **(b)** Method 2: Sample once introduces slightly more unstable behavior and prolongs convergence by a few iterations. **(c)** Method 3: Sample every iteration introduces the most oscillatory (noisy) behavior, but it converges within the same number of iterations as Method 2. **(d)** Method 4: Sample informative points takes the longest to converge but has very little noise.

Generally, the ideal number of samples appears to be somewhere near 200, in which Method 2 favors sparser sampling, while Method 3 favors slightly more dense sampling. The convergence behavior between Methods 1-3 appear consistent despite the slight noise observed during Method 3's

convergence. Meanwhile, Method 4 converges far more slowly; however, this may be a byproduct of ICP convergence dependency on k-means convergence. Regardless, with every ICP iteration performed, the RMS exponentially decays.

Initially, sampling per iteration at a sample size of 200 was deemed the most suitable choice as it yields the most accurate results (Figure 1).

2.2 Noise tolerance

Next, uniform distributed random noise was added between the minimum and maximum values of all three point clouds axes. In the Figure 3, the methods are compared for several percentages between 0% and 10% noise added. Method 4, informative regions sampling, is relatively insensitive for noise —something that is as expected considered its data selection method is inherently more constrained than any of its counterparts are. All random-selection based methods struggle with the addition of noise.

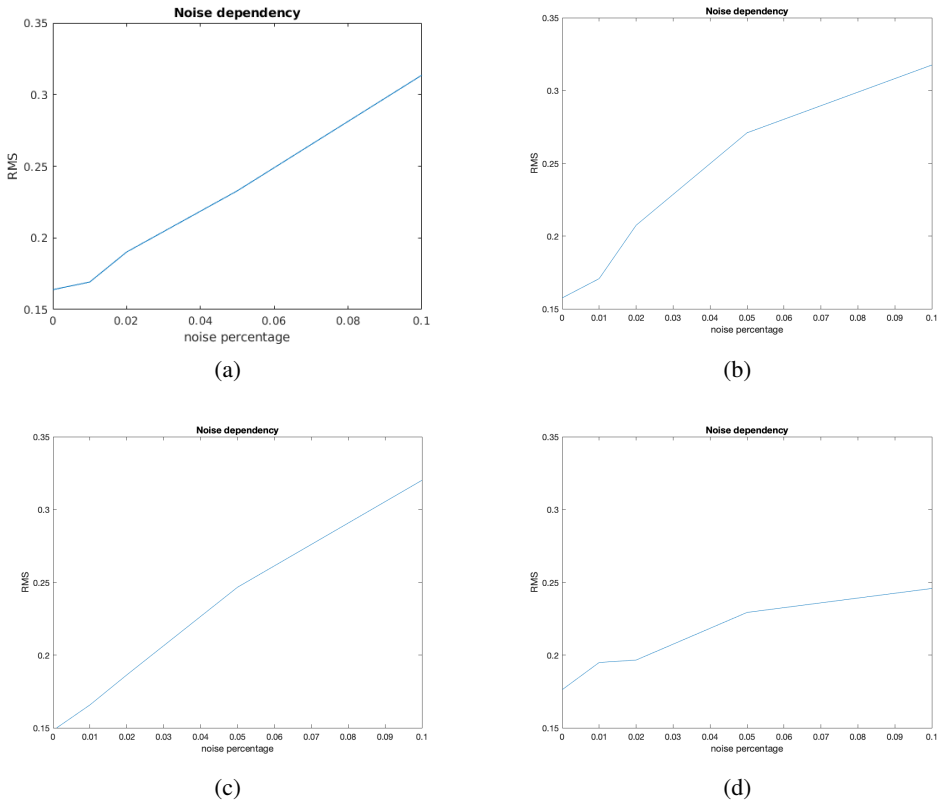


Figure 3: Noise tolerance. An increase in noise percentage results in an increase in RMS. **(a)** Method 1: No sampling. **(b)** Method 2: Sample once. **(c)** Method 3: Sample every iteration. **(d)** Method 4: Sample informative points.

3 Merging Scenes

Unfortunately, due to the excessive computational demands of this procedure, it was deemed infeasible and unrealistic to merge the *entire* data set of point clouds provided within the given time frame. As a result, only a small selection of point clouds was selected to be merged to recreate the 3D object originally photographed, since it stands to reason that if more computational power and time were given, this approach could easily be extended to the entire data set. Thus, the reader is encouraged to think of the results obtained within this section as proof of concept rather than completed implementations.

3.1 The Consecutive Merging of Frames

By consecutively merging frames, we aim to merge each frame into (only) its predecessor, which ensures that each frame remains well-fitted against its neighbors and minimizes distant frame interference. Additionally, this approach ensures that the ICP algorithm only needs to find its closest point from one point cloud instead of many. Unfortunately, this approach creates a dependency between the current camera position estimation and all previous estimations, making this method vulnerable to “error propagation", in which the earlier made errors exert more strongly affect the point cloud merging process.

This method can be performed with different step sizes between each consecutive frame. We compare the RMS and computation time of this approach with step sizes 1, 2, 4, and 10 (see Table 2). As well as a visual interpretation of the quality of the merging.

For all runs, the previous section's Method 3 was used. Even though previously findings suggested an ideal sample number of 200, such a sample is too small to accurately represent the enormity of the point clouds provided. A sample size of 500 was empirically found to fairly represent the point clouds. The ICP algorithm was given 500 iterations to converge before termination, where the definition convergence is that of the previous section. [Once again, parameter tuning between the merging of the two test point clouds failed to be representative of the merging many point clouds.]

As shown in Table 2, the mean RMS decreases alongside a reduction in the sampling rate. Most likely, a small sampling rate skews represents the point clouds towards being more similar than they truly are —where a large sampling rate can be considered a more faithful representation. The time per iteration is not dependent on the sampling rate, but the total computational cost decreases with a higher sampling rate as there are less frames to be considered.

Visually we can see that a larger sampling rate decreases the quality of the merging quite drastically, which can be explained by the fact that the point clouds are less similar (Figure 4). We hypothesize that this effect can be mitigated through an increase in the the point sampling size of the ICP method, and by implementing the proposed improvement in Section 5.

3.2 The Iterative Merging of Frames

The iterative approach of merging the frames is far less sensitive to error propagation when compared to the consecutive approach (Table 2). However, it is computationally more taxing since each point that is considered by the ICP needs to be compared with an ever growing number of points in the target cloud. The iterative approach yields a mean RMS that is considerably smaller than that of the consecutive method. This decrease in RMS comes at the cost of computation time and can be explained by the larger target cloud that needs to be matched. Figures 4(g), 4(h), and 4(i) show that the clouds do merge well to create a smooth and cohesive 3D object representation. The first ten clouds have been merged into one picture (once again, merging all clouds was computationally to expensive for this project). Visually, there is little difference between the iterative method and the consecutive method when a sampling rate of one is used; however, since the RMS of the iterative method is lower than that of the consecutive method, we can conclude that the iterative method is still more precise.

method	sampling rate	mean RMS	s/it
consecutive	1	0.035356	**111.20**
consecutive	2	0.039696	126.17
consecutive	4	0.043000	122.68
consecutive	10	0.06098	119.82
iterative	N/A	**0.02827**	515.7426

Table 2: Merging the frames using different settings. The iterative method significantly provides better mean RMS values, but does so at the cost of computation time.

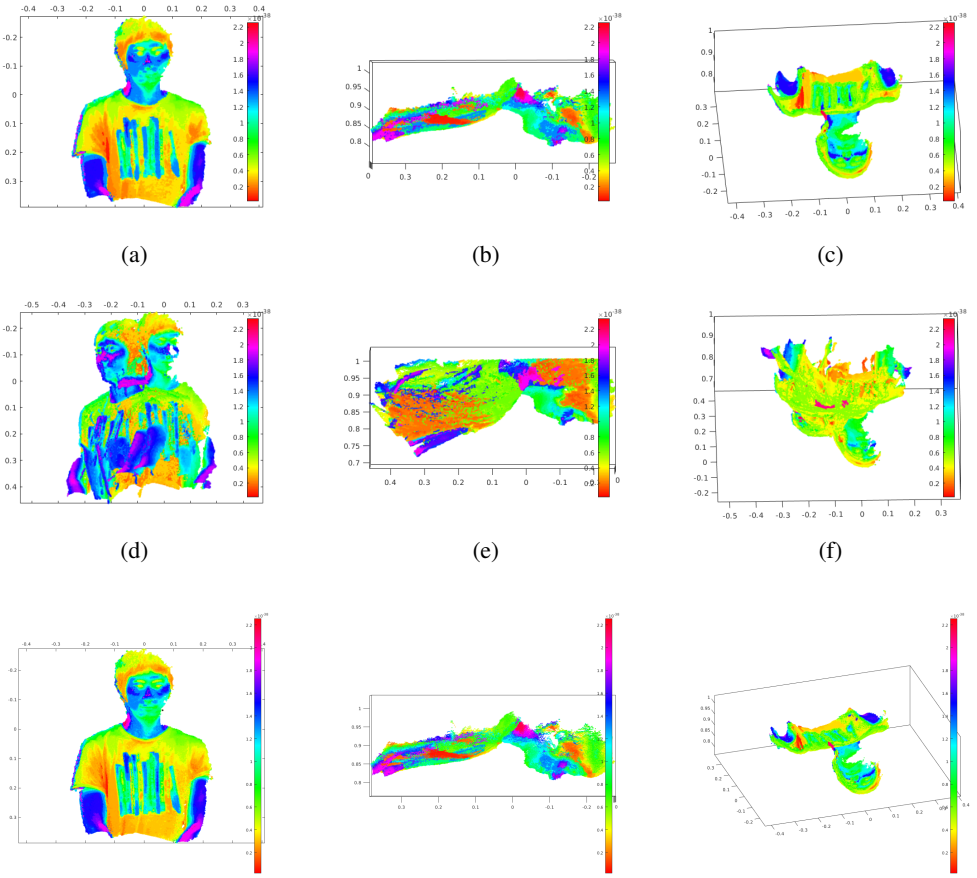


Figure 4: ICP merging output with respects to various point cloud merging methodologies. Figures **(a)**, **(b)**, and **(c)** show different perspectives on the consecutive merging of the first 10 point clouds when using a sampling rate of 1. Figures **(d)**, **(e)**, and **(f)** show different perspectives on the first 10 point clouds with the consecutive method and a sampling rate of 10. Figures **(g)**, **(h)**, and **(i)** show different perspectives on the first 10 point clouds that are obtained via the iterative merging method.

4 Conceptual Questions

1. *What are the drawbacks of the ICP algorithm?*

   By nature, the search for closest point pairs between two point clouds is a time-consuming and slowly converging process. Additionally the ICP initially proposed requires an initial guess —in which, a poor initial guess yields undesirable results. Without the testing and development of any heuristics, such as the penalty suggested by Besl and McKay [1992], much time can be wasted in an attempt to escape a local optimum. Finally, the original process is highly sensitive to outliers (Chetverikov et al. [2002]).

2. *How do you think the ICP algorithm can be improved, beside the techniques mentioned in Rusinkiewicz and Levoy [2001], in terms of efficiency and accuracy?*

   Buiding upon Rusinkiewicz and Levoy's initial introduction of point selection via the maximization of the normals distribution, Gelfand et al. [2003] provide a new stability analysis method (covariance sampling). Stability analysis can be thought of as a check to see if the points selected are satisfactory for registration, where in this particular method point sample selection uncertainty is minimized. Meanwhile, Segal et al. [2009] merge the point-to-point and point-to-plane approaches discussed by Besl and McKay [1992] to

obtain a single probabilistic framework. The combined algorithms outperforms both of its constituents, and is considered more robust and easier to tune. As there is always progress to be made, it will always be possible to improve ICP performance. The true question is: when are the results we obtain good enough that there's no further incentive to drive this search towards improvement? The continual onslaught of article publishings suggests that this will not be the case anytime soon.

5 Additional Improvement

In lieu of Rusinkiewicz and Levoy [2001]'s suggested but not implemented ICP improvement, we chose to implement a heuristic that rejects point-pairs based on their color similarity, or rather, based on their dissimilarity. Rejecting pairs based on color can significantly improve the matching between points because it takes into account not only the position of the point, but also its value. However, due to the time-consuming process required to obtain the results for all subsequent sections, this improvement was not able to be brought to fruition. As a result, this section serves as a guide on how to implement these improvements, and hypothesizes their resulting effects. The ICP procedure previously described becomes as follows:

Step 1. Initialize rotation and transformation matrices R and t.

Step 2. Using brute force, find the closest data point from target point cloud for all data points present in the source point cloud.

Step 3. Compare color values of the two points in the pair. If the difference in color value is above a pre-defined threshold go back to *Step 1.* If it is below the threshold, continue.

Step 4. Refine matrices R and t using Singular Value Decomposition (SVD).

Step 5. Repeat Steps $1 - 4$ until the RMS ceases to change.

The additional *Step 3* should aid in the aversion of mismatched cloud production that is seen in Figures 4(d) to 4(f).

6 Conclusion

Many more-advanced versions ond improvements of ICP have been introduced over the years; however, they all share the same objective: efficiently aligning three-dimensional surfaces (Segal et al. [2009]). This procedure —as well as the many improvements and variations of it that have been introduced throughout the years —serves as the basis for 3D object reconstruction tasks. Due to the sheer immensity of computational power required to recreate a fully 360 degrees reconstructed model from the data set provided, all the results illustrated provide proof of concept by only implementing this procedure to small subset of the data set provided. It is at the reader's behest if he or she wishes to execute the scripts provided to the entirety of the data set. Additionally, only some improvements were implemented and/or acknowledged as there exists a large body of research detailing the many ways to optimize ICP performance. The work of Rusinkiewicz and Levoy [2001] provides a clear and comprehensive framework for various methods of internally improving ICP components, while Segal et al. [2009] shows the power of ICP blended with similar models.

7 Appendix

7.1 Contributions

Peter Heemskerk designed the ICP function and testing scripts. Bella Nicholson supplemented and debugged both files in addition to conducting the experimentations. Guido Visser designed and executed the scripts to conduct many point cloud merges. All authors wrote with respects to the parts of the project they implemented, and peer-reviewed the rest of the paper.

References

Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152, 2001.

Natasha Gelfand, Leslie Ikemoto, Szymon Rusinkiewicz, and Marc Levoy. Geometrically stable sampling for the icp algorithm. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 260–267. IEEE, 2003.

Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548. IEEE, 2002.

Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435, 2009.