Computer Vision 2 - Assignment 1
Iterative Closes Point - ICP

Weitao Luo, Yijie Zhang and Xiaoxiao Wen
Faculty of Science, University of Amsterdam

1 Introduction

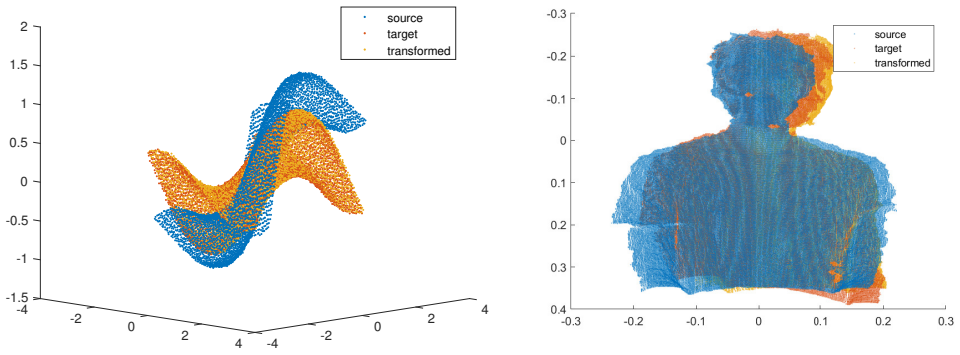
Designed for surfaces matching, Iterative closest point (ICP) is one of the most popular algorithms used to reconstruct 2D or 3D surfaces from different scans and the core idea behind ICP is to minimize the difference between two clouds of points. In this assignment, we will implement Iterative Closest Point (ICP) algorithm in Matlab and use it for merging scenes as application. In section 2, implementation of ICP is shown and discussed. In section 3, we will apply ICP for merging scenes using different parameters. And finally, in section 4, some questions about drawbacks and improvement of ICP will be discussed.

2 Iterative Closest Point

2.1 Basic Implementation

The basic ICP algorithm (sampling with all the points) is implemented according to the procedures and the method described in [1,2], where all the points in the source point cloud are used for estimating the rigid transformation R (rotation) and t (translation).

In Figure 1, the source, target and the transformed point clouds are plotted respectively to illustrate the quality of the estimated spatial transformation. As can be seen, for the toy example, the transformed point cloud almost perfectly lies on the target point cloud; for the frames (source: '0000000000', target: '0000000010'), the estimated spatial transformation can precisely transform the source point cloud to the target point cloud with relatively small misalignment.



(a) **(b)**

Figure 1: *Correctness check of the implementation of the ICP algorithm using all points for the toy example (left) and the frames (right).*

2.2 Variants Implementation

Including the basic implementation, there are four variants of the ICP algorithm developed differentiated by the point selection technique. The different variants have the the following selection choices from the source point cloud:

1. (Basic) All the points

For point matching from the source to the target point cloud, all the points from the source point cloud are used.

2. Uniform sub-sampling

   For point matching, a uniform sub-sampling is performed to select points from the source point cloud. This selection is fixed for all the iterations of the ICP algorithm. The number of selected points is a hyperparameter k which is empirically determined beforehand.

3. Random sub-sampling in each iteration

   In each iteration, before point matching, a random sub-sampling (uniform) is performed on the source point cloud that randomly select points for the point matching. This point selection is only valid for this iteration of the algorithm. The number of selected points is a hyperparameter k which is empirically determined beforehand.

4. sub-sampling more from informative regions (normal space sampling)

   For point matching, the points are selected from the source point cloud according to their corresponding normal vectors. The normal vectors are first divided into a certain number of 2D bins that is a hyperparameter in the angular space of the spherical coordinate system in the direction of azimuthal angle ϕ and polar angle θ, where an index to the 2D bins is computed in column-major order combining the two 1D bins. Then, a distribution of the normal vectors in the angular space is computed using the 2D bins indices, from which again a uniform sub-sampling is performed to select the points which have a high variance in terms of the distribution of the normal vectors.

The empirically determined hyperparameters used in this report are $k = 1000$ as the sample size for all sampling methods, $\epsilon = 1e^{-5}$ as the slack for convergence check and $bins = 500$ as the bin size for the normal space sampling method.

2.3 Comparisons

The different variants of the ICP algorithm are compared against four metrics: (a) accuracy, (b) speed, (c) stability and (d) tolerance to noise.

To compare the algorithm with respect to accuracy, the final RMS (Root Mean Square) level after convergence is regarded as the proxy. After executing the algorithm to estimate the transformation from the source to the target, the RMS after convergence is computed between the transformed source point cloud (including all the points) and the target point cloud taking the point matchings between the two into consideration. If the RMS after convergence is lower, then the accuracy of the estimation is higher, because the distance between the transformed point cloud and the target point cloud is smaller given the estimation of this algorithm.

As for comparing the speed of the variants, it is straightforward to use the number of iterations the algorithm takes until convergence as the metric. As the differences in computational complexity for the four variants before or during the execution are trivially similar, the speed of the algorithm is positively correlated with the number of iterations, and the number of iterations, thus, serves as a good proxy.

The stability of the algorithm is analyzed by first creating an augmented target point cloud from the source point cloud using a randomly generated rigid transformation ($\mathcal{N}(0,1)$) and then observing the convergence of the algorithm with this random initial condition or disturbance. If the algorithm is able to reject the the disturbance and, therefore, converges smoother given the same random initial condition, then it is considered to be more stable.

On the other hand, the tolerance of the algorithm to noise is analyzed by comparing the convergence of the algorithm between being given a clean source and a noisy source. Given the same noise, if the algorithm converges faster, then it is more robust or tolerant to noise. In reality, the noise in the measurements originates from the sensors, which is simulated by a Gaussian noise ($\mathcal{N}(0, 0.01)$).

The following comparisons are made using two inconsecutive frames from the dataset (source: '0000000000', target: '0000000030').

2.3.1 Accuracy and Speed

The results of the comparison is shown in Table **??**. As can be observed, in terms of accuracy, the random sub-sampling variant has the best performance of 0.029903 close to that of using all the points for

point selection. As for speed, the normal space sampling variant exceeds with 20.3 iterations on average. These results show that first the random sub-sampling variant has a similar accuracy as using all the points given that it runs for marginally more iterations such that it is able to refine estimation by oscillating without saturating at convergence. Furthermore, the normal space sampling variant is able to decrease the number of iterations needed to converge since it exploits more informative regions which boosts the efficiency. Nevertheless, once it tends to a certain level of RMS, it cannot easily escape the local minimum and converges slowly.

Table 1: *Comparisons in terms of accuracy and speed. The results are the average values of 10 experiments.*

Variant	RMS	Iterations
1	0.029906	27
2	0.057075	27.7
3	**0.029903**	112
4	0.031522	**20.3**

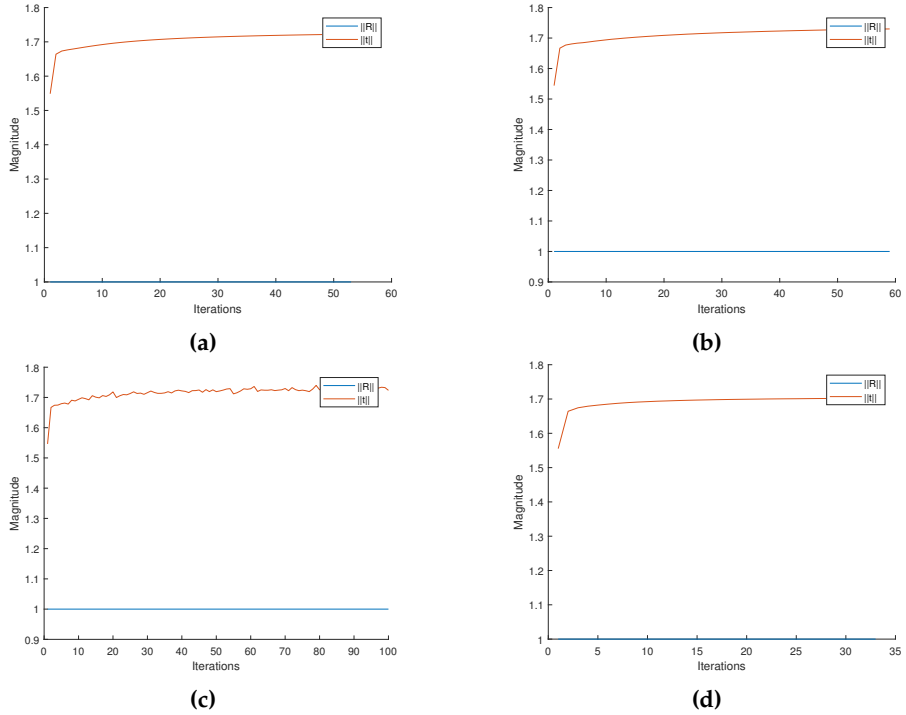


Figure 2: *Comparison in terms of stability for the four variants: (a) all the points, (b) uniform sub-sampling, (c) random sub-sampling and (d) normal space sampling.*

The comparison for stability is illustrated in Figure 2. The magnitudes for both R and t are plotted against the number of iterations before convergence. The normal space sampling variant of ICP has the best performance in terms of stability against disturbance/random initial condition. It is able to stabilize the magnitude of t to the required level in the shortest time, while the other variants are other relatively slower or cannot manage to stabilize. This can be explained by the advantage of using normal space sampling that it makes use of the more informative region in the source point cloud. As the crucial/interesting points are pinpointed easily, using these points to estimate the transformation is advantageous for random initial conditions.

The comparison for tolerance to noise is shown in Figure 3. It can be observed that the random sub-sampling variant has the best robustness against noise as it converges most quickly to a similar level of RMS as without noise. This is due to the large stochasticity introduced by random sub-sampling per iteration. Furthermore, we can observe that with additional noise, the normal space sampling variant is also capable of converge in relatively small number of iterations but the final accuracy is deviated by large compared to the case without noise. This can be explained by the fact that as external information, normal vectors, are used to supplement the point selection from more informative regions, these selected

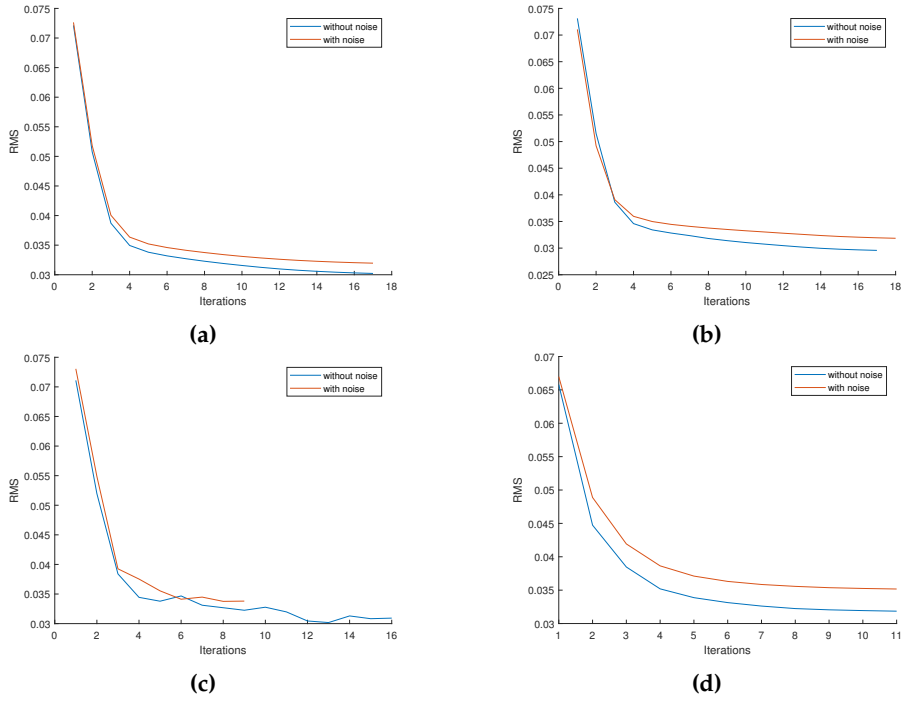


Figure 3: *Comparison in terms of tolerance to noise for the four variants: (a) all the points, (b) uniform sub-sampling, (c) random sub-sampling and (d) normal space sampling.*

points are also more crucial or interesting in terms of defining the point cloud. Hence, with additional noise at these crucial points, the estimated transformation becomes fragile.

3 Merging Scenes

For the section of Merging Scenes, we experimented with two strategies for merging scenes.

3.1

In the first strategy, we estimate the camera poses using two consecutive frames. Using the poses we estimated, we can merge all the scenes into one point cloud. The results for the first strategy are shown in figure 4. The point clouds in the results of this report are down-sampled with a sample rate of 100 to give better views. As we can observe, the point clouds we obtained can roughly depict the outline of the target.

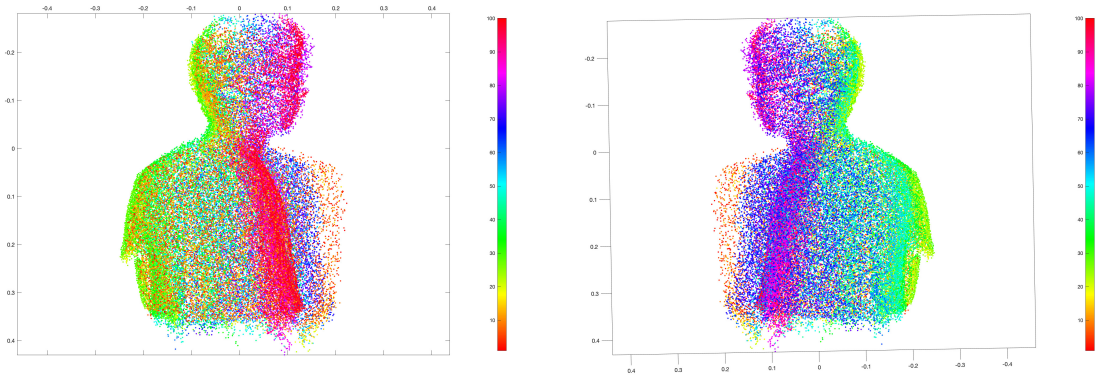


Figure 4: *Merged scenes for the first strategy*

When estimating the camera poses, we can pick the frames with a step size. In figure 5, we can see the merged point-clouds for step size 2,4 and 10. When the step size is relatively small, such as n=2, the result is quite similar and camera poses stay unchanged. But as we increase the step size, the model gradually deforms and becomes more unstable.

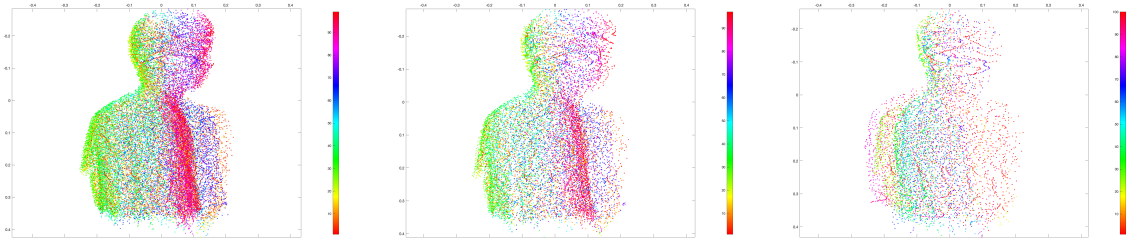


Figure 5: *Merged scenes for different step size (n=2,4,10)*

3.2

In the second approach, instead of estimating the camera poses using two consecutive frames, we use the merged frame as a target to obtain the transformation parameters. And the next given frame is then merged into the merged frame to get a new merged frame. The new merged frame is then the target of the next iteration. After merging all the point-clouds, the results are then given in figure 6. Compared to Figure 4, the outline of the target is more clear and vivid.

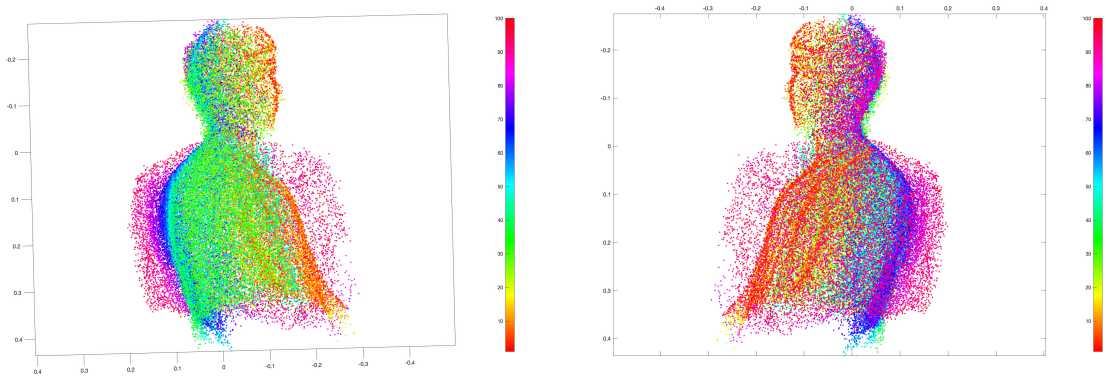


Figure 6: *Merged scenes for the second strategy*

4 Questions

1. What are the drawbacks of the ICP algorithm?

There are basically three major drawbacks of ICP algorithms, namely:

1. A large effort for data preprocessing is needed.
2. A good initial estimation of the relative pose is needed.
3. Algorithm convergence can be slow because finding the point pairs with smallest distance using brute force can be expensive.
4. Stability. If the surfaces are flat or nearly flat, ICP could fail to find a unique solution.

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

The paper mentioned had introduced and compared improvements from all phases of the algorithm from the selection and matching of points to the minimization strategy. So here we are going to introduce one technique that may improve the model performance by making good use of data preprocessing.

To make the algorithm more robust and reduce the error, one useful technique is to eliminate the data outliers. The most intuitive way is to set up some kind of threshold to define and remove outliers. A good candidate of thresholds can be the percentile of the data in the distribution and the data points outside the threshold percentile could be given the threshold value for smoothing. There is also other technique that may help, for example, a 3d boolean algorithm can be used to preprocess wrong matches for better accuracy.

5 Additional improvements

As described in [3], rejection of 10% of the worst pairs in terms of distance to eliminate outliers is implemented as additional improvement on the ICP algorithm. To reduce stochasticity, during comparison between using rejection and without using rejection, the deterministic selection of all the points is used. In terms of accuracy and speed, the results are $RMS = 0.029906$, Iterations = 27 without the rejection and $RMS = 0.030313$, Iterations = 44 with the rejection. The results, however, show that the improvement does not help with the performance but degrades it.

In order to improve the computational complexity, instead of brute force, k-nearest neighbors method is used for pairing the points between the source point cloud and the target point cloud. This improvement cannot be measured by the metric we introduce in this report (number of iterations), but is obvious in terms of real-world execution time.

6 Conclusion

This report is about the implementation of the ICP algorithm and its variants. Starting from the mathematical definitions, we have successfully implemented the basic ICP algorithm, some of its variants as well as tested and experimented some of the most important aspects for model performance like point selection techniques. We also show that ICP can be good in merging scenes and multiple frame sampling rate is experimented. Finally, drawbacks and potential improvement for ICP are discussed and some of the improvements are tested in real data.

A Self-Evaluation

Throughout the assignment, we distribute our work fairly in terms of both experiment implementation and documentation.

References

[1] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 239–256, feb 1992.

[2] O. Sorkine and M. Rabinovich, "Least-squares rigid motion using svd," Tech. Rep. February, 2009.

[3] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, pp. 145–152, IEEE Comput. Soc, 2001.