Assignment1: Iterative Closest points

Diego van der Mast 12222933
Vasiliki Vasileiou 13640496
Sameer Ambekar 13616064

Abstract

We implement the ICP algorithm for small scale datasets and explore how to optimize it using various techniques and also apply it to high dimensional data.

1 Introduction

In this assignment, we perform a wide analysis of the ICP algorithm [4] and some of its variants [3, 9, 10]. Iterative closest point (ICP) is an algorithm employed to minimize the difference between two clouds of points (RMS value), by consecutive calculations of the Rotation (R) and translation (t) matrices. As a second direction in this assignment we implement different techniques that have been proposed in order to improve the original ICP algorithm in terms of i) accuracy, ii) speed, iii) stability and iv) tolerance to noise. In section 3, we transfer our evaluation from the synthetic (idealized) data to the real-world data, which are mostly noisy and contain artefacts. The object of this part is the reconstruction of the 3D representation of a man from 2D frames by examining two cases, the merging of all pose estimations at the end and the merging every N frames.

2 ICP Implementation

2.1 Algorithm Steps

This section details the workings of the standard Iterative Closest Point (ICP) algorithm [4]. ICP aims to minimize the distance between source pointcloud A_1 and target pointcloud A_2 in d-dimensional space by computing rotation matrix R and translation vector t that translate A_1. The distance is often quantified using the root mean square (RMS):

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

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

where ψ maps the points of A_1 to A_2. This is required to find which points correspond to each other. Additionally, A_1 and A_2 do not necessarily have the same number of points. ψ ensures that only points that have a correspondence are used. In our baseline implementation, ψ computes the Euclidean distance between points in A_1 and A_2, selecting the points in A_2 with the lowest distance for each point in A_1 as correspondences. To initialize ICP, we set R to $\mathbf{I}$ and t to $\mathbf{0}$ using d dimensions. The algorithm iteratively runs until the RMS converges within a specified threshold or it runs for a set number of iterations. In each iteration, we match the points of A_1 to A_2 and calculate the RMS, after which we update R and t using Singular Value Decomposition as outlined in [13]. We then use them to translate A_1, which subsequently is used in the next iteration such that it approaches A_2 incrementally in every step. This means that R and t are also different in each step. In order to keep track of the global spatial translation, we additionally update a rototranslation matrix that aggregates all incremental updates of R and t of each iteration.

2.2 Improving Speed & Quality

Several adjustments can be made to the algorithm to improve its speed and matching quality. The adjustments discussed in this report can be categorized in matching functions ψ and methods to sample the point cloud data. The latter can be done in every iteration or once before the algorithm runs. The approaches discussed in the paper are described in the sections below.

2.2.1 Sampling

We compare the following sampling methods: No sampling, uniform sub-sampling, random sub-sampling in each iteration, multi-resolution sub-sampling and sub-sampling from high-density regions.

In uniform sub-sampling, the source cloud is uniformly sampled once at a specified ratio before the algorithm starts. In the random sub-sampling case, this sub-sampling happens in every iteration. In multi-resolution sub-sampling [9], the initial number of samples used is low, after which they are increased when a criterion is reached. In our case, this criterion is set to a ratio of the RMS value in the first iteration of the algorithm. The principle is that the matching increases in granularity from coarse to fine. For coarse matching, not as many samples are needed, which theoretically should increase the speed of the algorithm in earlier steps. In density sampling we refer to the given paper and implement it accordingly. We initially use a Kernel density function to obtain the points that are clustered/placed together based on density. We then score them based on this criteria, followed by giving set of n indices such that the points are returned based on value of n. We follow this method because this helps the matching algorithm to focus better on high volume data points rather than outliers.

2.2.2 kd-tree

We make use of kd-tree to speed up the matching process. A kd-tree consists of data where data in each node is a K-Dimensional point in space. In our experiments, we make use of $k = 1$ where we try to enforce matching points which have minimum k=1 number of neighbors. The kd-tree returns the indices of the points that satisfy this constraint and thus a better matching is performed.

We also observed that the RMS values converged faster and the execution time of the program was lesser than the execution time without kd-tree.

2.2.3 z-buffer

In order to be able to accelerate the ICP algorithm we are examining the use of z-buffer. Our implementation is based on [3], where the two cloud-points share a common referential, so it is possible to perform 2D image processing techniques on 3D data.

In more details, 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 assignment, the z-buffer referential will be the coordinate system of A_2. So for our implementation for the matching step of each iteration in ICP, we are following the next steps:

- Step 1: We stack the two cloud points in order to get their union.
- Step 2: For the set of the two cloud points we now find the minimum and the maximum coordinate in the x-axis and y-axis - the two first dimensions of the union described in step 1. Considering that the minimum bounding box is a rectangular, these four nodes define the bounding box.
- Step 3: We initialize the bounding box as a grid of size $H \times W$. In order to divide the bounding box in rectangular $H \times W$ cells, we calculate the dimensions of the bounding box as the maximum distance of the horizontal and vertical distances and divide them by the number of desired cells in horizontal and vertical axis of the grid.
- Step 4: We initialize two same buffers of size $H \times W \times 3$ in order to keep all the information of points of the source and target point clouds. For each point in each point-cloud, we check which cell of the grid is nearest based on the xy-coordinates. If we have more than one

combination, we keep the the point with the lowest z-coordinate. In that we ensure we will have a $1-1$ mapping. At the end of this step we will have the source and target buffer filled.

- Step 5: For each cell c at coordinate h, w in the source buffer, we find the closest match in a $N \times N$ neighborhood centered around coordinate h, w in the target buffer using mapping function ψ.

The mono z-buffer approach works well especially when the overlap between the two surfaces is sufficiently planar as it happens in case of the wave dummy data. In the case of strongly curved overlaps, experiments show that the matching algorithm does not converge properly, as is the case with the bunny dummy data, because the surfaces are not uniformly re-sampled in the twin z-buffer structure. To improve this approach the multi-z-buffer technique is proposed, which produces a more homogeneous re-sampling. In a single twin z-buffer structure, the direction of the projection is chosen by hand. In the multi-z-buffer approach these directions must be determined automatically, which required knowledge of the normal orientation of the object surface at each sampled point. The registration is still processed in two stages. First, the multi-twin z-buffers are optimized in order to efficiently define only the overlaps of the surfaces; this is obtained by clustering the normal directions through a Gauss sphere tessellation. Then the registration process is iterated while the correspondence problem is rapidly solved inside each individual twin z-buffer structure.

2.3 Comparisons

The algorithm and its variations were compared using the wave dummy data. We let ICP run for a maximum of 500 iterations with a convergence threshold of $10e^{-5}$. For sub-sampling methods that can use different sample rates, we used ratios $\{2, 4, 8, 16\}$. The variants were compared across 4 metrics: Speed, accuracy, stability and tolerance to noise. The speed was measured in seconds using the *time* library in Python on a standard range laptop. All remaining metrics were measured using RMS.

2.3.1 Speed

Figure 1 contains the speed measurements for each algorithm adaptation, including the baseline with no sub-sampling and Euclidean distance-based matching. The latter is denoted as BF, short for brute-force, in the figure. All sub-sampling methods perform better than the baseline in terms of speed, especially when combined with kd-tree matching. Overall, uniform sub-sampling and random-sub-sampling perform best. However, multi-resolution sampling performs comparably to uniform sub-sampling with BF matching. Multi-resolution sampling and density based sampling take a longer time to converge than others, however, they have other advantages that are discussed in the next section. Generally speaking, higher sampling ratios in uniform and random sub-sampling increase convergence speed, with the exception of random sub-sampling with BF matching. In the figure, the value for z-buffer matching was cropped to avoid skewing. In actuality, it took 1072 seconds to converge. It becomes clear why this is the case in the next section.

2.3.2 Accuracy

The accuracy for each method is displayed in figure 3. In the figure SR represents sampling rate. Uniform sub-sampling performs as expected. ICP benefits from more samples, assuming they are accurate. This is reflected by the diminishing accuracy when the sampling rate is increased. The baseline uses all samples, and because the wave data does not contain any noise, it performs the best (as with noisy data, it might attempt to match noisy samples to structured data points). Along with this, runs with less samples seem to converge quicker, as runs with higher sampling rates stop after fewer iterations. Random sub-sampling performs comparably across all sampling rates, but converge at slightly different times. They perform noticeably worse than the baseline algorithm with no sub-sampling. In contrast, density-based sampling performs extremely comparable to the baseline algorithm. This might be the case because the number of samples in the density-based approach adapts itself to the data. Since the wave-data is distributed relatively uniformly, most points from the original data are present in the sub-sampled version, as visualised in figure 2. Multi-resolution sampling perform comparable to the baseline as well. We observe that the RMS plateaus around 15 iterations. As per section 2.2.1, the criterion to increase the number of used samples was probably met. As more samples are included, this changes the RMS calculation quite abruptly. kd-tree matching

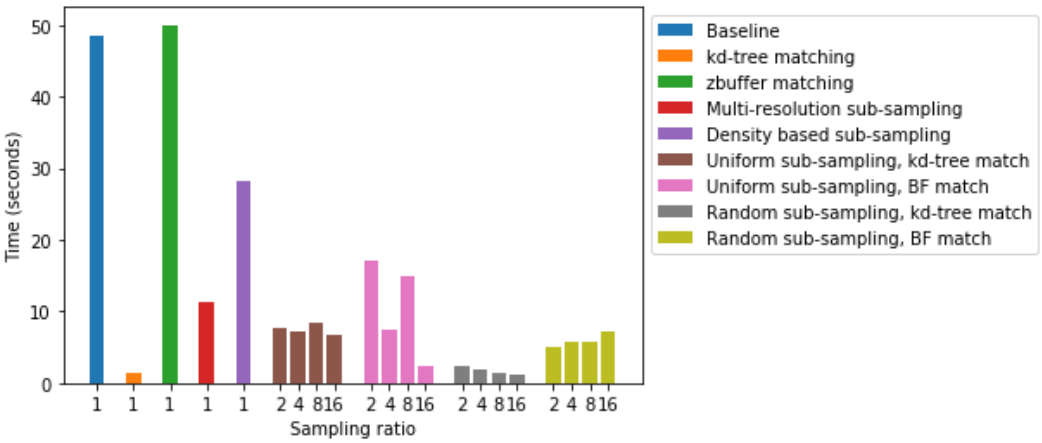

Figure 1: Time required to run algorithm in milliseconds for each sampling method across different sampling ratios.

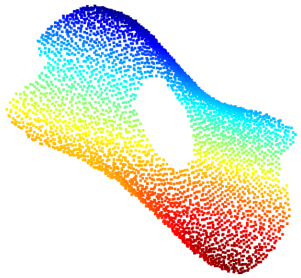

Figure 2: Wave point cloud with density-based sub-sampling

performs exactly the same as the baseline, which is to be expected, as the main optimization difference between kd-tree matching is in computational speed, as described in section 2.3.1. When z-buffer matching is used, the RMS decreases drastically in the first couple iterations, but the algorithm does not converge within 500 iterations. This explains why it took longer for this adaptation to finish running compared to other approaches.

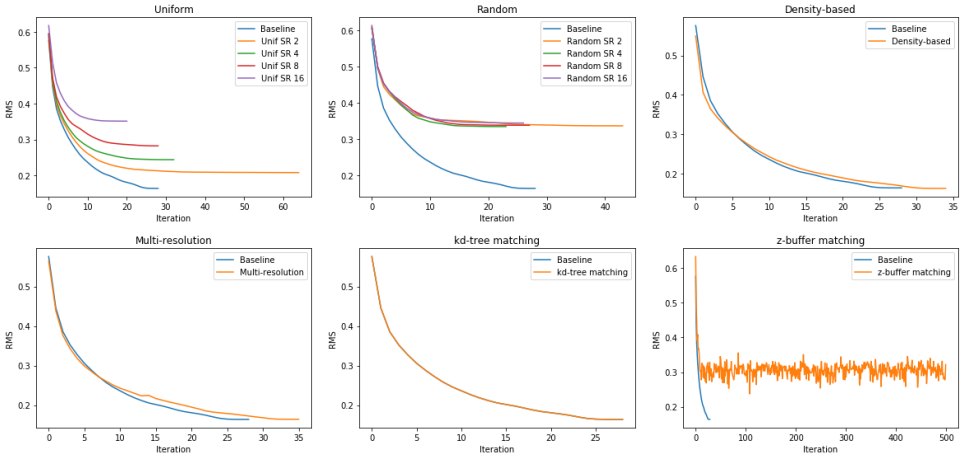

Figure 3: Convergence/accuracy over iterations for each alteration, expressed in RMS

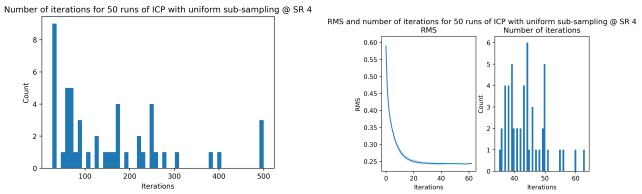

Figure 4: RMS and number of iterations for various ICP adaptations

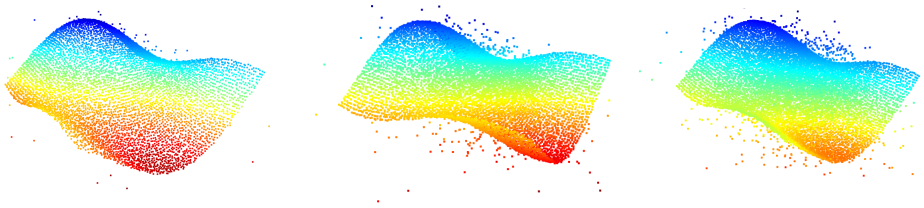

Figure 5: Visualization of wave data with 500, 2000 and 4000 noise samples

2.3.3 Stability

The stability was assessed by running each algorithm 50 times, with the exception of the z-buffer algorithm as it would take too long. z-buffer was run 10 times. Uniform and random sub-sampling were assessed with a sampling ration of 4. Every adaption, except for uniform and random sub-sampling, returned the same results for each run, indicating that they are stable. As none of these algorithms employ any randomness as part of their sampling and matching methods, this is to be expected. Uniform sub-sampling took a different amount of iterations to converge per run, but it performed similarly for each run in terms of RMS, with marginal amounts of variance. This also holds for random sub-sampling, however, the amount of iterations per run differed wildly, some even reaching the maximum amount of iterations. Both of these results are visualized in figure 4.

2.3.4 Tolerance to noise

To evaluate performance on noisy data, we add several amounts of noise (500, 2000 and 4000 samples) to the wave data with a variance of 0.5. The resulting data is visualised in figure 5. The performance of all adaptations is plotted in figure 6. Uniform sub-sampling performs similarly when using no noise, 500 noise samples and 2000 noise samples, and decreases in accuracy with 4000 noise samples. This is also the case for random sub-sampling. A curiosity for density-based matching is that it performs better than the baseline method when noise is introduced. This can be explained by the fact that high density regions are less likely to appear in the sparse noise than in the structured data. This sampling method thus effectively filters noise. Multi-resolution sub-sampling performs comparably to the baseline except when 2000 noise samples are introduced. This could be an anomaly in which data was selected, as there is a chance more noise samples are selected than samples part of the structured data. The variance of z-buffer matching remains very high, and still does not converge when using noise. kd-tree matching performs the same as the baseline method.

3 Global Registration

By calculating R and t for multiple pairs of point clouds using ICP, subjects can be reconstructed by stitching together the point clouds using the achieved rotation matrix and translation vector. In this report, this has been done for data of a man turning around in front of a Xbox Kinect sensor. We evaluate the performance of the algorithm for two different stitching methods and different frame step sizes. Due to noise present in the original data, samples with a distance further than 2 meters from the origin were removed.

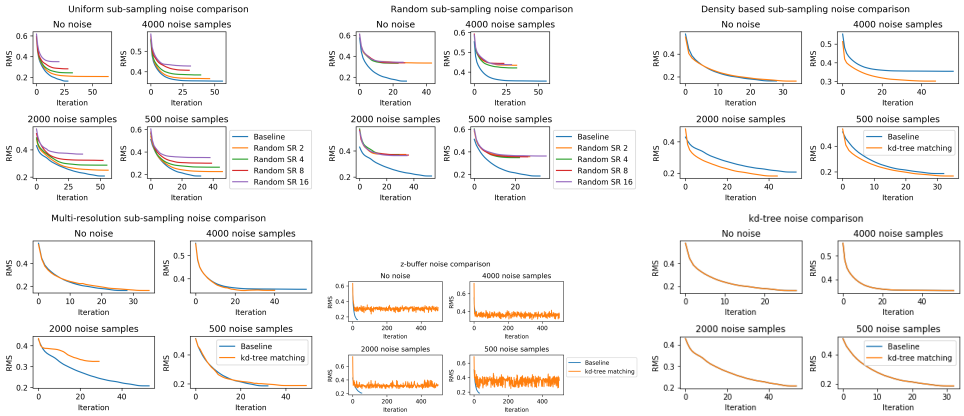

Figure 6: Performance of several algorithm with different amounts of noise

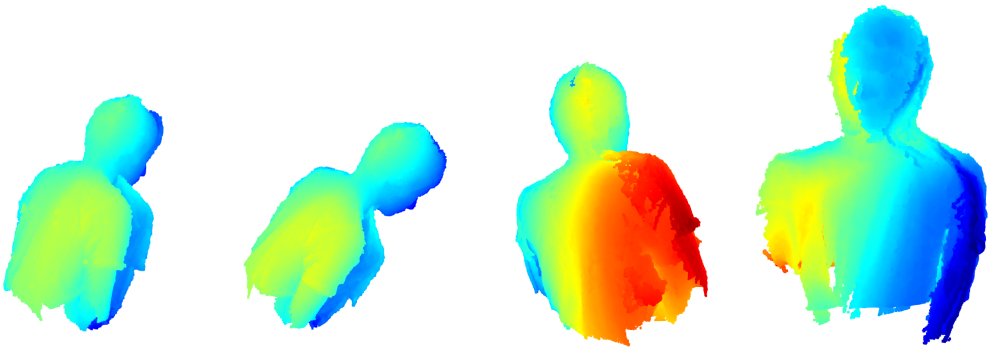

Figure 7: Images back of stitched model with 1, 2, 4 and 10 steps between frames for exercise 3.1

3.1 Estimate pose every N frames & merge at the end

In this approach, we compute R and t for every frame pair separately. After they are computed for each pair, we use them to stitch together the global image using frames and their corresponding R and t. In the results you can see that the quality of the stitching diminishes with larger step sizes. This is to be expected, as the data gathered by the sensor is quite noisy. For smaller steps, the point correspondences line up decently well, but for larger step sizes, there are not as many correspondences left. When combined with noise, this leads to subpar performance of the ICP algorithm. ‘ We observe from Fig. 7 that since we are skipping the intermediate frames the algorithm is not able to find all the matching points between the given two frames and merge all at the end.

3.2 Estimate pose and merge every N frames

For this section, we make use of different step sizes and we use every 4, 10th frame such that R,t values are updated accordingly. Additionally, we append the translated source to the target point cloud and use this to compare the next frame to. As such, the target is an aggregate of all previously translated frames. The small errors made in each step are exacerbated by a large amount and lead to a messy stitching result. This gets compounded by the fact that the target point cloud contains the errors of every previous frame, and thus next frames build upon an erroneous point cloud.

We observe from Fig. 8 where we make use of every 4th frame that since there is less information due to the fact that we only have every 4th and 10th frame as the input, the point clouds ware not merged effectively.

Similarly, for Fig. 9 where we take every 10th frame the visual results deteriorate even further when compared to previous figure where every 4th frame has been used.

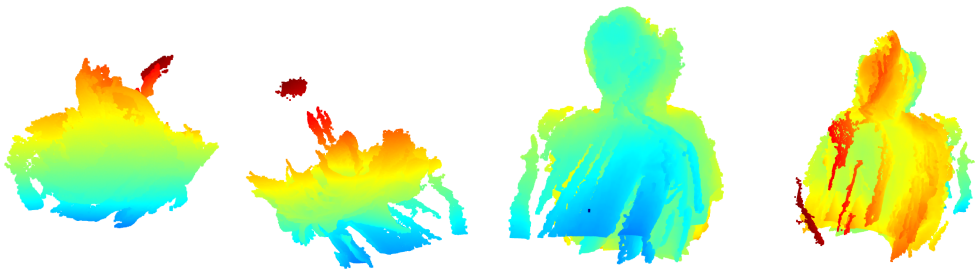

Figure 8: Back, bottom, front and top of stitched model with step size 4.

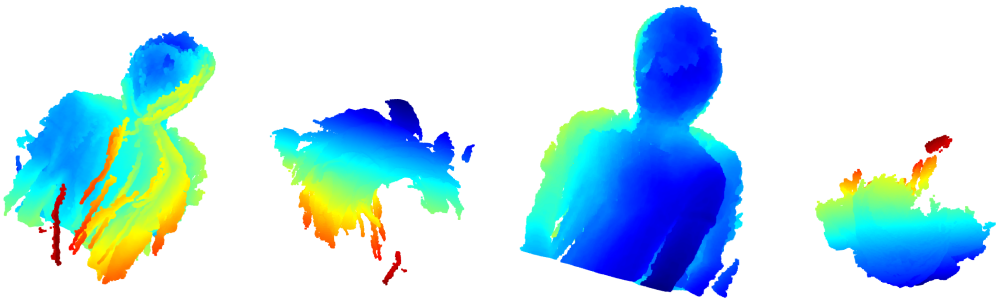

Figure 9: Back, bottom, front and top of stitched model with step size 10.

In addition, we plot the RMS values of the method as shown in Fig. 10in this section to observe closely how ICP reacts when frames are skipped by 4 or 10 times. As expected the RMS values increase slightly after 1 or 2 iterations.

4 Additional Questions

From Fig. 11 observe the RMS values for every algorithm/method that has been used in Section 2 to optimize ICP. It is evident from Fig. 11 that for uniform sampling since the data is now sampled in equal batch sizes the RMS values converges steady and smooth across iterations. For the case of random sampling we observe that the RMS values converges to better values than uniform possibly due to state of randomness. For multi resolution, the RMS values slope goes on decreasing and converges to a good value due to its multi resolution way of using the data after each iteration.
For informative sub-sampling we observe that the RMS values converge for very low values possibly because the informative regions alone are not enough for ICP. Since kd-tree improves the matching the RMS values for ICP with kd-tree converge to good values as expected.

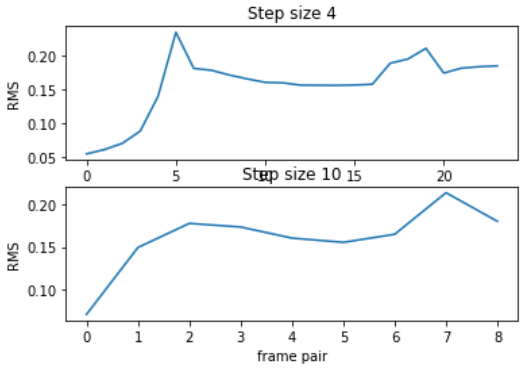

Figure 10: RMS values for every 4th frame and every 10th frame method.

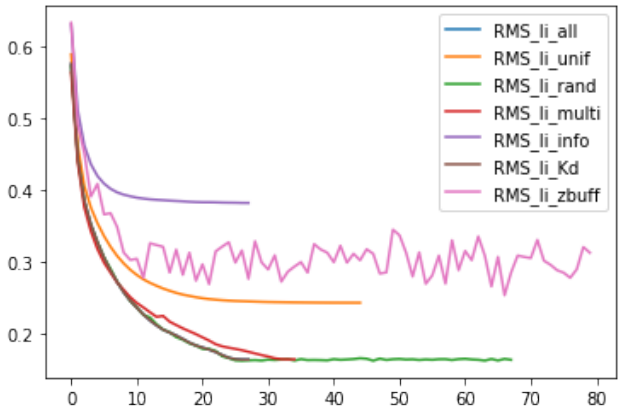

Figure 11: Comparison of the various algorithms that have been used so far with ICP implementation.

4.1 What are the drawbacks of the ICP algorithm? Please provide proof to back up your claims.

ICP suffers from one of the main drawbacks that gradient descent does as well. As an algorithm, it can assure you that it will converge in a minimum, but it cannot guarantee you that it is going to be the global minimum. Combined to that, the minimum that this algorithm finds differs widely with respect to the input. In other words, the algorithm is not robust enough and is seriously affected by outliers. Finally, while it converges to at least a local minimum, it does that not fast enough due to the slow point-matching technique.

4.2 How do you think the ICP algorithm can be improved, besides the techniques mentioned in [11], in terms of efficiency and accuracy?

The ICP algorithm can be improved in terms of efficiency and accuracy by adjusting some steps of the original version. Specifically,

- *Point selection*: The ICP algorithm needs to find the nearest point of each point in the current point set at the point of the other point set in each iteration, so the computation is complicated. This process can be accelerated by down-sampling the original point set [14]. Furthermore by down-sampling the outliers are also removed.
- *Finding corresponding points*: The ICP algorithm needs to find the nearest point from another point set as the corresponding point of the current point. By using the kd-tree data structure, projection, invariant feature search algorithm [12, 1, 8, 2] to effectively find the corresponding relationship between the two point sets, we can speed up the search process and improve the corresponding precision.
- *Point pair exclusion* The appropriate error points in the exclusion method can improve the point cloud data stitching accuracy and stability [7]. We can also use the gravity of center of the matching points that are obtained. This would enusre better matching.
- *Specifying error metrics function and minimizing errors* Specifying the appropriate error metric function can improve the accuracy of point cloud registration [5, 6]

5 Conclusion

In this report, we worked on the Iterative Closest Point algorithm and some variations on it to examine their effect on the accuracy, the speed, the stability, and the tolerance to noise. To note the effect on accuracy and speed, we performed various sampling and matching methods. We notice that different methods have different effects on accuracy, speed, or both. Therefore, our choice each time depends on whether we want to put more weight on the accuracy or the calculation time. In any case, however, there should be a balance between the two. Unsurprisingly, noise tolerance is better when accuracy is greater. Therefore, in real-world problems, we must also take this parameter into account.

Finally, stability's metric is the ability of the model to converge or not, which also depends on the sampling applied. Eventually, we apply the algorithm to both real and specifically successive frames of a person's rotation. We notice that also in this case the algorithm works quite well as we have a complete reconstruction of the human. But this is not the case when we cut out intermediate frames and the calls between the images are long.

6 Self-Evaluation

We shared the points equally and worked collaboratively on questions that we found difficult to implement individually.

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust pairwise surface registration. *ACM Trans. Graph.*, 27(3):1–10, aug 2008.

[2] K.-H. Bae. Evaluation of the convergence region of an automated registration method for 3d laser scanner point clouds. *Sensors*, 9(1):355–375, 2009.

[3] R. Benjemaa and F. Schmitt. Fast global registration of 3d sampled surfaces using a multi-z-buffer technique. In *Proceedings. International Conference on Recent Advances in 3-D Digital Imaging and Modeling (Cat. No.97TB100134)*, pages 113–120, 1997.

[4] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] J. Chen, X. Wu, M. Yu Wang, and X. Li. 3d shape modeling using a self-developed hand-held 3d laser scanner and an efficient ht-icp point cloud registration algorithm. *Optics Laser Technology*, 45:414–423, 2013.

[6] J. Dong, Y. Peng, S. Ying, and Z. Hu. Lietricp: An improvement of trimmed iterative closest point algorithm. *Neurocomputing*, 140:67–76, 2014.

[7] S. Du, J. Dong, G. Xu, B. Bi, and Z. Cai. An improvement of affine iterative closest point algorithm for partial registration. In *Proceedings of the International Conference on Internet Multimedia Computing and Service*, ICIMCS'16, page 72–75, New York, NY, USA, 2016. Association for Computing Machinery.

[8] X. Ge. Non-rigid registration of 3d point clouds under isometric deformation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 121:192–202, 2016.

[9] 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*, pages 540–543, 2002.

[10] S. Rusinkiewicz. A symmetric objective function for icp. *ACM Trans. Graph.*, 38(4), jul 2019.

[11] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[12] G. Sharp, S. Lee, and D. Wehe. Icp registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102, 2002.

[13] O. Sorkine-Hornung and M. Rabinovich. Least-squares rigid motion using svd. *Computing*, 1(1):1–5, 2017.

[14] Y. Wu, W. Wang, K. Lu, Y. Wei, and Z. Chen. A new method for registration of 3d point sets with low overlapping ratios. *Procedia CIRP*, 27:202–206, 2015. 13th CIRP conference on Computer Aided Tolerancing.