Computer Vision 2 assignment 1

Emma Quist 11706724, Haohui Zhang 14124262, Faye Raaijmakers 11848669

April 22, 2022

1 Introduction

The Iterative Closest Point algorithm was firstly proposed by Besl and McKay (1992). The ICP algorithm was originally described as a novel method to accurately register 3-D shapes in a computationally efficient way. Applied to alligning two different point clouds (A1 = source, A2 = target), the pseudo algorithm looks as follows:

1. Initialize rotation matrix R = I (identity matrix) and translation vector t = 0.
2. Transform A1 using the current values of R and t.
3. For each point in A1, find the corresponding closest point in A2 (using brute force).
4. Solve R and t using Singular Value Decomposition.
5. Repeat this process until convergence.

The ICP algorithm is rather effective. However, as can be derived from the above pseudo-algorithm, the algorithm lacks in efficiency. As the closest points are found through brute force, the ICP algorithm runs in $O(N^2)$ which implies long processing time for larger point clouds. In order to reduce the time complexity, several adaptations on the original algorithm have been proposed. In this report, we will implement the ICP algorithm and its adaptations, and compare the the results in terms of accuracy, speed and stability.

2 Iterative closest point

First of all we implemented the regular ICP, as described in the introduction. The source point cloud we used, contained of 40256 data-points. Each iteration of the algorithm took about 2 minutes and 15 seconds. The amount of iterations it took for the algorithm to converge depended on the defined stop criterion. In our implementation, we calculated the error for each iteration. The error was defined as the euclidean distance between the source cloud and the target cloud. Once the difference between error in consecutive epochs was lower than 3e-6, we accepted our algorithm to be converged. The final error in this situation was 0.12. A visualisation of the results is shown in figure 1, in which the target points are yellow, the original source points are pink, and the transformed source points (result of the algorithm) are blue.

Figure 1: Regular ICP results

3 Improving Speed & Quality

3.1 Sampling

As mentioned in the introduction, the ICP lacks efficiency due its brute force approach in closest point matching. This was also proven by the processing time as mentioned in the previous paragraph. In order to increase this efficiency, several sampling methods have been proposed. As we now only have to brute force match a sampled data-set, the processing time decreases. The next paragraph will go over four different implemented sampling methods. All sampling methods were implemented within the regular ICP, meaning we used the same error, convergence criterion and visualisation technique.

The first method used uniform sub-sampling. Instead of brute force matching all data-points, we sub-sample at the beginning of the algorithm. The chosen sample size in this implementation was set to be 6000. Sampling was done uniformly, so that the data-points are evenly distributed across the source and target. The uniform sub-sampling reduced the processing time per epoch to 3 seconds, which is significantly less than the regular ICP. As we uniformly sampled, the convergence rate depended on the initial sample and thus was not consistent. The final error for one run was 0.09. Even though the errors differed per run, all errors were of the same order of magnitude. A visualisation of the results is shown in figure 2

The second method implemented random sub-sampling in each iteration. Whereas the previous described method used the same sample during all iterations, this method sampled new points for each iteration. In order to keep track of the progress of the algorithm (thus keep track of the roto-translation matrix), the source was transformed by the current roto-translation matrix before sampling. Again, the sample size had been set to 6000. The time per epoch was 3.3 seconds, which is about the same as the uniform sub-sampling. This makes sense as the amount of data-points that have to be matched are equal. Again, the rate of convergence was not consistent, even less than for the sub-sampling method. This is because now each new sample influences the rate of convergence. The final error for one specific run was 0.08. A visualisation of the results is shown in figure 3.

Figure 2: Uniform sub-sampling ICP results

Figure 3: Random sub-sampling ICP results

The third sampling method was called the multi-resolution sub-sampling, based on Jost and Hugli (2002). The key idea of their sampling method was a coarse to fine resolution approach. In other words; we take a very small sample size (100 datapoints) and iterate until defined stop criterion. The stop criterion was for the error to decrease less than 1e-2. Once the stop criterion is met, we increase the sample size by factor N (N=4). The processing time of one epoch was not consistent, as this of course depends on the current sample size. The final error was 0.08 and a visualisation of the result is shown in figure 4.

Figure 4: Multi-resolution sub-sampling ICP results

The final sampling method, sampled over informative regions. In other words: instead of considering the entire point cloud to sample from, only informative regions were sampled. We decided on informative regions by classifying the keypoints in the point clouds. 450 keypoints were found using the Open3D library. As the amount of keypoints is significantly less than in the other sampling methods, the processing time per epoch was also significantly lower (¿1 sec). A visualisation of the results is shown in figure 5. The final error was 0.12, which is higher than the previously mentioned sampling methods. However, the results are very impressing as the processing time was very obviously the smallest and it still delivered high accuracy. In addition to this, this sampling method was the only method to converge within 12 epochs, whereas the others did not meet the convergence criterion within 40 epochs.

Figure 5: Information region sampling ICP results

3.2 Matching methods

As described in the introduction, for each point in the source cloud, the ICP algorithm tries to find the closest point in the target cloud using brute force. The following section will discuss two other implementations that can be used to match closest points.

The first method uses a k-d-tree. A k-d-tree is a binary search tree. Each node in the tree represents a partition of the k-dimensional space (Greenspan and Yurick, 2003). In other words, the k-d-tree is a data structure for storage of information (Bentley, 1975). The tree can be accessed by queries, and will in this case return the closest points in the target cloud. The k-d-tree was implemented using the scipy library. The k-d-tree decreased the processing time per epoch significantly. For example, one epoch on the full dataset using brute-force took about 2 minutes and 15 seconds, whereas it took the k-d-tree about 1.08 second. This means that combining the k-d-tree with a sampling method would be extra efficient. Using k-d-tree in combination with the informative region sampling method, resulted in a processing time of 0.01 seconds per epoch, and no increase of error.

4 Global Registration

4.1 Merging at the end

We implemented an algorithm which was able to construct a 3d model based on 100 overlapping point clouds. We did so by calculating the rotation matrix (R) and translation vector (t) between every N of the given frames, where $N \in \{1, 2, 4, 10\}$. As the provided data-set was not synthetical, the data-set contained a lot of noise. Therefore, before applying the ICP algorithm, the data was cleaned. We removed the data-points that had a depth 2. The R and t values between the frames were found using the ICP algorithm. The values for each of the frames were saved and then reverse applied to the first frame to achieve the total transformation. Tabel 1 shows the time it took for the different sampling methods to retrieve all the R and t values for N = 1, using kdtree.

The second method we implemented is called the z-buffer. The z-buffer has the advantage of being relatively simple and intuitive. In addition to this, the order in which objects appear on the screen is not of importance anymore, which is beneficial for hardware implementations. Disadvantages of the z-buffer implementation is that it requires a lot of memory, which makes it less efficient. In addition to this, the correlation and continuity of the point clouds are unused, which of course should be of big importance. When implementing, we found out that the ICP algorithm with z-buffer adaption was really slow, only with 40 iterations, 20 x20 rectangular cells, 5x5 selecting window and using the fastest informative regions subsampling method still need more than one and a half minutes. If using other sampling method like uniform or random, it will take more than 15 minutes when testing the bunny data. Besides, it is important to note that this could be influenced by our own implementation. Our implementation included a lot of for-loops, which could most likely be vectorized.

Figure 6: Visualisation for uniform sampling with N=1

Sample method	Time
Informative region	3 min
Multi-resolution	60 min
Uniform	7 min
Random	6 min 20 seconds

Table 1: Calculation time different sampling methods on N=1.

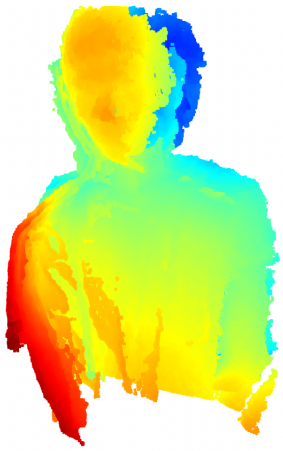

Figure 7: Visualisation for uniform sampling with N=10

What we can derive from 1, the multi-resolution sampling took very long to run, implying low efficiency. On the contrast, random- and uniform sampling are about 10 times faster. The fastest sampling method is informative region, however the performance of this subsampling method is much worse than other three. The obtained results looked very similar on the eye, thus not implying significant differences in accuracy. Therefore, it follows that the random- and uniform sampling are favoured as the processing time is way more efficient.

When increasing N, we see that the processing time for all sampling methods decrease, as less translations have to be retrieved. However, increasing N also decreases accuracy drastically for all methods. For N=10, the final model is not recognizable as a person. Visualisations are shown in figure 6 and 7, where a clear difference is visible.

4.2 Intermediate merging

Following, we implemented a very similar method to construct a 3d model based on 100 overlapping point clouds. However, in contrast to the previous method, we now apply the retrieved transformation on the image after every frame (note: in the previous method, we applied the full transformation at the end). For N=4, the random, uniform and informative region sampling improved their accuracy with respect to the previous method. Intermediate merging also results in higher accuracy when increasing N for all sampling methods. Especially for N=10 this stands out. The time inefficiency for informative region- and multi-resolution sampling still exist in this method. Visualisations for the uniform sampling for N=1 and N=10 are shown in figure 8 and 9.

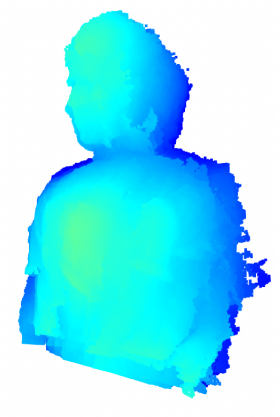

Figure 8: Visualisation for uniform sampling with N=1

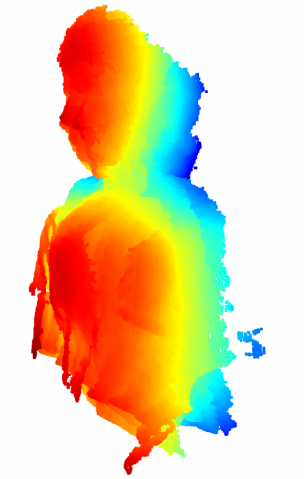

Figure 9: Visualisation for uniform sampling with N=10

As can be observed when comparing figure 6, 9, 7 and 8, we see that the second method produces best results. Especially for N=10, we see a major increase in the result. However, it is hard to visualize a 3d object in a 2d image, so for further inspection, we added the point cloud files to our submission as well.

5 Additional Questions

5.1 Drawbacks of the ICP

This report has mentioned a few drawbacks of the ICP algorithm, and will summarize them in the upcoming paragraph. First of all, the processing time of the regular algorithm is very poor. The regular algorithm runs in $O(n^2)$, which is very inefficient once working with a big data-set. The experiment as executed in section 2 proved this: one epoch took over two minutes to run. However, several methods have been proposed to overcome this problem, among which sample methods. The drawbacks of the sampling-including-ICP algorithm is that the final result is dependent on the initial samples. One can imagine, especially when sampling non-uniform, that the distribution of the data-points influences the final result.

Another drawback of the algorithm, which has not been tackled in this report, is the dependence on sufficient mutual overlap of the two point clouds (Salhi et al., 2019). If the initial alignment of the point clouds are very deviating, finding the closest points will result in incorrect closest point matching, in magnitudes that will alter the entire course of the algorithm. In addition to this, the ICP algorithm

requires a somehow evenly distributed point cloud. It follows logically that if point density in certain areas is way higher than in other areas, the algorithm will not be able to correctly track down the closest points in the target.

5.2 Improvement of the ICP

Many different techniques have been proposed to improve the ICP algorithm in terms of efficiency and accuracy, besides the methods already described in this research. Two of which will be discussed in the following section.

Firstly, the efficiency of the ICP algorithm can be improved by using a weighted ICP algorithm. Where matched closest point are weighted differently according to their significance. Meaning that matched points are assigned a higher weight when laying in more interesting regions and lower when laying in less important regions. These regions could for for example be obtained by using SIFT features, as proposed by Shin and Ho (2016). An other method is proposed by Rusinkiewicz and Levoy (2001), they obtain the weights by subtracting the distance between a matched point pair divided by the maximum distance of all matched pairs from 1. Which results in a higher weighted matches that lay close to each other, i.e. a weight close to one, and lower weights (close to 0) for points that lay far away.

Secondly, also proposed by Rusinkiewicz and Levoy (2001), outlier detection could be a way to improve accuracy. Their best performing implementation involves, rejecting a fixed percentage of matches that are classified as outliers, meaning that those points will not be taken in consideration when calculating the rotation and translation matrix.

6 Conclusion

This report focused on the implementation of the ICP algorithm. We found that the ICP algorithm is rather effective, but lacks in efficiency. Several sampling methods have been intertwined within the algorithm to decrease processing time. Sampling methods did increase efficiency while keeping the accuracy in tact. In addition to this, we also implemented a k-d-tree in the ICP algorithm, to increase the process of matching closest points. We found that the k-d-tree significantly lowered processing time, especially when used in combination with one of the sampling methods.

The second part of the report focused on using the ICP algorithm in order to construct a 3d model based on 100 overlapping points. Again it stood out that sampling increased efficiency, however not all sampling methods performed well. The informative region- and multi-resolution sampling took longer than an hour, while uniform-and random sampling finished within 7 minutes while delivering the same accuracy. In addition to this, we also found that translating the frame after every retrieved translation severely increases the accuracy, with respect to translating the frame at the very end.

All in all, the ICP has proven to be an effective algorithm and many adaptions have been done to improve both efficiency and accuracy.

7 Self-evaluation

For this assignment, Emma and Faye took care of writing the code covering the first and second part of the assignment ('ICP implementation' and 'Improving Speed & Quality'). Haohui coded the z-buffer and the third part of the assignment (Global Registration).

References

Bentley, J. L. (1975), 'Multidimensional binary search trees used for associative searching', *Communications of the ACM* **18**(9), 509–517.

Besl, P. J. and McKay, N. D. (1992), Method for registration of 3-d shapes, *in* 'Sensor fusion IV: control paradigms and data structures', Vol. 1611, Spie, pp. 586–606.

Greenspan, M. and Yurick, M. (2003), Approximate kd tree search for efficient icp, *in* 'Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.', IEEE, pp. 442–448.

Jost, T. and Hugli, H. (2002), A multi-resolution scheme icp algorithm for fast shape registration, *in* 'Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission', IEEE, pp. 540–543.

Rusinkiewicz, S. and Levoy, M. (2001), Efficient variants of the icp algorithm, *in* 'Proceedings third international conference on 3-D digital imaging and modeling', IEEE, pp. 145–152.

Salhi, I., Poreba, M., Piriou, E., Gouet-Brunet, V. and Ojail, M. (2019), Multimodal localization for embedded systems: A survey, *in* 'Multimodal Scene Understanding', Elsevier, pp. 199–278.

Shin, D.-W. and Ho, Y.-S. (2016), Iterative closest points method based on photometric weight for 3d object reconstruction, *in* '2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)', IEEE, pp. 1–4.