Computer Vision 2 - Assignment 1
Iterative Closest Point - ICP

Yke Rusticus
yke.rusticus@student.uva.nl
11306386

Ewoud Vermeij
ewoud.vermeij@student.uva.nl
11348860

April 18, 2020

1 Introduction

The first assignment is mainly focused on the Iterative Closest Point (ICP) algorithm first introduced by Besl and Mckay Besl and McKay (1992). This method iteratively tries to find a spatial transformation that minimizes the distance between two point clouds. There are different ways to improve the accuracy and efficiency of ICP, which will be discussed in the methods section in this work.

We have been given a dataset consisting of 100 images, with corresponding data files containing the point clouds, in which a person slightly turns in every successive image. All images combined represent a person turning 360 degrees.

In the experiment section we will attempt to merge the point clouds of all images to create a 3 dimensional visualization of the person in all images. In the conclusion we will discuss ICP, its drawbacks and how it can be improved.

2 Methods

For the ICP algorithm, we use Root Mean Square Error (RMS) to minimize the distance between two point clouds. RMS is defined as follows:

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

We need to find the rotation matrix R and translation vector t which minimizes:

$$min_{\psi: A_1 \rightarrow A_2, t \in \mathbb{R}^d} \sum_{a \in A_1, b \in A_2} ||Ra - t - \psi(b)||^2 \tag{2}$$

where ψ creates one-to-one correspondence between the elements of A_1 and A_2

2.1 Implementation

For ICP, various point sampling techniques will be analyzed to address the accuracy and efficiency of the ICP algorithm. These sampling techniques are:

- All points

- Random sub-sampling in each iteration
- Uniform sub-sampling
- Sub-sampling more from informative regions

For all sampling techniques we analyze the accuracy, speed, stability and tolerance to noise. Implementation details for each sampling method are given in subsections below.

Our ICP algorithm will is based on the steps provided by the assignment, with some adjustments. ICP will start with sampling the base and target point cloud by specified sampling technique. Then each iteration the following is applied:

1. If sampling technique is random, take a new sample.
2. Find the closest point in the target point cloud for each point in the base point cloud.
3. Derive optimal rotation R and translation t using SVD (Singular Value Decomposition, see Sorkine (2009)) to merge the base to the target.
4. Transform the sampled base cloud with R and t.
5. Calculate RMS based on the sampled points and stop if RMS is unchanged (change $<$ threshold).

When the iteration stops, the accumulated rotation and translation transformations are stored into a final rotation R_f and translation t_f. These values are returned by the algorithm, as well as the final transformed base point cloud. Our ICP function also has an evaluation flag, that will calculate the final RMS based on all points of the new base and target, instead of the sampled points. This way we can analyze the RMS of different sampling techniques more accurate.

In the next section (following the subsections) we will analyze the results of the different sampling techniques and will conduct various experiments with the provide images and their point clouds.

2.2 Sampling implementation: 'all'

Sampling all points is straightforward. In this case we take all points from both point clouds into account when running ICP.

2.3 Sampling implementation: 'random'

In the random setting, we sample a new subset of a given size from both point clouds at each iteration. These subsets will be used to estimate the optimal transformation parameters. Due to random nature of this method, we expect to see instabilities in the results, when using this for ICP.

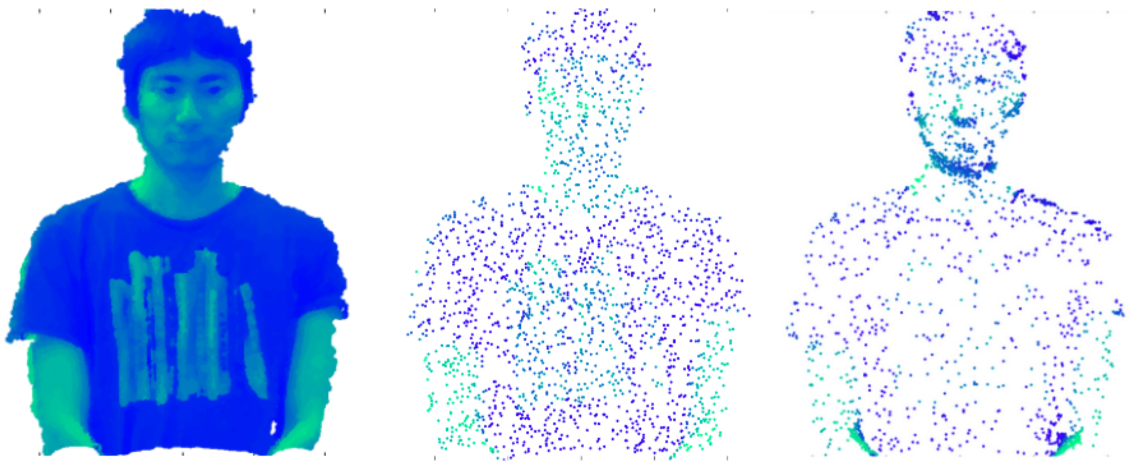


Figure 1: Left: all points in the point cloud (point cloud index 0). Middle: 2000 randomly sampled points. Right: 2000 points sampled from more informative regions.

2.4 Sampling implementation: 'uniform'

A random subset of each point cloud is also taken in this case. However, we only do this once at the start. The initial subsets are used to run the full ICP algorithm.

2.5 Sampling implementation: 'informed'

This method samples once at the start subsets from both point clouds. The aim of this method is to sample these subsets from more informative regions, rather than sampling fully random points. The way this is done is as follows (similar to Rusinkiewicz and Levoy (2001)): Consider having point cloud A, with corresponding surface normals N. Each $n \in N$ is a unit vector in three dimensions. We aim to group the normals together in bins, to be able to sample as uniformly as possible over the distribution of normals. In order to do so, we divide a cube bounding the unit sphere into equal sized blocks, or bins, and only consider the bins in which normal vectors fall. We then sample an equal number of points from each bin (corresponding to the surface normals). Some bins might have fewer points in them than the number of points we sample from each bin. Therefore, all "leftover" samples are taken fully random from the point cloud. Figure 1 shows the difference between random sampling and informed sampling for 2000 points, in comparison with all points in the point cloud. This figure shows the informed sampling method samples more points around small detailed regions such as the nose, eyes and chin, compared to fully random sampling.

3 Experiments

All experiments were conducted with the following parameters: an RMS change threshold of 0.0001; and 5000 sampling points, if not all points are selected.

Sampling	RMS - mean	RMS - std	Time - mean	Time - std
All	1.638e-3	~0	384	98
Random	1.71e-3	0.07e-3	0.8	0.5
Uniform	1.65e-3	0.02e-3	1.1	0.1
Informed	1.638e-3	0.007e-3	0.947	0.004

Table 1: ICP efficiency and accuracy, merging distance: 2 consecutive frames. Time is measured in seconds.

3.1 Efficiency and accuracy

To analyze the difference in accuracy, speed, stability and tolerance to noise we ran each ICP sampling algorithm 3 times with 2 consecutive frames of our dataset. The results are shown in table 1. It is shown that, particularly among RMS scores, the differences are very modest. Looking at the RMS mean and standard deviation, we can see that informed performs best on average and that sampling method "all" is most stable. This last point is expected, as each run using all points is the same. Compared to informed sampling, using all points performs equally well regarding mean RMS, but is much slower than informed sampling. Looking at the random sampling method, taking a new sample every iteration, we can conclude that it is still relatively tolerant to noise. Although, we do notice more standard deviation in RMS for random sampling, compared to the other methods. In terms of speed, taking all points is out of the question. Uniform, random and informed have a slight difference in average speed, but informed shows clearly most stable times. Given the results, informed sampling is most stable and accurate and therefore used for further experiments. One could be surprised by the high standard deviation in time for sampling method "all". This result is however likely explained by the fact that multiple programs (e.g. browser) were running at the time measuring. Sometimes these programs require more computing power and sometimes less, such that the execution times of our experiment varied as a result.

In order to compare accuracy across sampling techniques more generally, we ran the ICP-algorithm several times for different merging distances. Following the indices given in the data set, we compared the accuracy of merging point cloud 0 to 1, 0 to 2, 0 to 4, and 0 to 10, for each sampling method. The results are visualized in Figure 2. This figure shows that each sampling method shows similar results. Each method seems to struggle more with larger merging distances, which is not unexpected. For large merging distances, this figure suggests that informed and random sampling struggle slightly more than uniform sampling and taking all points into account.

We also compared the average number of iterations needed till convergence for each sampling method for varying merging distance. The results are shown in figure 3. We see that informed sampling takes on average less iterations to converge, compared to the others. Random sampling shows a sharp peak, which is again an example of the instability of this method.

Lastly, we show the RMS decrease over the number of iterations per sampling method

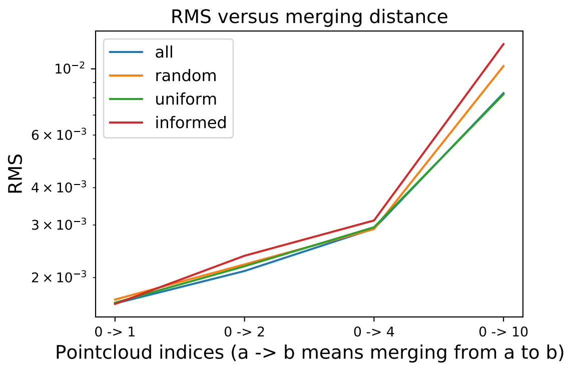


Figure 2: RMS as function of merging distance evaluated on all points in the point clouds, plotted for different sampling methods. Results are averaged over three runs.

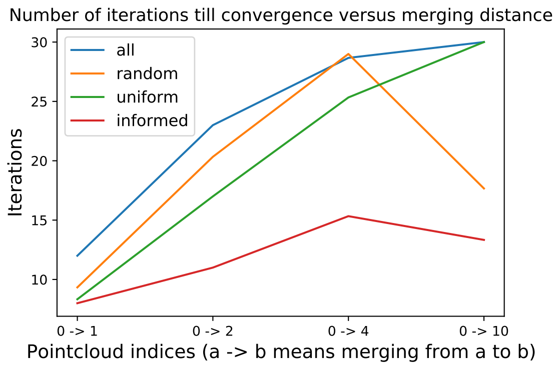


Figure 3: Iterations needed till convergence for the different sampling methods, as function of merging distance. The cut-off was at 30 iterations (so values that show 30 iterations could in fact be larger). Results are averaged over three runs.

for a single run in Figure 4. In this case we see again the relatively quick convergence for informed sampling. Random sampling took the maximum number of iterations (cut-off at 30), probably because it kept fluctuating significantly enough to not fall under the RMS change threshold. This shows that using this method requires a more sophisticated way of measuring convergence, instead of only taking the previous RMS value into account. The RMS value at which each method converges does not say much in this figure, as the RMS was measured on the sampled point clouds, which were different in each case.

3.2 Merging scenes

For our first merging scenes experiment, we will perform the following algorithm on the the 100 point clouds:

1. Take the first frame (A) as reference point
2. Take the next frame (B) and perform ICP to get R and t, transforming the next frame to the previous (transformed) frame.
3. Transform (B) towards A with R and t.

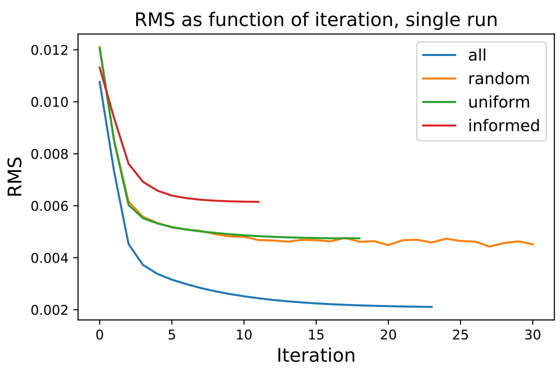


Figure 4: Convergence of the ICP-algorithm for the different sampling methods for a single run of merging point cloud 0 to 2. RMS was evaluated on the sampled point clouds, so these values might differ from RMS evaluated on all points in the point clouds.

4. Go to point 2 with the next frame (C). Do this until all frames have been processed.

3.2.1 Transform and merge

Figure 5 shows that the the merge of all frames are in some situations ”off” of the original reference point. This is probably caused at some run of ICP iterations where a frame got ”caught” in a local minimum which happens to be very far away from the optimal minimum in terms of RMS. When using informed sampling, the merging scene algorithm shows the same error every run, meaning that some frame has a certain starting position that partially causes it.

Instead of taking every successive point cloud, we also ran the merging algorithm taking every second, fourth and tenth frame/point cloud. The results in figure 6 are different compared to figure 5, but show a similar problem. Taking every fourth or tenth point cloud means a larger difference in the angle of the turning person between the point clouds, making it rather difficult than easy to find the perfect R and t. We never expected skipping point clouds to perform better than taking consecutive point clouds.

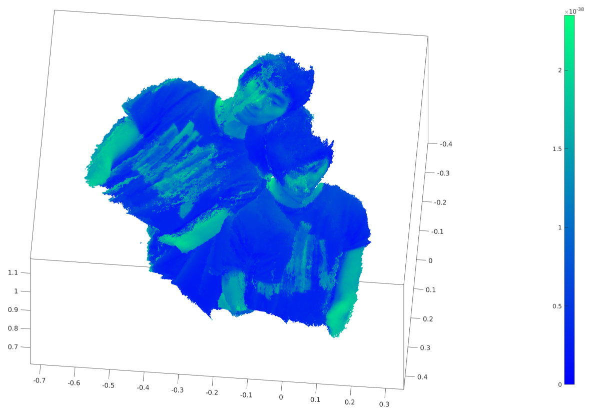
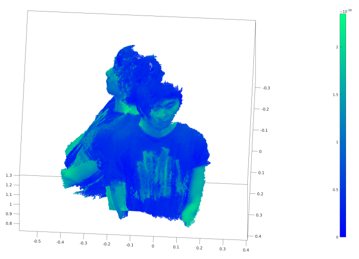
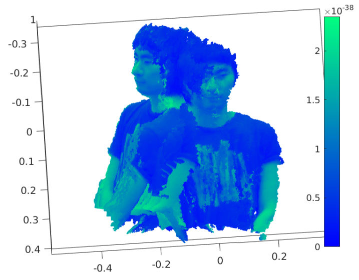


Figure 5: Merging scences: taking each consecutive frame.

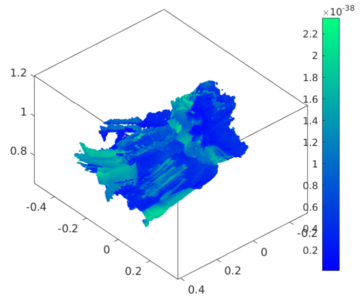
3.1b



(a) Every 2nd

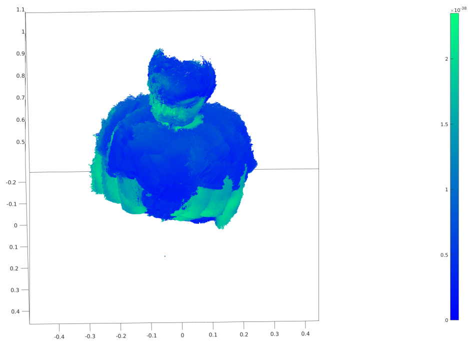


(b) Every 4th

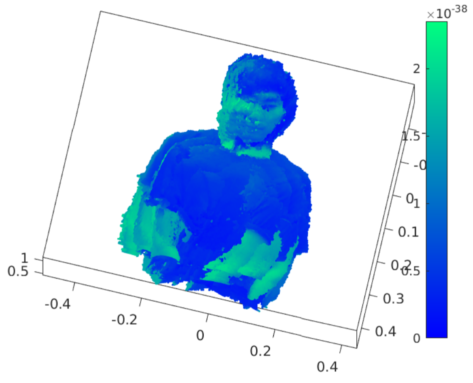


(c) Every 10th

Figure 6: Merging every 2nd, 4th and 10th frame.



(a) Angle 1



(b) Angle 2

Figure 7: Merg and add.

3.2.2 Adding merged frames to base

Figure 7a and figure 7b show the results of our last scene merging algorithm. The algorithm is similar to the one used before, with one additional step:

1. Take the first frame (A) as reference point
2. Take the next frame (B) and perform ICP to get R and t, transforming the next frame to the previous (transformed) frame.
3. Transform (B) towards A with R and t.
4. **Merge the transformed frame (B) with the previous frame (A)**
5. Go to point 2 with the next frame (C). Do this until all frames have been processed.

Instead of taking the previous transformed frame as a target for the next frame, we merge every processed frame into the base such that, for every next frame, the base grows bigger and bigger. When performing ICP, the next new frame is less likely to be transformed far off from the current point cloud, since the whole current point cloud will be considered. The difference is clearly visible when looking at figure 5, 6 and 7. In figure 7 all point clouds are somewhat merged into the shape of one human being. Figure 5, 6 show that that one bad transformed frame can hurt the next, having more than one human shaped point clouds.

4 Conclusion

The ICP algorithm is a relatively simple and straight forward. When two point clouds are similar in shape and size (scale), it is very capable of finding the rotation and translation parameters R and t to transform one point cloud into the other. However, ICP also has its drawbacks. The first drawback can be found in the algorithm itself. Each iteration it searches for the closest point in cloud B for each point in cloud A and rotates cloud A towards those points in B with the retrieved parameters R and t. Since there are no restrictions on retrieving the closest point, it could well be that the to be transformed point cloud gets stuck in a local minimum in terms of RMS.

Another drawback of ICP, as used in this work, is that small errors could accumulate in merging large number of point clouds, resulting in unrealistic 3-dimensional reconstructions. This occurs if each time a new point cloud is added to scene, the merging only depends on the previously merged point cloud. Errors increasingly accumulate this way, similar to in the Chinese Whispers game[1]. However, whether or not this was the reason why our results turned out as presented is uncertain.

[1] `https://en.wikipedia.org/wiki/Chinese_whispers`

The ICP algorithm could possibly show better results if point distances are explicitly taken into account. We could set a threshold, such that we only take points (from different point clouds) into account that lie closer to each other than the given threshold. The motivation for this is that far-away points are less likely to be corresponding points than points that have a smaller distance to each other. The desired effect is that only corresponding points are taken into account, and that as a result less errors occur that could accumulate over large merging operations.

References

Besl, P., & McKay, N. D. (1992). A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence*(14), 239–256.

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

Sorkine, O. (2009). Least-squares rigid motion using svd. *Technical notes*, *120*(3), 52.

A Team member contributions

Ewoud Vermeij:

- Implementation of basis ICP algorithm.
- Implementation of ICP analytics demo.
- Implementation of half the visualizations.
- 50% of report.

Yke Rusticus:

- Extension of ICP algorithm to all sampling methods.
- Fine tuning ICP, evaluations RMS.
- Implementation of half the visualizations.
- 50% of report.