Computer Vision 2 - Assignment 1

Bob Borsboom (10802975), Louis van Zutphen (11851910), Reinier Bekkenutte (13438557)

April 19, 2021

Link to Github: https://github.com/RBekkenutte/CV2

Introduction

In this assignment the Iterative Closest Point (ICP) algorithm is implemented and performed on two different datasets. The first dataset, the toy data, is a plane in 3D space (a point cloud). The planes are created using a mathematical function. The second dataset is a video of a person who is rotating. We have point clouds from this person where there is a small difference in time between the point clouds. With the camera pose estimation from the ICP algorithm we tried to merge the frames together and try to create a 3D reconstruction.

2 ICP

2.1 ICP implementation

The ICP algorithm is implemented following the steps described in the assignment. The following point selection techniques are implemented:

1. All the points
2. Uniform sub-sampling
3. Random sub-sampling in each iteration
4. Informative regions (K-means)

First, the differences between the point selection techniques are described below.

2.1.2 Point selection technique (all the points)

The point selection technique for computing the distances between points in the point clouds is calculated as follows. Loop over all the points in the source point cloud. Per point check which point in the target point cloud is the closest (euclidean distance), whereby all the points in the target point cloud are available.

2.1.3 Point selection technique (uniform sub-sampling)

The point selection technique for computing the distances between points in the point clouds is calculated as follows. Not all the points from the target point cloud are available. Before starting the ICP algorithm a set percentage of the points are uniformly sampled from the source point cloud. For the results, We choose to run it with either 10 or 30 percent of the data. Then during all the iterations of the ICP algorithm this same subset of the source point cloud is used

2.1.4 Point selection techniques (random sub-sampling in each iteration)

The difference between this method and unifom sub-sampling is that for every iteration during the ICP algorithm you select different points from the ICP algorithm. Namely, every iteration you uniformly select a given amount of the source point cloud (independent of the points which have been selected before). We choose to run it with 10 or 30 percent.

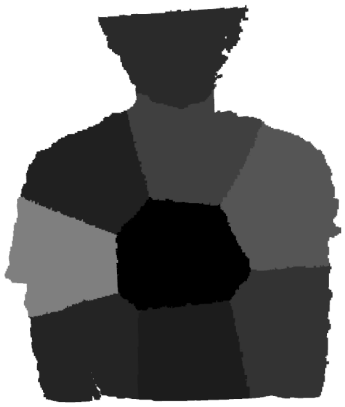
2.1.5 Point selection techniques (from informative regions (K-means))

An attempt was made at using K-means in order to sample from more informative regions. The main idea is to split up the target image uniformly in order to reduce the search space. K-means is used to find a configurable number of clusters in the point cloud and split the target image into multiple *bins*. A bin is then assigned to each point from the source point cloud and only the target points of the corresponding bin are compared to find the closest point. This reduces the search space significantly as only a subset of the target is searched through, and should not increase computation time too much as the clusters only need to be found once. The difference between this subset and a subset for one of the random methods is that here the subset encompasses a specific region of the target image instead of just random points of the source image. The division in bins for one of the source images can be seen in figure 1.

The final implementation ended up not working correctly. As a consequence, the results do not contain data for the K-means implementation as these did not add anything.

K-means Clusters

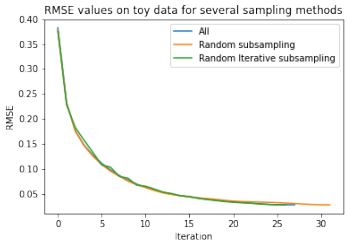
Figure 1: K-means clusters of the target image.



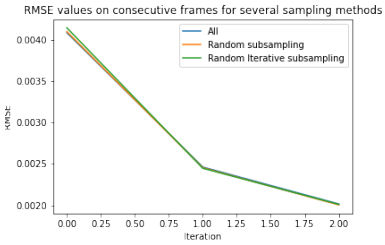
Results point selection techniques

Root Mean Squared Error

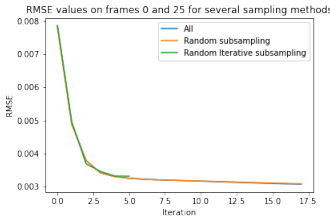
The random sub sampling methods used 10% of the source data. Experiments were also done with 30%, but these resulted in almost the exact same performance. As 10% of the data results in faster computation, this was also used for the results. Figure 2 shows the Root Mean Squared Error of each iteration for the ICP algorithm for different sampling methods and datasets without added gaussian noise. As can be seen in figure 2a, the RMSE converges rapidly for all three sampling methods and for the toy dataset. The difference in performance between sampling methods is small in this case.
Figure 2b shows the RMSE when using the first and second frame of the video. Because the cloud points are already relatively close to each other, the algorithm converges rapidly. Similar to the RMSE for the toy data, the three sub sampling methods follow roughly the same loss. The final figure, figure 2c, shows the RMSE for frames 0 and frame 25 of the video. The difference between the two point clouds is larger, requiring more iterations to converge than in figure 2b. The Random Iterative Sub sampling method converges sooner and has a higher RMSE than the two other methods. It should be noted that, due to the nature of randomness, this might not always be the case, especially for this implementation. The performance depends on the initial split of the data and might therefore not always result in good performance, especially when a percentage of the point cloud is small.



(a)

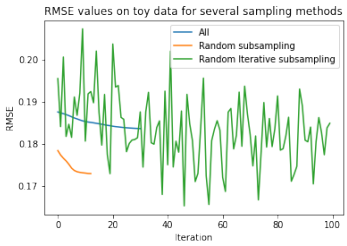


(b)

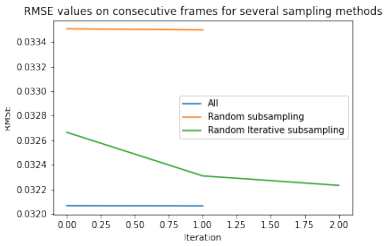


(c)

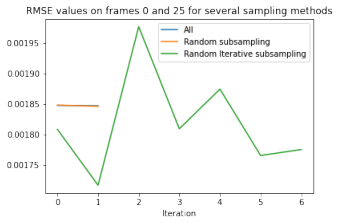
Figure 2: RMSE values for each iteration of the ICP algorithm for different pairs of images



(a)



(b)



(c)

Figure 3: RMSE values for each iteration of the ICP algorithm for different pair of images with added Gaussian noise

Shown in figure 3 above, the RMSE values differ quite a lot when adding gaussian noise to the respective datasets, when comparing them to the values that we get without adding gaussian noise. This is of course expected, since adding noise changes the image considerably. As noise we opted for random Gaussian noise, with the mean and standard deviation the same as the dataset in question. We can observe that, especially for the toy data, the iterative subsampling method has a harder time in converging towards a lower RMSE value. This is due to the fact that at every iterations it randomly selects 10 percent of the data (independently of the previous iterations) and it add new random gaussian noise. In this approach the algorithm sees to much different data to converge. Looking at figure 3b the RMSE values are higher then the RMSE scores in figure 3a. This is due to the fact that the point clouds doesn't differ to much since they have consecutive frames. There is almost no difference between the sampling methods. When looking at figure 3c it can be seen that the RMSE scores are lower than when the same data without gaussian noise is used. Thus it can be said that the gaussion noise is helping the algorithm for these frames. When looking at the different sampling methods it can be seen that the "all points" method and "random subsampling" method perform exactly the same. The random subsampling per iteration needs a few more iteration but also performs similar.

Execution Time

The execution times for the different datasets and sampling methods can be seen in table 1. The first notable result is the difference in execution time between the normal and noisy data. This can be explained with the fact that the ICP algorithms converged much faster due to the noise, resulting in lower times. An expected result is the difference in execution time when using all the points and when taking a random subset. These results, combined with the results for the RMSE for the same data tells us that Random Sub sampling should be chosen over using all the points when the two frames are close, as the RMSE is more or less the same in this case, but the execution time is lower. When the two frames are far apart from one another, a decision should be made between a faster but slightly worse version or slower but slightly better performance.

2.2 Improved ICP - explaination KD-tree

KD-tree stands for K dimension search tree. It is a useful method when solving multidimensional search keys like for example a nearest neighbor approach. Since point clouds also contain a lot of

Table 1: Execution times of the ICP algorithm for different datasets and subsampling methods in seconds.

	All (s)	Random Subsampling (s)	Random Iterative Subsampling (s)
Toy Data	0.270	0.0401	0.0400
Close Frames	0.377	0.0680	0.0830
Far Frames	23.305	2.745	1.080
Toy Data - Noise	0.303	0.0200	0.127
Close Frames - Noise	0.285	0.0640	0.0830
Far Frames - Noise	0.282	0.0510	0.117

points (high dimension data), KD-tree can be useful for speeding up the ICP algorithm. The idea of the KD-Tree is that it splits non-leaf nodes (hyperplanes) into two parts, half-spaces. Whereby every leaf node is a k-dimensional point. In our case this is a point in the point cloud. All the points left of the hyperplane are put in the left subtree of that node. All the points right of the hyperplane are put in the right subtree of that node.
For example if for a specific split in the tree the y axis is chosen, then the points with a larger y values will be put in the right subtree. All the points with smaller y values will be put in the left subtree. In this way the tree is split up in smaller sub trees and this makes it easier for the ICP.

3.0 Merging scenes

3.1.a

In figure 4, 5 and 6 the merged point clouds can be seen from the front, middle and side view. The difference between the plots is the sampling rate (every 1st, 4th and 10th frame). The sampling rate is the number of which we skip the number of frames in between. So the higher the number, the more frames we skip and the more difficult it becomes to merge properly because the ICP algorithm has a more difficult task. This is because the point clouds are more different. When looking at the figures it can be seen that the merging result is similar for sampling size 1 and 4 but performs worse when sampling size is 10. Again, this is probably due to the facts that the ICP receives a more different point cloud than when compared to sampling size 1 and 4. In all the plots there are still some artifacts left and it can be seen that the merging can still be improved. This is probably due to the fact that there are outliers in the data and ICP is very sensitive to these outliers (**?**).

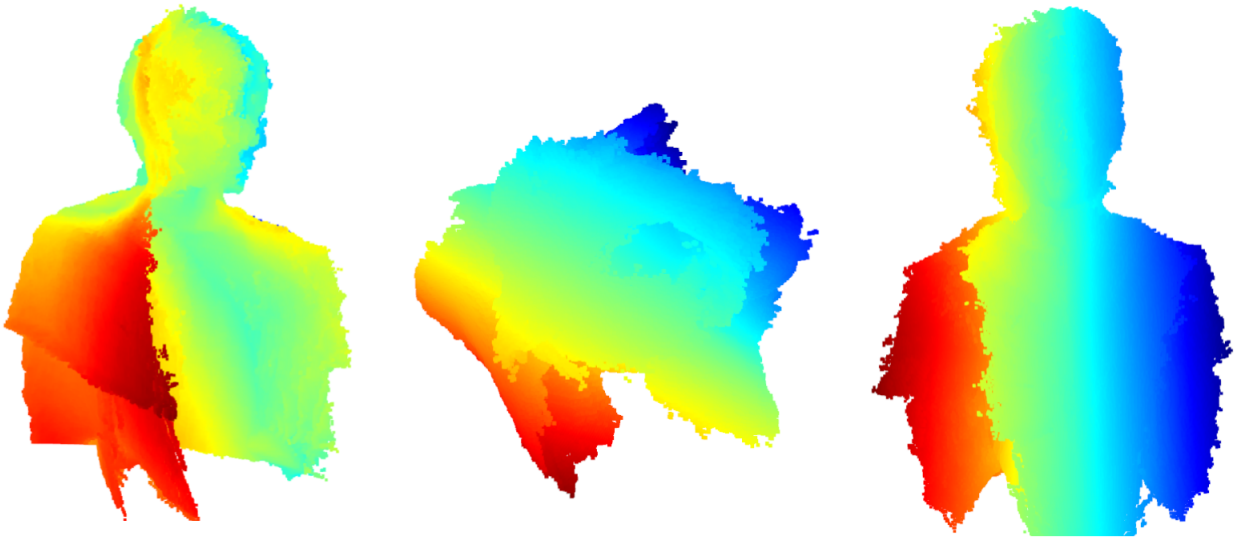


Figure 4: Front view (left), top view (middle) and side view (right) of the merged point cloud with sampling size = 1

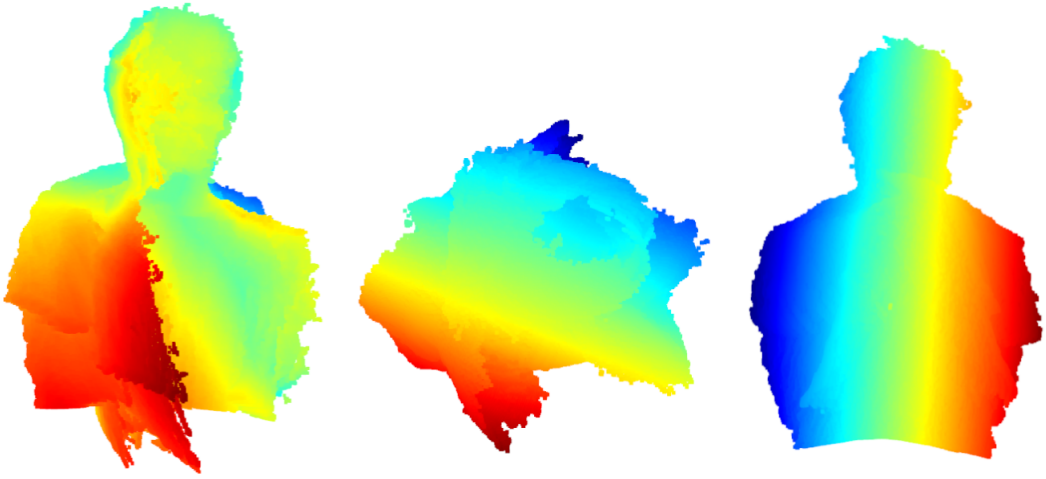


Figure 5: Front view (left), top view (middle) and side view (right) of the merged point cloud with sampling size = 4

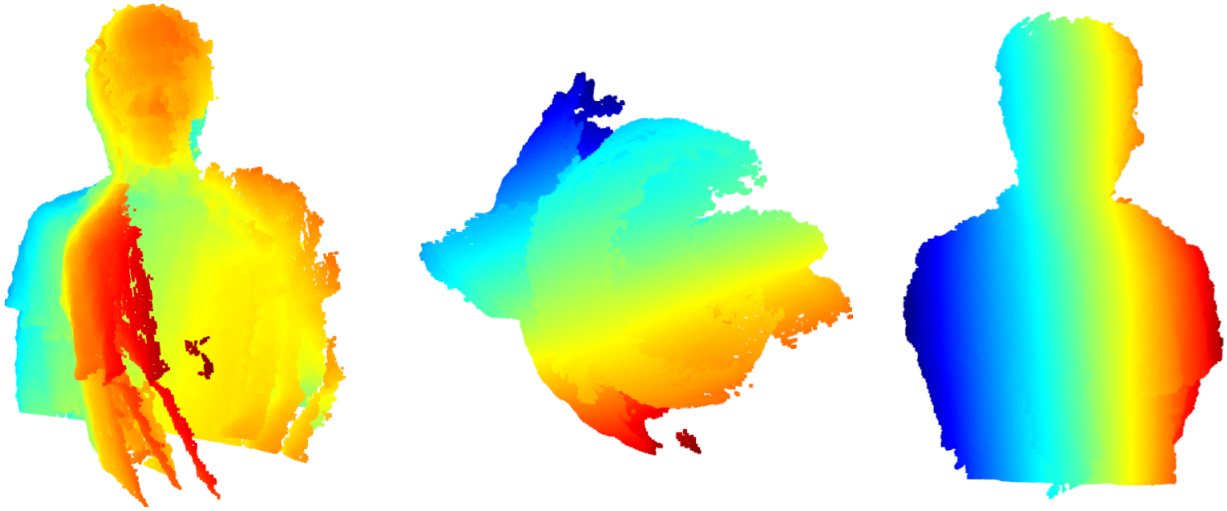


Figure 6: Front view (left), top view (middle) and side view (right) of the merged point cloud with sampling size = 10

3.2

In figure 7 the top and side view of the merged point cloud can be seen. When comparing this result with the results from figure 4, 5 and 6 it can be seen that this method is performing worse. This is probably due to the facts that the target point cloud is fixed and the ICP algorithm has to calculate a bigger distance between the point clouds. Another difficult thing is that the point clouds are combined and the ICP algorithm contains more points when training takes place. When a point cloud contains more points it is harder for the ICP algorithm to estimate the best pose parameters between the two point clouds.

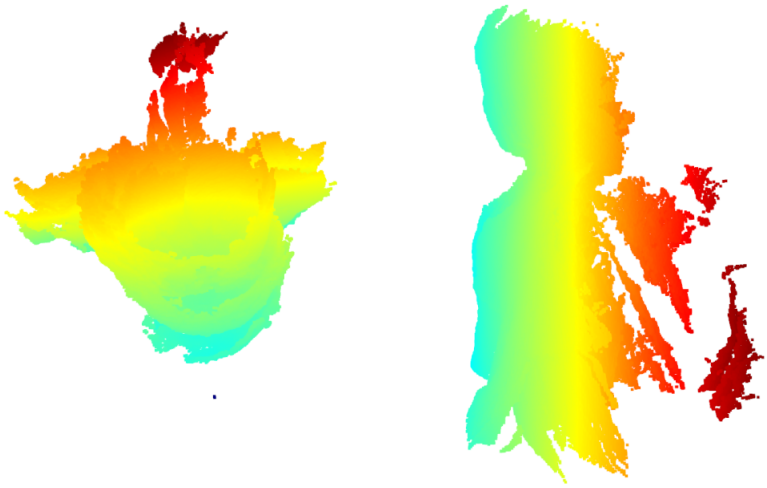


Figure 7: Top view (left) and side view (right) of the merged point cloud

4 Questions

1. What are the drawbacks of the ICP algorithm?

   (a) One of the drawbacks of the ICP algorithm is that it is sensitive to outliers. The algorithm converges must faster when removing these outliers (1).

   (b) The objective of finding for each point in the source cloud the closest point in the target cloud is slow. Especially for larger point clouds this "problem" grows fast (1).

2. How do you think the ICP algorithm can be improved, beside the techniques mentioned in [4], in terms of efficiency and accuracy?

   The idea is to divide 10 buckets uniformly over the point cloud. Then, points from the point cloud are filled in the buckets based on which bucket is the closest for that point (Euclidean distance). After this approach some buckets will have more points than others. Afterwards, you sample 10 percent of the total points in the point cloud for running the ICP algorithm. This portion of the data is sampled over the buckets where buckets with more points have a higher probability. In this way you can "force" the ICP algorithm to perform well in the regions where you sample more points from. This can be useful around the face for example, since there are more edges and corners and this makes it harder for the ICP algorithm to solve. This approach is efficient because you select only a subset of the point cloud and it is accurate since you sample relatively more points from more difficult regions.

5 Additional improvements

One way to improve the ICP algorithm, in speed but potentially also in accuracy, is to sample from more informative regions. This is in line with what we tried using K-means, however, it did not yield good results for us. More informative regions include be sampling from certain body parts (for this dataset that is) which, we know, have a great number of datapoints. Other informative data could be the surface normals or perhaps even the movement between two frames, using optical flow.

6 Self-Evaluation

We did all the work together. We met up via discord and zoom. Screenshare was used to do all the coding together. Then afterwards we together wrote the report.

Conclusion

In this assignment we implemented the ICP algorithm on two datasets, the toy data and the dataset where a person is rotating. The ICP algorithm was implemented with different point selection techniques, namely all datapoints, uniform sub-sampling, random sub-sampling in each iterations and K-means. Next to this implementation we tested the algorithm with adding random gaussian noise to the data. The results are analyzed based on the execution time, RMSE score and stability. We used the camera estimation parameters from the ICP algorithm for scene merging using the "rotating data".

References

[1] Pan, Y.; Yang, B.L.F.: Iterative global similarity points : A robust coarse-to-fine integration solution for pairwise 3d point cloud registration. https://arxiv.org/pdf/1808.03899v1.pdf (2018)