Computer Vision 2
Assignment 1: Iterative Closest Point

Riaan Zoetmulder
MSc Artificial Intelligence
University of Amsterdam
`riaanzoetmulder@gmail.com`

Minh Ngo
MSc Artificial Intelligence
University of Amsterdam
`nlminhtl@gmail.com`

1 ICP

The iterative closest point algorithm matches attempts to match surfaces. It does so by firstly calculating for each point the nearest neighbor, and creating pairs with corresponding nearest neighbors. It than proceeds by calculating the average centers of mass for each point cloud, and subtracting these from the closest point pairs. It than takes the outer product of the centered closest points and performs a singular value decomposition on the resulting matrix A. The V and the U^T components are than multiplied to obtain the rotation matrix. If the determinant of this rotation matrix equals -1, it is transformed such that its determinant equals 1. The translation matrix is calculated by taking the average of the target point cloud and subtracting the rotation matrix times the average of the source point cloud. This is done for several iterations, until the Mean square error is sufficiently low.

1.1 Experiments

We ran several different experiments with the ICP, using different types of sub-sampling. We used uniform sampling from all the points, we used re-sampling after every iteration of ICP and we had a sampling method based on k-means.

1.1.1 Results

For all the different manners of sampling we looked at the time it took to converge and we looked at the MSE after convergence. Firstly we shall look at the results for the uniform sampling. We plotted Means Squared Error and Time respectively in figure 1 and 2.
We looked at different samples from the source and target cloud. As can be seen, the larger the sample, the lower the MSE, but, unsurprisingly, the longer it took to converge.

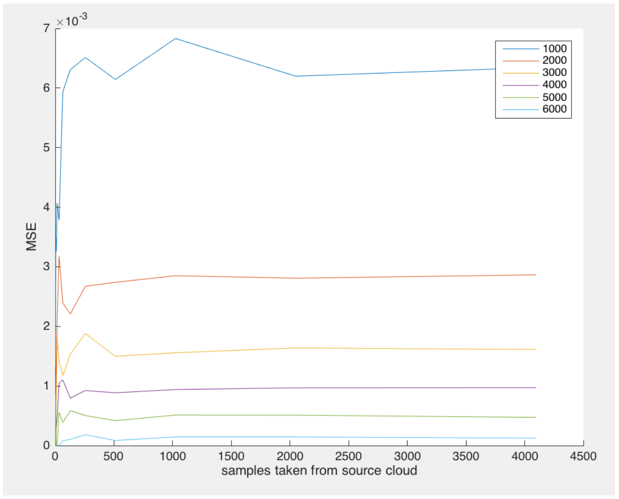

Figure 1: Mean squared error for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis.

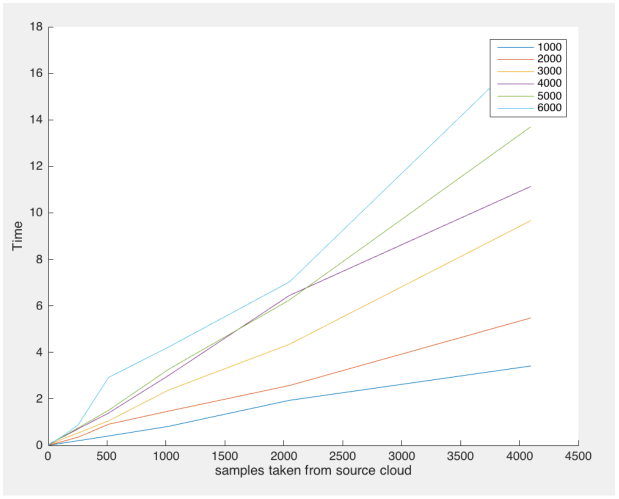

Figure 2: Runtime for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis.

Secondly, we will look at the results for the continuous subsampling. This methodology resamples the target cloud uniformly in each ICP step. The results for the Mean Square error and Runtime are plotted in Figures 3 and 4 respectively.

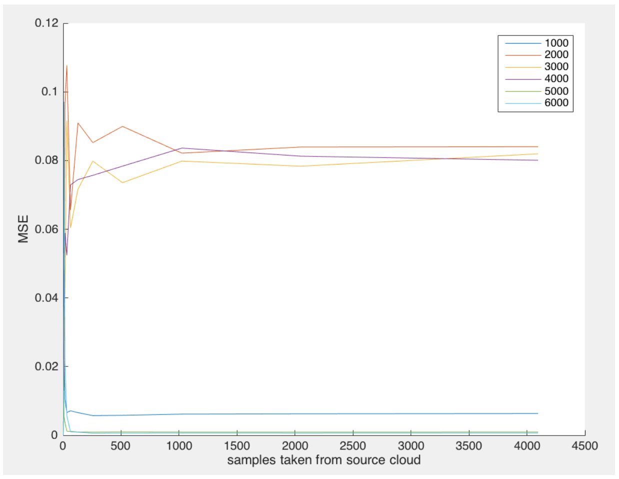

Figure 3: Mean squared error for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis. Resampling method; continuous uniform resampling of the target cloud.

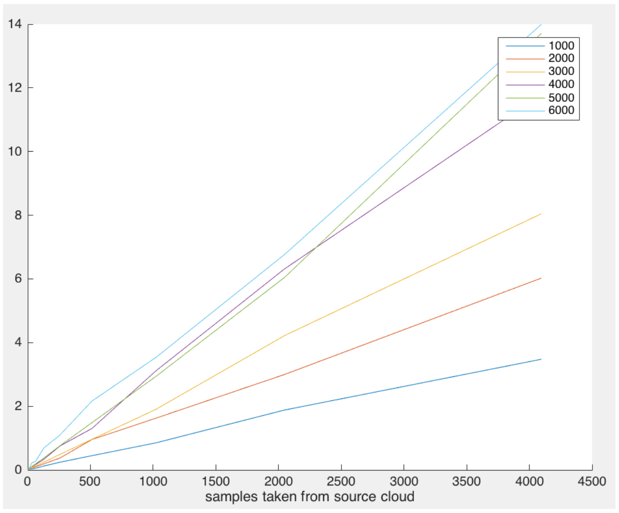

Figure 4: Runtime for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis. Resampling method; continuous uniform resampling of the target cloud.

Thirdly, we present the results for the sampling method based on k-means. Our results for the mean square error and runtime for 50 centroids and different amounts of points sampled around the centroids are respectively shown in figures 5 and 6.

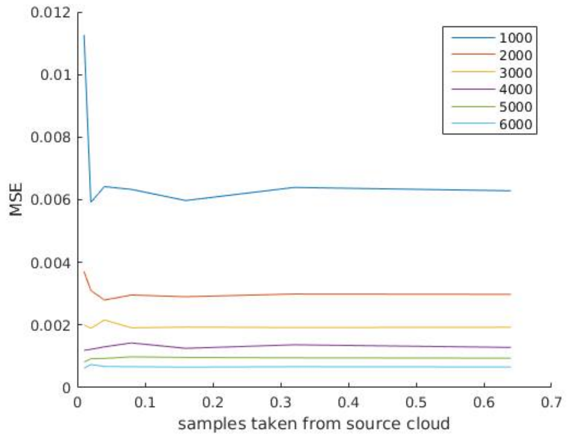

Figure 5: Mean squared error for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis. sampling method; sampling around 50 k-means centroids.

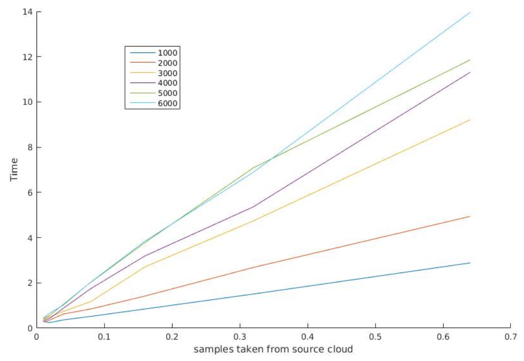

Figure 6: Runtime for different sample sizes of the target cloud, plotted in different colors. Sample sizes of source cloud shown on x-axis. Resampling method; sampling around 50 k-means centroids.

We believed that k-means can extract meaningful points because the centroids will move to dense and less dense regions. By extracting a certain percentage from the points belonging to a certain centroid, more densely populated regions will have a higher influence, when compared to more sparsely populated regions. As such the point clouds will be matched using these regions. However we found that, k-means with 50 centroids performed slightly worse than random sampling.

2 Merging Scenes

In this section a real data set of a scanned body has been used. We experimented with two strategies for merging scenes.

In the first strategy we are picking consecutive frames i-th, (i+1)-th and then estimate a rotation matrix and a translation vector to match (i+1)-th frame to i-th. To merge (i + 1)-th point cloud to the rest, we apply estimated rotation matrices and translation vectors from all (k, k + 1) pairs where $k \leq i$ in the reverse order. We have experimented this approach with each 1, 2, 4, 10 frames. To speed up our algorithm we have subsampled 1000 points from the target surface and 1000 points from the source surface as an input for the ICP algorithm. The result of merging all frames together is presented in the figure 8.

To compare merged results we have introduced a measurement called Average Mean Square Error that is an average value of MSE obtained by merging each consecutive frames. Results are shown in the figure 7. It can be seen that AMSE measure increases with decreasing amount of frames to consider.

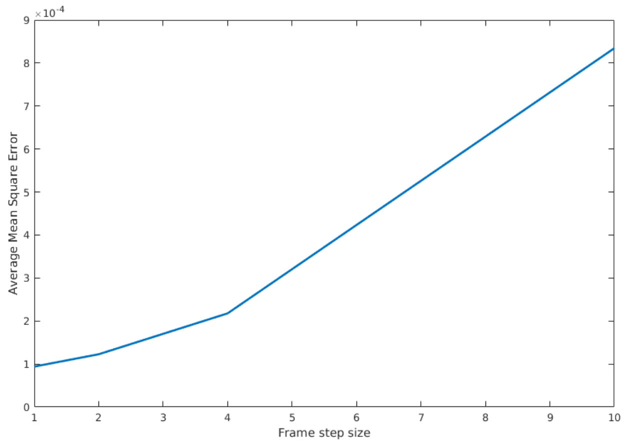

Figure 7: Average Mean Square Error for the first merge scene strategy with frame step size of 1, 2, 4, 10

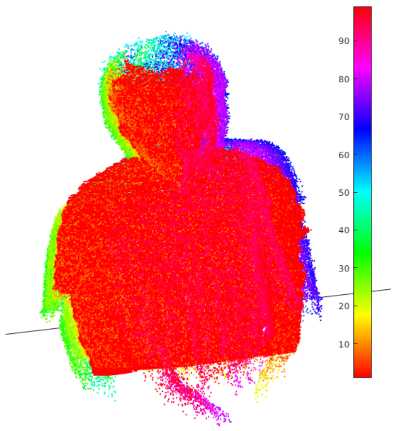

Figure 8: Merged scenes for the first merging strategy. Point clouds of different frames are visualized in different color.

In the second strategy, a camera pose is estimated between 0-th and 1-th frames, two frames are merged together for estimating a camera pose between it and the 2-nd frame and this process continues until we merge all point clouds together.

We experimented with two ways of merging point clouds. In the first we map the i-th point cloud to the coordinate system of the merged point cloud, in the second we do in the opposite way.

It has been noticed that the first solution didn't work for the real case scenario. In this case a back surface and a frontal surface will be initially aligned together since a depth camera has a fixed position at the same time when a human is rotated around, and since ICP converges to local optimum the result will be even more matched to a position of the frontal surface. As a result a half of body surface will be obtained.

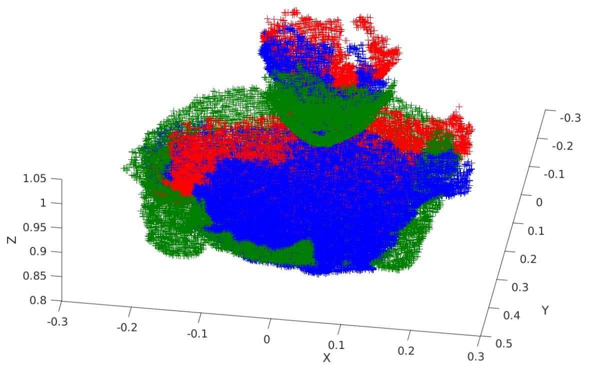

Figure 9: Possible alignment problem of the second merge scene strategy. Back surface (red) is initially aligned to the frontal surface (green). After matching it with ICP the result (blue) will be more stitched to the frontal part.

In the second approach the desired result has been obtained [Fig 10]. The i-th frame will be initially aligned to the (i-1)-th frame since we map a coordinate system of the merged point cloud of all frames from 0-th to (i - 1)-th frames to the (i - 1)-th frame coordinate system and because of that i-th frame will be aligned to a local minimum situated near the i-th frame point cloud.

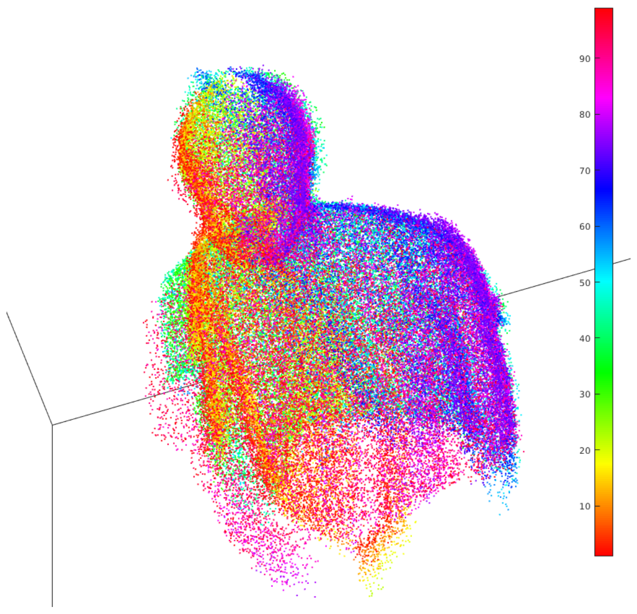

Figure 10: Merged point cloud provided by the second merge strategy.

3 Conclusion

During this assignment we set out to answer two questions, firstly we wish to evaluate what the drawbacks of the ICP algorithm are. One problem that can arise with the ICP algorithm is that when the surfaces are nearly flat (planar). Than no unique solution may exist. Another problem with ICP is that until recently it was quite slow. Lastly, when implementing the algorithm, there is no requirement that closest points are matched uniquely. This means that multiple source points may be matched to one target point. The differences that may arise due to this, can be negligible. Secondly, we wish to give an idea as to how the ICP algorithm can be improved.

One way to make ICP more robust is to find a way that reduces the amount of error. One approach could be creating criteria to reject certain points. For example given that a point is located in a certain percentile, we could throw it out of the data set. If however we do believe that this point is a legitimate data point, we could windsorize it. This means that all the points that lie beyond a certain predetermined maximum percentile will be given that percentiles value.

Wrong matches can be also preprocessed later via the 3d boolean algorithm to remove surfaces that have both side touched to the air.