Computer Vision 2 - Assignment 1

Emma Hokken - 10572090
emmahokken@gmail.com
University of Amsterdam

Tim van Loenhout - 10741577
timvanloenhout@gmail.com
University of Amsterdam

1 INTRODUCTION

In this report, we note our findings for Assignment 1 of Computer Vision 2. We were asked to implement the Iterative Closest Point (ICP) algorithm, introduced by Besl and McKay [1]. ICP is an algorithm that, given two point clouds, attempts to find a rotation and translation matrix such that the distance between the two point clouds (pcds) is minimized. This algorithm can then be used to merge multiple pcds. In a way, the ICP algorithm finds the camera angle and position for each point cloud, and is able to transform a pcd such that the camera angle and position is the same for both clouds.

In Section 2 we describe the implemented algorithm and perform experiments on its accuracy, speed/efficiency, stability, and tolerance to noise for different point selection techniques. In Section 3 we attempt to find the camera angle and position of each pcd using ICP, after which we try to merge these pcds in order to create a 3D reconstruction of 2D images. We perform further experiments on this, using a different step size and base pcd for the ICP algorithm.

2 ITERATIVE CLOSEST POINT - ICP

2.1 Algorithm

Here, we will briefly describe our iterative closes point function. For this description we will use the uniform sampling approach, which when applied with a sampling ratio (r) of 1.0 is similar to iterative closest point without sampling. First of all, the function takes a base point cloud (pcd) of dimension (3xm) and a target pcd of dimension (3xn) as input arguments. These pcds are then sampled down to subsets of size rn.

Subsequently, we iteratively update the base pcd until a threshold is reached where the standard deviation over the RMS scores of the 10 previous iterations is reduced to below 0.00001. For each of these iterations, the RMS score is computed by for every point in the base pcd, computing the distance to the closest point in the target pcd, which is found using the SKlearn nearest neighbors algorithm.

After computing the current RMS, the base pcd is transformed to match the same camera angle as the target pcd by first taking the dot product with the rotation matrix R and then subtracting the translation matrix t. These matrices are computed by first centering both pcds around the origin, then taking their dot product to get the covariance matrix, which is then decomposed into matrix U, S and V using singular value decomposition. By taking the dot product of the two square matrices U and V, we get the (3x3) rotation matrix R. Using this matrix to rotate the centered base pcd, we only have to subtract the target point cloud from it to obtain the translation matrix t.

2.2 Point selection techniques

Four different sampling techniques are used: no sampling, uniform sampling, random sampling, and sampling of more informative points (from here on out called normal sampling, as surface normals are used). For both the uniform and random sampling, the point cloud is reduced by taking a set of random integers within the range (0, total number of points), of which the corresponding points are used for the sampled point cloud. The only difference between the two methods is that while the uniform approach uses the same subset of points during the entire ICP process, the random approach creates a different subset for each iteration.

For the normal sampling, the subset is created in a different manner. Each pcd file has a corresponding file with surface normals. These surface normals are sorted into buckets, where normals with similar values are grouped together. As each pcd point has a corresponding surface normal, only the indices of the normals are stored in the buckets. Then, an equal number of indices is drawn from each bucket, of which the corresponding points are used in the subset. Points for which no surface normal is available (surface normal is 'nan') are not used.

2.3 Comparison

For comparing the different sampling methods, we tested using 5 different sampling ratio's; 1.0, 0.75, 0.5, 0.25 and 0.1. Furthermore, we used the first two frames of the human pcds as the waveform pcds do not contain the surface normals which are required for testing the normal sampling method. Nevertheless, for the remaining three sampling methods, the same results as discussed below for the human pcds hold for these wave form,

Testing was done on four aspects; accuracy, stability, speed and tolerance to noise.

2.3.1 Accuracy. For measuring the accuracy, we look at the RMS after convergence (table 1). As anticipated, we observe that there is a positive correlation between the used number of points and the accuracy. Consequently, using the entire pcd holds the lowest error. Nevertheless, uniform or random sampling with a sampling ratio of 0.75 or 0.5 yield an almost equal accuracy. Furthermore, despite scoring slightly lower in terms of accuracy, both sampling methods gain a significant increase in speed, which will be discussed more elaborately in the next section. Normal sampling on the other hand, performs significantly worse, even when using 75 percent of the points.

	1	**0.75**	**0.5**	**0.25**	**0.1**
no sampling	0.0016				
uniform		0.0017	0.0019	0.0024	0.0033
random		0.0017	0.0019	0.0024	0.0034
normal		0.0026	0.0027	0.0030	0.0036

Table 1: RMS value for every sampling method, using different sampling ratios

2.3.2 Speed. For speed, we looked at the time until convergence, using a sampling ratio of 0.5 for the uniform, random and normal method (table 2). From these results, we can observe that uniform, random and normal sampling increase the speed of a single iteration opposed to using no sampling. The results also indicate that uniform sampling takes less seconds for the complete optimization than random sampling, despite both using the same random point selection. The difference might therefore be caused by the fact that the former only requires one sampling operation, opposed to the multiple sampling operations of the latter.

Furthermore, normal sampling performing similar to uniform and random sampling on a single iteration is not the result we anticipated as this method uses the more informative regions, which we expected to result in a faster finding of the closest points. On top of that, the total time until convergence using this method is even larger than when using all points. This added time could be the result of the extra operations required for this sampling method. That is, in order to create the buckets containing the normal vectors, the operation needs to loop over all points in the point cloud. Finally, from table 2, it can be seen that the number of iterations required for convergence does not differ much across the sampling methods.

	single iteration	**until convergence**
no sampling	0.24 seconds	5.00 seconds
uniform	0.10 seconds	1.97 seconds
random	0.10 seconds	2.07 seconds
normal	0.14 seconds	7.50 seconds

Table 2: Seconds until convergence for each sampling method, with the sampling ratio set to 0.5

2.3.3 Stability. For stability, we looked at the convergence trend (figure 2). From this figure, we observe that for the uniform, normal and no sampling methods, the RMS decreases without any fluctuations, indicating a stable convergence. For the random sampling method we observe a similar trend for the cases with a high sampling probability, whereas more fluctuations arise when sampling down to less number of points. This could be caused by the fact that this method uses a different sample for each iteration. For larger samples, the difference between these pcds is relatively small, however, for smaller samples, the variance increases. Consequently, when using random sampling with a low sampling probability, the algorithm has to use significantly different pcds for every iteration.

2.3.4 Tolerance to noise. In order to test the tolerance to noise, we added different levels of Gaussian noise to the point clouds. From the results (figure 3), we observe this significantly impacts the performance of the algorithm when using uniform sampling, random sampling and no sampling. Normal sampling on the other hand, seems to be much more tolerant to the extra noise. This might be due to the fact that the added points are significantly less informative. Consequently, the normal sampling is able to emphasize on filtering out the noise, whereas the other methods do not make the distinction.

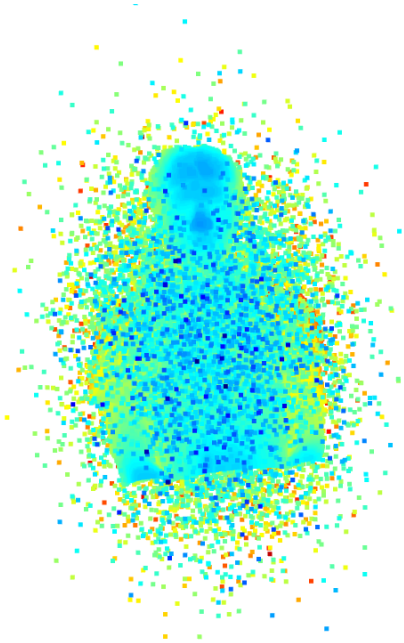
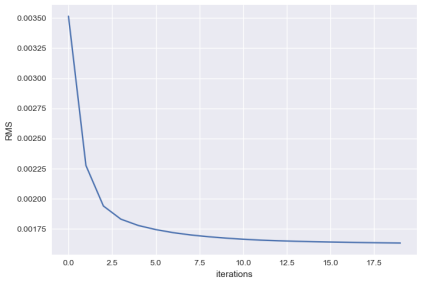


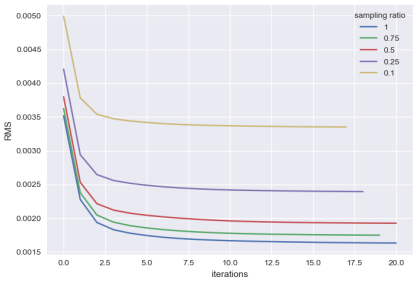
Figure 1: Point cloud with 25 percent added Gaussian noise.

2.4 Conclusion

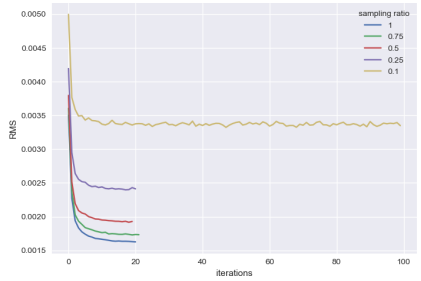
In terms of accuracy, no sampling obtains the high score, followed by uniform and random sampling. However, despite the latter obtain slightly worse scores, they require less time, with the uniform sampling method converging fastest. On top of that, uniform sampling displays a more stable convergence than random sampling. Hence, for accuracy, no sampling yields the best results, however uniform sampling provides similar results in considerably less time. Due to time not being an issue, and in order to gain the best results possible, we will apply the ICP algorithm without sampling for the merging operation, discussed in the next section.



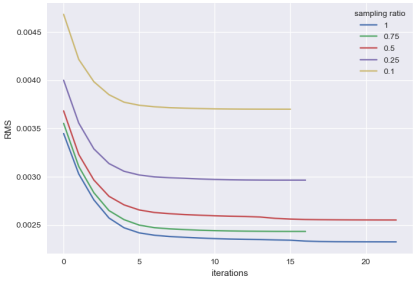
(a) No sampling



(b) Uniform

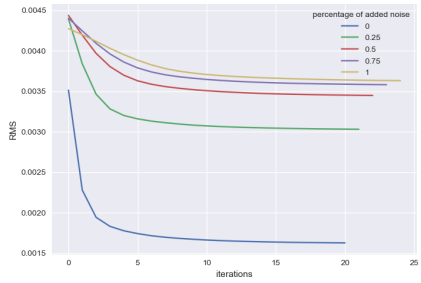


(c) Random

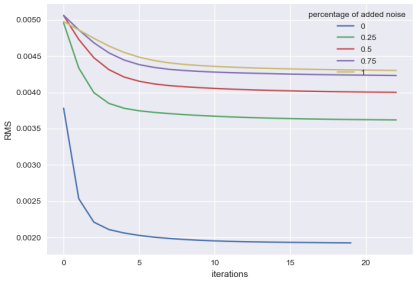


(d) Normal

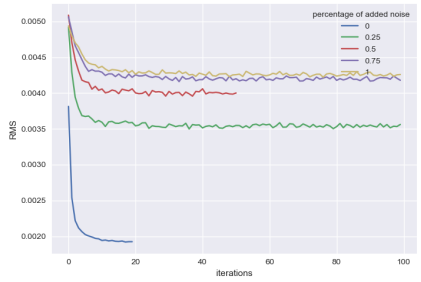
Figure 2: Convergence of each sampling method, using different sampling ratios



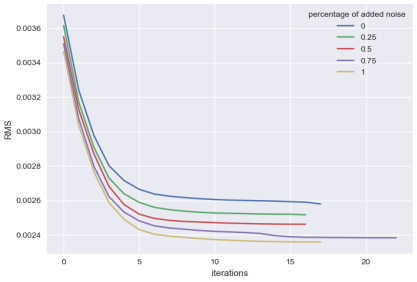
(a) No sampling



(b) Uniform



(c) Random



(d) Normal

Figure 3: Convergence of each sampling method, using different amounts of added noise

3 MERGING SCENES

3.1 Method

With the use of the ICP algorithm, two images can be merged into one. When done with mutliple images, this can create a 3D representation from a set of 2D images. The following sections describe two ways of applying this process over the a set of pcds using a given step size.

3.2 Merging using consecutive frames (3.1a)

For the first process, the first step consists of finding the rotation and translation matrix between the first two pcds. In order to merge these pcds, the resulting matrices are used to transform the base pcd to the same angle as the target icp, after which the adjusted base and target are stacked. Subsequently, for the next steps the old target becomes the new base, on which the ICP algorithm is applied to transform it to match the next target. During each iteration of the ICP optimization, the same transformation is also applied to the previously created merged pcd. Consequently, after convergence, the merged pcd can be updated by stacking the new target as they contain the same camera angle. This process is repeated until all pcds in the set are merged together.

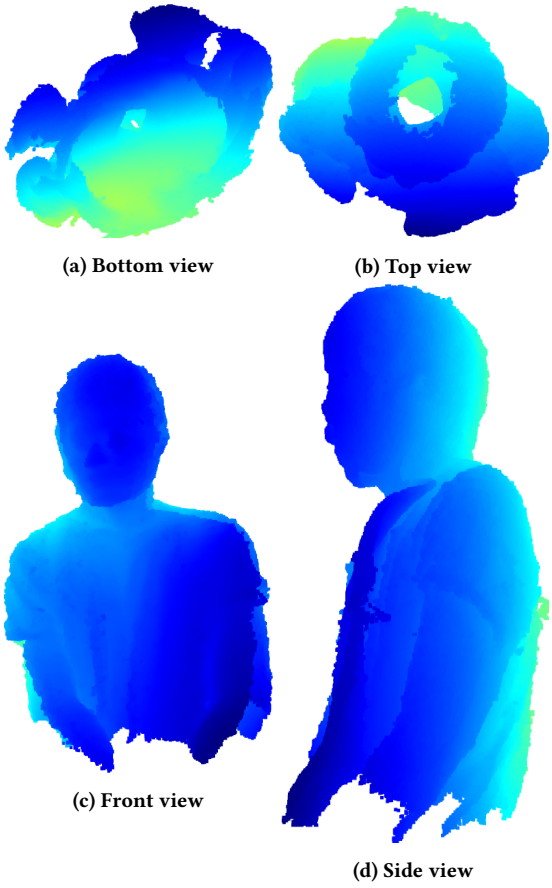
Results of this merging process using consecutive frames can be found in Figure 4. The pcd files were visualized using Open3D.

As can be seen in Figure 4, this reconstruction is not perfect. For instance, the bottom view shows that the arms are located at several locations. This discrepancy might be caused by small errors in the transformations during some steps. Figure 5, which shows the RMS after convergence for each point cloud pair, also points towards this suspicion. After one such mishap, the merged matrix is updated in a slightly incorrect manner. Sometimes the algorithm comes back from this quite easily (such as around the mishap after 20 iterations), but sometimes it does not (such as around 65 iterations). In such cases, the error accumulates, as the merged matrix does not contain the same camera angle anymore as the pcd which is used to compute the transformation matrices. In order to test this suspected accumulation of errors, we wanted to investigate whether the results would be better when prematurely interrupting the process. For this, we ran the algorithm until 50 frames, of which the results can be found in Figure 6. Whilst the resulting merged pcd is not complete, due to only using the first 50 frames, it does not show the same discrepancies observed in the full merged pcd. Hence, this strengthens the hypothesis that the error does build up over time.

3.3 Merging using 2nd, 4d and 10th frames (3.1b)

In this section, we test the impact of using different step sizes. That is, instead of using every consecutive frame, we use every 2nd, 4th, or 10th frame. The results for each step size can be found in Figures 8, 9, and 10, respectively.

The figures show that, when increasing the step size, the resulting figure decreases in quality. The ICP algorithm seems to be able to handle skipping every other frame, but already shows significantly worse results when only using every 4th frame. The distance from frame A to frame B simply becomes too large, due to which the corresponding rotation and translation matrices become less accurate.



(a) Bottom view (b) Top view

(c) Front view (d) Side view

Figure 4: Visualization of point clouds (using Open3D) using all frames in ICP

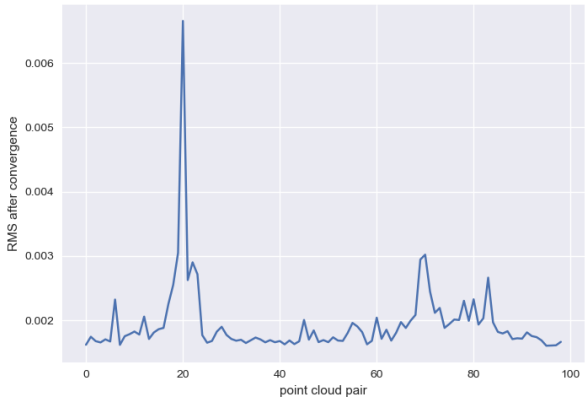
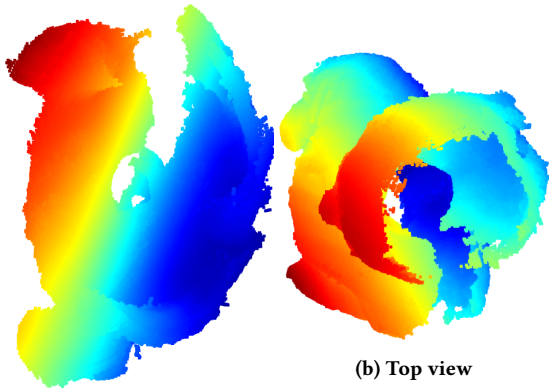
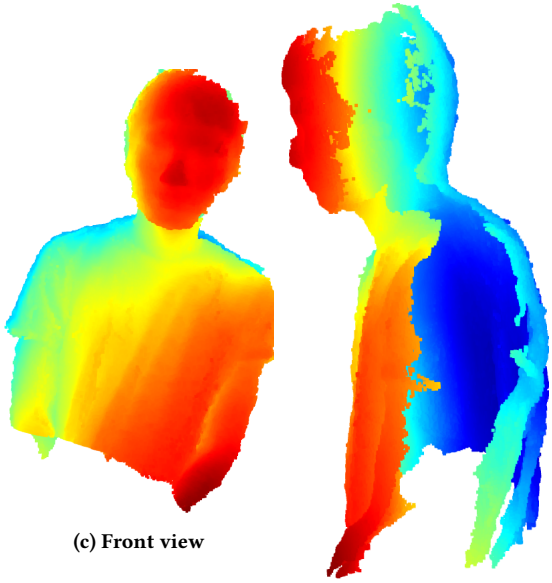


Figure 5: RMS after convergence for each point cloud pair.



(a) Bottom view

(b) Top view



(c) Front view

(d) Side view

Figure 6: Visualization of point clouds (using Open3D) using frames 1 to 50 in ICP

3.4 Merging using consecutive frames, ICP on merge (3.2)

For this approach, the merged point cloud (which is the transformed base and target stacked on top of each other) is used as the next base point cloud, based on which the transformation matrices are computed. The resulting 3D reconstruction can be found in Figure 11. This figure shows that the ICP algorithm has some trouble with this task. The bottom view does not show any body-like shape, and the front view shows almost two human figures. Together with the results discussed in section 3.3, this relatively poor performance indicates that while the ICP algorithm is quite good at finding the camera angle and position between two relatively similar pcds, it is less optimal for finding this transformation when the pcds are more deviated. That is, during this process, the base consists of increasingly more points, for which all the nearest point in the target pcd has to be found. Hence, as the base increases in size,

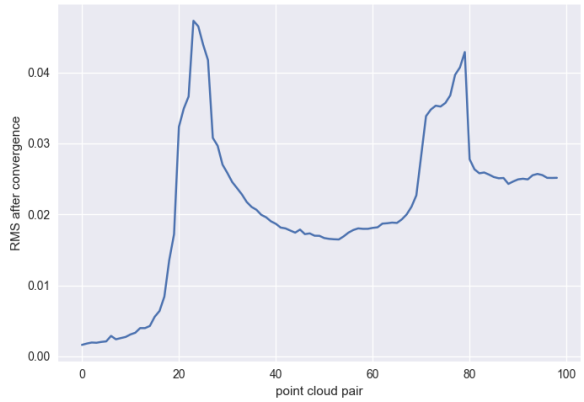
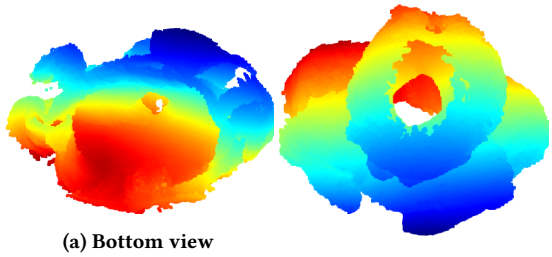
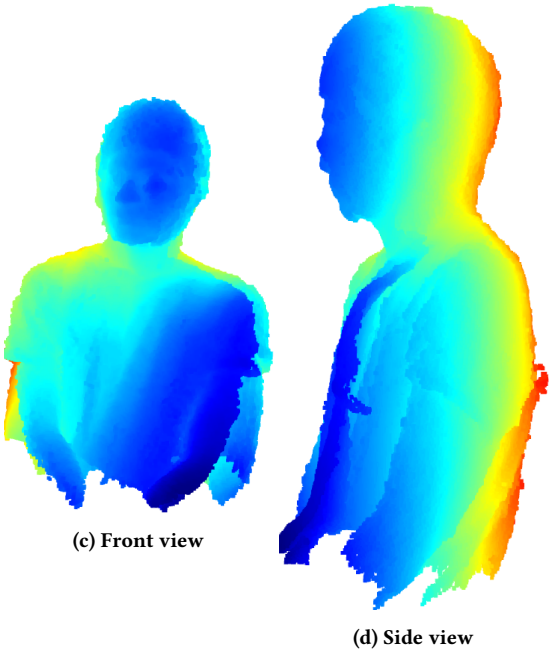


Figure 7: RMS after convergence for each point cloud pair, with previously merged point clouds as the base



(a) Bottom view

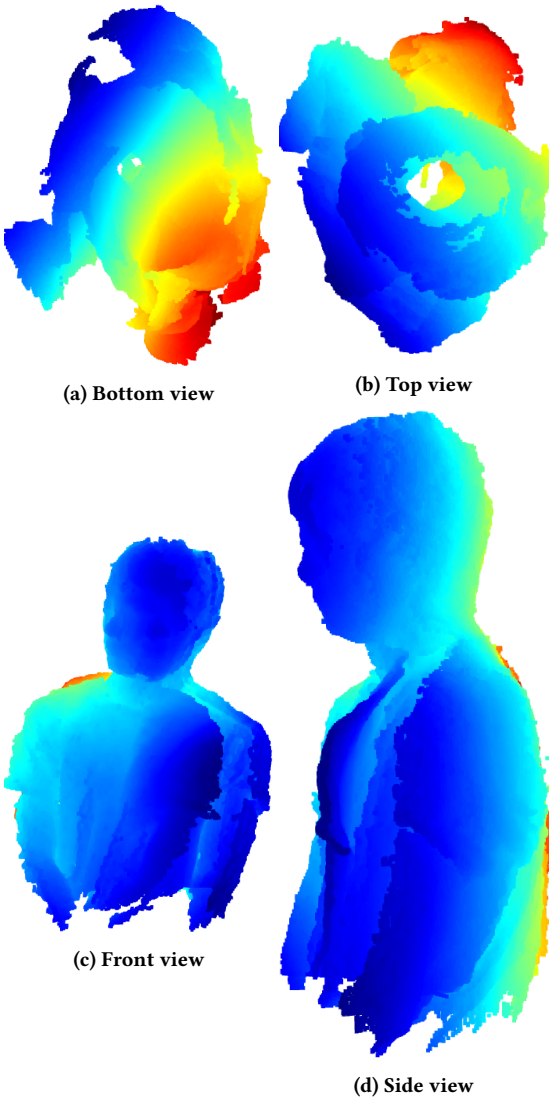
(b) Top view



(c) Front view

(d) Side view

Figure 8: Visualization of merged point cloud (using Open3D) using every 2nd frame.



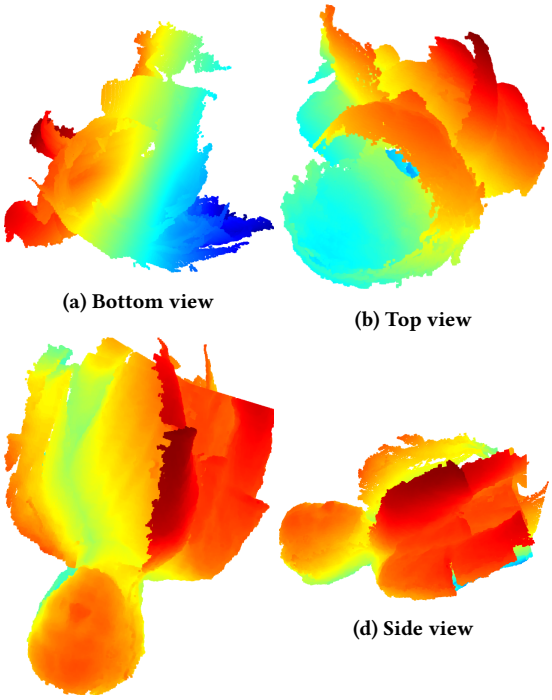
(a) Bottom view (b) Top view

(c) Front view (d) Side view

Figure 9: Visualization of merged point cloud (using Open3D) using every 4th frame.

increasingly many of its points do not correspond to any of the points in the target, consequently negatively impacting the accuracy of the optimization.

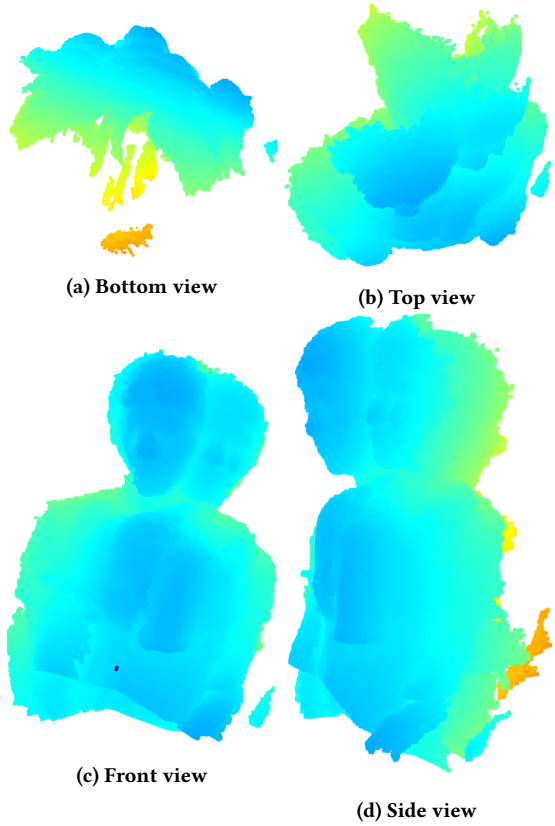
Figure 7 shows the RMS values after convergence for each iteration. At first, it looks quite similar to Figure 5, with the same increase in the RMS around iteration 20. However, unlike in Figure 5, the current implementation is not able to recover from this error, and instead displays a continual increase in RMS values.



(a) Bottom view (b) Top view

(c) Front view (d) Side view

Figure 10: Visualization of the merged point cloud (using Open3D) using every 10th frame.



(a) Bottom view (b) Top view

(c) Front view (d) Side view

Figure 11: Visualization of point clouds (using Open3D) using merged frame as base for next iteration in ICP

4 QUESTIONS

4.1 What are the drawbacks of the ICP algorithm?

One downside of the ICP algorithm is that it assumes that there are no outliers in the data [2]. When outliers are present, the algorithm fails to perform as it should. In essence, ICP is not very robust [2]. Furthermore, the ICP algorithm is guaranteed to converge to a minimum, but it is not guaranteed that that minimum is the global minimum. [3]. Convergence also tends to be a bit slower than desired [2], which is mainly due to the used point-matching technique. Therefore, various attempts have been made at speeding up the algorithm by experimenting with different approaches to the point matching process (see Section 5).

4.2 How do you think the ICP algorithm can be improved in terms of efficiency and accuracy?

A possible improvement in terms of accuracy could be found in the following; when a small transformation error occurs, this error will accumulate and increase over time. A fix for this would be to skip steps that result in a high RMS. However, a drawback of this approach is that the ICP algorithm then has to match frames that are further apart, a task this algorithm is not necessarily good at.

An improvement in terms of efficiency could be found in parallelisation, i.e. performing ICP on multiple point clouds at the same time. Due to the nature of the ICP algorithm, it is an ideal candidate for this. That is, for instance, finding the transformation matrix of frame 26 to frame 27 does not depend on frame 3 and 4. Parallelisation would greatly improve execution time, provided that it is done properly.

5 ADDITIONAL IMPROVEMENTS

As mentioned in [4], the execution time of the ICP algorithm can be improved by using *k-d* trees as the point matching technique. We attempted to increase the execution time of the ICP algorithm by using K-D tree search instead of Nearest Neighbours. However, when using uniform sampling with a sampling rate of 0.5, the program took almost 50 times longer to run (from 358 seconds for nearest neighbors to 17787 seconds for K-D trees). We think this might be caused by the underlying mechanism of used package (scipy.spatial.KDTree), as this longer execution time is the opposite to what various literature reports ([4], [2]).

6 CONCLUSION

The Iterative Closest Point algorithm is a fairly simple way to find the camera angle and position between two point clouds. In this report, we have performed several experiments with this algorithm using different sampling methods. The results show that using all points yields the best results in terms of accuracy. However, using up to half of the points using uniform or random sampling still results in only a minor decrease in accuracy, while gaining an significant increase in speed. Normal sampling on the other hand, performs worse than the other methods, which is against our expectations, as this method samples more from the informative regions in the point cloud. However, this method displays a higher tolerance against noise. Finally, when testing on stability, the ICP algorithm shows a stable convergence when using uniform, normal or no sampling, whereas it shows more fluctuations when using random sampling..

The ICP algorithm can also be used for merging several point clouds together. We have shown that this works reasonably well when the differences between the point clouds does not become too large. That is, the results decrease in quality when using only every 10th point cloud. Finally, computing the transformations based on the entire merged point cloud and a target point cloud results in a significantly worse reconstruction than when using only the the previous target for computing this transformation.

A SELF-EVALUATION

We divided the work fairly evenly among the two of us. Tim worked on the implementation for the ICP algorithm, with Emma providing the implementation for the normal sampling. Emma also focused on reading literature. For the merging of the point clouds we mainly worked together.

REFERENCES

[1] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. International Society for Optics and Photonics, 586–606.

[2] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. 2002. The trimmed iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, Vol. 3. IEEE, 545–548.

[3] Andrew W Fitzgibbon. 2003. Robust registration of 2D and 3D point sets. *Image and vision computing* 21, 13-14 (2003), 1145–1153.

[4] David A Simon et al. 1996. *Fast and accurate shape-based registration.* Carnegie Mellon University Pittsburgh, PA.