UNIVERSITY OF AMSTERDAM

Assignment 1 Report

Luca Simonetto - 11413522
Edgar Schönfeld - 11398272
Computer Vision 2

April 24, 2017

Iterative Closest Point - ICP

In this section the details on the implementation of the ICP algorithm, along with results and optimizations, are presented and discussed.

As the general workflow of the ICP algorithm has already been presented in the assignment instructions, it will not be discussed here.

Subsequent transformations accumulation

As the given algorithm determines the optimal rotation and translation matrices in an iterative way, the first issue that emerged was the output of the final rotation matrix R, along with the final translation matrix t. In order to solve this problem, the optimal matrices R_{opt} t_{opt} at each iteration have been combined with the previous ones using a transformation matrix T, defined ad iteration i as

$$T^{(i)} = \begin{bmatrix} R^{(i)} & t^{(i)} \\ 0 & 1 \end{bmatrix}$$

where $R^{(i)}$ is the 3×3 rotation matrix accumulated until iteration i

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

and $t^{(i)}$ is the 1×3 translation matrix accumulated until iteration i

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

The new transformation matrix $T^{(i+1)}$ will then be determined as

$$T^{(i+1)} = T^{(i)} \cdot T^{opt}$$

where T^{opt} is the transformation matrix created as before, using R_{opt} and t_{opt}.

Sampling methods

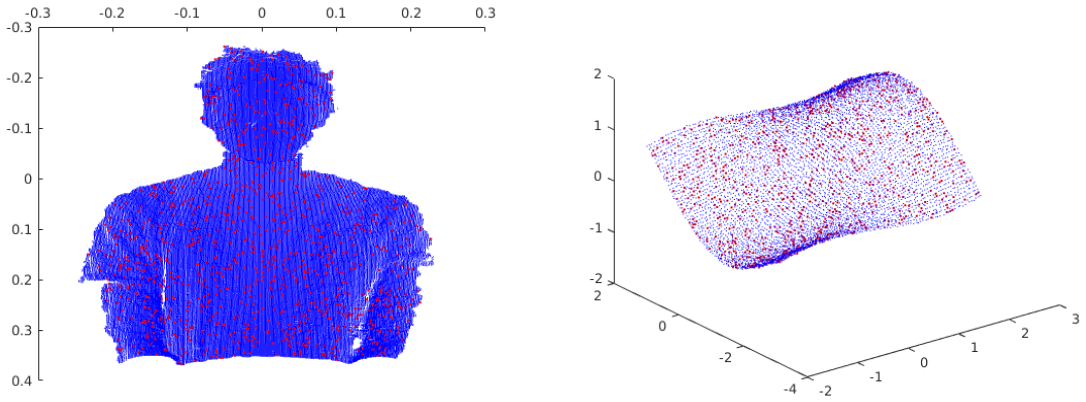
At the start of every ICP iteration, the source point cloud is sampled in order to retrieve a number of candidate points to be matched with the target point cloud. As multiple methods for sampling exist, the more widely used ones have been implemented, along with a custom sampling created using normals vectors knowledge (of the source points).

No sampling

The first sampling method used is just taking all the points in the source, and matching each one with a point in the target. This method requires the highest number of iterations in order to converge, but gives the best results in terms of root mean square (RMS).

Random sampling

In order to increase the speed of the algorithm, sampling of the source points has to be preferred, in order to compute less matchings even if the final result is more approximated. This method simply samples at random the points of the source, in order to output k candidates, where k is a hyperparameter of the model. Figure 1 shows 1000 candidate points extracted using this method.



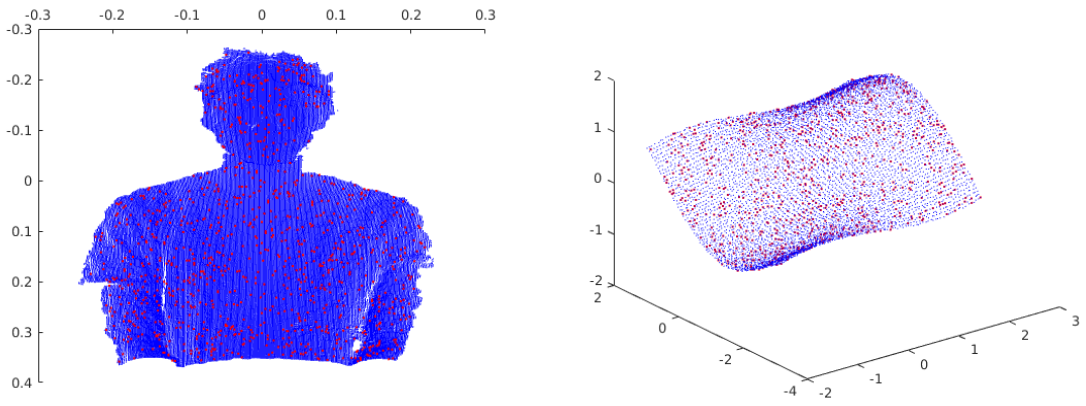
(a) (b)

Figure 1: Sampled points using random sampling.

Uniform sampling

Uniform sampling is similar to random sampling, but the candidates are extracted using a random distribution. This method has the advantage of giving more equally spaced candidates, in order to cover the source cloud of points in the best way possible. This sampling, however, does not take into account the spatial

properties of the object, so if a surface is very inclined, the extracted candidates may not cover well the whole area. This downside of the sampling method reduces its applicability to surfaces that are mostly flat and regular. Figure 2 shows 1000 candidate points extracted using uniform sampling.



(a) (b)

Figure 2: Sampled points using uniform sampling.

Informative regions sampling

As the data provided also includes the 3d normals of the cloud of points (vectors indicating the slope of the surface at each source point), it has been decided to implement a sampling method that uses this information in order to sample from regions of the source where more distinctive features are present, in this case under the form of edges, wrinkles etc. This method firstly samples uniformly k points from the source using the method discussed above, where k is the number of required candidates.

$$candidates = uniformSampling(source\_points, k)$$

When the k points are determined, a k-nearest neighbor algorithm is used to find $n\_neighbors$ number of neighbors for each candidate, in order to determine the closest normals to consider. The knn algorithm outputs the indexes of the chosen points, so the normals to consider are just the elements with the same index.

$$candidate\_neighbors = knnSearch(candidate, n\_neighbors)$$
$$normals\_neighbors = points(candidate\_neighbors)$$

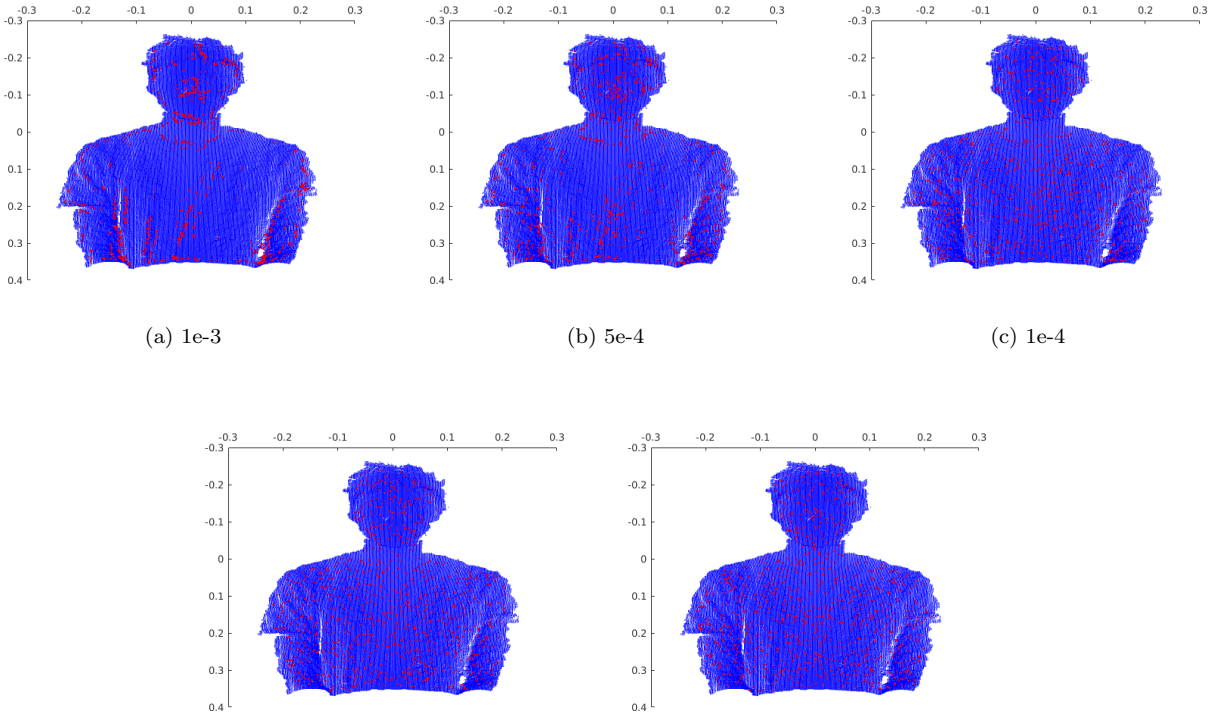
An important element to consider, is that the normals have their origin subtracted at the start of the algorithm, in order to be all centered in 0. This is necessary for the correct execution of the following steps.

Having determined the normals to consider, their total variance is calculated as the trace of their covariance. This value is normalized by dividing it by the total number of neighbors, in order to allow different numbers of neighbors to be used.

$$total\_variance = Tr(cov(normals\_neighbors))/n\_neighbors$$

where Tr is the trace operation and cov is the covariance matrix calculated on the 3D points. The total variance is then checked against a threshold, and the candidate point is chosen if the total variance of its neighbors normals is greater than it. After every candidate is checked, the execution is repeated if the total candidates are less than the required number.

Higher values for the threshold increase the selectivity of the sampling method as higher variance of the normals is required, but slow down the algorithm as more samples have to be checked until good candidates are found. Lower values for the threshold will result in a sampling that will be less discriminative and closer to uniform sampling but increase the overall speed of the method. Figure 3 shows 1000 candidates chosen with different values for the threshold.



(a) 1e-3 (b) 5e-4 (c) 1e-4

(a) 5e-5 (b) 1e-5

Figure 3: Sampled points using informative regions sampling. The numeric value indicates the total variance threshold

It can be seen that no results are present for the test point cloud, as no normals have been provided. Also, this sampling method would not give good results, as the surface is smooth and lacks imperfections.

Results

In this subsection the results are presented and discussed. Experiments have been done using frame 0-3 and 0-10, in order to analyze if some methods perform better or worse when the point clouds are less similar. Each experiment has been done multiple times, and the presented results are the averaged in order to be more reliable.

By looking at Table 1, it can be seen that the number of iterations vary greatly based on the sampling method used, and the highest number of iterations are done when considering all the points for each step. This shouldn't be a surprise, as this method balances high number of iterations with the lowest RMS error. Random sampling, along with uniform sampling converge faster and in a fraction of the time, but give

slightly worse results. When considering normals sampling at different thresholds the elapsed time increases to over two seconds for each run, but manages to reach results that are always better than random and uniform sampling. Different thresholds don't give enormous differences in RMS between each other (25% of maximum increase in performance) but can reduce the execution times by tens of seconds in the best case.

Sampling method	**Iterations**	**RMS**	**Elapsed time**
No sampling	36	**0.002037**	2.64
Random sampling	7	0.002717	**0.15**
Uniform sampling	**6**	0.002772	**0.15**
Normals sampling (1e-3)	10	0.002182	12.10
Normals sampling (5e-4)	11	0.002077	6.06
Normals sampling (1e-4)	8	0.002501	2.31
Normals sampling (5e-5)	9	0.002319	2.56
Normals sampling (1e-5)	12	0.002193	3.43

Table 1: Results using different sampling strategies on frames 0-3. The value in parenthesis for normal sampling is the total variance threshold.

Table 2 gives almost the same results, with no sampling being the best choice in terms of RMS. As the frames are farther apart the error increased, but the trends are kept, with random and uniform sampling being a lot faster with worse results, and normals sampling giving better results with higher execution times and similar iterations. In this case, however, when the threshold is set to 1e-3 and 5e-4 the results are worse than any other method.

Sampling method	**Iterations**	**RMS**	**Elapsed time**
No sampling	57	**0.005056**	7.11
Random sampling	7	0.008854	0.17
Uniform sampling	6	0.009366	**0.16**
Normals sampling (1e-3)	**5**	0.011534	7.14
Normals sampling (5e-4)	7	0.010186	4.57
Normals sampling (1e-4)	7	0.008942	2.15
Normals sampling (5e-5)	7	0.008661	2.12
Normals sampling (1e-5)	7	0.008514	2.19

Table 2: Results using different sampling strategies on frames 0-10. The value in parenthesis for normal sampling is the total variance threshold.

Given the above analysis the best method can be chosen based on the user preferences: if time is a concern the most efficient samplings are random and uniform, while if the minimum error is to be preferred then no sampling is the best choice. While normals sampling seems to give sub-par results, it could be seen as a valid alternative when using point clouds with a lot of non-uniform surfaces. In the second part of the project uniform sampling will be used, to achieve a good balance between error and elapsed time.

Merging Scenes

Given a dataset of point clouds, the task was to merge the given point clouds into a 3D reconstruction of the recorded object. Two different methods of merging frames es are tested, which will be referred to as method 1 and 2, corresponding to problem 2.1 and 2.2, respectively.

Method 1

The algorithm was implemented as described in Problem 2.1, using frame rates of 1, 2, 4 and 10. It should be noted that, in order to merge the accumulated point cloud with the newest merged frame (during the iterative 3D reconstruction), the rotation matrix and translation vector were computed for the newest frame with the accumulated frames as target. The accumulated frames were then matched to the newest frame via the inverse rotation and translation (see code for details). For all frame rates, the full object could only be reconstructed if all points were used (no sampling). The outcomes got worse with increasing frame rate. In fact, only a frame rate of 1 and 2 produces a satisfying result (Fig. 4). The 3D reconstruction is complete for a frame rate of 1 and 2, and also looks completely identical. The only imperfection are two little "wings" on the reconstructed person's chest. Higher frame rates produce insufficient results, because two frames are too dissimilar if the camera angle is too big. From Fig. 4 it is also evident that the camera position estimate changes, since the reconstructed point cloud is rotated along the y-axis. The reason for this is that the algorithm assumes that the camera is positioned on an axis going through the middle plane of the last two merged frames, since the accumulated point cloud is iteratively matched to the most recent two merged frames.

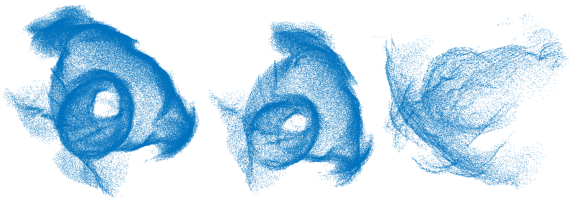


Figure 4: Superior view (X-Z-plane) on the 3D-reconstructed human body. From left to right, the frame rate is 2,4 and 10. The quality of the reconstruction decreases with increasing frame rate. In these pictures, the point cloud density is reduced by 90%, in order to make the geometry of the body visible. The head is in the center, the shoulders are left and right, the back is up and the chest is facing down.

Method 2

This method was implemented as described in the assignment description. Similar to method 1, to merge the accumulated point cloud with the newest frame, the rotation matrix and translation vector were computed for the newest frame with the accumulated frames as target. The accumulated frames were then matched to the newest frame via the inverse rotation and translation (see code). The algorithm only produced a satisfying outcome if a frame rate of 1 was used, as well as no sampling (all points). The outcome of the algorithm is depicted in figure 5. In effect, the quality of the resulting reconstruction is comparable to the outcome of method 1. Method 1, however, also produced a satisfying result for a frame rate of 2. The camera position is also slightly different. Overall, method 1 was faster to compute, since it allowed a higher frame rate.

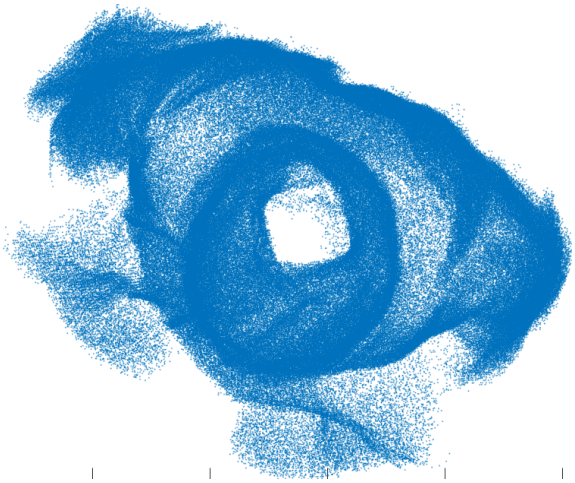


Figure 5: Superior view (X-Z axis) of the resulting reconstruction via the 2nd method. For this image, the point density was reduced to 10% after the final point cloud was computed, in order to make the contours more visible.

Questions

What are the drawbacks of the ICP algorithm? The ICP algorithm implemented in this project has multiple drawbacks, that will be discussed here:

- *long computation times*, as every iteration requires a sampling and matching step, that are computationally intensive. After the sampling method, that has been shown to take a considerate amount of time in some cases, is followed by a point-to-point matching that if not well optimized could become a bottleneck.
- *priori knowledge of the problem*, in order to choose the best sampling method to use. As explained in the **Informative regions sampling** subsection, if a point cloud does not have a high number of imperfections the normals sampling, for example, does not perform well. Uniform sampling, similarly, does not give good results in non flat regions (as implemented in this project), so an object with multiple inclinations is not well suited for it.

How do you think the ICP algorithm can be improved in terms of efficiency and accuracy? In order to increase the algorithm performance, some useful techniques can be implemented to increase its speed: for example, as implemented in this project, a KD tree data structure can be built on the data in order to give considerable speedups. When using the normals sampling method, all the normals have been centered in zero, in order to skip the centering at every matching step of the algorithm. Also, other interesting sampling methods can be implemented, by analyzing the properties of the input cloud to extract features that are always fixed. Alternatively, principal component analysis on the points could give interesting insights and a sampling method built on it may give good results.