Computer Vision 2 - Iterative Closest Point - ICP (lab 1)

Kassapis, Elias **Verdenius, Stijn** **Ayoughi, Melika**

1 Introduction

In this assignment, we will implement the ICP algorithm and some of its variants to estimate the camera poses for given point-clouds. We will analyze various aspects and design experimental setup. Finally, using consecutive frames from a person, we recontruct the 3D image of that person by merging the scenes, obtaining a qualitative representation of our implementation.

2 Iterative Closest Point - ICP

2.1 Implementation

We implement the ICP algorithm using Sorkine-Hornung and Rabinovich (2017). We start by applying a form of sub-sampling of k=40% of the points. The method of subsampling is one of the following:

- All points, which is self explanatory
- Uniform sub-sampling, which sub-samples points before iterations and sticks with those points
- Uniform sub-sampling in each iteration, which does the same as uniform sub-sampling but at each iteration
- Sub-sampling from informative regions in each iteration. Which is identical to the previous technique but not randomly chosen. For this we use Jana and Dalibor (2018), in which we take the dot product between points and their normals and then choose the kth largest ones. This is because these points are more likely to be identical according to Rusinkiewicz and Levoy (2001).

Thereafter, we will perform iterations of the ICP algorithm. We define an error threshold of $\epsilon = 0.00025$ and while the `RMSE` between the selected points of the source point clouds and targets is lower than ϵ we do the following: First we find the best matching points given a euclidean distance measure, then we fit a rotation matrix R and translation vector t using `SVD` given in (Sorkine-Hornung and Rabinovich, 2017) and finally we transform the source cloud points with the rotation matrix and translation vector and continue until the error goes under the threshold value of ϵ.

2.2 Analysis

To evaluate the performance of our ICP implementation, we measure accuracy, speed of convergence, stability and tolerance to noise. Accuracy is defined as the root mean squared error between source and target cloud points. Convergence is measured using number of iterations as well as total run-time. Tolerance to noise is examined by adding Gaussian noise to source point clouds and stability is examined at 45, 90 and 180 degrees rotation of point clouds with respect to x, y and z and translation with respect to x, y, z and all axes at the same time. We then see how the results are changed qualitatively. We obtained the following results (see Tables 1, 2 & 3) upon testing.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) R_x (b) R_y (c) R_x

Figure 1: Rotation matrices

By analyzing Table 1 we observe that the ICP is stable with different rotation and translations in different directions. The first line of the table represents the basic setting with no rotation and translation. The longest run-time belongs to 180 degrees rotation with respect to z axis and a translation of 1 in all directions.

Time(s)	Accuracy	Rotation	Translation
5.54	1.33e-05	0	0
5.86	1.32e-05	0	1s
5.70	1.33e-05	0	1-x
5.84	1.32e-05	0	1-y
5.94	1.33e-05	0	1-z
6.02	1.32e-05	45-x	0
5.90	1.32e-05	45-x	1s
5.89	1.33e-05	45-x	1-x
5.87	1.33e-05	45-x	1-y
5.61	1.32e-05	45-x	1-z
5.71	1.15e-05	45-y	0
5.59	1.14e-05	45-y	1s
5.58	1.16e-05	45-y	1-x
5.54	1.14e-05	45-y	1-y
5.64	1.16e-05	45-y	1-z
5.58	1.31e-05	45-z	0
5.56	1.29e-05	45-z	1s
5.56	1.30e-05	45-z	1-x
5.52	1.32e-05	45-z	1-y
5.59	1.31e-05	45-z	1-z
5.56	1.18e-05	90-x	0
5.76	1.18e-05	90-x	1s
5.76	1.19e-05	90-x	1-x
5.60	1.17e-05	90-x	1-y
5.55	1.17e-05	90-x	1-z
5.56	1.16e-05	90-y	0
5.60	1.15e-05	90-y	1s
5.57	1.18e-05	90-y	1-x
5.57	1.17e-05	90-y	1-y
5.65	1.15e-05	90-y	1-z
5.53	1.32e-05	90-z	0
5.54	1.33e-05	90-z	1s
5.58	1.31e-05	90-z	1-x
5.53	1.33e-05	90-z	1-y
5.53	1.31e-05	90-z	1-z
5.53	1.24e-05	180-x	0
5.57	1.24e-05	180-x	1s
5.57	1.25e-05	180-x	1-x
5.56	1.25e-05	180-x	1-y
5.53	1.24e-05	180-x	1-z
5.56	1.31e-05	180-y	0
5.65	1.30e-05	180-y	1s
5.60	1.30e-05	180-y	1-x
5.55	1.31e-05	180-y	1-y
5.55	1.31e-05	180-y	1-z
5.50	1.35e-05	180-z	0
6.44	1.30e-05	180-z	1s
5.62	1.32e-05	180-z	1-x
5.51	1.30e-05	180-z	1-y
5.51	1.33e-05	180-z	1-z

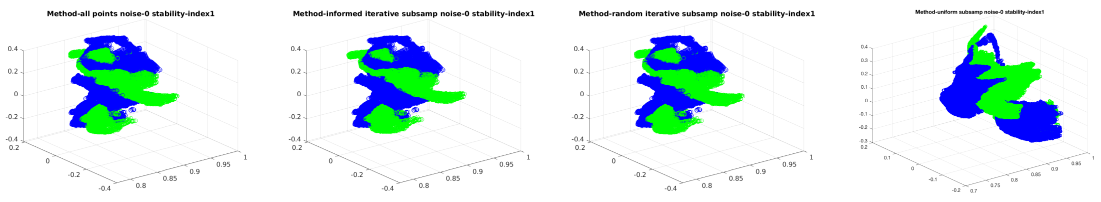
Table 1: Time and accuracy of ICP with the `uniform sampling` method and no noise with different rotation and translations.

Time(s)	Iteration	Accuracy	noise
74.53	1	0.001	0
220.52	3	0.001	1
213.22	3	0.001	2

Table 2: Time and accuracy of ICP with the `uniform sampling` method, no initial transformation and a Guassian noise with mean 0 and variance 0, 1 or 2.

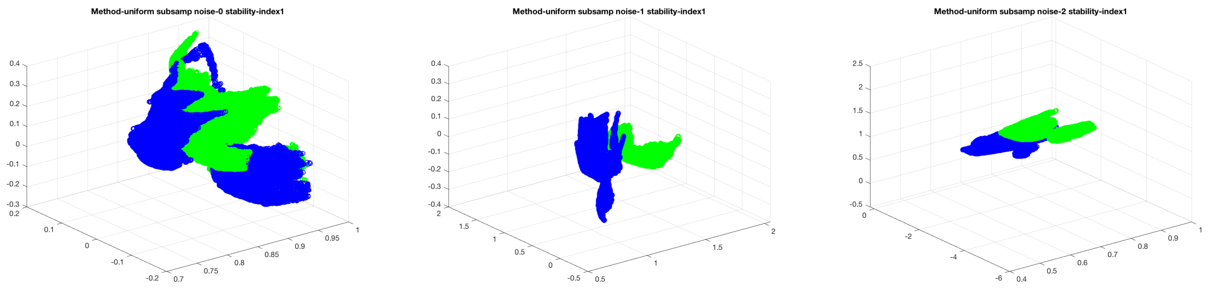
Time(s)	Accuracy	Subsampling
41.43	0.000184	random-subsample
40.19	0.000183	iterative-random-subsample
40.04	0.000164	informed-subsample
100.1	0.000116	all-points

Table 3: Time and accuracy of ICP with no noise, no initial transformation and different subsampling methods.



(a) All points (b) Informed iterative (c) Random iterative (d) Uniform

Figure 2: A comparison of different subsampling techniques.



(a) No noise (b) Guassian noise, mean=1 (c) Guassian noise, mean=2

Figure 3: A comparison of different guassian noises.

Unlike stability, the ICP performs worse as the guassian noise increases. This can be seen in Table 2 and in Figure 3. An interesting observation is that the accuracy at these points is almost 0. Another interesting observation is that the more the noise, the more time it takes to convergence. Number of iterations was not specified in any other table because it was 1 for all rows except for this table.

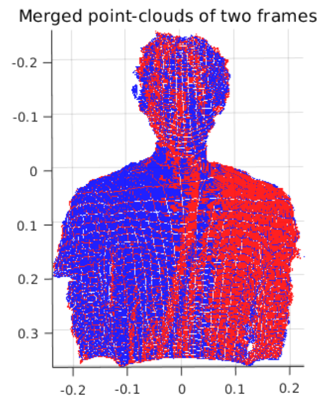
3 Merging Scenes

3.1

(a) For this part of the assignment, we used the ICP algorithm designed and implemented in section 2.1, in order to estimate the camera poses using each two consecutive frames of the given data. Subsequently, we used the estimated camera poses to align all the frames into an optimal alignment used to recreate the 3D structure that is the subject of the images. We tried to do this by propagating the corresponding transformation required to align each frame to the selected target frame, from the latter through the intermediate frames, all the way to the former. Using the final tranformations derived for each frame, we implemented these, and merged the point clouds by stacking their matrix representations in the column dimension. Our aim was to align all the frames to the last frame of the dataset. An illustration of merging the point clouds of two consecutive images, and an image showing the 3D plot of the result of merging the point clouds of all frames using the described method are shown below in Figure 4.

As we can see, the merged image on the right of Figure 6 is not very clear, providing an insufficient result. It appears like the transformations are not tranfered well to the frames by propagating from the target frame to all the other frames to be rotated. This is obvious by the fact that the merged image appear to be of the same subject; that is, a person in this case, but the point clouds indicate that the subject is taken at different angles. Therefore, our result illustrates that the camera pose estimation is not sufficiently accurate to allow 3D reconstruction of the subject of the image, as the derived transformations are not good enough to correctly align all the frames.

(b) Next, we estimate the camera pose and merge the results by sampling every 2nd, 4th, and 10th frame, as described in the assignment file. In this case, we iteratively merged the frames by starting from an initial base frame, and sampling the target frame as determined by the sampling rate. We derive the R and t vectors that reflect the change in camera pose and use them to transform the point cloud and normals of our current base frame to become aligned with our current target frame. These are then merged to form one point cloud as described in part (a). Now our target frame becomes our base frame and the next sampled frame becomes the new sample frame. We determine the change in camera pose in the same way as before, however, in this paradigm, we apply the



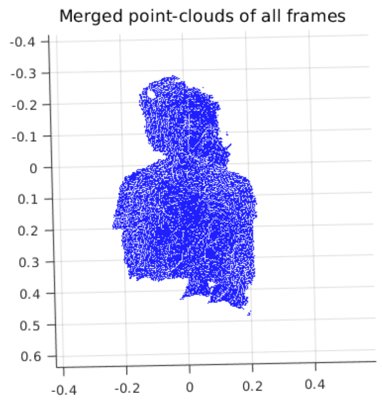


Figure 4: Left - Image showing the 3D plot of the merged point clouds of two consecutive frames from our dataset. The blue dots are from the first frame, and the red dots are from the second frame.
Right - Image showing the merged point clouds of all the frames using the method described above.

transformation using the newly derived R matrix and t vector on the point cloud that resulted from the merging implemented in the previous step. This proccess is iteratively carried out untill we reach the last frame in the dataset. For this part, we found that the informed iterative subsampling method worked best, and sampling 10% in order to speed up the ICP process. We ended up with using an ICP threshold of 0.005 because estimating the difference in camera pose when using a large framerate did not allow the ICP algorithm to converge if we used a lower threshold (which would result in better alignment). Hence, to avoid any bias in our results other than frame sampling rate we used the same threshold for all of our experiments in this part of the report. The results for the sampling rates of every 2nd, 4th and 10th frame are shown below in Figure 5, and we also show the results for sampling all images in the same method in figure 6.

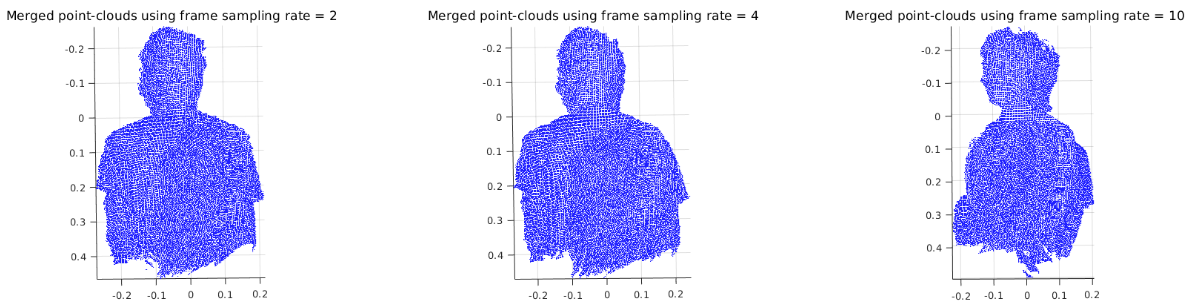


Figure 5: The 3D plots of the merged point clouds of images sampled from the data set using a sampling rate of 2 (left), 4 (centre), and 10 (right)). All these are from the same orientation.

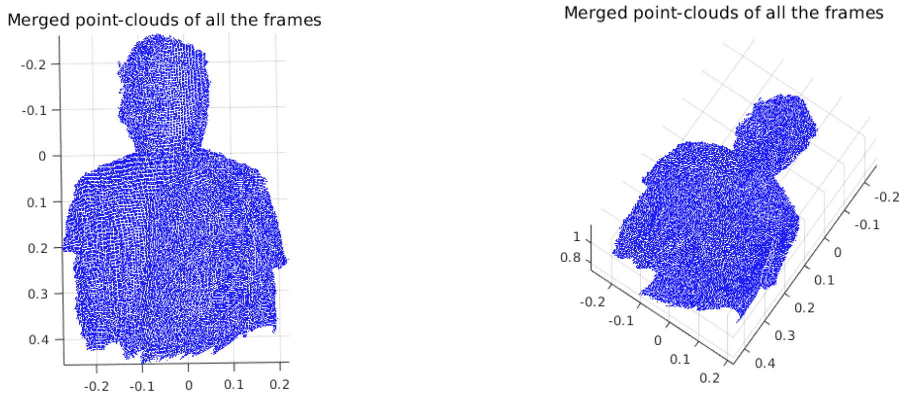


Figure 6: Two angles of the 3D plot constructed from merging the point clouds of all frames in the dataset with the method described above.

As we can see from the images above, as the frame sampling rate increases, our 3D model becomes more and more refined. This is probably because larger differences in alignment between frames used for camera pose estimation, increase it's error, which is only amplified when using the individual frames for its computation instead of the current merged point clouds. This point also agrees with what we have observed with the relationship of ICP threshold values and frame sampling rates, were low thresholds do not interact well with high sampling rates. This method of point cloud merging does however have the advantage of decreased computational expense.

3.2

For the final section of scene merging, we were instructed to iteratively merge and estimate the camera poses for the consecutive frames by using the same procedure as in the previous section, but instead of estimating the change in camera pose using the individual frames at each step and then transforming the aggregated merged cloud, we directly measure the camera pose between the next target frame and the current merged point cloud containing the aggregated point clouds from all the previously sampled frames. This approach took considerably longer as merging the point clouds by stacking their matrix representations in the 2nd dimension means that at every iteration the merged cloud matrix increases in size in a high rate, which increases the time required for the ICP algorithm to detect the R and t vectors defining the change in camera pose that satisfy the set threshold. To counter this effect we deleted the coordinates of the matrix that were almost identical by using the inbuilt MATLAB function *uniquetol*, on the assumption that after transforming the current merged point cloud to make it aligned to the current target frame, many points will be shared between the merged point clouds. Thereby, deleting these points will reduce the size of the merged cloud point matrix propagated to the next step, while keeping the important information gained by each merge. Unfortunately using this approach on all the data did not confer good results as illustrated in Figure 7. We therefore applied this merging method to only 10 consecutive frames instead of using all of the frames in our dataset. Our results are illustrated in Figure 7

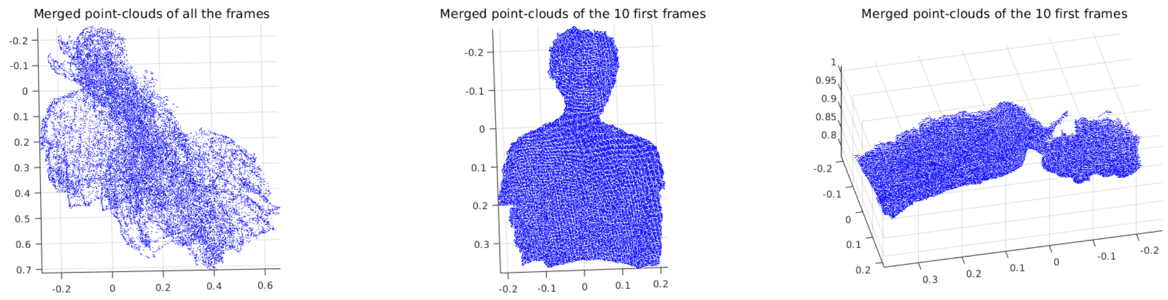


Figure 7: The result of plotting the merged point clouds of all the frames (left), and 10 frames (centre and right) from our dataset.

Our results at this part of the assignment are somewhat ambiguous. When we use this method for camera pose estimation, we can see that when using the whole data set we get what appears to be an elongated streak originating from the original 3D shape of the subject of the picture, which can be clearly viewed when using only the first 10 frames of the image, the results of which are shown in the central and rightmost images of Figure 7. We would expect the results of merging the point clouds of all of the frames to appear like these, as estimating the camera pose directly from the difference between the current merged point cloud, and current target frame point cloud, rather than using the previous target frame point cloud as a proxy to determine the change in camera pose would intuitively give a more accurate result. Although these 10 first frames do look promising, the fact that consecutive frames in the dataset are more closely aligned than sparser frames, does not allow us to make a solid conclusion about why the difference between the merged point clouds full set and 10 first frames occurs, and therefore what is the effect of this merging method to camera pose estimation. At the present time by weighting the evidence, we conclude that, counter-intuitively, this method increases the error in camera pose estimation, and this error is amplified as more transformations are measured and implemented. Unfortunately we did properly dissect the matter due to time constraints.

4 Questions

1. The ICP algorithm has several drawbacks (Halchenko) that we encounter some of them while implementing it. First, it is computationally demanding to identify the closest point for each source point at each iteration. Second, depending on the threshold value, convergence could be really slow; Thus, we need to find the perfect threshold for our experiments. Another important drawback that we observe is its sensitivity to outliers, so we achieve much better results only by ignoring them. Lastly, based on (Halchenko) the algorithm converges to local minima with some random initialization, while it could converge to global minima with a better initialization; Although we did not experience this in our experiments.

2. As explained in (Rusinkiewicz and Levoy, 2001), there are 6 possible family of variations to the original *ICP* algorithm that make it more accurate and faster to converge. These variations include changes in selection, matching, weighting, rejecting and the error metric. In this assignment, we experiment with different selection and rejecting mechanisms that will be explained in Section 5. Apart from the techniques mentioned in the paper, we can also use K-d trees and dynamic caching to speed up the closest point selection. We can also do stochastic ICP and simulated annealing. Another approach suggested by (Halchenko) is to use *L1* norm as the error function.

5 Additional Improvements

We make a few changes to the original *ICP* implementation to acheive better accuracy and faster convergence that will be explained in here:

- **Rejecting:** Normal and the point clouds with *NaN* values are filtered. In addition, the points with depth higher than 2 are also filtered. In order to speed-up the *ICP* iterations, we filtered out foreground points that according to our assumption, almost stay in the same place and don't give extra information about the transformation. The point matches that their mean distance was higher than 0.01 are also filtered. All these techniques reduce the number of outliers, thus leading to better results.
- **Selection:** We experimented with 4 different subsampling techniques: a) all the points b) uniform subsampling c) random subsampling d) informative subsampling. We observe that the *uniform subsampling* achieved adequate results in a short time.

6 Conclusion

In this assignment, we implemented the ICP algorithm and some of its variants to estimate the camera poses for given point-clouds. We analyzed various aspects and reported on them. Finally, using consecutive image frames from a person, we recontructed the 3D image of that person by merging the scenes.

References

Halchenko, Y. O. Iterative closest point algorithm. l1 error approach via linear programming.

Jana, P. and Dalibor, M. (2018). Notes on iterative closest point algorithm. In *Object recognition supported by user interaction for service robots*, volume 3, pages 545–548. Slovak University of Technology.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *3dim*, volume 1, pages 145–152.

Sorkine-Hornung, O. and Rabinovich, M. (2017). Least-squares rigid motion using svd. *Computing*, 1(1).

Workload:

- Elias: Merging Scenes and analysis
- Melika: ICP and analysis and additional improvements
- Stijn: ICP algorithm and test setup