# ICP
# Computer Vision 2

Nichita Diaconu (11737980), Radu Sibechi(11808527),
Ruben-Laurentiu Grab(11609923)

May 6, 2019

## 1 Introduction

Point cloud alignment is a fundamental problem for many applications in robotics and computer vision. Finding the global transformation is generally hard: point-to-point correspondences typically do not exist, the point clouds might only have partial overlap and the underlying objects themselves are often non-convex, leading to a potentially large number of alignment local minima.

Iterative closest point (ICP) is an algorithm that minimizes the difference between two clouds of points and as such can be used in order to align two clouds of points. The ICP algorithm always converges monotonically to the nearest local minimum of a mean square distance metric, and experiments have shown that the rate of convergence is rapid during the first few iterations.

We will analyze different ways to improve the efficiency and the accuracy of ICP metrics such as accuracy, speed, stability and tolerance to noise by changing the point selection technique. Furthermore, we will make use of the algorithm in order to estimate camera poses between two consecutive frames of the given dataset. Using these estimated camera poses, we will merge the point-clouds of all the scenes into one point cloud.

## 2 ICP

In the Iterative Closest Point, one point cloud, the target, is kept fixed, while the other one, the source, is transformed to best match the reference. The algorithm iteratively revises the spatial transformation (combination of translation and rotation) needed to minimize an error metric, such as the sum of squared differences between the correspondences of the matched pairs. The ICP algorithm was first introduced by Besl and McKay [1].

Essentialy, the algorithm steps can be reduced to:

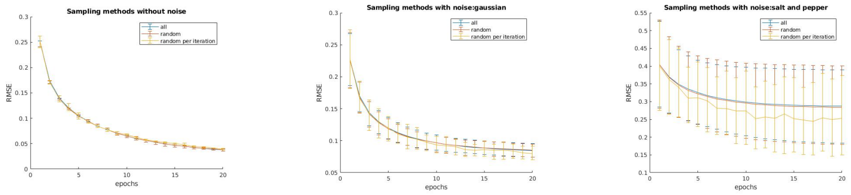
- for each point in the source point cloud, match the closest point in the target point cloud using the brute-force approach
- estimate the combination of rotation and translation using Singular Value Decomposition in order to minimize the square root distances between all matching points
- re-iterate until the RMS does not change

The algorithm solves the problem of finding correspondences by assuming that closest points correspond to each other and they are used in order to compute the best transformation. As such, the algorithm only converges if starting positions are "close enough".

## 2.1 ICP experiments

The algorithm gives us the liberty to choose the sampling method of the points in the point clouds. From here on, "all" means all the points in a point cloud are sampled, "random" means 10% of the points are randomly sampled, "random per iteration" means 10% of the points are sampled at every iteration of the ICP, "informative" means close to 10% of the points are sampled based on the surface normal.

We first evaluated our algorithm on 2 generated surfaces in order to see how the algorithm converges, results are shown in Figure 2.1.1. The error bars are computed over 6 different experiments of the same setting. As the results looked well, we added noise, Gaussian noise and salt and pepper noise to see how robust the method was, before applying it to real world data, the results can be also seen in Figure 2.1.1. Following [2] we also added a method for outlier detection in order to see wether it was stable and how well it performed, the outlier detection enhances the results when salt and pepper noise is applied, as we show in Figure 2.1.2.



(a) Sampling methods with no noise added

(b) Sampling methods with gaussian noise added

(c) Sampling methods with salt and pepper noise added

Figure 2.1.1 RMSE of various settings.

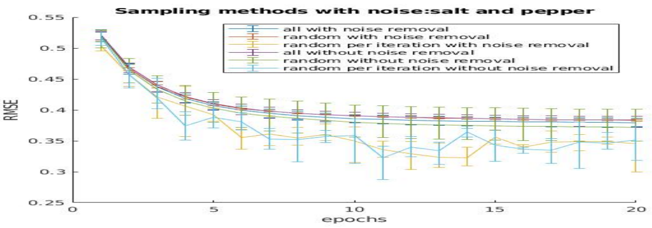
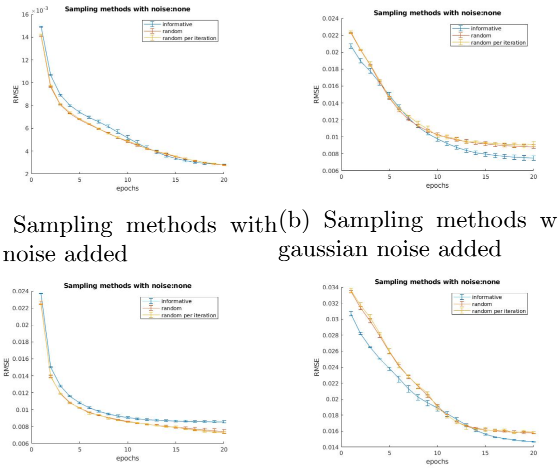


Figure 2.1.2 RMSE of various sampling methods on data with added salt and pepper noise.

We then tested the sampling methods on the real world data of a human. The sampling method "all" is not considered, as all the points are too demanding for our systems. Moreover, we only looked at a couple of pairs of frames to match. The results vor 3 different sampling methods are shown in Figure 2.1.3.



(a) Sampling methods with no noise added

(b) Sampling methods with gaussian noise added

(c) Sampling methods with salt and pepper noise added

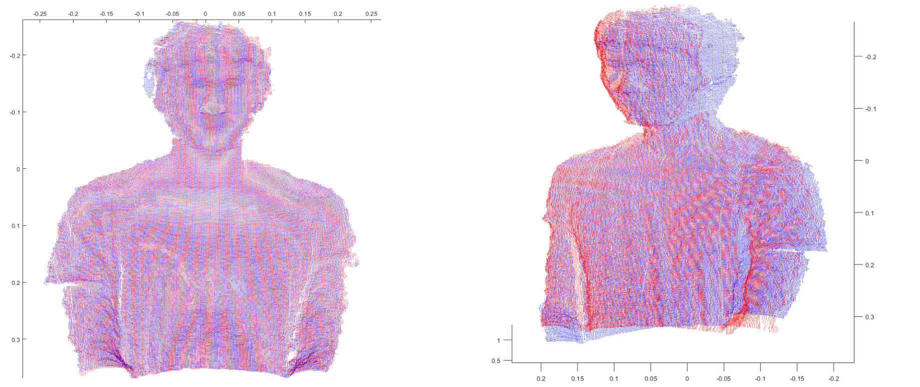
(d) Sampling methods with salt and pepper noise added

Figure 2.1.3 ICP results on frames 5-10, 15-20 ( top row from left to right ), 0-10, 10-20 (bottom row from left to right) for the 3 different sampling techniques.

# 3 Merging

## 3.1 Frame by frame pose estimation

Before running ICP algorithm on the whole dataset, we ran it on two frames, but with different distances between them. Figure 4 shows the result of our experiments. Figure 4a shows that the algorithm performed well, managing to find a rotation matrix and a transformation that approximates the pose change accurately. Indeed, the region of intersection between the two point clouds is very large, thus the good results obtained. Figure 4b shows that the algorithm is able to find a relatively accurate approximation of the camera pose change even when we skip a couple of frames, and the region of intersection is smaller. One can observe that the two point clouds(red cloud and blue could) capture the depth map of the human from different perspectives(i.e. the left side of the head is captured only by the blue point cloud). However, we can observe that there is is a small error in the latter approximation. We suspect that errors like this will accumulate over longer sequences of frames, yielding in poorer results.



(a) Two consecutive frames

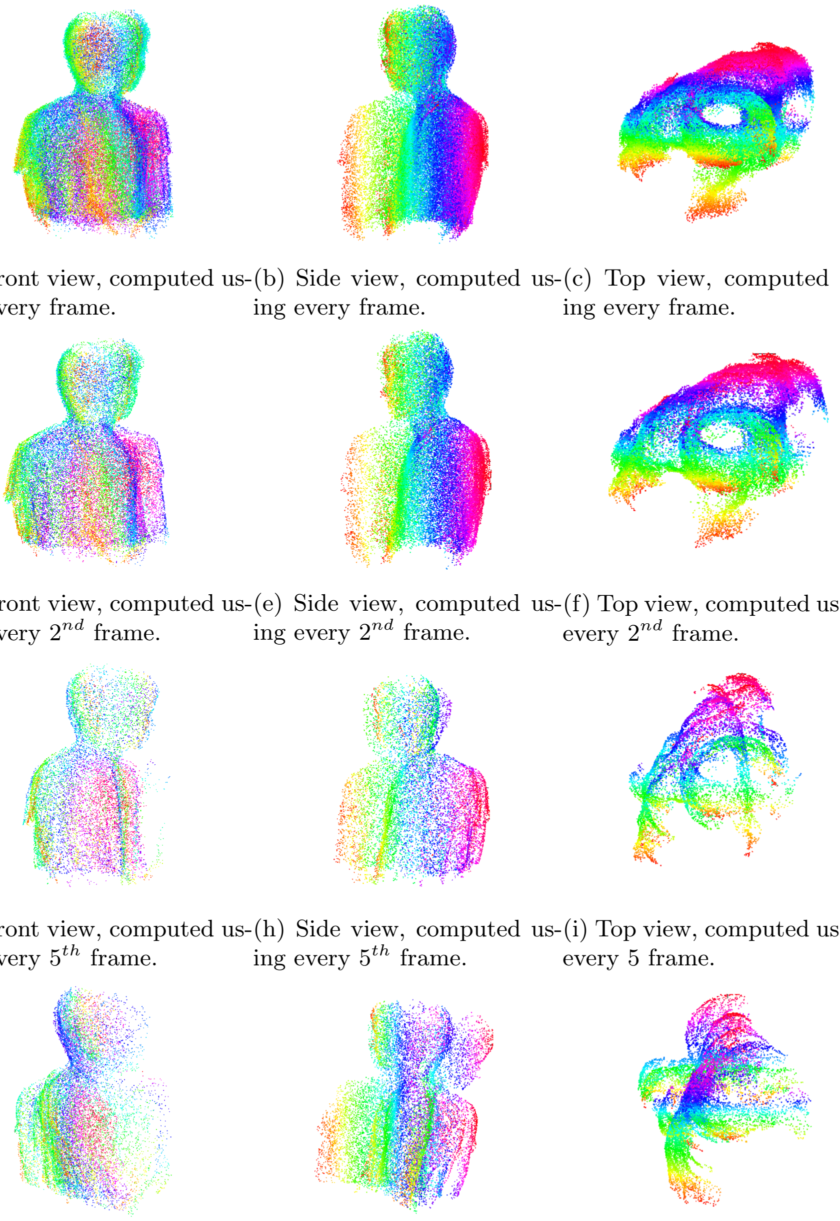
(b) Two frames at a distance of ten frames

Figure 4: Alignment results for two frames, at different distances between them

As required, we also ran experiments and tried to merge the point clouds(also referred to as frames) in one single mesh. We tried different setups, and used at first every frame. We further explored by choosing different frame sampling rates: we used ever second, every fifth and every tenth frame to compute the final mesh. The results can be seen in figure 5. We observed that for the last frames used in computing the rotation and translation, there were some errors in final rotation and translation. These can best be seen in the top view of the final mesh. We suspect that although the ICA algorithm performs relatively well, the error accumulates and the accuracy of the last frames' rotation and translation, will suffer the most.

Another aspect that has to be mentioned, and that is visible in the images, is that the more we skip frames, the more quality of the resulting mesh drops. We expected to see that for an either lower or higher number of skipped frames, the final mesh won't look as desired, and for about 5 frames, the result will satisfy our qualitative standards. However, we were surprised to see that computing the transformation for every frame or for every second frame yielded in the best results.

Besides using random sampling, we tried to use uniform sampling, with respect to the normals of the points. However, with our implementation of sampling the points from the mesh, the results didn't improve considerably, and the results were omitted from this paper.



(a) Front view, computed using every frame.

(b) Side view, computed using every frame.

(c) Top view, computed using every frame.

(d) Front view, computed using every $2^{nd}$ frame.

(e) Side view, computed using every $2^{nd}$ frame.

(f) Top view, computed using every $2^{nd}$ frame.

(g) Front view, computed using every $5^{th}$ frame.

(h) Side view, computed using every $5^{th}$ frame.

(i) Top view, computed using every 5 frame.

(j) Front view, computed using every $10^{th}$ frame.

(k) Side view, computed using every $10^{th}$ frame.

(l) Top view, computed using every $10^{th}$ frame.

Figure 5: Results showing the merging of frames for different settings of the frame step used, for random sampling of the points. For better display, for one frame, only 2000 points were randomly chosen to be shown in the plots, and out of all the frames used for computing the transformation parameters, every third frame was displayed.

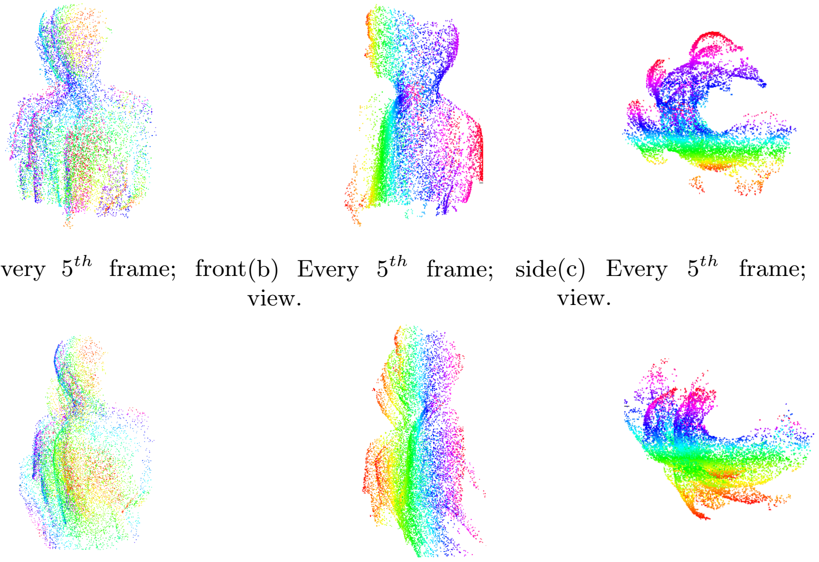
## 3.2 Cumulative frames pose estimation

When estimating the camera poses using the cumulative merging of the previous frames, the results change. Unfortunately, we did not see major improvements. In some cases, the results of the merge was even worse that when using just 2 frames for estimating the pose change. We believe that the error accumulates from two reasons.

Firstly, as mentioned in subsection **??** the parameters for rotation and translation accumulate estimation errors, and for the last frames, the error greatly hinders the result.

Secondly, because of the accumulated error the target frame that is made of the previous frames merged together, is erroneous and further hinders a good match with the new frame. Although we expected this technique to overcome the disadvantage of skipping frames, the errors introduces weight more.

Because at every new frame, we had to do the matching on all the previous points, and we lack the computational and time resources, we decided to experiment with cumulative frames only taking every $5^{th}$ and every $10^{th}$ frame. The results can be seen in figure 6.



(a) Every $5^{th}$ frame; front view.

(b) Every $5^{th}$ frame; side view.

(c) Every $5^{th}$ frame; top view.

(d) Every $10^{th}$ frame; front view.

(e) Every $10^{th}$ frame; side view.

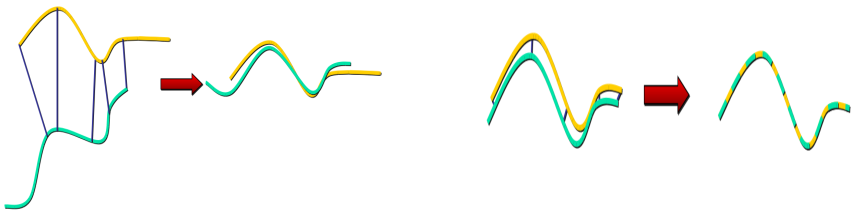
(f) Every $10^{th}$ frame; top view.

Figure 6: Results using cumulative frames. Due to limited computational resources, only the first half of frames was taken into consideration.

# 4 Questions

## 4.1 Drawbacks of ICP

- Only converges if starting position are "close enough" (see figure [7])



(a) "Bad" alignment

(b) "Good" alignment

Figure 7: Alignment results based on starting position

- Needs preprocessing steps (for triangulation, for mesh simplification, for 3D-trees)
- Finding of closest point pairs is a bottleneck and negatively affects overall algorithm speed

## 4.2 ICP further improvements

In their paper, Szymon Rusinkiewicz and Marc Levoy mentioned that improvements to this algorithm can be made at different stages, and regarding different processes of the algorithm. They mention that improvements can be made at the selection of the set of points in the meshes, at their matching and the level of importance of the points(points can have different weights), but also at the step of choosing a correct set of correspondences. Finally, they also state that the error metric can also play an important role at the robustness and speed of convergence of the algorithm, and they propose an improved metric.

### Sampling points

Although there are several policies of picking a subset of data points from the meshes(such as picking all available points, uniform sampling, random sampling, selection of points with high intensity gradients, etc), Szymon Rusinkiewicz and Marc Levoy introduce a new strategy: the sampling of points such that the distribution of normals among selected points is as large as possible. They mentioned that choosing sample points from both meshes can improve the algorithm.

### Matching points

As for matching points, they mention several strategies that can be used: finding the closest point in the other mesh, a strategy named “normal shooting” and another which involves the projection of the source point onto the destination mesh, from the point of view of the destination mesh’s range camera.

It is regarding to this process that we thought at a possible improvement. Although they to mention the use of colour and intensity for the matching gog points, we believe that the use of the albedo or different colour spaces can also improve the accuracy of the algorithm, at least for clouds of points with large flat regions.

### Rejection of certain matchings

Regarding the rejection of some matching pairs, the paper mentioned the discarding of pairs whose distance exceeds a certain threshold, or the worst n% of pairs based on some metric.

As a potential improvement to this step, we believe that restricting a point to be matched by only one other point can lead to better results. However, certain heuristics must be implemented in order to ensure that this policy doesn’t break or cancel some of the improvements brought by other strategies.

## 5 Conclusion

ICP is a powerful algorithm for calculating the displacement between point clouds. One major drawback of the algorithm is to determine the matching between points in the source cloud to points in the target cloud. Once these matchings have been determined, the transformation can be computed efficiently using SVD. We tried to match point clouds of a human from different angles and managed to do this to some degree. On the one hand, the algorithm performs more smoothly when, frames are close to each other, but the error accumulates and there is also a part of the human, close to the end of the frames that seems to be missing. On the other hand, if we choose sparser frames, there is less error to accumulate and the gap is note there anymore at the end of the human, but the algorithm performs worse ( quite badly when we sample every 10 frames ) on a frame to frame basis.

## 6 Self-contribution

We all feel that the work has been shared equally among ourselves.

# 7 References

[1] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 14:239–256, 1992.

[2] Pulli, K. “Multiview Registration for Large Data Sets,” Proc. 3DIM, 1999.