Computer Vision 2 Assignment 1

Kasper Bouwens Wolf Vos

April 2016

1 Iterative Closest Point

In this assignment we were given the task to combine point clouds, retrieved with a 3d camera, into a full 3d model of the person being photographed. To achieve this task, the Iterative Closest Point (ICP) algorithm is implemented. This section describes the examination of the ICP algorithm with different parameters and settings. These first tests were performed on the provided data test.mat and source.mat

The ICP algorithm tries to find the translation and rotation between two 3-dimentional shapes. Because our algorithm never reached a mean distance metrix between matches, smaller than 0.0012 we used an other convergence criterium. If the mean distance between matched points did not go down with by a number greater than 0.0001 our ICP algortihm would stop.

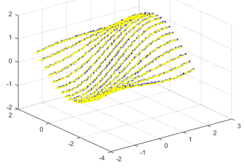
We have implemented multiple sampling methods to find matches. The first method we used was uniform subsampling with different interval lengths. We listed them in table 1. As you can see the average distance get's bigger when a larger interval is used. This captures the intuition that subsampling results in worse matching because a large portion of the points that could turn out to be good matches are being omitted.

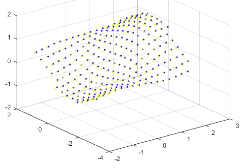
The second method we used was random subsampling. We tried this with 1280 random points 320 random points and 128 random points because these correspond to the number of point that were sampled with 5, 20 and 50 point intervals respectively. As seen in table 1 random subsampling performs much worse than uniform subsampling.

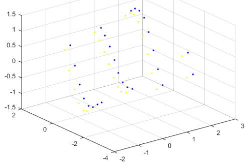
Finally we devised our own type of point matching which, like the baseline method matched each point from the base cloud to a point in the target cloud. Then, we looked at each point in the base cloud and only used the match with each point that had the shortest distance.

Length of intervals (number of points)	number of iterations	average distance
all points used	27	0.0268
5 (1280)	23	0.0268
20 (320)	23	0.0273
50 (128)	7	0.0873

Table 1: Uniform Subsampling


(a) 5 point interval


(b) 20 point interval


(c) 50 point interval

number of random samples	number of iterations	average distance
1280	27	0.0675
320	23	0.1267
128	17	0.2189

Table 2: Random Subsampling

number of matches	number of iterations	average distance
5622	39	0.0252

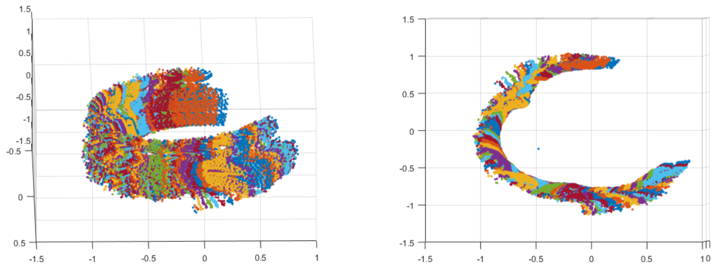
Table 3: Our Matching method

2 Merging scenes

2.1 Frame by frame

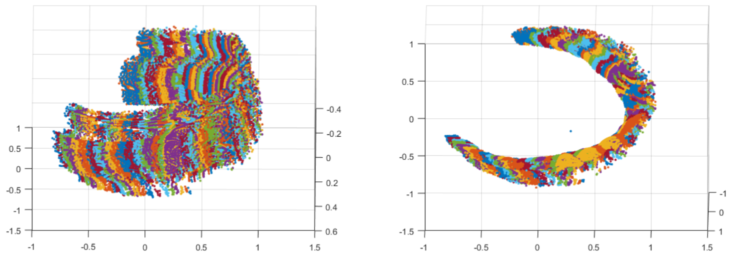
In an attempt to merge the points clouds the first step is to find the rotation matrix $\mathbf{R}$ and the translation vector $\vec{t}$ between two frames A_i (base) and A_{i+1} (target) with ICP.

In our algorithm the merge does not provide sufficient results as you can see in figure 2 and figure 3.



(a) Facing out (b) Facing out from the top

Figure 2: Rotation matrix inversed



(a) Facing in (b) Facing in from the top

Figure 3: Rotation matrix not inversed

This probably has to do with the fact that the error is accumulating over rotation matrices. Every rotation matrix has a small error which results in large errors at the final frame. This error is also increased because we used the inverse of the rotation matrix, which will (because of float computations) introduce even more errors. An attempt to solve the non closing loop can be done by using a loop-closure algorithm, see figure 4[1]. When using less frames (i.e. skipping some in between) the result will be more error prone.

[1]http://cogrob.ensta-paristech.fr/loopclosure.html

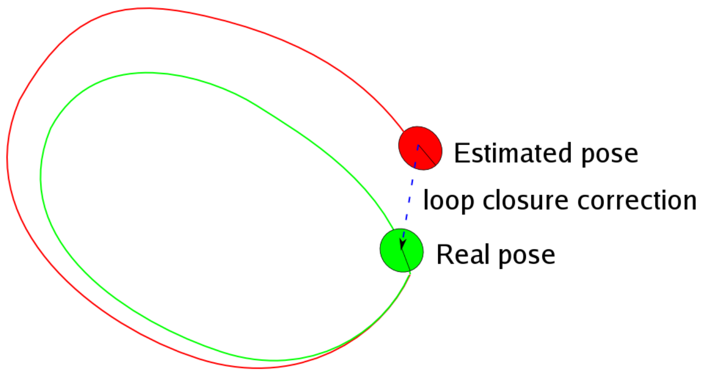
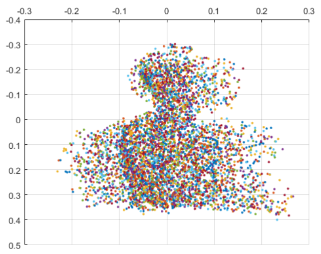


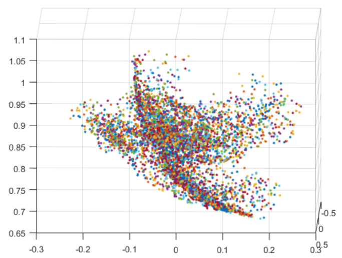
Figure 4: Illustration of loop closure correction

2.2 Cumulative

We have attempted two different approaches to iteratively merge and estimate the camera poses for consecutive frames. First we made *frame1* the global coordinate system. Then we matched *frame2* to *frame1* and saved the coordinates from *frame2* in the global coordinate system. Then we continued with the transformed point cloud of *frame2*, *frame2t*. We then match *frame3* on *frame2t* and so on until we reached the final frame. This worked for the first 20 frames but after that the error that was accumulated within the point cloud made matching very difficult and thus resulting in bad rotation and translation matrices. This can be observed in figure 5 (Only the first 50 frames are plotted otherwise it is just a big mess).



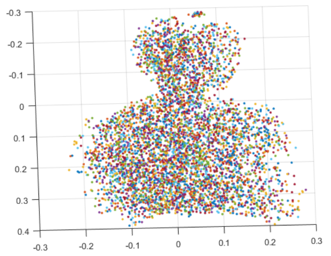
(a) Facing out



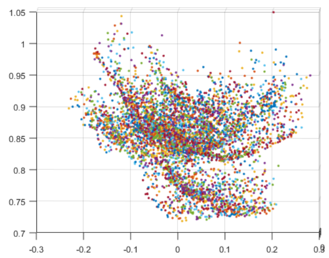
(b) Facing out from the top

Figure 5: Rotation matrix inversed

After that we attempted to do this in a accumulative manner. This means that we match *frame2* to *frame1* and add the points to the point cloud of *frame1*. Then we perform ICP to match *frame3* to *frame12*. This did not give better results, the points are a little bit different.



(a) Facing in



(b) Facing in from the top

Figure 6: Rotation matrix not inversed

3 Questions

What are the drawbacks of the ICP algorithm?

The drawbacks of ICP are that it will fail when you try to model a non rigid object or if there are big changes in the object location or camera pose.

How do you think the ICP algorithm can be improved, beside the techniques mentioned in [2], in terms of efficiency and accuracy?

We think that adding texture features could help match point clouds that are far away from eachother (i.e. too far for ICP to match them). Also loop-closure algorithms can be used to get better results.