3D Photography ICP Project

Steven Alsheimer

October 23, 2019

1 Practicing Rotations

The first part of testing the ICP algorithm is to test the direct translation between exact corresponding points. We choose a point cloud (this time a reduced version of gh17.ptx) and manually rotate it with a certain (proper) rotation matrix and then we select pairs of directly corresponding points and see if we can get back the translation and rotation we just gave to create gh17', a second point cloud rotated.

We applied a rotation of: $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

We applied a translation of: $\begin{bmatrix} 2 & 3 & 4 \end{bmatrix}$

Then we got the following image of two point clouds.

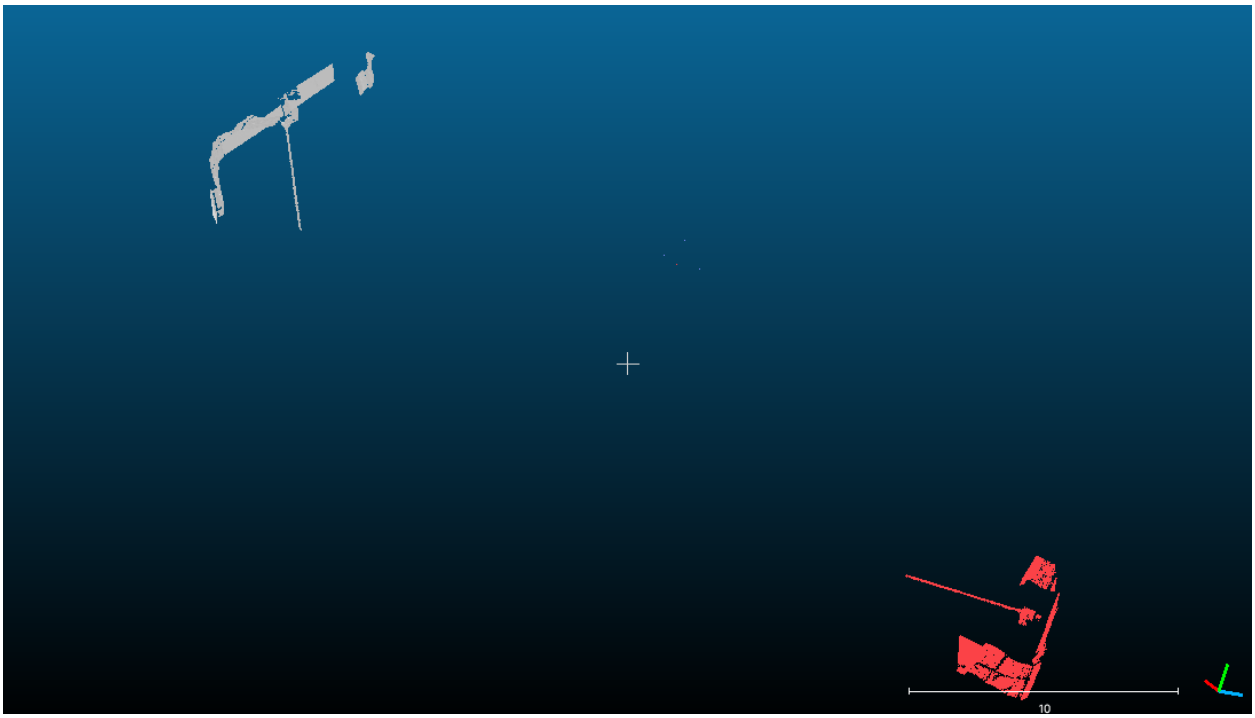

Figure 1: The red is the color after the manual rotation had been applied

After this we found 3 corresponding points in the grey point cloud and 3 in the red point cloud. We wanted to see if using the methods for directly translating and rotating a plane would we get back the rotation and translation we gave the point cloud to get the red point cloud.

So we applied the principle of:

$$R = R_r R_g^T \tag{1}$$

Where r is red and g is grey. After doing so with three points we knew were corresponding we got back our rotation and translation wit only some minor floating point errors.

2 Classic ICP

After knowing the direct corresponding points, and knowing how to perform proper transformations on these data sets we wanted to test this out on noisy data, or even better, data where we did not know the direct corresponding point. This means we have to guess the closest point, and iterate until we get the correct one. This is the essence of the Iterative Closest Point Method (ICP). Here we will guess corresponding points by assuming that the closest point between two data sets is the corresponding point. We know it wont be, and so we iterate, minimizing the RMS Squared Distance Error until we do choose the right point and the process converges.

2.1 The Algorithm

We sweet the following into a while loop until the process reaches a maximum number of iterations (usually 25) or converges (The difference in RMS Error is less than a threshold, 0.001). So: While Max i ¡ threshold and diff RMSE ¿ threshold:

1. Find 1000 random points from the left data set (sometimes more).
2. Find closest point between the 1000 and the right data set
3. Subtract the centroids from these data sets
4. Calculate the co-variance matrix of these sets, H
5. Minimise the RMSE by using SVD Decomposition to maximize the last term in the distance equation (math not shown here). We then find that maximizing the last term, thus minimizing RMSE means that R will be $R = VU^T$. Where V and U are unitary matrices that one can get from the SVD of H.
6. Find the rotation after RMSE minimization, and then apply that rotation to the set
7. Find the RMSE of the data set, find the difference of the RMSE from the last iteration
8. If the RMSE is not lower than 0.001 and we have not reached 25 iterations, we continue the loop.

2.2 Importance of Proper Parameters

There are a few parameters one must manually give to the script before running, these are

- The minimum RMSE change before convergence
- The cutoff distance beyond which points are thrown away
- Maximum Number of allowable iterations
- The number of randomly selected points

For the RMSE diff threshold we set it to 0.001, however on noisier or larger data sets we will set this to 0.005. This is the number that monitors when the algorithm has converged. If this number is too high it will converge too early, if too low it will never converge. The following numbers seemed to work well.

The Maximum number of allowable iterations is more depend on how much time you can space, for large data sets 25 iterations can take up to 40 minutes to run on large data sets (900k points).

The number of randomly selected points is very important to the statistical side effects one will get from this algorithm. Besl and Mckay said that theoretically the convergence should be monotonic. As you will see below it is often not. This is because as the data set gets larger the 1000 points sampling becomes a shockingly small subset of the point cloud. At 900k points, 1000 is a ridiculously small percentage of the point cloud. Thus this will lead to many errors, especially if the distance threshold is wrong. If you are noticing a very erratic convergence it is best to increase the sampling size to 3000 or in serious cases 5000. This is not to go over 5000, as the larger samples add time exponentially to the already costliest part of the algorithm.

This distance threshold is very important as well. A very low threshold will show some convergence will be stopped at a certain point. It will also not appear monotonic but erratic at too small distances (likely because convergence has already been reached for points at these distances). However, at too large a distance outlier points that do not have corresponding points will be linked to a point and yield inaccurate transformations. We found the going into the point cloud on cloud compare, and manually finding the distance between what appeared to be corresponding points and then increasing that some was the best method. This can likely be automated at a later point by taking all the distances of the corresponding points and finding the mean, and then taking a Gaussian distribution and sampling out the outliers after 3 standard deviations or so.

2.3 The Bunny Data

In order to properly test the ICP algorithm we used the MIT Bunny data set. This data set is only 40,000 points. The data set also has differently scaled ICP data for the bunny and the bunny rotated at 45 degrees. We did not solve for that here and that will thus lead to

some error after ICP is performed. We also manually rotated one of the bunnies, changing the pose only slightly and using a much smaller distance for the ICP distance threshold to see which did better.

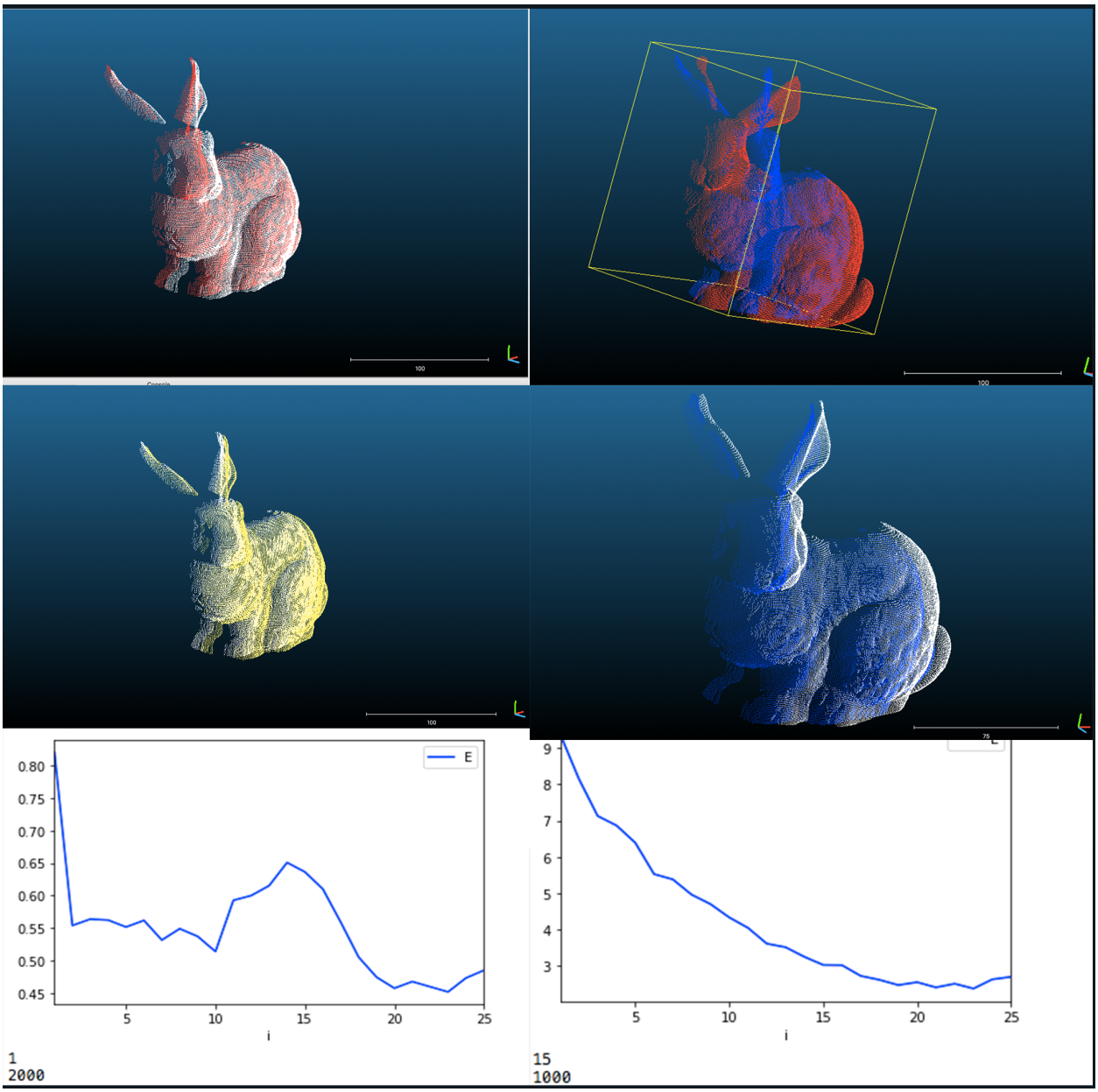

Figure 2: The right hand side is the bunny offset by only a slight pose and a small distance threshold. The left is the larger distance threshold and the 45 degree angle difference. The bottom plots show the convergences of the RMS Error.

It is clear that the larger data set converged more monotonically or smoothly, it also found a very similar fit to the one on the left. This is despite the fact that the left started with a much closer pose. Interestingly the middle hump on the bottom left graph is because as the points converged on the small distance threshold more points started to get added and

this increased the RMSE until those points were then properly transformed. This did not happen on the right plot since the distance was much greater. The lefts distance threshold was 1, while the rights was 15 (15 was determined using the method described previously). Another problem with choosing a very small distance threshold is that you must choose a larger point sampling and this often increases the cost and thus the time it takes to run the algorithm. IT is determined, although not decisively, that a larger threshold has a faster, monotonic convergence to the same pose that a smaller threshold and more points would, thus larger is better here.

2.4 Shepard Hall Data

The Shepard Hall Data consisted of three different data sets or point clouds (Gh 16, 17 and 23). We were given a rotation for 17 to 23 and so we used that to align 17 and 23 for ICP. We found that after the initial alignment the pose was so good that alignment occurred in only 2 iterations. After aligning 17 and 23 as per the given rotation we wanted to see if we could get

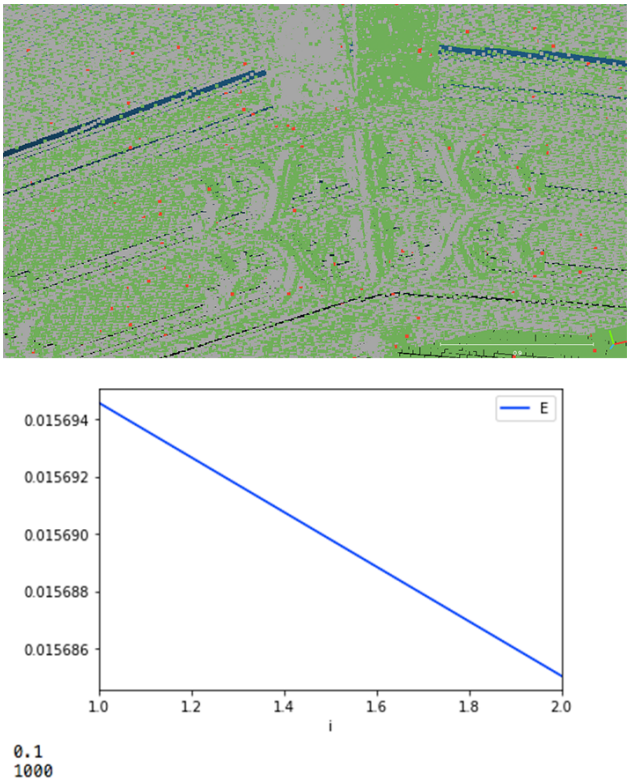

Figure 3: We only show the converged set here to conserve space and since the different was minimal. The red dots are the randomly sampled corresponding points that were chosen. The lower plot showed the rapid convergence of the 17 and 23 data sets after the given rotation was applied.

our ICP for a non-perfect pose but a good pose. So we went back into the CLoudCompare software and rotated and translated our point cloud until we felt it was sufficiently close, this took about 2 minutes. Then we used ICP to perfectly align the point cloud. This gave

much more noticeable results and took a few iterations to reach convergence.

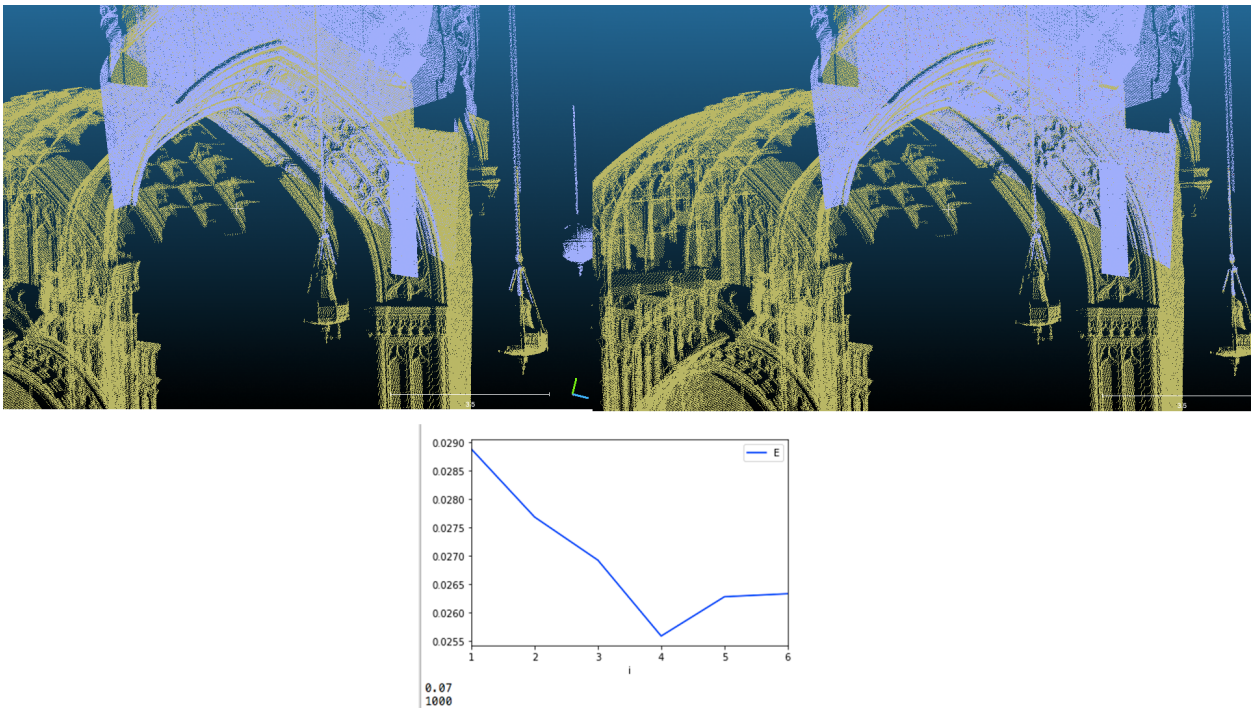

Figure 4: Left image is before ICP was done, right is after ICP. Bottom image shows the convergence plot of the RMS Error.

The above images show the point cloud of the combined GH 17 and 23 dataset that we merged after the ICP calculation done above. Then after manually translating and rotating gh17-23 toward GH 16 we ended up with the left image, where is is clear that the point clouds are not in perfect alignment since there is some grey on the wall of the upper arch/nave which should be all blue. This indicates some rotation of GH 17-23 relative to its optimal alignment position. So we calculated ICP on these two point clouds, and after six iterations the algorithm reached a convergence. The right image shows the result, as you can see the back arch is now evenly blue and even some detail in the decorative Gothic arch is now more crisp. Also the lamp in the lower right now has a much better alignment.

We do note that the convergence was a little erratic, this is due to the fact that we were choosing 1000 points at 0.07 distance in between two point clouds of over 800k points each, this lead to a higher variability of points to choose. So points chosen earlier may have been closer by happenstance then points chosen later on, before the convergence. At 1000 sample points Gh 16 and GH 17-23 took about 20 minutes to converge with a Python based ICP code.

We did label the red points here for corresponding points as well, however they are harder to see at this distance.