AER1515 Assignment 2 Report

Kevin Hu

November 17, 2023

1 Feature Point Detection and Correspondences

1.1 Feature Detection

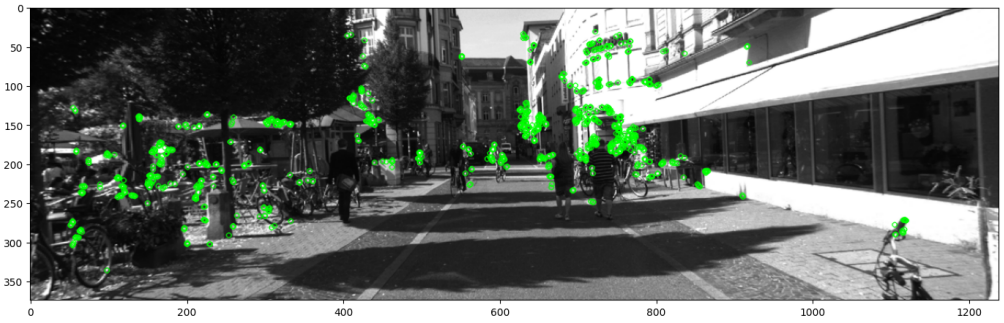

Figure 1: 1000 Keypoints for Test Image 1

Some of the features on the people and some of the features on the left may be hard to match since they may not be corners and the lighting is not the best on the left side. Some of the features far away may also be hard to detect and have an incorrect depth value.

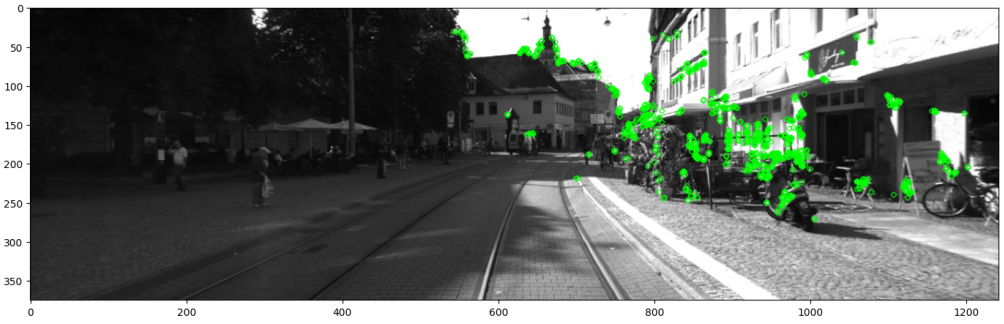

Figure 2: 1000 Keypoints for Test Image 2

Not many difficult features to identify here but some of the leaves may be challenging since there are some many of them and they may be quite similar.

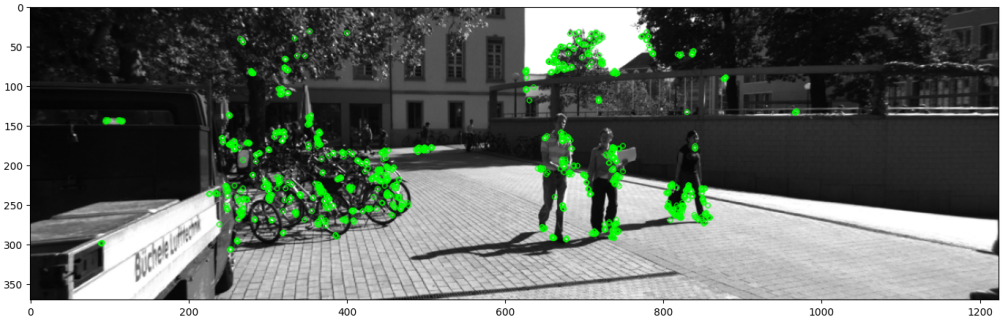

Figure 3: 1000 Keypoints for Test Image 3

Some of the features on the bikes may be hard to identify since the lighting may not be the best and some of the features are not corners.

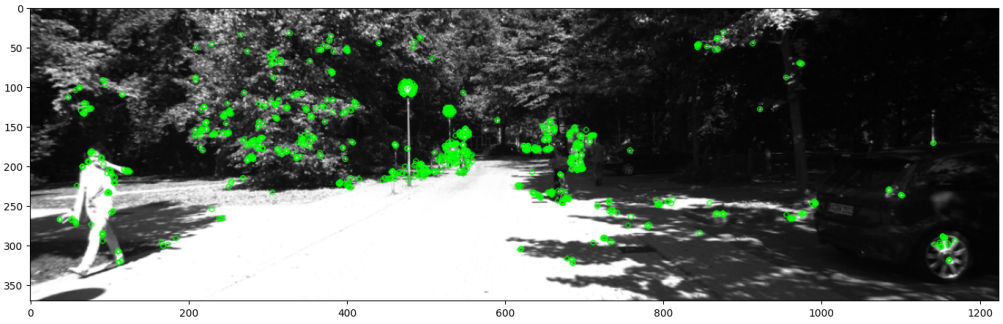

Figure 4: 1000 Keypoints for Test Image 4

Not too many difficult features here but the large number of leaf features may be hard to distinguish from each other.

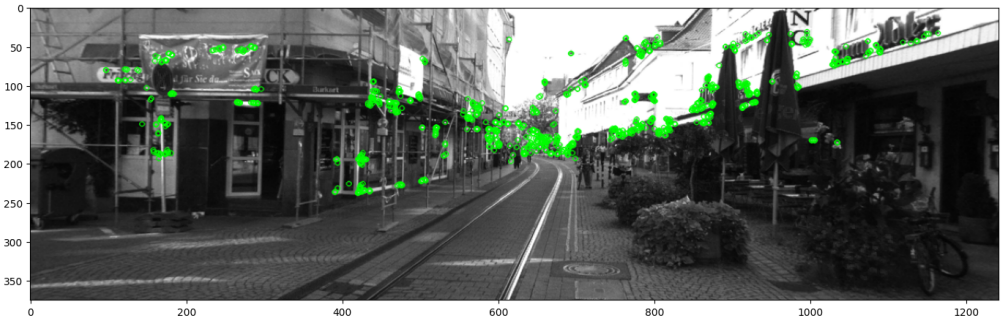

Figure 5: 1000 Keypoints for Test Image 5

The features far away may be difficult to identify and cause the depth to be inaccurate.

1.2 Feature Matching

The features from an ORB feature detector were matching using a brute force matcher matching Hamming distance with cross-check enabled meaning that it will only be a match if the feature on the left matches the feature on the right with the lowest Hamming distance and vice versa, instead of one way. An epipolar constraint was applied by making sure both features had the same y coordinate when matching and that the depth would not be inaccurate.

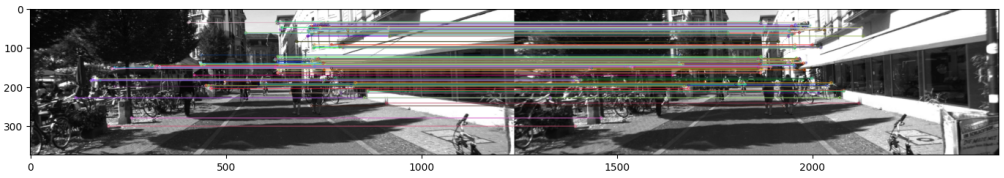

Figure 6: Feature Matches for Test Image 1

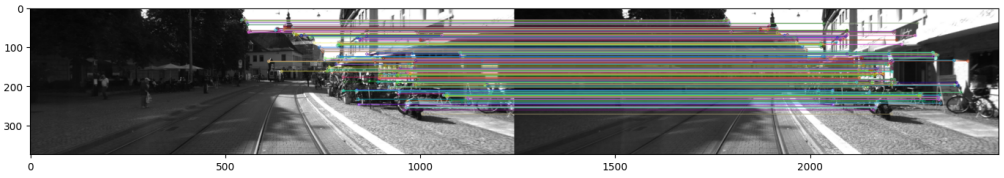

Figure 7: Feature Matches for Test Image 2

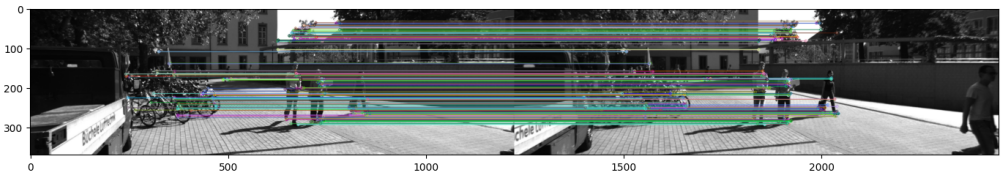

Figure 8: Feature Matches for Test Image 3

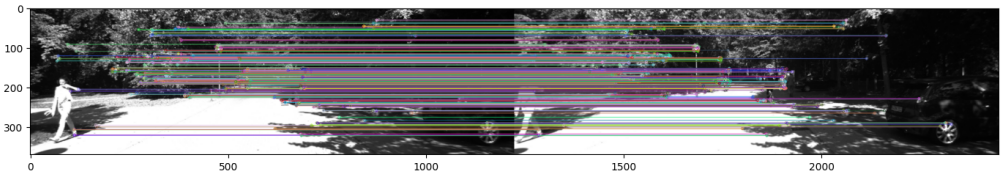

Figure 9: Feature Matches for Test Image 4

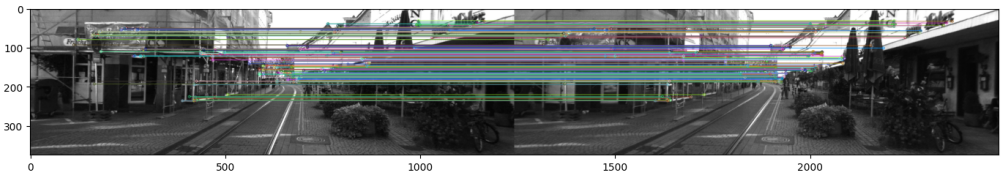

Figure 10: Feature Matches for Test Image 5

1.3 Outlier Rejection

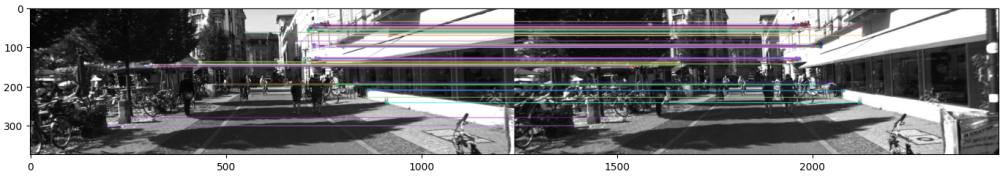

Figure 11: Feature Matches after Outlier Rejection for Test Image 1

Differences between outlier rejection algorithm and epipolar constraint is that that most of the matches on the left are gone and the matches far away are gone.

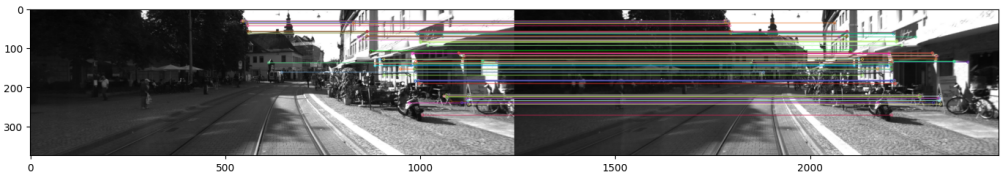

Figure 12: Feature Matches after Outlier Rejection for Test Image 2

Differences between the outlier rejection algorithm and the epipolar constraint is that most of the matches on the building are gone as well as the ones on the leaves/plant.

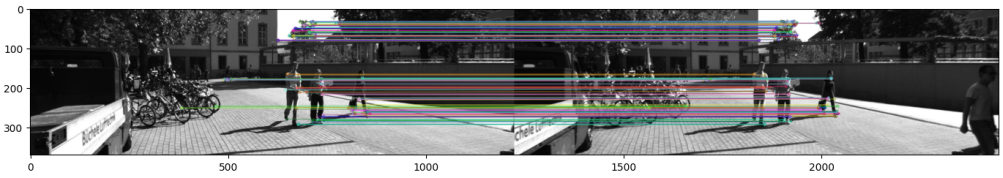

Figure 13: Feature Matches after Outlier Rejection for Test Image 3

Differences between the outlier rejection algorithm and the epipolar constraint is that most of the matches on the bikes on the left are gone.

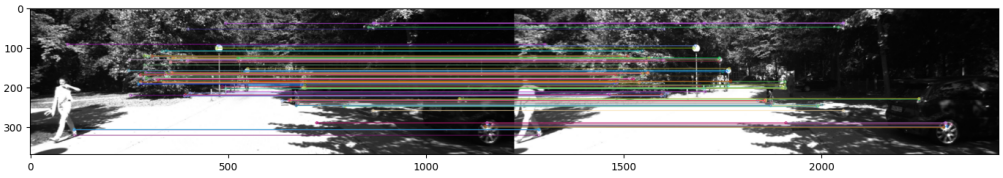

Figure 14: Feature Matches after Outlier Rejection for Test Image 4

Differences between the outlier rejection algorithm and the epipolar constraint is that most of the matches on the bushes behind the bush at the front are gone and some of the features on the signs are gone.

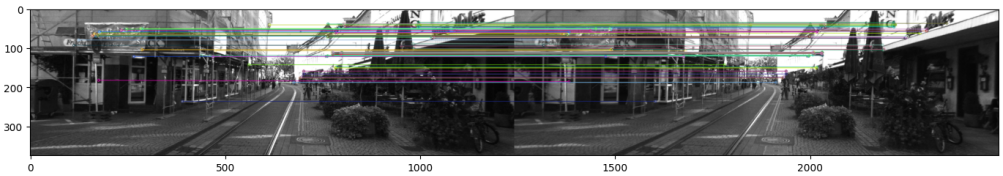

Figure 15: Feature Matches after Outlier Rejection for Test Image 5

Differences between the outlier rejection algorithm and the epipolar constraint is that the matches on the trees far away are gone.

The algorithm used for outlier rejection included doing the ratio test from D.Lowe paper on SIFT. What this algorithm does is that for each matching key point on the left it would find the best two matches on the right. If the best (lowest) Hamming distance is less than 0.7 times the second best (second lowest) Hamming distance it would be an inlier. Otherwise it would be an outlier. Additional steps for the outlier rejection included sorting the resulting matches by the Hamming distance difference between the matches and only taking the 130 best matches. After that there is a depth constraint of 80 meters that removes matches that would have inaccurate depths.

The performance of the outlier rejection algorithm is similar to the performance using the epipolar constraint since it seems to have gotten rid of all the outliers on a different y coordinate of the image. The difference is that there are a lot less matches and some of the difficult to match features like edges and leaves were rejected and the feature far away were rejected.

2 3D Point Cloud Registration

2.1 Nearest Neighbour Search

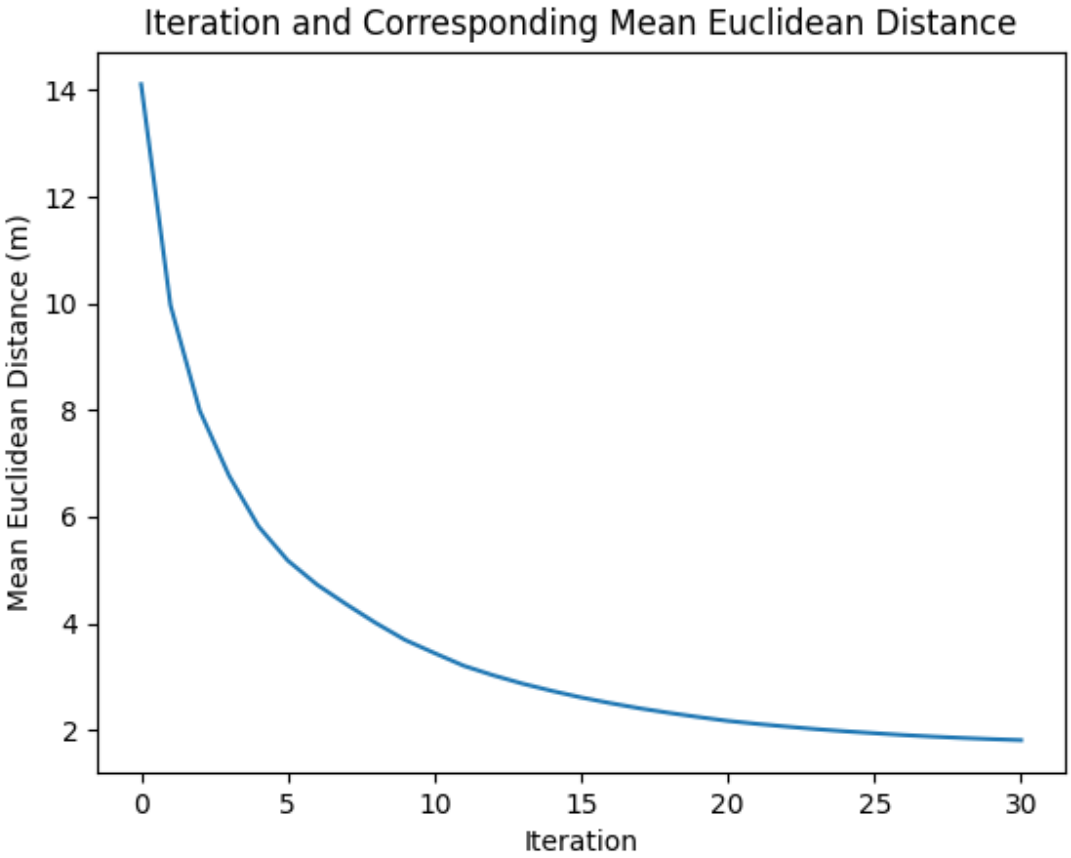

Figure 16: ICP Loss Curve for First Training Example

The ICP loss is monotonically decreasing, decreasing faster at the first few iterations and decreasing slower as the iteration number increases. The ICP loss converges to a value of 2. The reason for the loss trend because the correspondences are getting more accurate each iteration since the points are moving closer to their actual location and their nearest neighbour is more likely to be the correct location. Also the SVD method allows the transformation to get better by updating the transformation in the direction of least error according to a cost function each iteration.

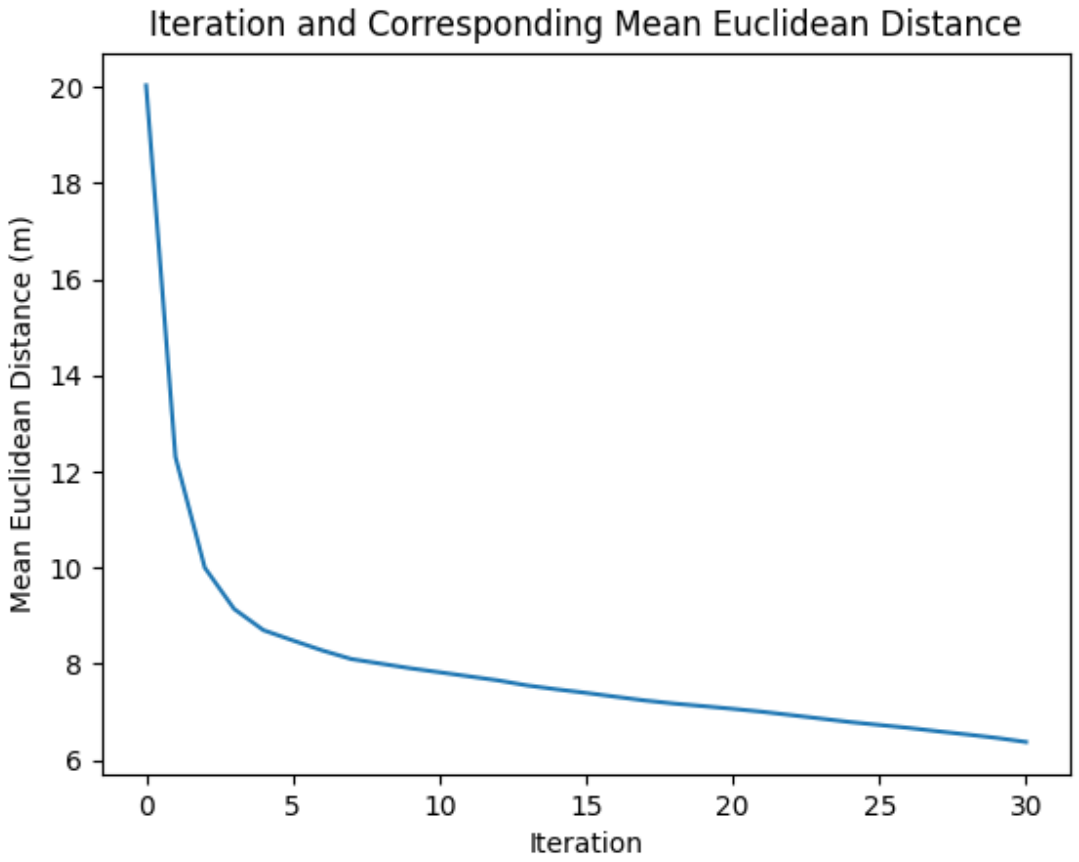

Figure 17: ICP Loss Curve for Second Training Example

The ICP loss is monotonically decreasing, decreasing faster at the first few iterations and decreasing slower as the iteration number increases. At around the 7th iteration, the loss starts decreasing almost linearly. By the 30th ICP iteration, the distance is still decreasing linearly and has not converged being at a loss of 7. The reason for the loss trend is the same as for the first training example.

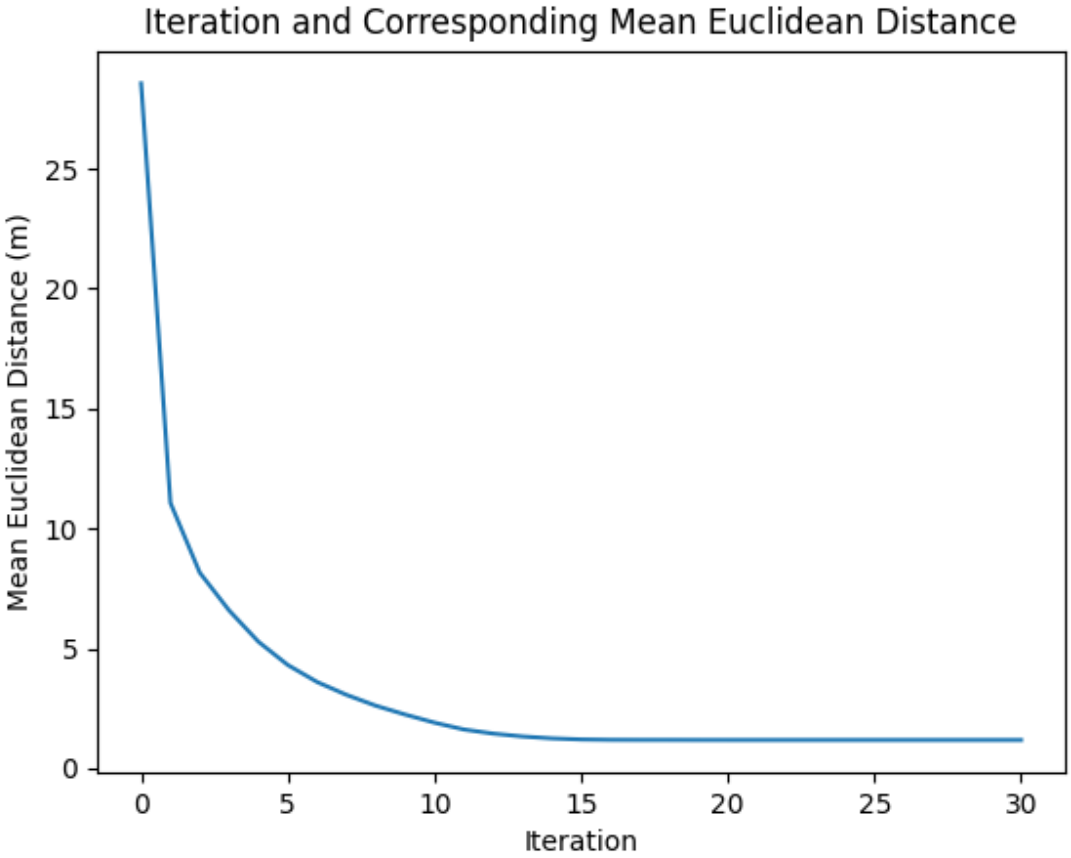

Figure 18: ICP Loss Curve for Test Data

The ICP loss is monotonically decreasing, decreasing faster at the first few iterations and decreasing slower as the iteration number increases. The ICP loss has completely converged to a value of around 2 by the 15th iteration. The reason for the loss trend is the same as the training examples.

2.2 Pose Estimation from the Correspondence

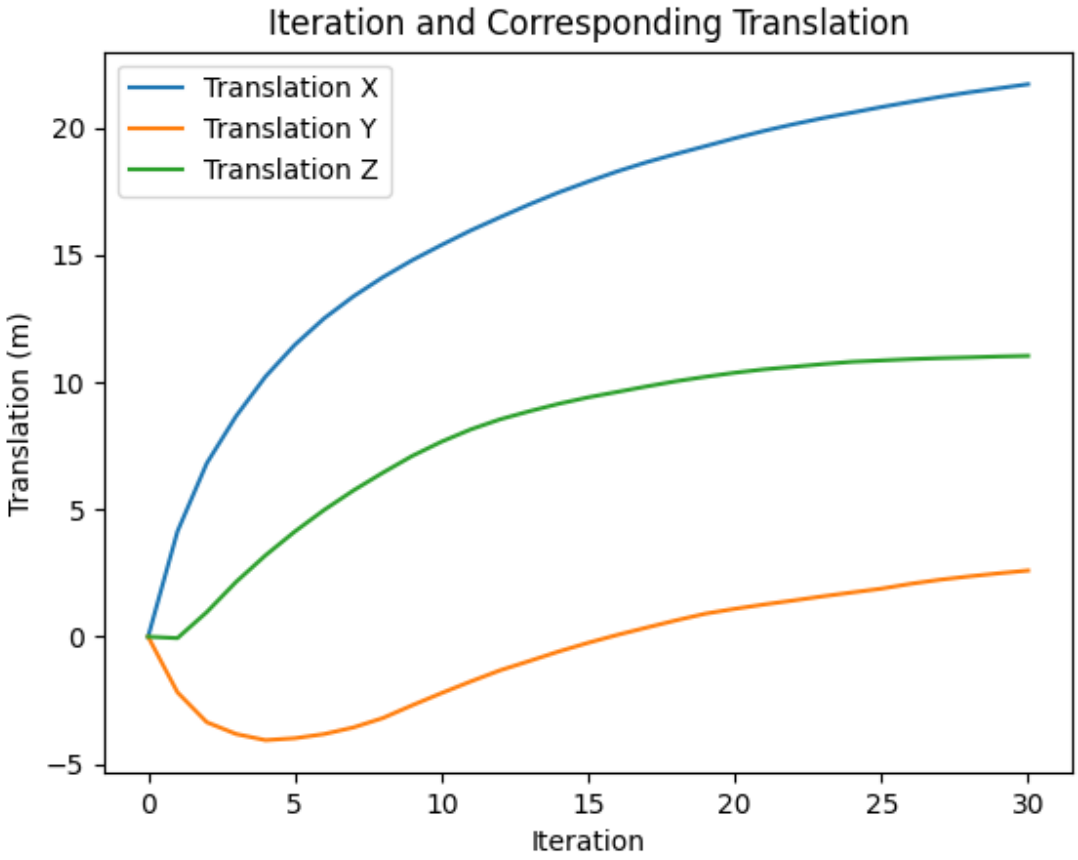

Figure 19: Estimated 3D Translation Per ICP Iteration for First Training Example

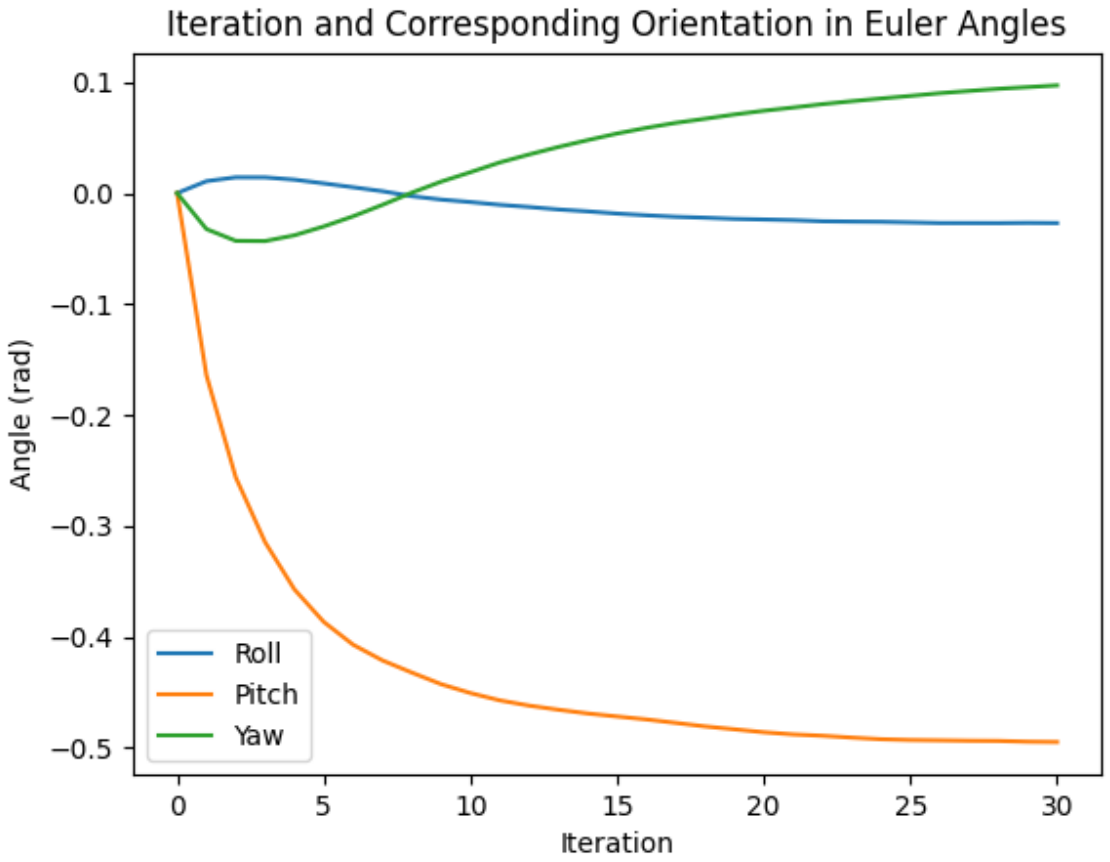

Figure 20: Estimated Rotation In Euler Angles Per ICP Iteration for First Training Example

The translation diverged in a smooth curve with the X and Z directions increasing and converging to 10 and over 20 while the Y translation first decreasing and then increasing until converging around 3. The pitch Euler angle decreased and converged to about -0.5 while the roll Euler angle increased slightly and then decreased and converged slightly below 0. The yaw Euler angle first decreased and then increased around the third or fourth iteration until converging to a value of around 0.1 which follows a similar trajectory as the Y translation. The reason for these poses changing directions is because at some point during ICP better correspondences were found, changing the direction of best convergence to the solution.

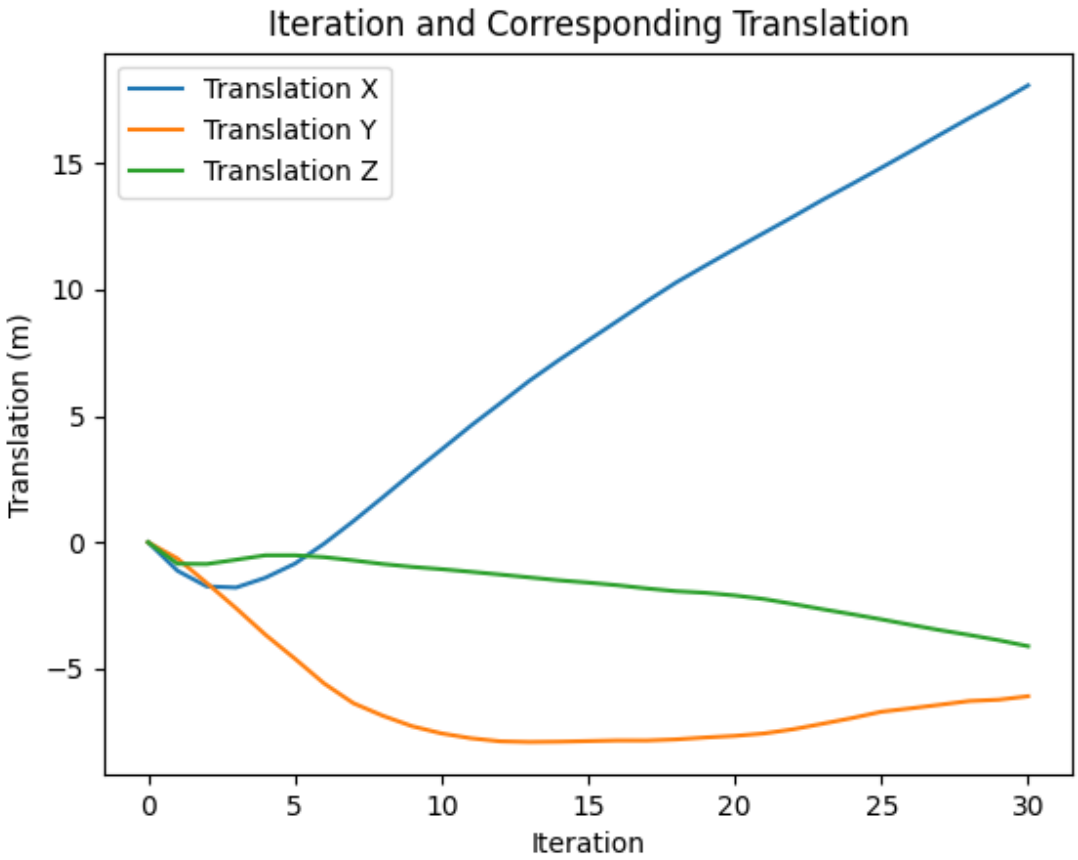

Figure 21: Estimated 3D Translation Per ICP Iteration for Second Training Example

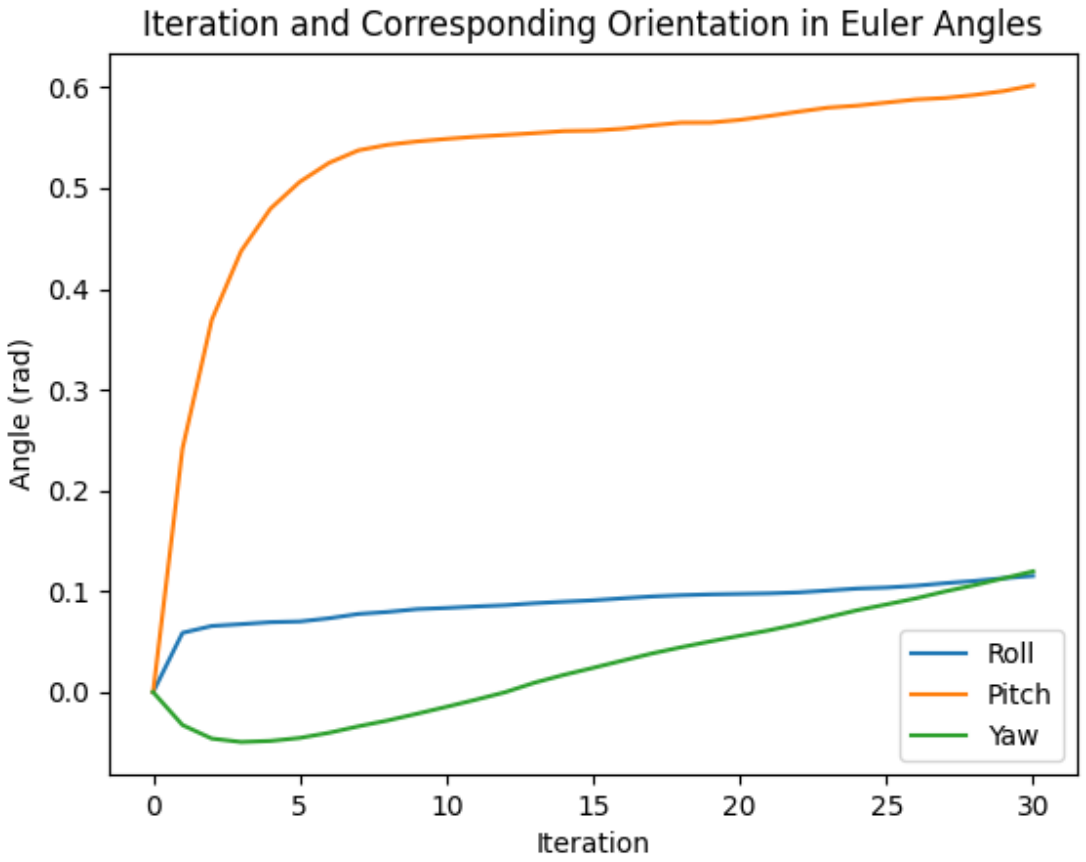

Figure 22: Estimated Rotation In Euler Angles Per ICP Iteration for Second Training Example

The X translation first decreased in the first three iterations and then started increasing linearly being around 20 with little signs of convergence. The Y translation decreased to around -9 by the 15th iteration and started increasing again up to around -6. The Z translation was decreasing at a slow rate at first but is decreasing at a faster rate by the 20th iteration down to -5 by the 30th iteration. The roll Euler angle, it is increasing slowly up to value of 0.1 rad. The pitch Euler angle first increased at a fast rate and is slowing down but is not converged yet at around 0.6 rad. The yaw Euler angle first decreased until the 3rd iteration and then started increasing up to 0.1 rad seeming to change direction at the same time as the X translation. The reason for the poses changing directions is the same as for the first training example.

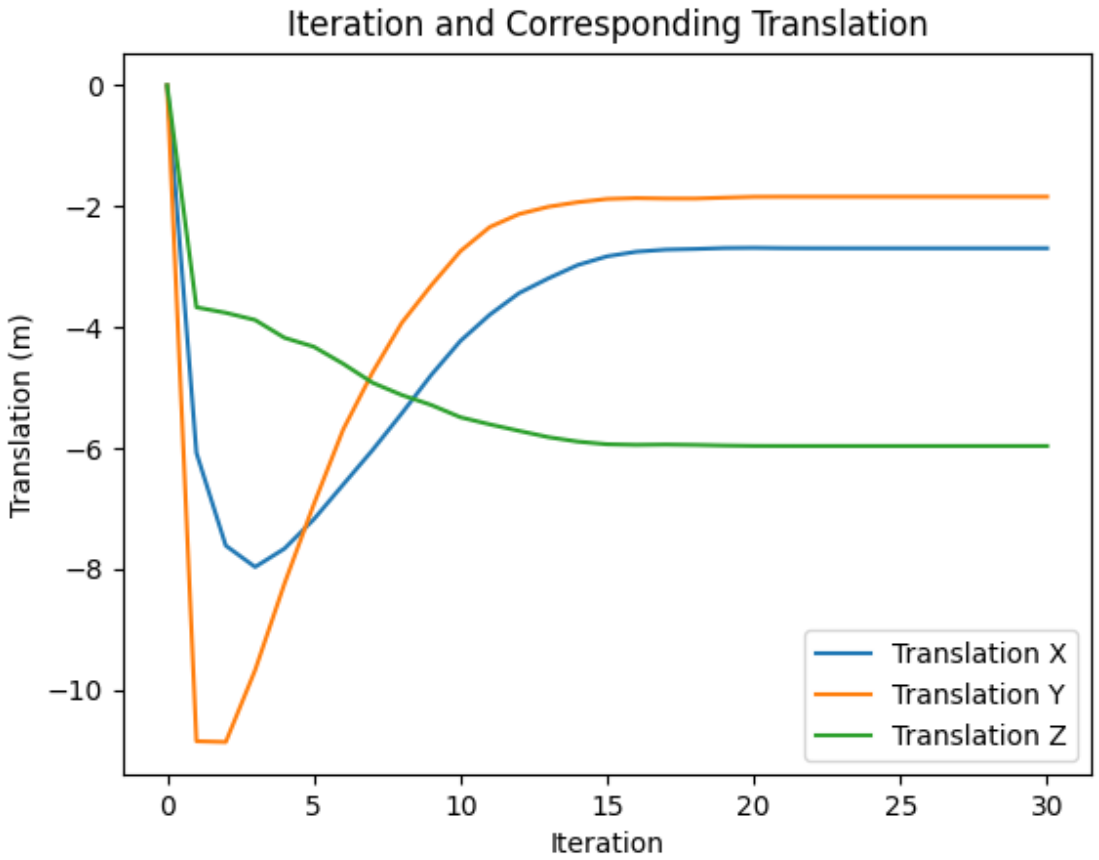

Figure 23: Estimated 3D Translation Per ICP Iteration for Test Data

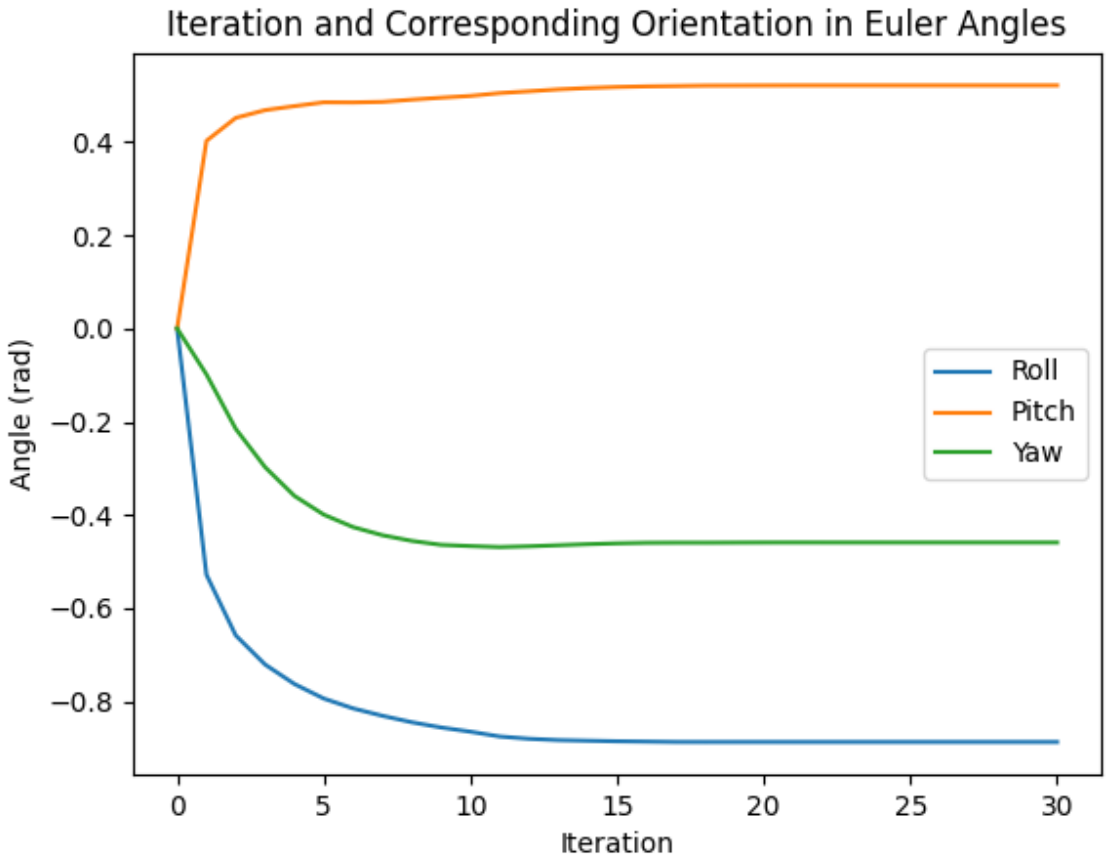

Figure 24: Estimated Rotation In Euler Angles Per ICP Iteration for Test Data

The translation for the test set is very interesting. All three translation directions decreased sharply for the first iteration down to around -4, -6, and -12 for the Z, X and Y directions respectively. The Z translation then started decreasing at a slower rate until converging to around -6 by iteration 15. The Y translation stayed the same for the 2nd iteration and then increased quickly until it converged to around -2 by the 15th iteration. The X translation kept decreasing after the first iteration but at a slower rate until the fourth iteration until -8 and then increased quickly until it converged to -3 by the 15th iteration. The rotation Euler angles behaved fairly normally with the roll and yaw angles decreasing and then converging to -0.9 and -0.5 respectively. The angle increased and then converged to around 0.5. The angles converged by around the 15th iteration as well. The reason for the poses changing directions is the same as the training examples.

2.3 Iterative Closest Point Registration

2.3.1 Visualizations

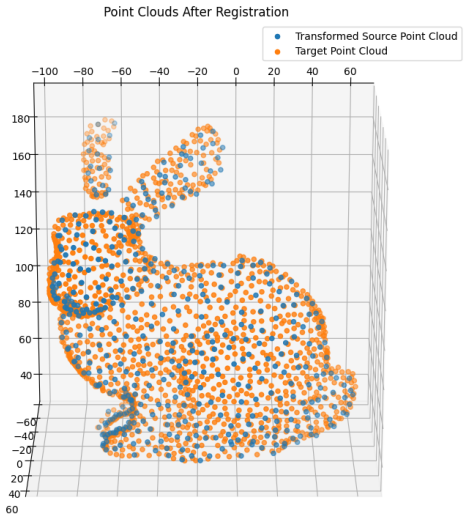

Figure 25: Visualization of the Registration Result for the Bunny

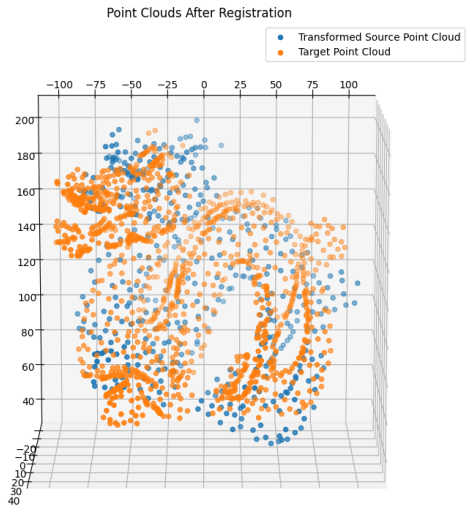

Figure 26: Visualization of the Registration Result for the Dragon

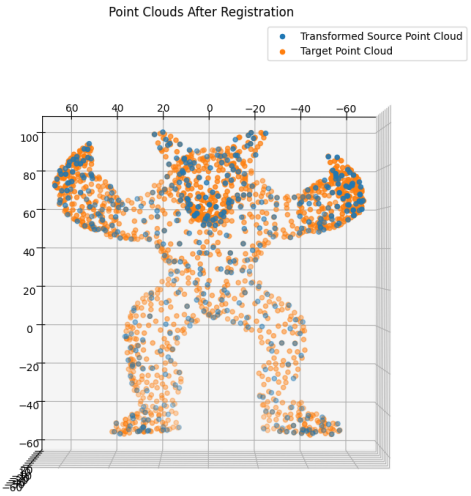

Figure 27: Visualization of the Registration Result for the Armadillo

2.3.2 Pose Error on the Training Sets

For the first training example the translation error were 3.04968597, 1.96330325, and -0.1694319 with a translation error norm of 3.63095721. The Euler angle rotation errors were 0.0154222, 0.00105453, and 0.02801828 with a rotation error norm of 0.0280805.

For the second training example the translation errors were 28.26918683, 9.41352565, and -2.3483948 with a translation error norm of 29.88772905. The Euler angle rotation errors were 0.03832554, 0.07725539, and 0.30320039 with a rotation error norm of 0.31522645.

2.3.3 Pose Output on the Test Set

The estimated pose on the test set is:

$$\begin{bmatrix} 0.77747525 & -0.06592832 & 0.6254484 & -2.69331663 \\ -0.38422036 & 0.73753312 & 0.55535539 & -1.84122822 \\ -0.49790256 & -0.67208507 & 0.54808275 & -5.96364711 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$