AER1515 Assignment 2

Alaa Hatoum
alaa.hatoum@mail.utoronto.ca
Student Number: 1004921855

October 2023

0.1 Part 1

Question 1

A feature detector in image processing identifies distinctive points or regions within an image, enabling tasks like object recognition or image stitching. In the context of the ORB (Oriented FAST and Rotated BRIEF) algorithm, feature detection involves the fusion of FAST keypoint detection and BRIEF descriptor generation, enhancing performance while addressing certain limitations.

ORB's key strength lies in its ability to handle rotation invariance. It utilizes FAST to locate keypoints and subsequently applies the Harris corner measure to select the most relevant ones. To address the issue of rotation invariance, ORB computes the orientation by determining the centroid of a patch around the corner point and utilizes moments within a circular region to enhance this invariance.

However, a potential issue arises with unreliable keypoints. These keypoints lack stability and can be erroneously matched to multiple unrelated features. For instance, consider a scenario where a keypoint on one leaf in an image might erroneously correspond to a different leaf, leading to inaccurate feature matching. Other Unreliable keypoints are any that are close to the edge of an image as some portion of the left image doesn't appear in the right image and vice versa and therefore cannot be matched.

Figure 1: Feature Detection for 011

Figure 2: Feature Detection for 012

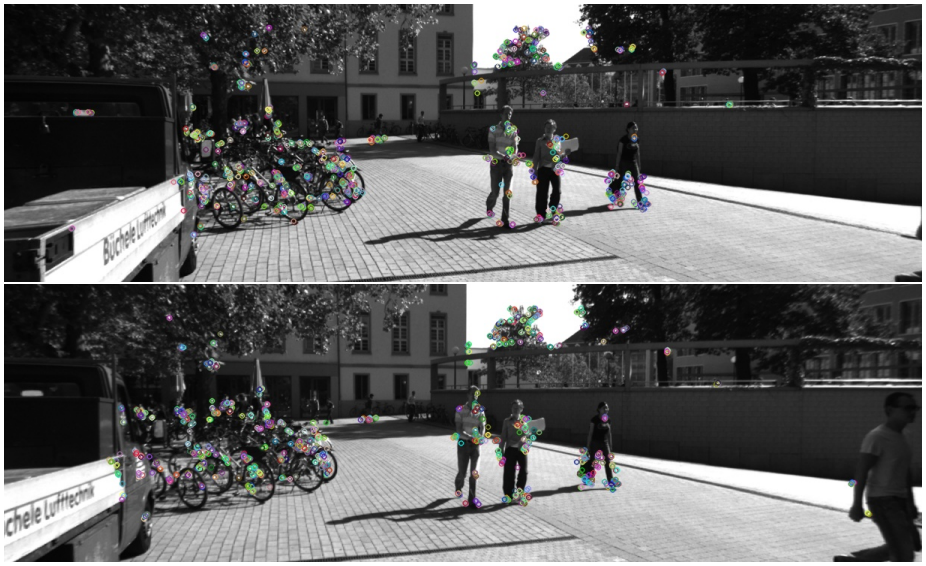

Figure 3: Feature Detection for 013

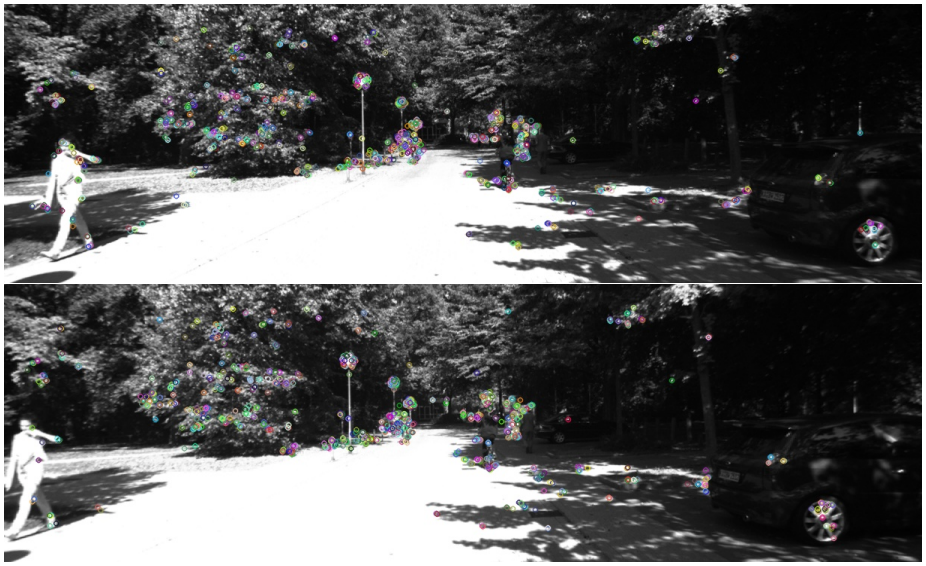

Figure 4: Feature Detection for 014

Figure 5: Feature Detection for 015

Question 2

he Brute-Force Matcher (BFMatcher) stands out as a fundamental and robust method for feature matching. The BFMatcher operates by exhaustively comparing descriptors of features from one set to those of another set and determining the best matches based on a specified distance metric.

The BFMatcher method conducts pairwise comparisons between descriptors extracted from two sets of features within different images. This process involves evaluating the similarity or dissimilarity between descriptors to establish potential correspondences.

BFMatcher employs various distance metrics to quantify the similarity between descriptors. For instance, in the case of binary string-based descriptors like ORB, BRIEF, or BRISK, the Hamming distance serves as the appropriate metric. This distance computation ensures robust matching, particularly for binary descriptors.

Optionally, BFMatcher integrates a cross-check mechanism, enabled by setting the crossCheck parameter to True. This feature refines the matching process by considering matches where both features from each set mutually agree as the best match. This enhances the consistency and accuracy of the matching results. This method is often used as an alternative to ratio test.

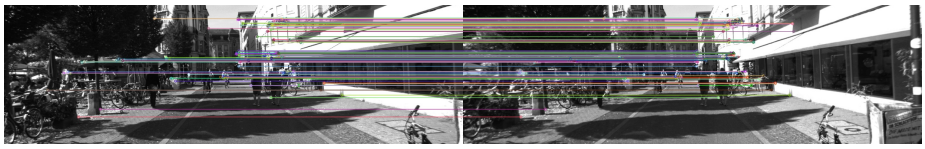

Figure 6: Images with non-outlier matches overlaid 011

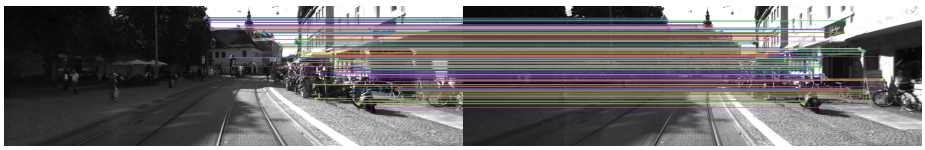

Figure 7: Images with non-outlier matches overlaid 012

Figure 8: Images with non-outlier matches overlaid 013

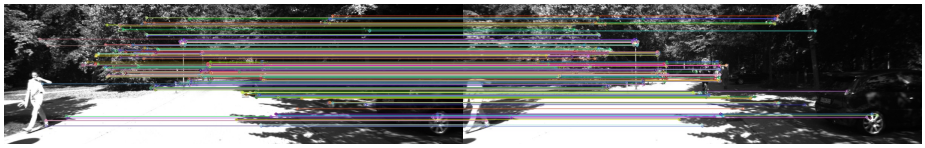

Figure 9: Images with non-outlier matches overlaid 014

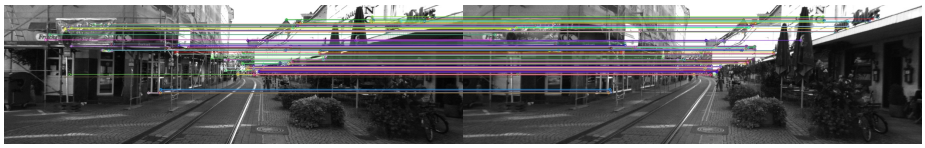

Figure 10: Images with non-outlier matches overlaid 015

Question 3

Random Sample Consensus (RANSAC), stands as a critical tool in handling inaccuracies during feature extraction and transformation estimation. RANSAC serves to robustly estimate model parameters in the presence of outliers or erroneous data points. This iterative algorithm functions by randomly selecting minimal subsets of data points and fitting models to these subsets, aiming to estimate parameters such as a geometric transformation matrix.

Specifically, when dealing with feature matching and object localization, RANSAC finds significant application in conjunction with functions like cv.findHomography() in the OpenCV library. This function is pivotal in determining the perspective transformation between images based on matched feature points. The estimation of a homography matrix is crucial in scenarios such as aligning or registering images, allowing for the localization of objects within different views or perspectives.

The concept of homography refers to the transformation matrix that maps points from one image to their corresponding points in another image in the same plane, assuming a perspective transformation. It encapsulates the relationship between the two images, enabling precise geometric adjustments or alignments. In the context of RANSAC, cv.findHomography() employs this matrix estimation while simultaneously utilizing RANSAC to enhance accuracy.

During the computation of the homography matrix, RANSAC is invoked within the cv.findHomography() function by setting specific flags, thereby facilitating outlier rejection and ensuring a more robust estimation of the transformation. This process involves identifying inliers and outliers among the matched feature points based on their consistency with the estimated transformation model. The inliers, representing the most reliable correspondences, contribute to the determination of an accurate homography matrix, while the outliers, which deviate significantly from the model, are disregarded.

In essence, the synergy between RANSAC and the homography estimation provided by functions like cv.findHomography() in OpenCV ensures the reliability and accuracy of geometric transformations between images. This proves essential in numerous applications, including object recognition, image alignment, and panoramic image stitching, where precise localization and alignment of features are paramount.

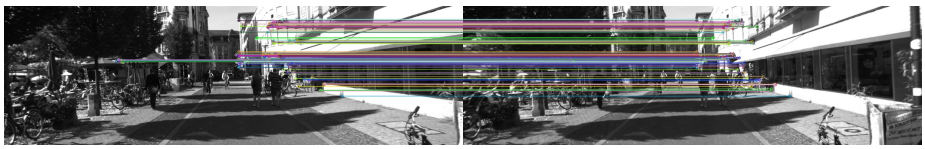

Figure 11: Images with non-outlier matches overlaid 011

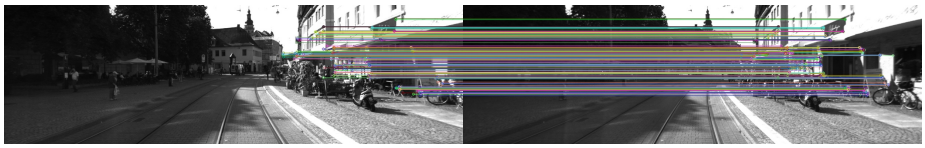

Figure 12: Images with non-outlier matches overlaid 012

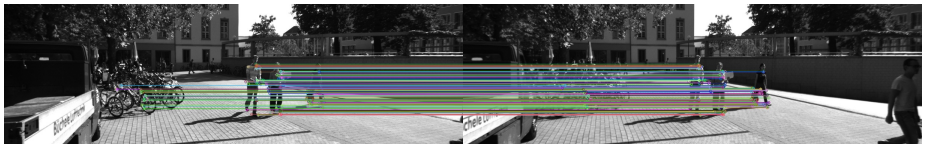

Figure 13: Images with non-outlier matches overlaid 013

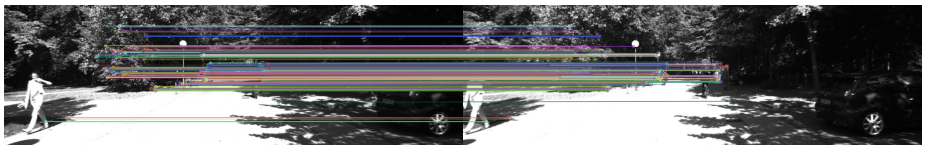

Figure 14: Images with non-outlier matches overlaid 014

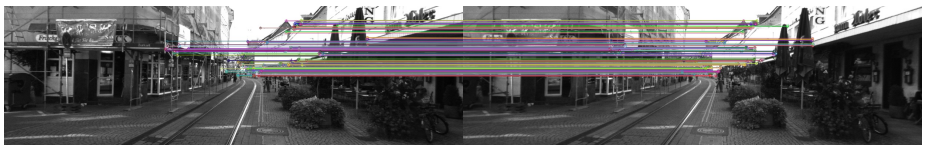

Figure 15: Images with non-outlier matches overlaid 015

0.2 Part 2

Question 1 and 2
The nearest search function performs a nearest neighbor search between points in the source and target point clouds. The original implementation uses a brute-force approach to find the nearest neighbor for each point in the source cloud within the target cloud, accumulating distances for mean calculation. However, this is later optimized using NearestNeighbors from Scikit-Learn to efficiently find the nearest neighbors.

The estimate pose function computes the pose transformation between corresponding points from the source and target point clouds. It computes the centroids of the corresponding points, then derives the transformation matrix using Singular Value Decomposition (SVD) to find the rotation and translation components.

The icp function implements the main ICP iteration loop. It repeatedly performs nearest neighbor search, estimates the pose, applies the transformation to the source cloud, and records mean distances and translations. This iterative process continues for a specified number of max iterations.

The mean distances list accumulates the mean distances between corresponding points in consecutive iterations. This metric serves as an indicator of alignment progress throughout the iterations. Lower mean distances imply better alignment between the source and target point clouds.

The computed translations represent the translation components extracted from the estimated pose at each iteration. These translations denote how much the source point cloud needs to be moved to align with the target point cloud.

Overall, the ICP algorithm aims to minimize the mean distances between corresponding points in successive iterations, ultimately aligning the source point cloud with the target point cloud. The process refines the transformation until convergence, providing an accurate alignment that can be used in various applications such as 3D reconstruction, object recognition, and robot navigation.

In the graphs bellow we can see that the X-Y-Z translations don't change after a few iterations of the ICP for the 2 training and 1 test set meaning that the converge to the correct position. If we were to plot the iterative change over time it would converge to zero. Meaning that it is closer to the target.

The estimated 6D pose refers to the determination of an object's precise position and orientation in a 3D space, encompassing translation along three axes (x, y, z) and rotation about these axes (roll, pitch, yaw). In computer vision and robotics, estimating the 6D pose is crucial for tasks like object localization, manipulation, and augmented reality applications. Several methods can estimate this pose, often involving geometric calculations, feature matching, or deep learning-based approaches.

Estimating the 6D pose typically deriving a transformation matrix that accounts for both translation and rotation. The translation components (x, y, z) denote the object's displacement in a 3D coordinate system, while the rotation components (roll, pitch, yaw) represent its orientation. The 6D pose matrix

representation:

$$P = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

Where:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

is a 3x3 rotation matrix representing orientation, and

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

is a 3x1 translation vector representing position in 3D space.

The 6D pose estimation and the Iterative Closest Point (ICP) algorithm are closely intertwined in solving the problem of aligning and estimating the position and orientation of objects or point clouds in a 3D space.

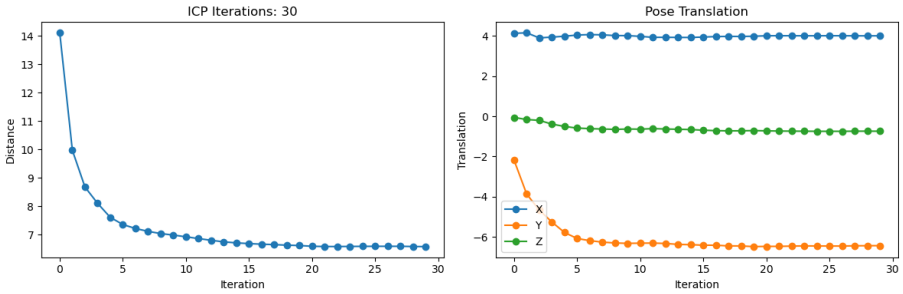

Figure 16: ICP loss plot and absolute 3D translation, Bunny

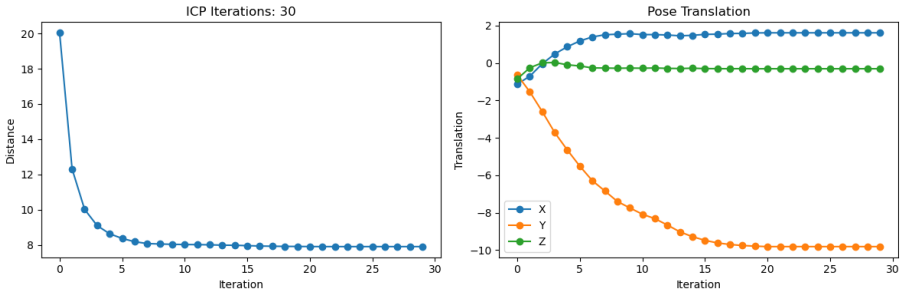

Figure 17: ICP loss plot and absolute 3D translation, Dragon

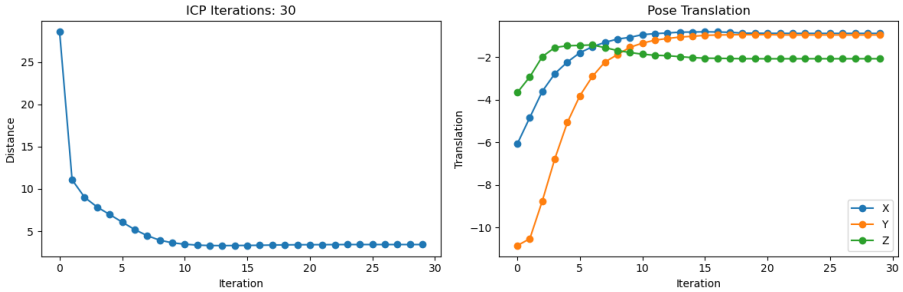

Figure 18: ICP loss plot and absolute 3D translation, Armadillo

Question 3

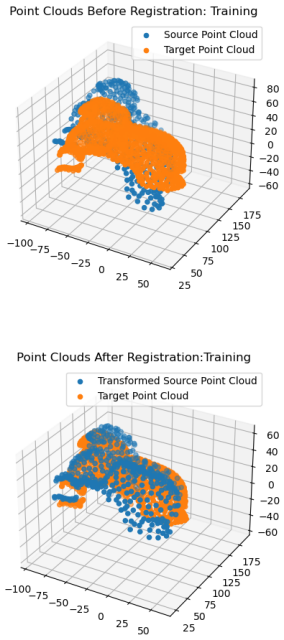

Figure 19: Point Clouds Before and After Registration, Bunny

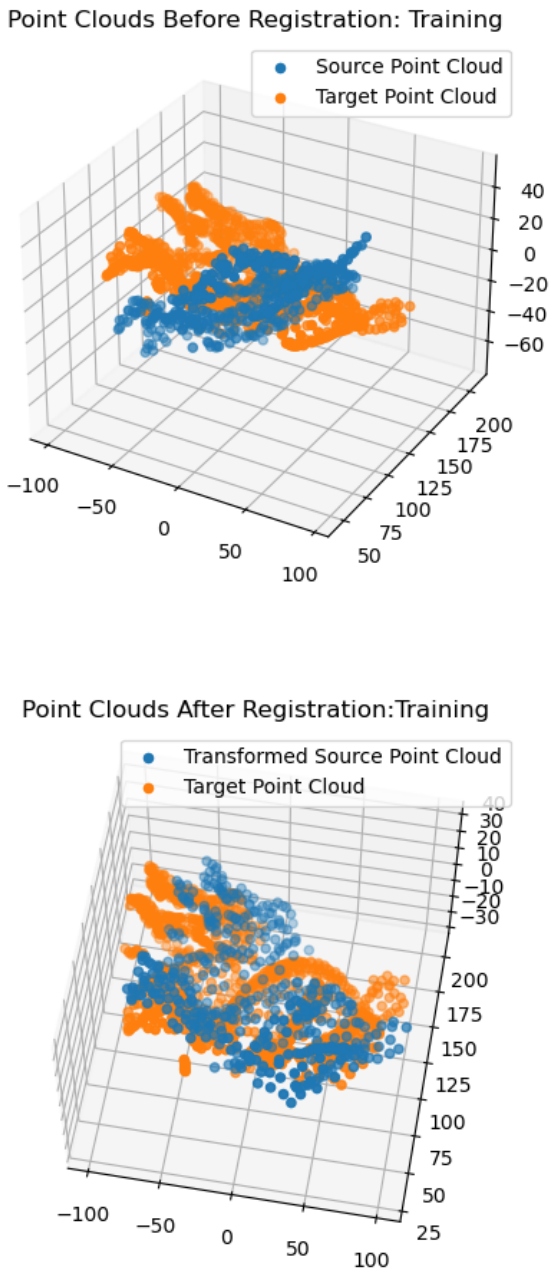

Figure 20: Point Clouds Before and After Registration, Dragon

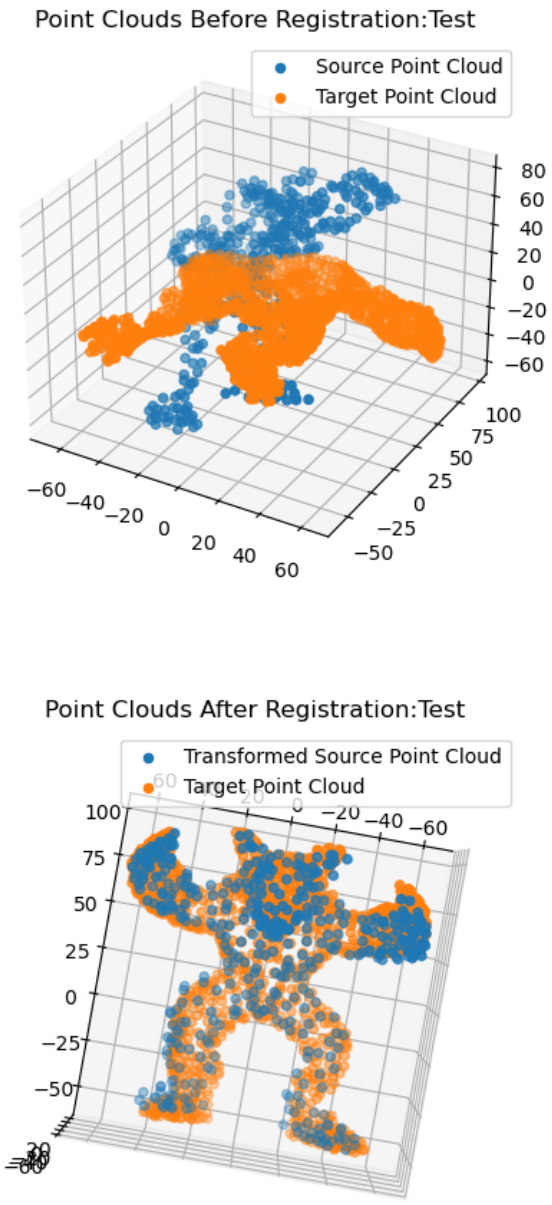

Figure 21: Point Clouds Before and After Registration, Armadillo

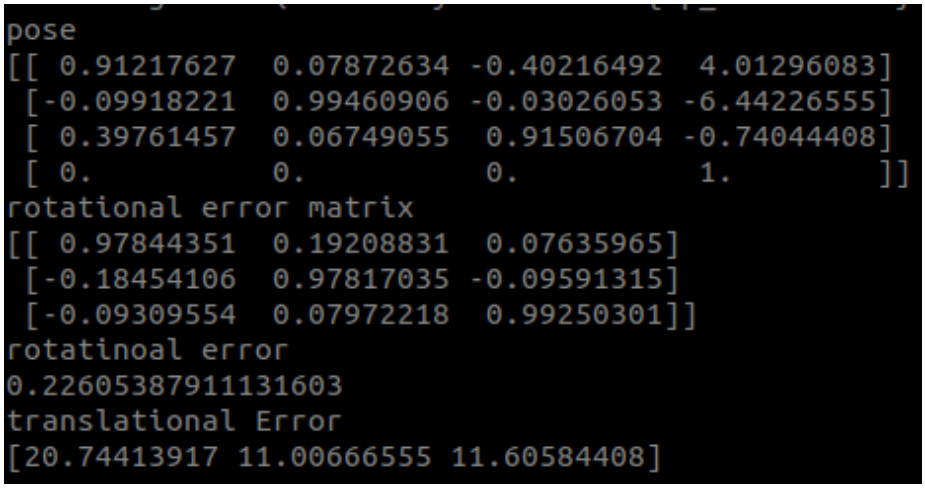

Figure 22: 6D Pose Error on training set, Bunny

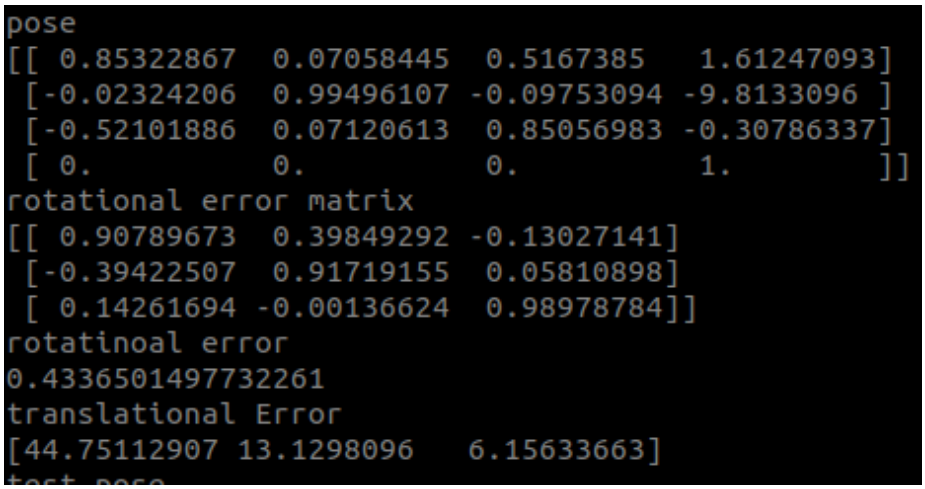

Figure 23: 6D Pose Error on training set, Dragon

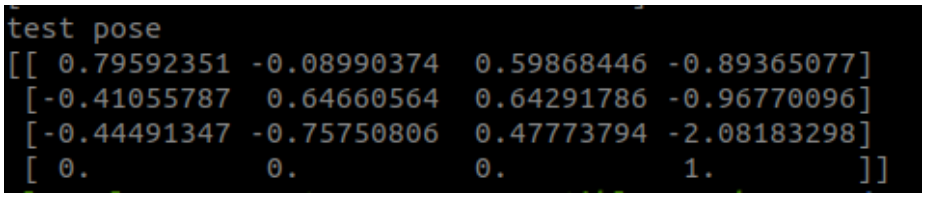

Figure 24: 6D Pose on test set, Armadillo