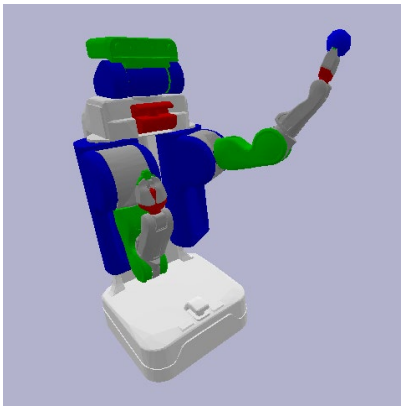
Q1. IK

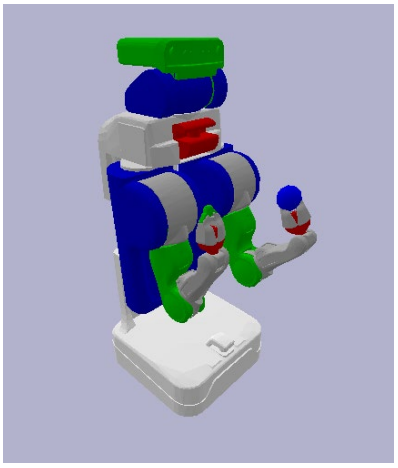
c)

target point 0:



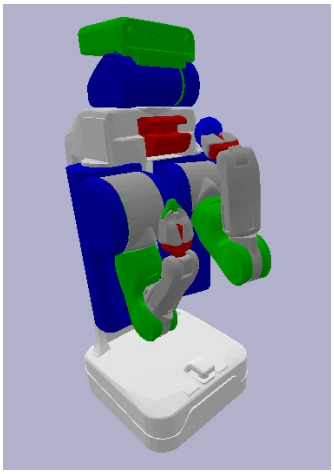
```
Reach target point 0
Configuration of robot: [[ 0.45009185 -0.22976254 -0.11359675 -0.66635122  0.         -0.36027897
   0.        ]]
```

target point 1:



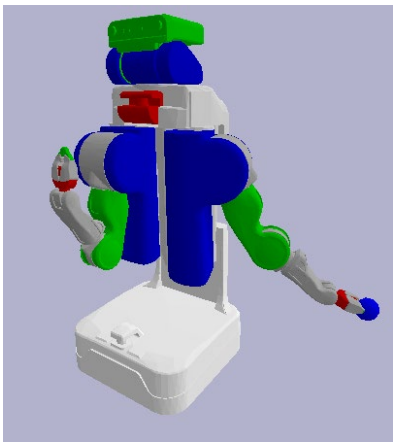
```
Reach target point 1
Configuration of robot: [[-2.09624059e-05  1.04058238e+00 -2.64334954e-05 -1.80550642e+00
   0.00000000e+00 -1.32649585e+00  0.00000000e+00]]
```

target point 2:



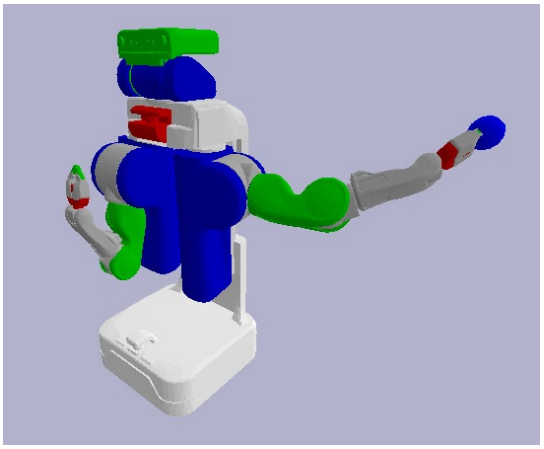
```
Reach target point 2
Configuration of robot: [[-0.29501559  0.38153069  0.09968505 -2.08335905  0.         -1.03928796
   0.        ]]
```

target point 3:



```
Reach target point 3
Configuration of robot: [[ 1.93593225e+00  9.17227496e-01 -3.78036460e-05 -2.70373083e-01
   0.00000000e+00 -2.32517219e-01  0.00000000e+00]]
```

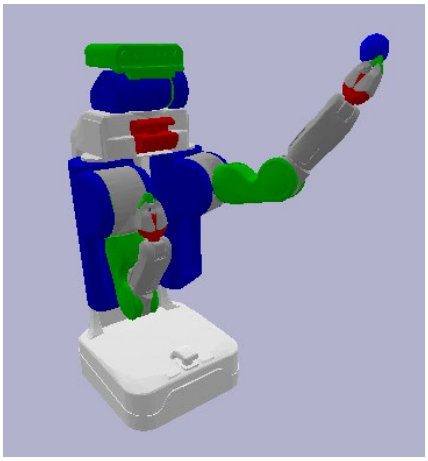
target point 4:



```
Reach target point 4
Configuration of robot: [[ 1.71428586 -0.01235743 -0.61900837 -0.56040263  0.         -0.30594294
   0.        ]]
```

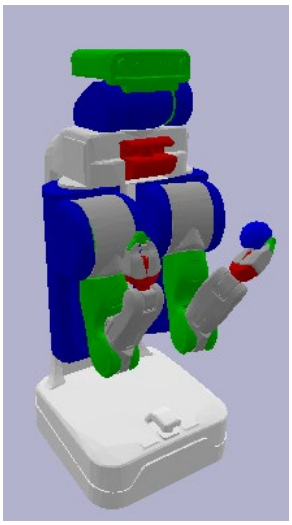
d)

target point 0:



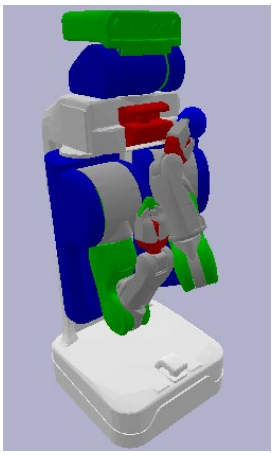
```
Reach target point 0
Configuration of robot: [[ 0.36149363 -0.20207633 -0.07483566 -0.72139276 -1.         -0.36237043
  -1.        ]]
```

target point 1:



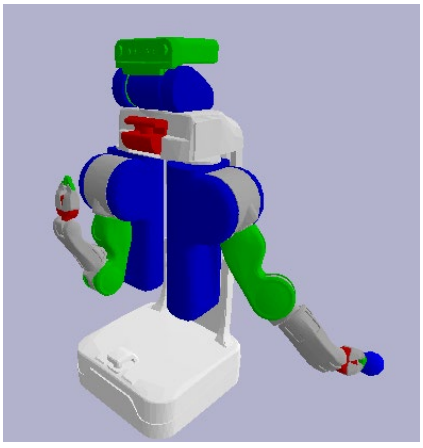
```
Reach target point 1
Configuration of robot: [[-0.28600978  1.0432296  -0.01648656 -1.98708431 -1.         -1.11917034
  -1.        ]]
```

target point 2:



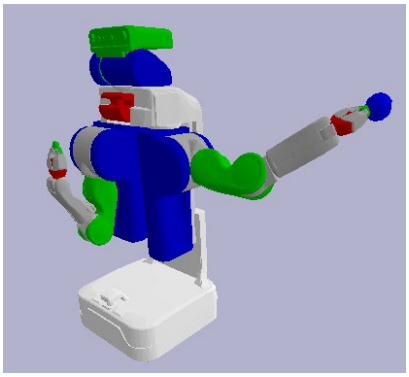
```
Reach target point 2
Configuration of robot: [[-0.51468303  0.42288619  0.31856158 -2.3213     -1.         -0.87228218
  -1.        ]]
```

target point 3:



```
Reach target point 3
Configuration of robot: [[ 1.64606285  0.85518207 -0.490582   -0.21057492 -1.         -0.86798997
  -1.        ]]
```

target point 4:



```
Reach target point 4
Configuration of robot: [[ 1.60927927  0.00401772 -0.53629336 -0.69489144 -1.         -0.51997725
  -1.        ]]
```

Q2. Force closure evaluation

a)

```
Grasp 1 complete. Press <ENTER> to compute friction cone volumes and force closure.

Grasp 1 Contact point 1 Volumes:
True volume: 0.1227
4 edge volume: 0.0781
8 edge volume: 0.1105

Not in force closure
```

With fewer vectors, the discretization is coarse. Each vector has to cover a larger range of possible force directions. This simplification leads to larger errors in representation as the actual continuous nature of the friction cone isn't accurately captured. Increasing the number of vectors makes the discretization finer. Each vector represents a more specific direction of force, and collectively, they approximate the continuous nature of the friction cone more closely. This finer discretization reduces the error in representation.

b)

```
Grasp 2 complete. Press <ENTER> to compute force closure.

In force closure. Maximum radius: 0.0141
```

As convex hull function is designed to check for force closure in a set of wrenches using their convex hull. Here's a shortened explanation of the method:

1. Compute the Convex Hull: Determines the smallest convex set that contains all the wrench vectors using the `ConvexHull` function.

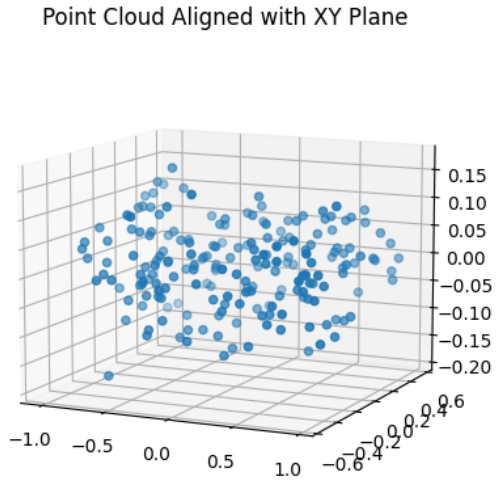
2. Check for Force Closure: Checks if the origin of the wrench space is inside the convex hull. If it is, then the object is in force closure, meaning the wrenches can collectively balance any external force.

3. Find Largest Hypersphere's Radius: If the object is in force closure, calculates the radius of the largest sphere that can fit inside the convex hull and is centered at the origin. This radius indicates the strength of the grasp; a larger radius suggests a more robust grasp.

Knowing the radius of the hypersphere is useful because it provides a quantitative measure of the grasp quality. A larger radius implies a stronger grasp that can resist larger disturbances. It gives an idea of the robustness of the grasp. If the hypersphere's radius is very small, it indicates that the grasp is only just achieving force closure and may not be very stable.

Q3. PCA algorithm

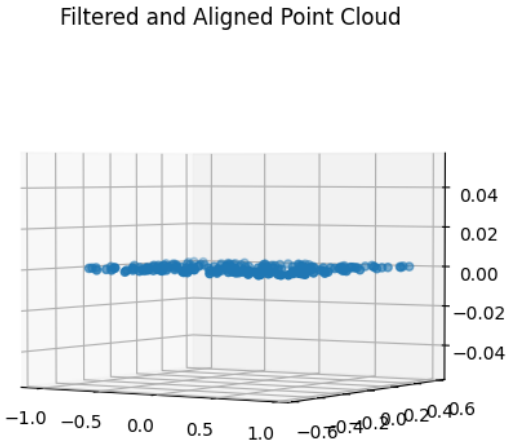
a) Rotated point cloud:



V^T matrix:

```
[[-0.52935026 -0.05809333  0.84641211]
 [-0.00253698 -0.99754007 -0.0700526 ]
 [ 0.84839959 -0.0392297   0.52790071]]
```

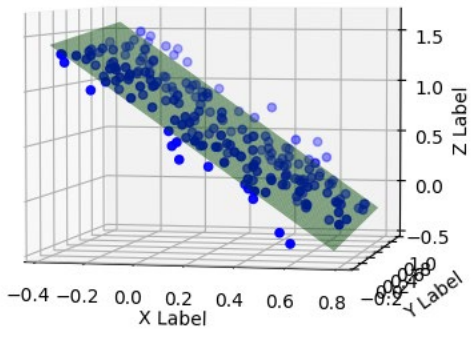
b) The point cloud after PCA:



Vs^T matrix:

```
[[-0.52935026 -0.05809333]
 [-0.00253698 -0.99754007]
 [ 0.84839959 -0.0392297 ]]
```

c) Use PCA to fit a plane to the cloud and draw that plane in green in a plot with the point cloud:

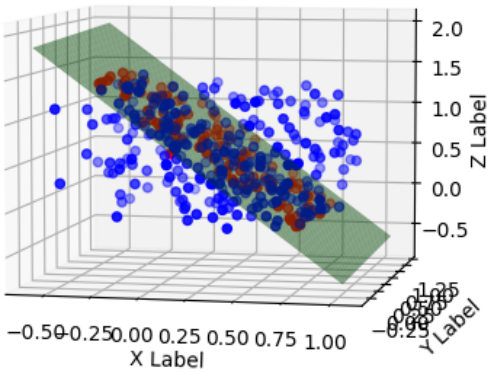


Q4. RANSAC algorithm

The equations for that plane:

```
The plane equation is: -0.85x + 0.03y + -0.53z + 0.51 = 0
```

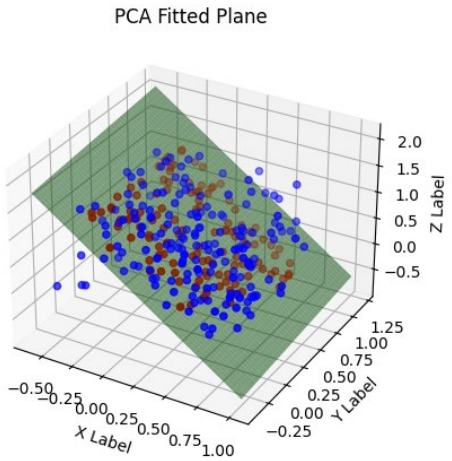
The plane in green fitting to the data, the inliers for that plane in red, and all other points in blue:



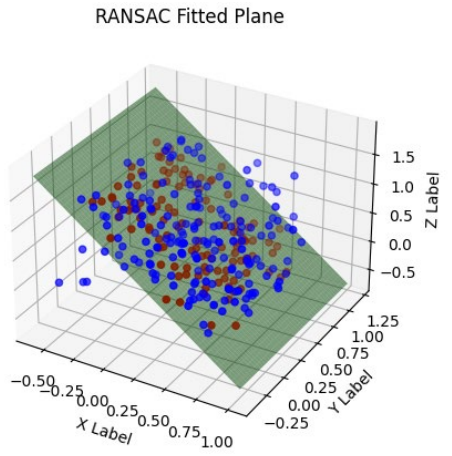
Q5. RANSAC versus PCA algorithm

a)

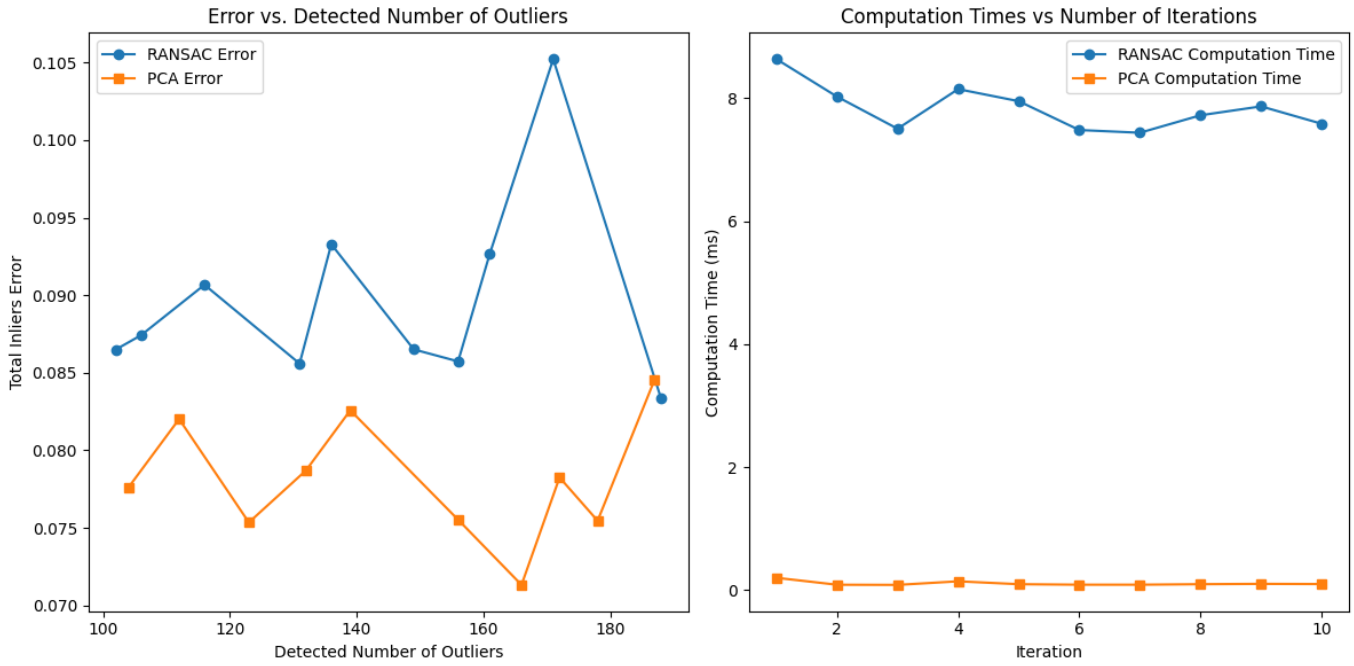
Plot1, PCA's plane fitting to the data in green, the inliers for that plane in red, and all other points in blue:



Plot2, RANSAC's plane fitting to the data in green, the inliers for that plane in red, and all other points in blue:



b) Error vs. Number of Outliers for both algorithms and computation times of PCA and RANSAC at each iteration.



c)

Error Performance: PCA consistently showed lower error rates than RANSAC, suggesting it was more effective at fitting a model in this context. RANSAC displayed variable error rates, likely due to its iterative and random sampling process.

Computation Time Performance: PCA had lower computation times across all iterations, indicating it was computationally more efficient. RANSAC had slightly higher and more variable computation times, reflective of its iterative nature.

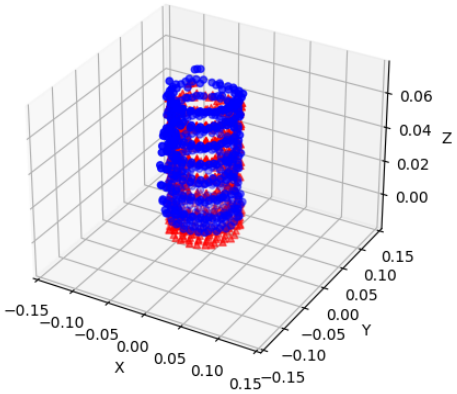
Generally, PCA outperformed RANSAC in both error metrics and computation time for the data and tasks presented in the plots. The reason why PCA might have outperformed RANSAC in this context could be due to several factors:

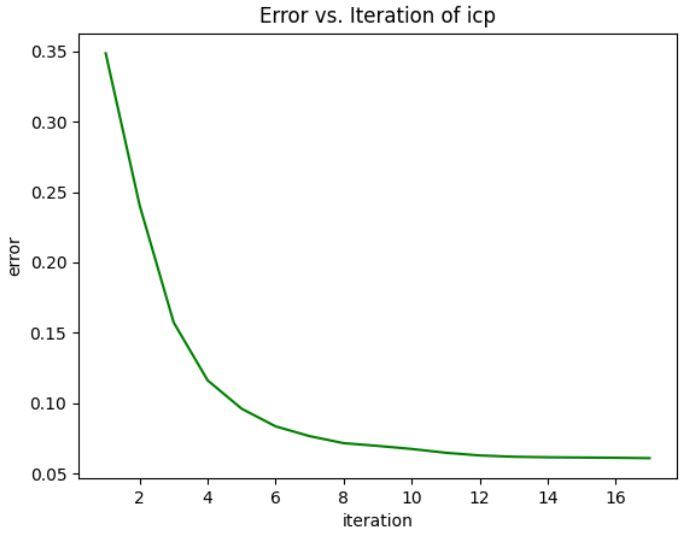
- **Data Characteristics**: The nature of the point cloud and the distribution of outliers may be such that the principal components found by PCA are robust to these outliers.
- **RANSAC Parameters**: If the parameters for RANSAC (like the number of iterations, the threshold for determining inliers, and the size of the sample) are not optimized for the data, its performance could be worse than PCA.
- **Computational Complexity**: PCA has a predictable computational complexity that is dependent on the size of the data matrix, whereas RANSAC's complexity can vary significantly with the number of required iterations to find a consensus set.

Q6. ICP algorithm

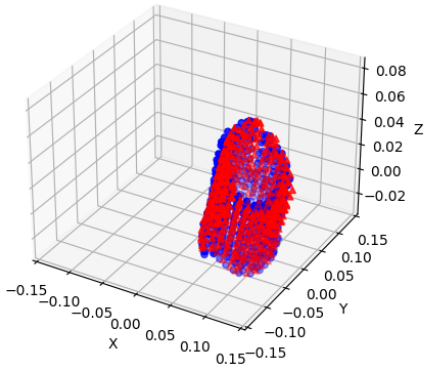
b)

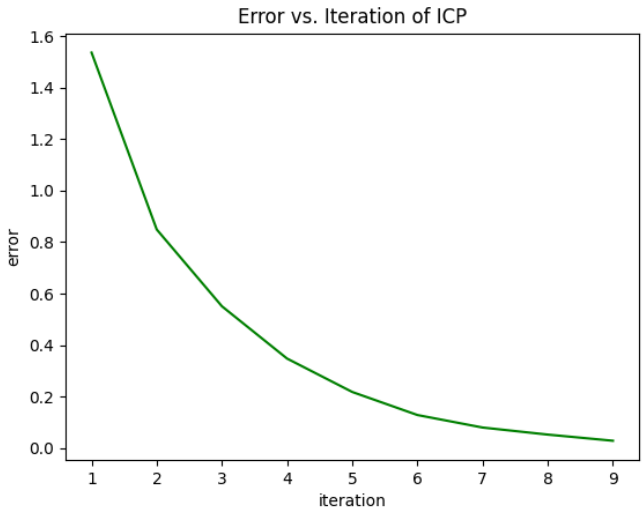
Target 0:



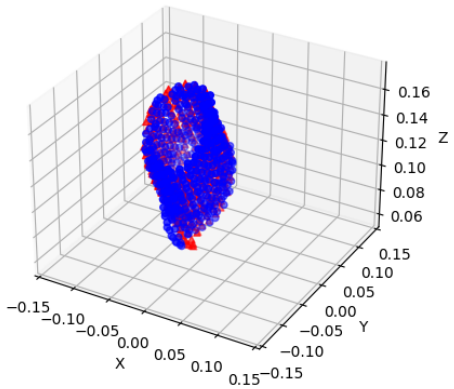


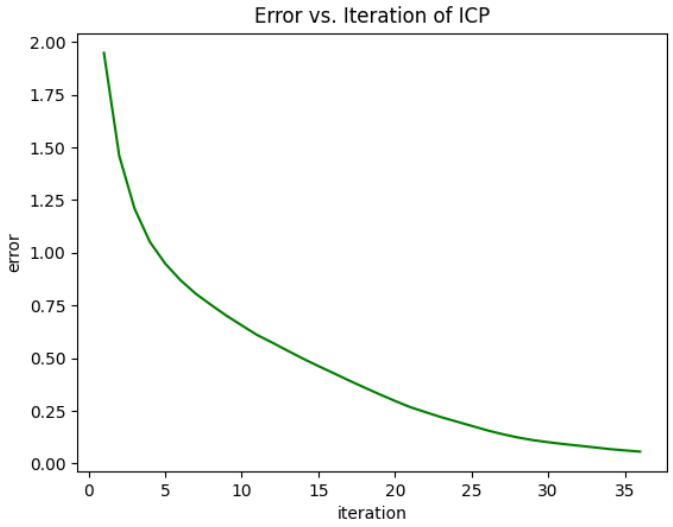
Target 1:



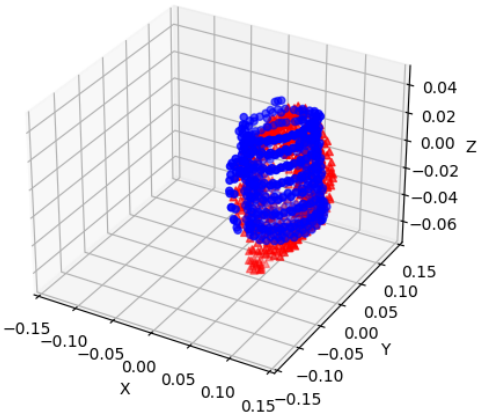


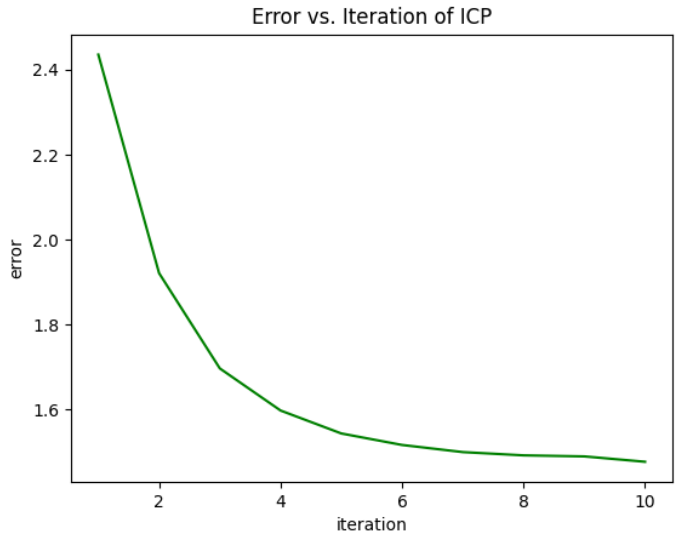
Target 2:





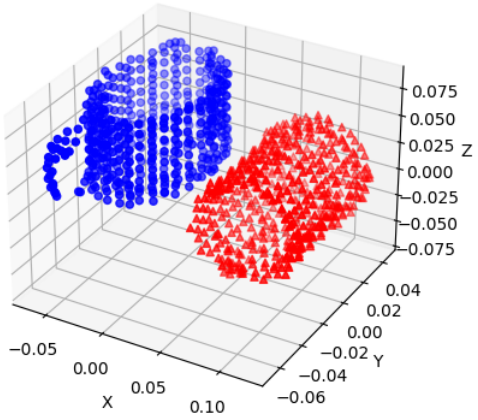
Target 3:





c)

From the observation of previous figures, it can be observed that the **Target 3** have the largest error for ICP. Therefore, it was considered that the more difficult for ICP.



The figure presents a more challenging scenario (Target 3) for the ICP algorithm mainly misalignment between the two-point clouds. The two-point clouds have different shapes, which could significantly increase the challenge to establish correspondence between two points clouds. Therefore, it becomes very difficult to align them. Apart from that, their large difference in orientation also makes correct point correspondence more difficult to establish, potentially leading to longer convergence times or convergence to an incorrect solution.