NAME : JOSE JOSEPH THANDAPRAL

NUID NO. : 002102407

COURSE CODE : CS 5335

COURSE NAME : ROBOTIC SCIENCE AND SYSTEMS

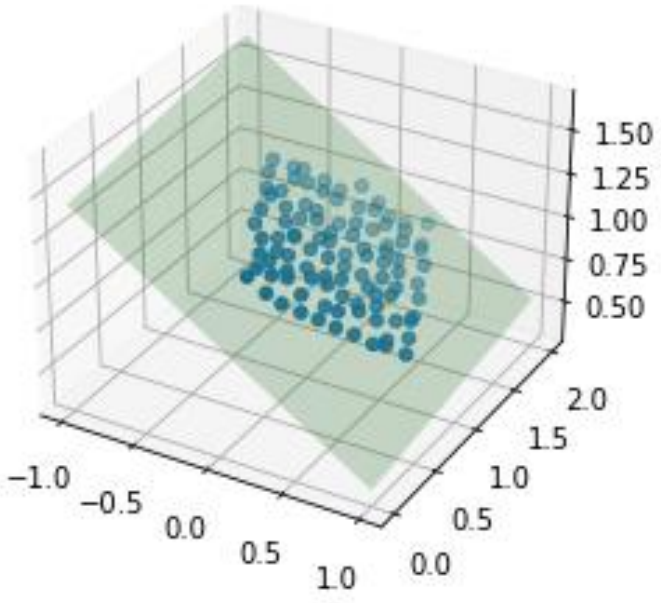
CRN : 38996

FACULTY : PROF. ROBERT PLATT

NORTHEASTERN UNIVERSITY, BOSTON

HW 4

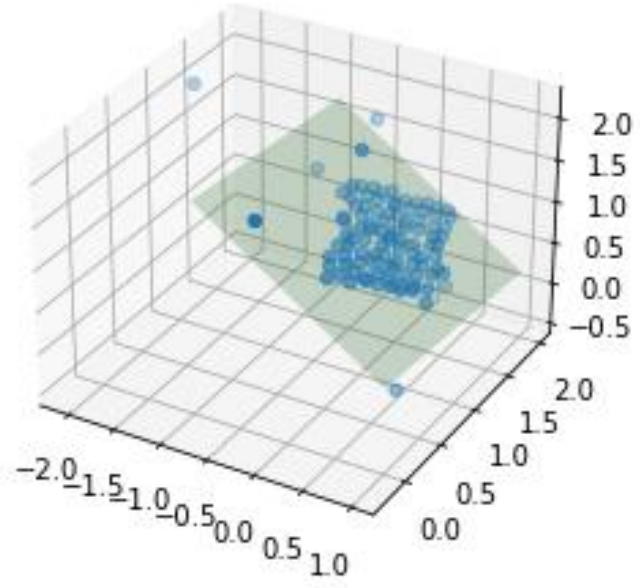
Q1.(a) Output of $python q1_a in the terminal:




```
In [54]: hw4.hw4("q1_a")
normal vector of plane of points=  [-0.48138664  0.08143396 -0.87271726]
center of the plane of point cloud =  [0.12765143 1.1703587  0.94010405]
```

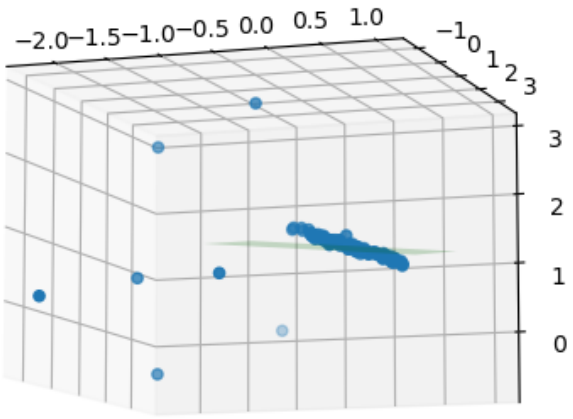
Terminal output

Q1.(b) Output of $python q1_b in the terminal:



```
In [55]: hw4.hw4("q1_b")
normal vector of plane of points=  [0.52358564 0.06385163 0.84957698]
center of the plane of point cloud =  [0.11853269 1.13240446 0.98492827]
.............Baised due to outlier points...........
```

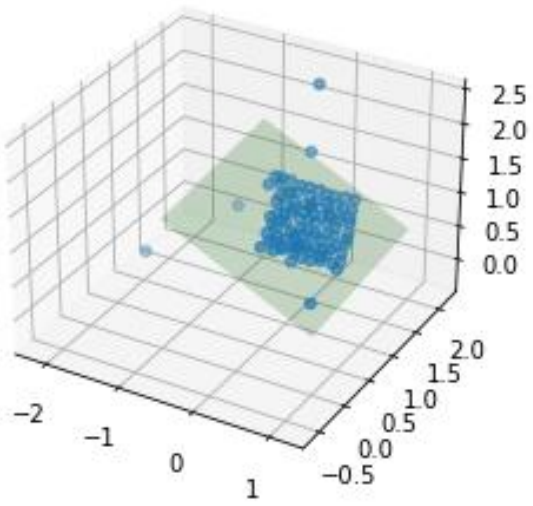
Terminal output

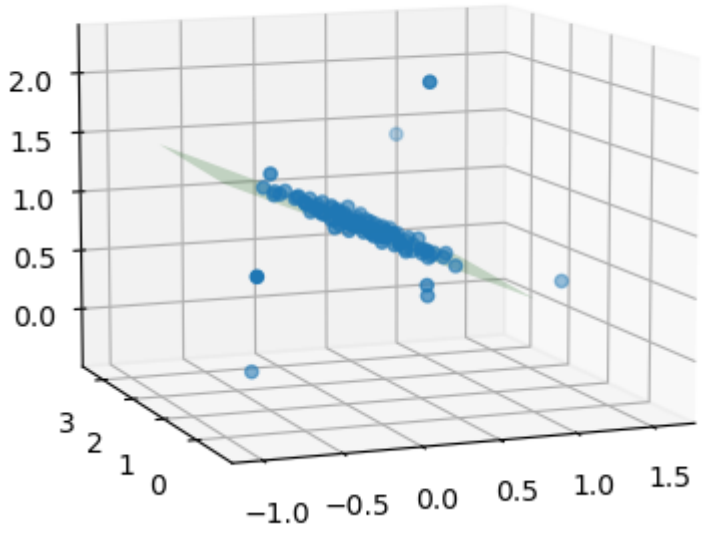


Alternate view angle

We see that as compared to the results from (1)(a), the Singular Value Decomposition(SVD) does not perfectly fit even the part of the point cloud that is along a perfect plane. This is because SVD is not robust to outliers and would take the outliers into it's calculation for a plane. Other methods like RANSAC would have to be used to consider optimal subsets in it's iterations to reach convergence at an optimal plane fit.

Q1.(c) Output of $python q1_c in the terminal:

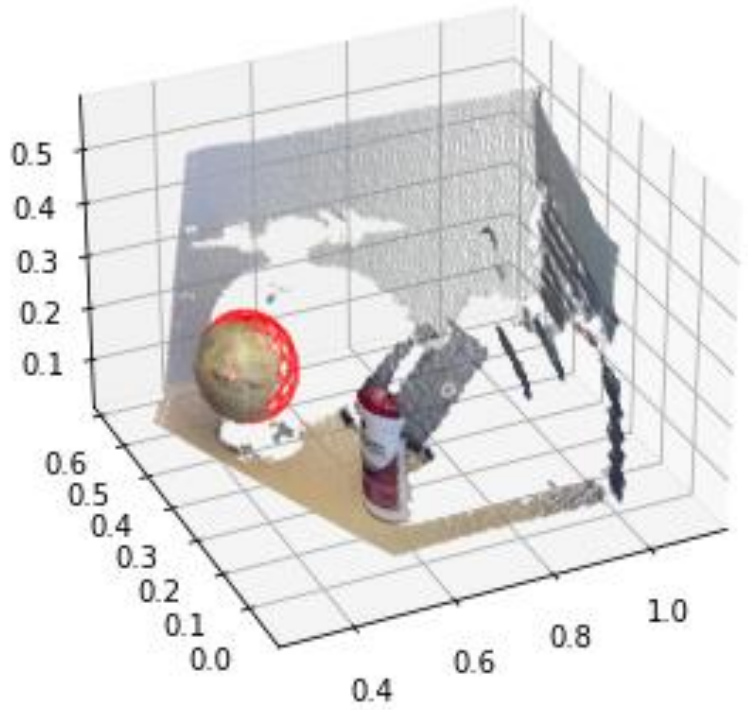





```
In [58]: hw4.hw4("q1_c")
Performing RANSAC over given point cloud
normal vector of plane of points=  [ 0.15500127 -0.03192707  0.30258754]
center of the plane of point cloud =  [0.10050219 1.13200093 0.90969527]
```

As we can see, the RANSAC approach is robust to outliers and gives a better fit according to the subset of point cloud that is close to a plane. SVD is easier in implementation but susceptible to outliers while RANSAC would be a bit more compex to implement while being resilient to outliers.

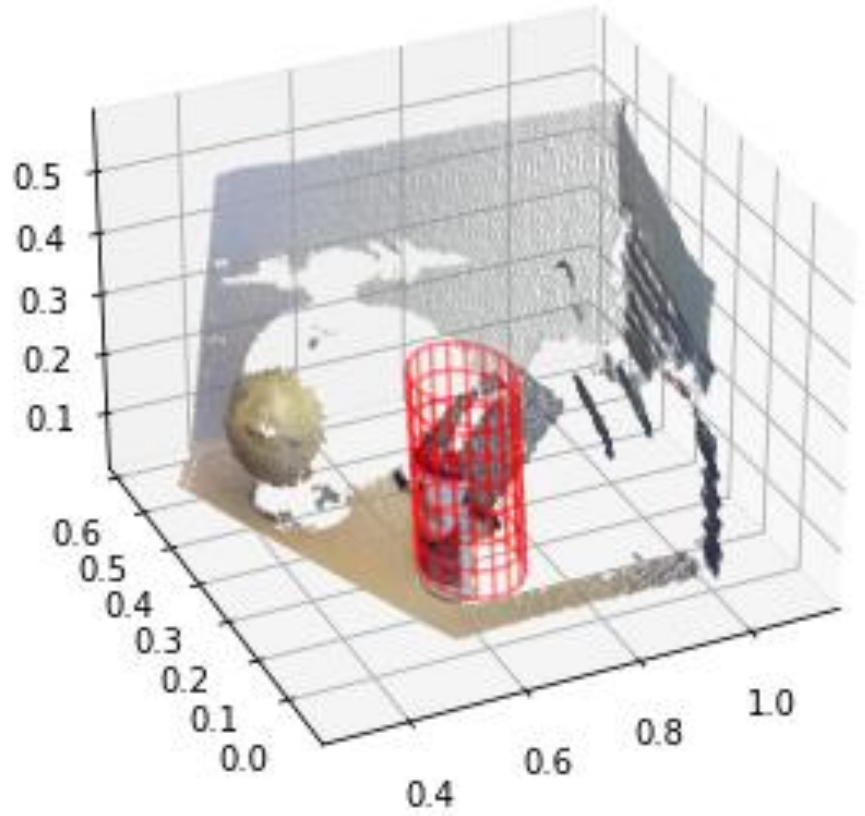
Q2. Output of $python q2 in the terminal:




```
In [59]: hw4.hw4("q2")
Performing RANSAC over given point cloud  for Sphere fitting
Iteration# 0
Iteration# 100
Iteration# 200
Iteration# 300
Iteration# 400
Iteration# 500
Iteration# 600
Iteration# 700
Iteration# 800
Iteration# 900

Fitted sphere's center: [0.43149544 0.34928713 0.21098412]
Fitted sphere's Radius: 0.09
Maximum number of inliers =  20385
```

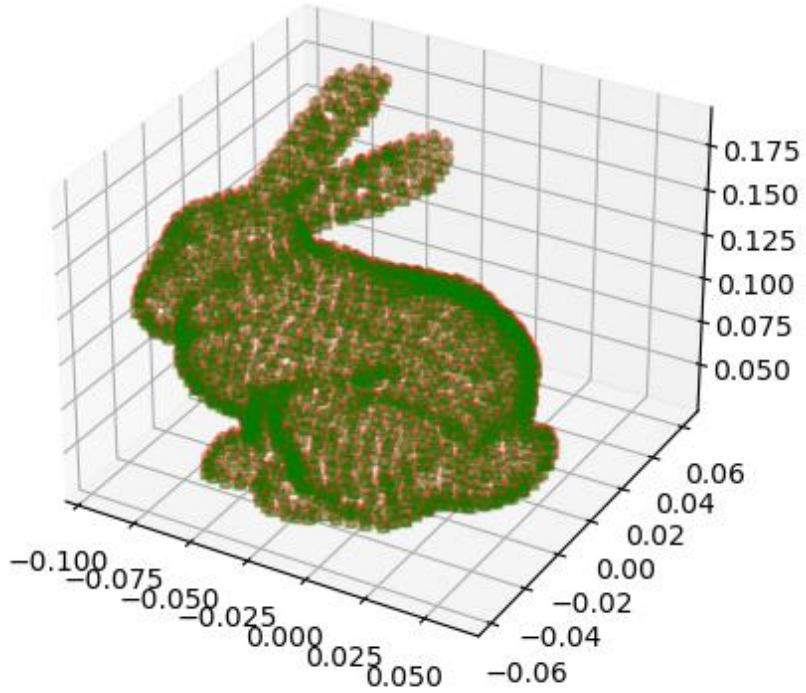
Q3. Output of $python q3 in the terminal:



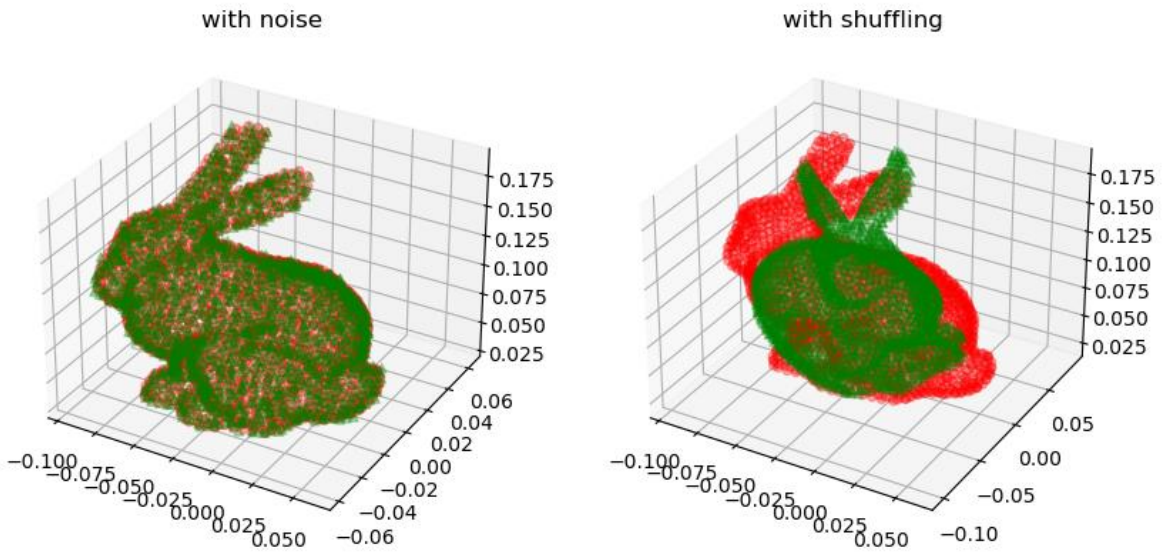

```
In [60]: hw4.hw4("q3")
Performing RANSAC over given point cloud  for Cylinder fitting
Processing iteration 0
Processing iteration 100
Processing iteration 200
Processing iteration 300
Processing iteration 400
Processing iteration 500
Processing iteration 600
Processing iteration 700
Processing iteration 800
Processing iteration 900

Fitted cylinder's center: [0.59601543 0.07631856 0.07315807]
Fitted cylinder's Radius: 0.06666666666666667
Maximum number of inliers =  16902
Axis of cylinder =  [ 0.02310822 -0.00648239 -0.82209009]
```

Q4.(a) Output of $python q4_a in the terminal:



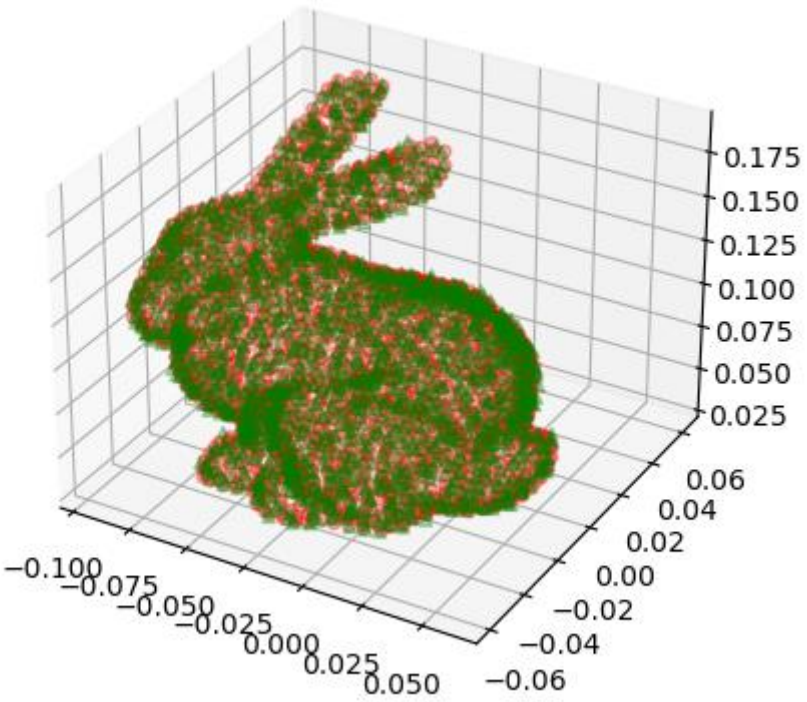
Q4.(b) Output of $python q4_b in the terminal:



Even when gaussian noise is introduced, as seen in Q4(a), the ICP algorithm gives an optimal result of finding correspondences as the underlying geomtery of the point cloud is not lost.

Hence, the ICP algorithm still finds the closest points in the point clouds for given alignment, over it's iterations, but only slightly affects the resulting correspondences. But when the indices are shuffled, the underlying geometry is lost and hence the correspondances calculated by ICP get skewed with consecutive iterations.

Q4.(c) Output of $python q4_b in the terminal:



ACKNOWLEDGEMENT

- Avinash Ayite –RANSAC and ICP implementation
- Tarun Srinivasan– RANSAC and ICP implementation