Project 1

ENPM673

Perception for Autonomous Robots

UID: 119061770

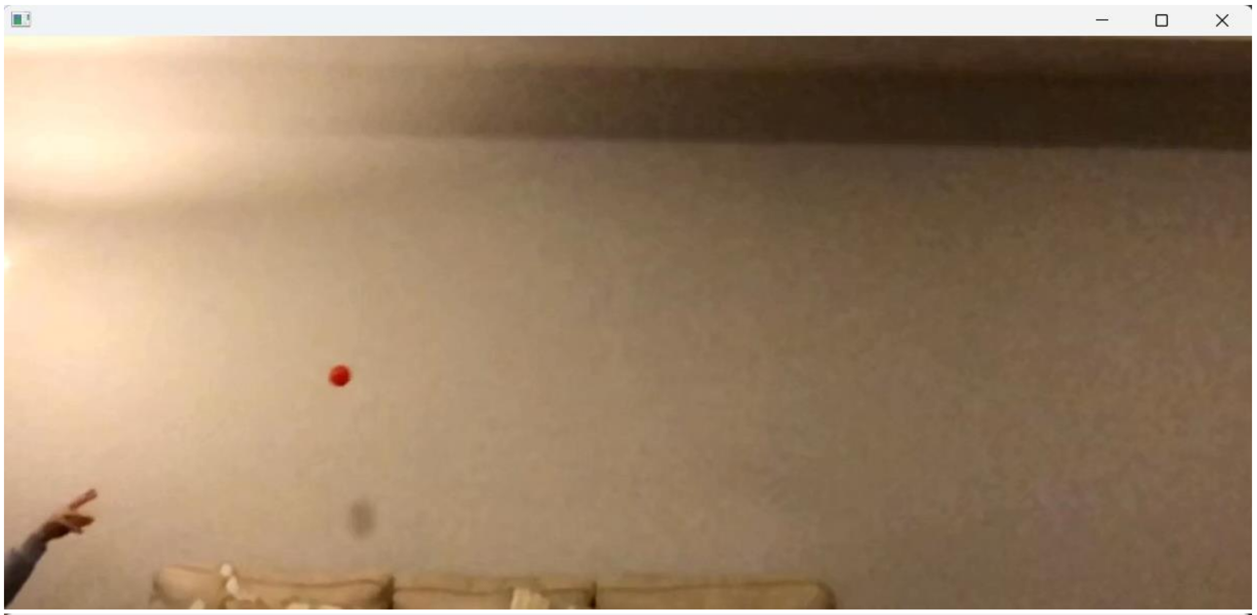
Name: Rohan Maan

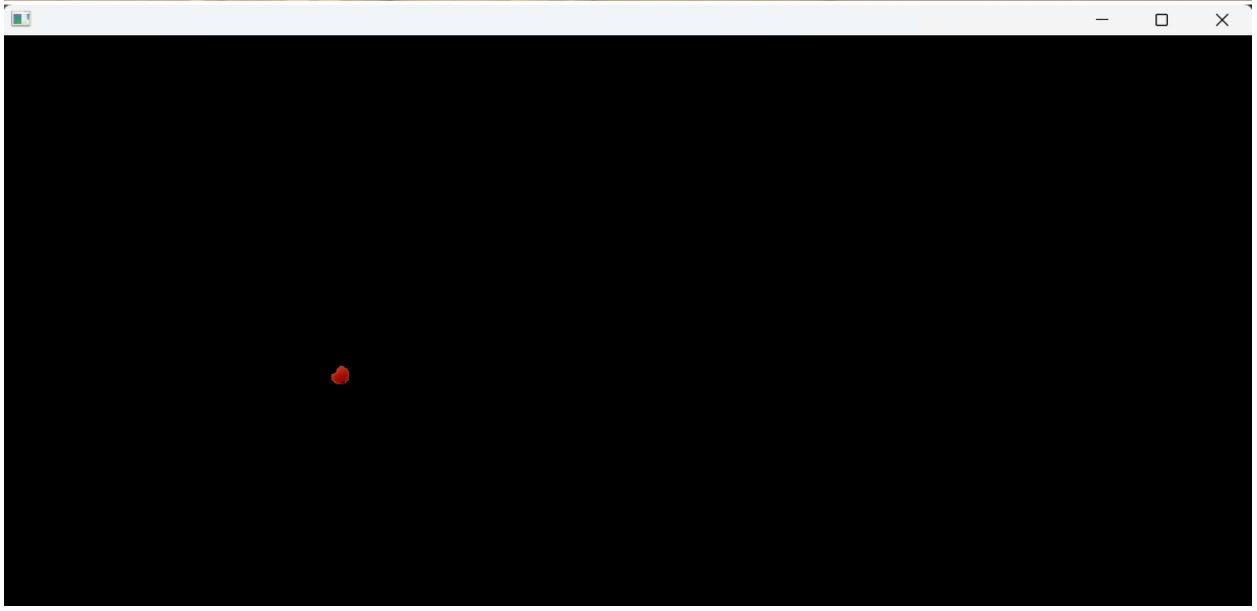
Problem 1:

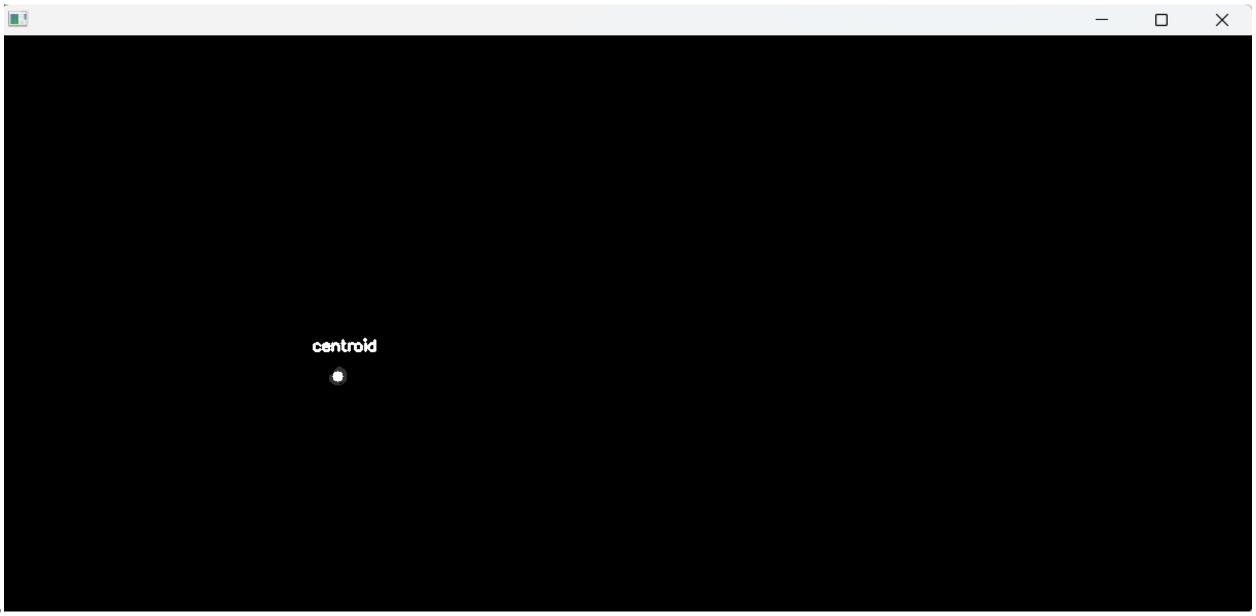
In the given video, a red ball is thrown against a wall. Assuming that the trajectory of the ball follows the equation of a parabola:

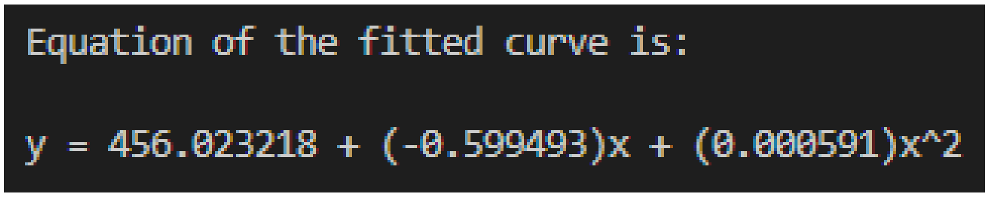
1. Detect and plot the pixel coordinates of the center point of the ball in the video.
   Ans. This program named Problem_1.py starts by importing libraries, most of which are standard libraries. The code begins by writing a function for calculating the centroid(named – *calculate_centroid*). This function takes input of image. I am passing a filtered image, where only the ball pixels are non zero values and rest are all 0(hence black color). The function takes average of x and y coordinates of non zero values(indicates ball region) and this average indicates the central position of the ball in each frame.

   Extracting Red ball:
   1. Load the image using cv2.imread (for sample)
   2. To read all frames from the video, we can use cv2.VideoCapture() function. In my code, I store all the frames in a list called *cv_img*.
   3. On each frame, we are supposed to iterate to find a red ball, which means in between a certain threshold, only the red color of the ball should be highlighted and the rest will be marked black.
   4. I filtered the red image by setting lower and higher threshold as follows: [0, 0, 130] and [65, 65, 255] indicating lower and upper values for RGB each.
   5. After this filter or mask is obtained, I used cv2.bitwise_and function so that the rest of the image value becomes black and ball can be identified and separated from the background.
   6. Post this, an additional threshold on gray image helps to be sure on the filtered image having just the ball in focus.
   7. A few additional cv functions allow us to mark a circle on the identified spot of the ball and this is also what we used to plot the final coordinates and hence used for plotting curve across the following points.

8. 

9. 

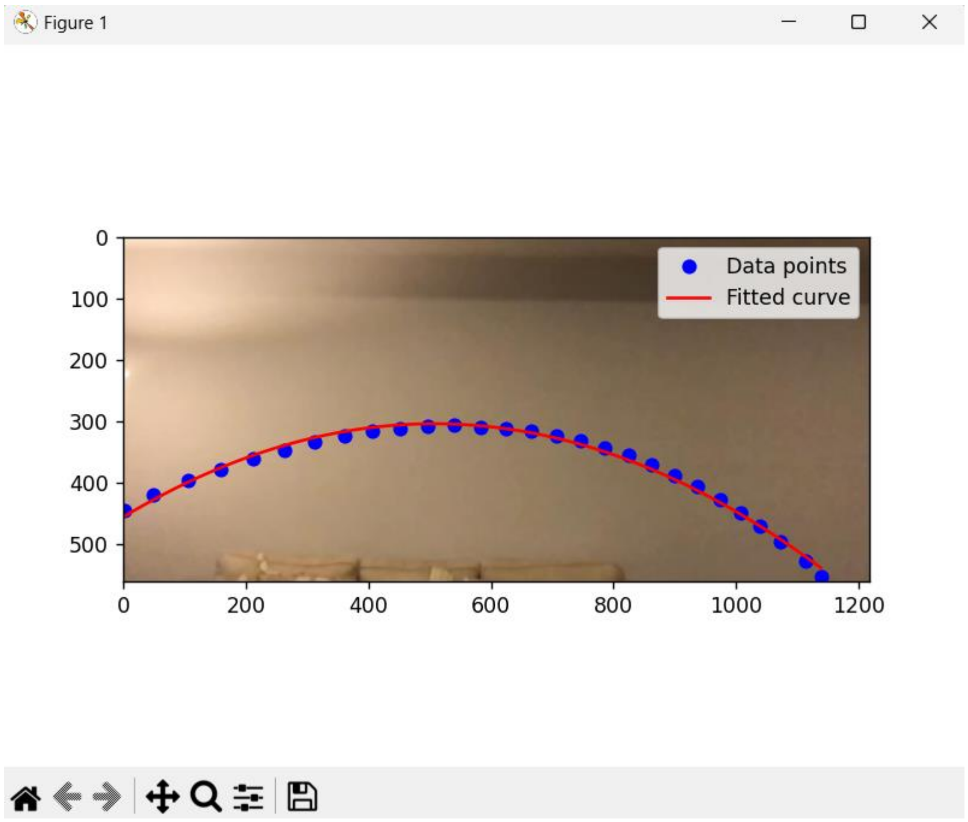
10. 

2. Standard Least Square was used to fit the curve for extracted coordinates. For this we have to first find the coefficients of equation of parabola using the x and y coordinate pair for various location of ball. The equation comes out to be:
   a. 

```
Equation of the fitted curve is:

y = 456.023218 + (-0.599493)x + (0.000591)x^2
```

   b.

   Here, we have plotted x and y points for every 6th frame, so the points alone don't show up as a line. On top of this the curve was plotted, for this we first need to use the coordinates and find the coefficients of the parabolic equation and later this is used to plot the curve making use of the available x values in our frame. I faced a lot of problem here mainly into finding the perfect threshold for the ball and particularly because the values are variable for each red, green and blue channel. Components of image other than the ball also have components of red and hence just filtering red was not enough.



3. Part 3 involves considering coordinate frame where the top left corner is the origin and this is why I set the coordinate to that before hand. The condition requires us to add 300 pixel to the entry y intercept. As per our equation, when x = 0, y = 456 and hence total becomes 456 + 300 = 756. At Y=756, we have to calculate the value of x. As we have a quadratic equation, we can find the value of X and that comes out to be -367 and 1381, according to our coordinate, we know that we are looking for the positive value and hence the X will be 1381 at Y = 756. The output from the code is as follows:

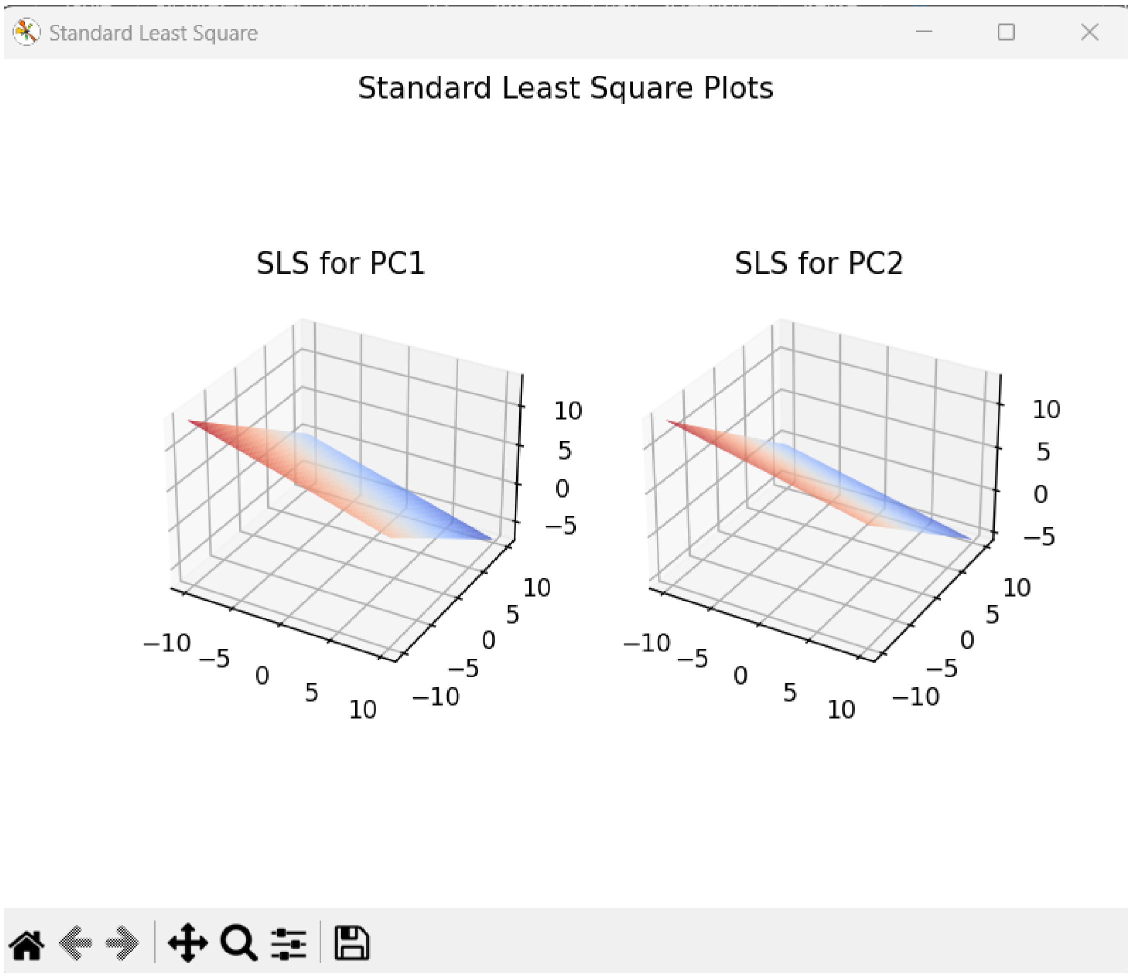
```
The roots are  (-367.3723922225899+0j)  and  (1381.7429505982245+0j)

X coordinate of Landing spot is  (1381.7429505982245+0j)
```

Problem 2:

I have assigned Problem_2.py to solve question 2 and this has multiple functions present in it each to solve curve for Standard Least Square, Total

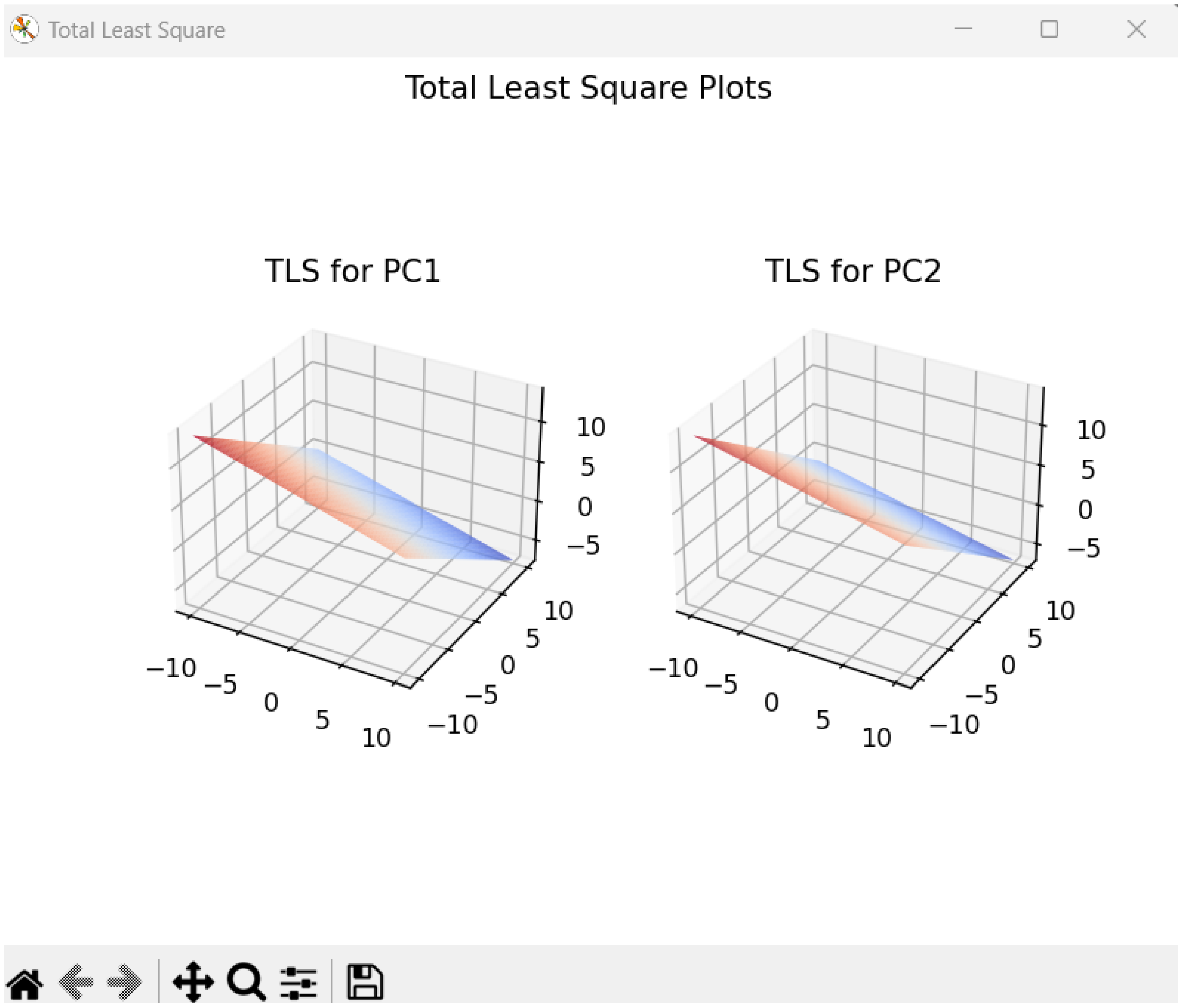
Least Square and RANSAC. Then these functions were called with values from PC1 and PC2 data lidar data points.

a. The code begins by importing some standard libraries including numpy, pandas and matplotlib.
b. The first step is reading the csv file and extracting data from it. This was done using pandas as they have very convenient functions available to perform reading data from csv.
c. I have stored the values obtained from PC1 into an array named dataFrame1 and PC2 coordinates into array named dataFrame2.
d. As asked in first question, to computer covariance matrix, I have written a function named find_covariance(dataFrame), which calculates covariance from scratch.
e. For part b, we can calculate the magnitude and direction of the surface normal by finding eigen_values and eigen_vector of the covariance matrix. The eigen vector corresponding to minimum eigen value denotes the direction of normal and the magnitude is denoted by the minimum eigen value.
f. In part 2, I have written three separate functions for Least Square, Total Least Square and RANSAC.
g. Standard Least Square function works in the same way, as indicated in question 1, by calculating the coefficients of a parabola, except for that it will now apply to 3D instead of 2D points



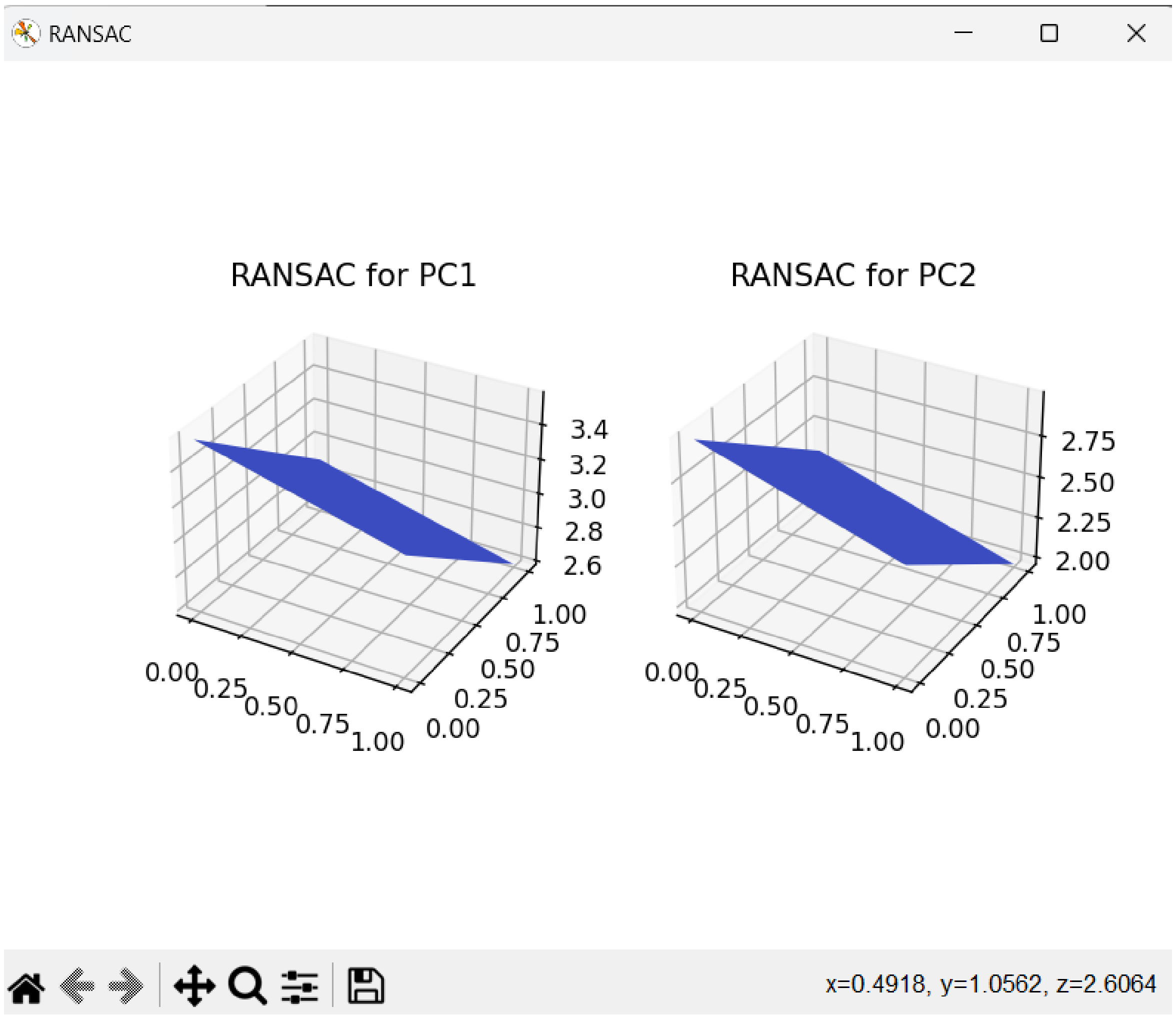
h.

i. Total Least Square function finds distance in both the x and y axis(diagonal) and tries to minimize the distance. This can be done by finding out the eigen values and eigen vectors, then extracting the one with least eigen value. We will utilize these eigenvector values to compute the coefficient and hence obtain the curve of the parabola.

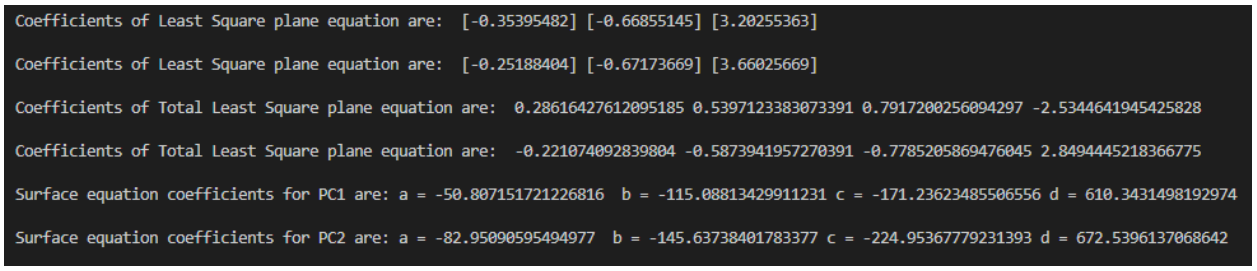


j.

k. Coming to RANSAC, we pick up any 3 random points and try to find inliers within a certain threshold(0.1 in our case) and this is used to find the best model which indicates the point with maximum number of inliers.



I.



```
Coefficients of Least Square plane equation are:  [-0.35395482] [-0.66855145] [3.20255363]

Coefficients of Least Square plane equation are:  [-0.25188404] [-0.67173669] [3.66025669]

Coefficients of Total Least Square plane equation are:  0.28616427612095185 0.5397123383073391 0.7917200256094297 -2.5344641945425828

Coefficients of Total Least Square plane equation are:  -0.221074092839804 -0.5873941957270391 -0.7785205869476045 2.8494445218366775

Surface equation coefficients for PC1 are: a = -50.807151721226816  b = -115.08813429911231 c = -171.23623485506556 d = 610.3431498192974

Surface equation coefficients for PC2 are: a = -82.95090595494977  b = -145.63738401783377 c = -224.95367779231393 d = 672.5396137068642
```

The above mentioned are the coefficients of each plane plotted for PC1 and PC2 respectively.

My conclusion from the model fitting tells me that RANSAC is most robust to outliers and Standard Least Square fitting is the most sensitive one, with total least square fitting existing somewhere in between.

I faced a lot of problem initially understanding how to code all three, though it sounds easier in theory, implementation has its own problems associated. Though the assignment was fun and I have learned a lot from this. Hopefully, I will be able this to the autonomous project I am working on.