Name : Vignesh Ravichandran Radhakrishnan
UID: 119144060
Directory ID: rr94

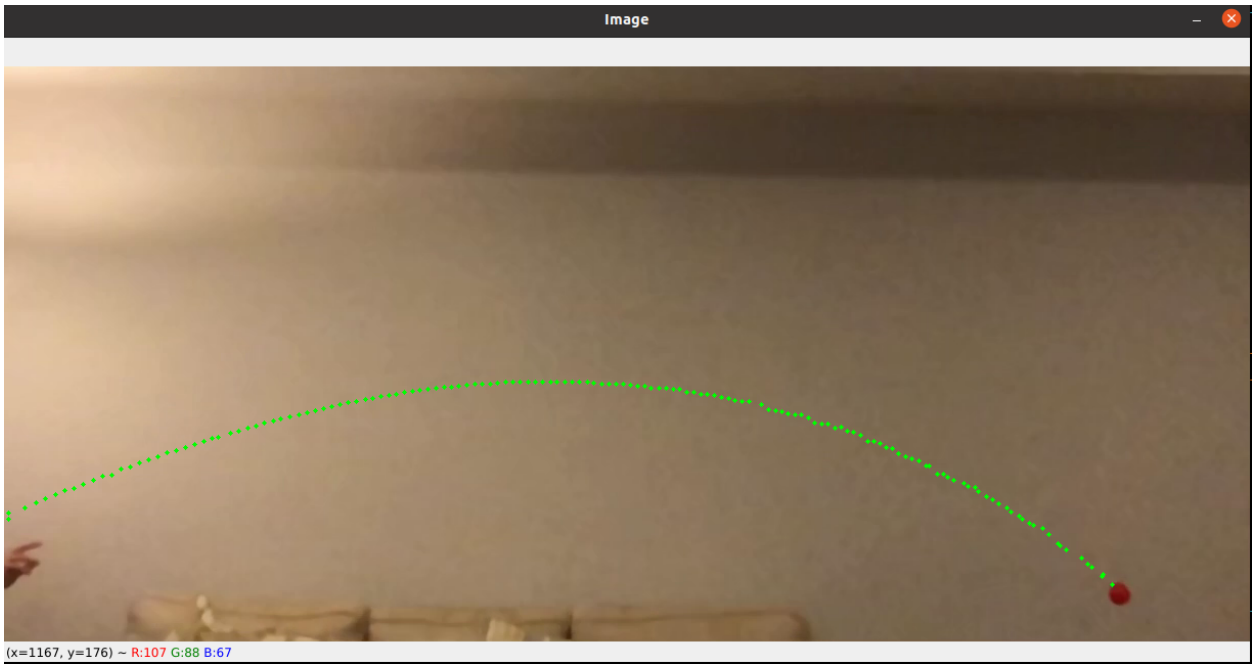
ENPM673 – Perception for Autonomous Robots
Project 1

Problem 1.1:

Solution:

Object detection (Ball) and plotting the pixel coordinates of its center point

- Read the video file frame by frame until end
- Use HSV mapping to obtain the upper and lower thresholds to mask the ball
- The contours are found by checking the mask where the pixel value is greater than 0 indicating the presence of the ball.
- The center is calculated by calculating the mean of x and y points in each frame.
- The center of the ball is plotted using cv2.circle function in a loop for all the x and y center coordinate points of the ball.



Problems encountered:

- Finding the mask required to track the ball with proper upper and lower red threshold values.
- Avoiding the other unintentional objects/pixel values which had similar RGB value

Problem 1.2:
Solution:

The Standard least square is obtained by calculating using the formula

$$A^T Ax = A^T b \text{ or } x = (A^T A)^{-1} A^T b = A^+ b$$

Where:
Matrix A -> x^2 , x and the constant (1) values of the center of the ball (order : n*3, where n is the number of points detected for center of the ball)

Matrix B -> y coordinate values of the center of the ball (order : n*1, where n is the number of points detected for center of the ball)

Matrix X -> gives the coefficients a,b,c in the equation of parabola of the form:

$y = ax^2 + bx + c$

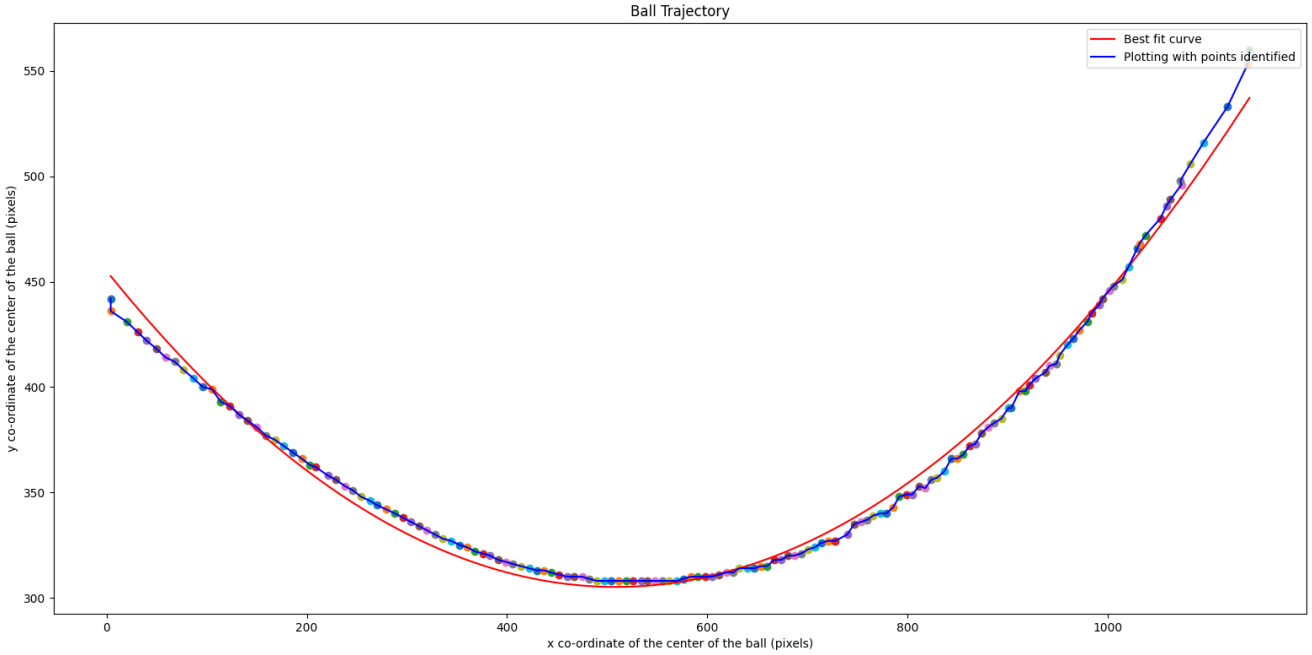
a) The Equation of the curve is:

$y = 0.00058\ x^2 - 0.589\ x + 454.97$

Output from code:

```
The equation of parabola is :
y = 0.00058 * x^2 -0.58900 * x + 454.97075
```

b) Data plotted : Comparison of Best fit curve (from equation) and the center points detected in the ball is shown below in the plot:



Problems encountered: None

Problem 1.3:

Solution:

The final landing y-coordinate of ball = $y_{initial} + 300$

The equation of parabola is given by,

$y = 0.00058\ x^2 - 0.589\ x + 454.97$

$y_{initial} = 452.624$ (Obtained by substituting x = x[0] which is the first detected location in the above equation)

The final landing x-coordinate of the the ball can be obtained by solving the below quadratic equation:

$y_{initial} + 300 = 0.00058\ x^2 - 0.589\ x + 454.97$

$\Rightarrow 0.00058\ x^2 - 0.589\ x - 297.654$

The x-coordinate of the landing position of ball in pixels is

x_{final} = 1388 (approximately in integer)

Problems encountered: None

Problem 2.1:

Solution:

a) The covariance can be calculated using the following steps:
   - The mean of each variable (x,y,z) is calculated and then subtracted from individual values to find the deviation.
   - The covariance matrix is calculated by using the below formula:

$$K = \begin{bmatrix} Var[X] & Cov[X,Y] & Cov[X,Z] \\ Cov[X,Y] & Var[Y] & Cov[Y,Z] \\ Cov[X,Z] & Cov[Y,Z] & Var[Z] \end{bmatrix}$$

Where

Cov|x,y| is given by,

$$COV(x,y) = \frac{1}{n}\sum (x_i - \bar{x})(y_i - \bar{y})$$

Var|x| is given by,

$$S = \frac{1}{n}\sum_i (x_i - \bar{x})(x_i - \bar{x})^T$$

The final Covariance matrix obtained from the code output is:

```
The covariance Matrix is :
[[ 33.6375584   -0.82238647 -11.3563684 ]
 [ -0.82238647  35.07487427 -23.15827057]
 [-11.3563684  -23.15827057  20.5588948 ]]
```

b) The magnitude (Eigenvalue) and the direction (Eigenvector) of the surface normal are obtained using the inbuilt function np.linalg.eig()

The output obtained from python code is as follows:
The magnitude of surface normal is : 0.6672780805108545
The direction of surface normal is : [0.28616428 0.53971234 0.79172003]

Problems faced: None

Problem 2.2

Solution 2.2

a) **Standard Least Square Method:**
The data is read from the csv files and the x,y and z values are separated. The x, y and z values are passed into a function to calculate the least square using the formula:

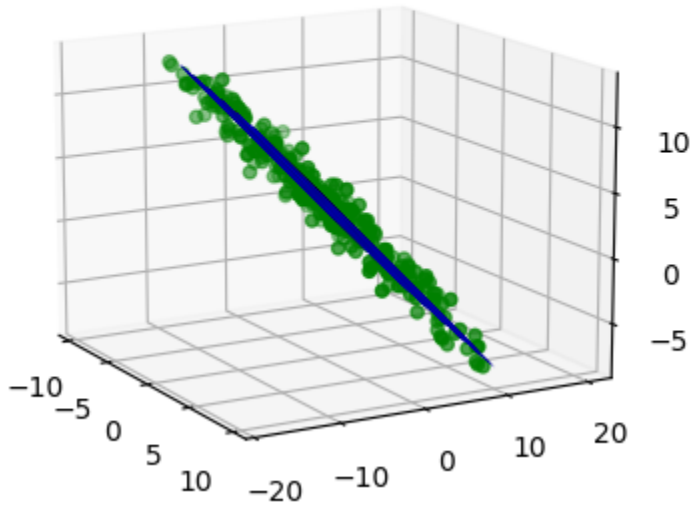
$$A^T Ax = A^T b \text{ or } x = (A^T A)^{-1} A^T b = A^+ b$$

Total Least Square Method:

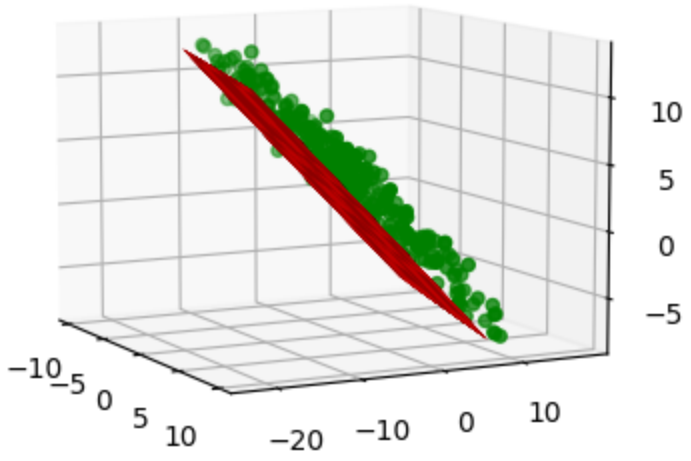
- The data is read from the csv files and the x,y and z values are separated.
- The x, y and z values are passed into a function to calculate the covariance using the user defined function.
- The eigen-values and eigen-vectors are calculated using the inbuilt function np.linalg.eig()
- The eigen-vector corresponding to the least eigenvalue represents the direction of normal and that eigenvalue represents the magnitude of the vector.

The surface plot for pc1.csv and pc2.csv using Standard least square and total least square methods are plotted below:

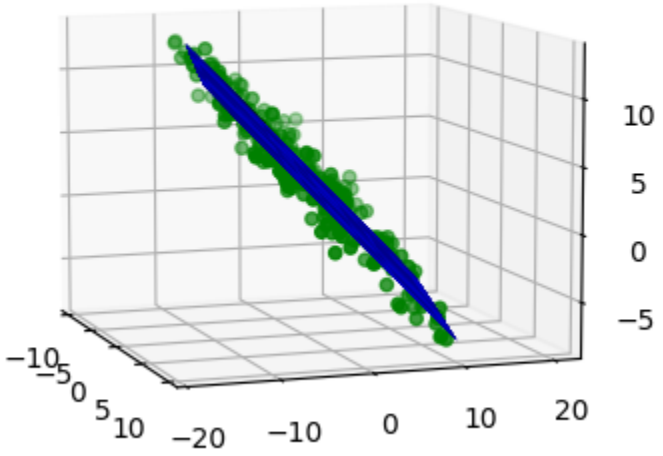
Plane estimated using standard least square for pc1.csv'

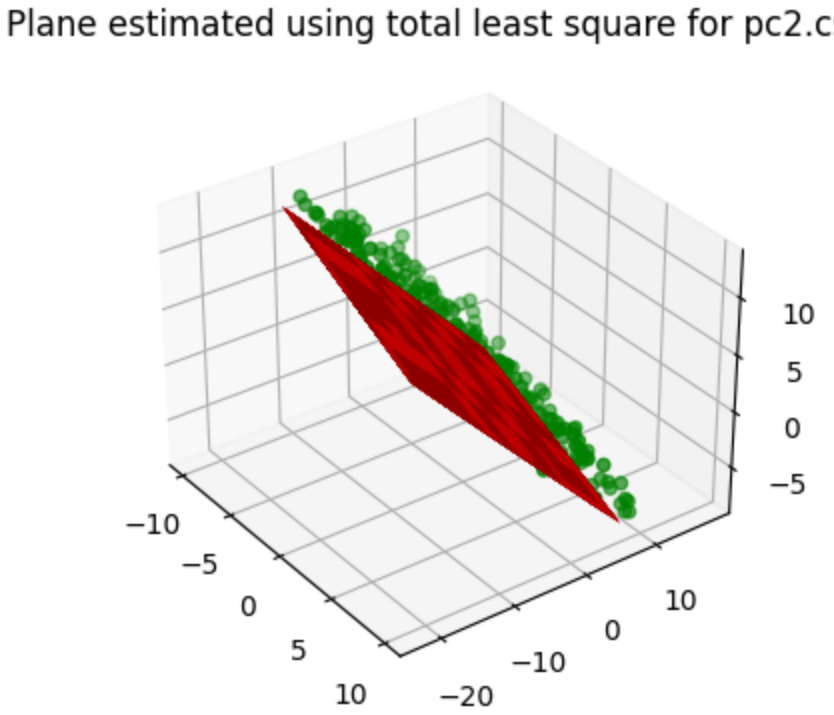


Plane estimated using total least square for pc1.csv



Plane estimated using standard least square for pc2.csv





Inference/Interpretation: From the above graphs, it is evident that the least square method provides the best surface fit for the given points in the data for both pc1.csv and pc2.csv data. The surface plot for least square covers or passes through the maximum number of points and hence considered the best fit when compared to the total least square method.

Problems encountered:
The linear matrix format was not able to be plotted and I had to convert it to a meshgrid to plot it to obtain the plane surface which fits the data points.

Solution 2.2 (b)

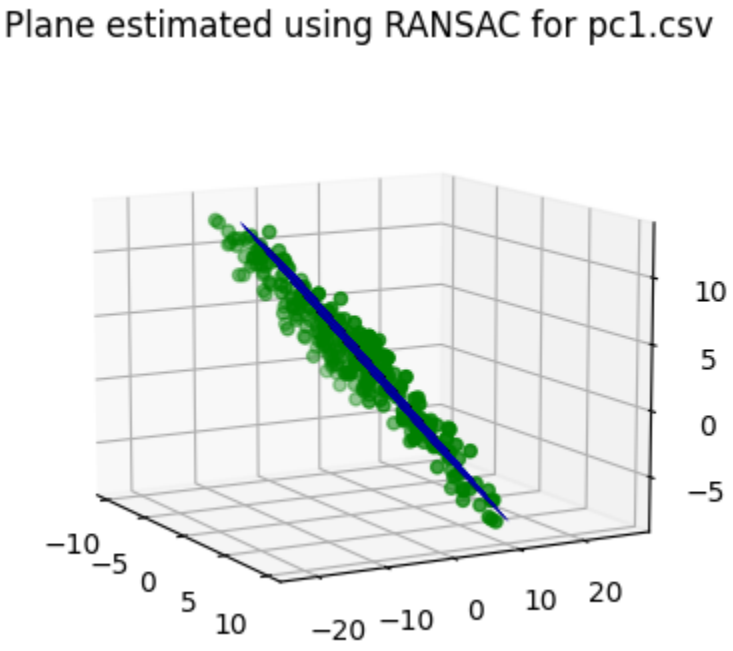
The surface fit for data sets pc1.csv and pc2.csv are plotted below using RANSAC (RANdom SAmple Consensus)

RANSAC Code:

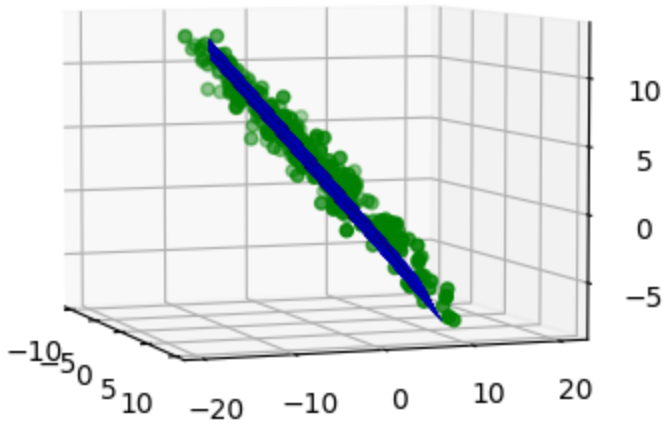
The function takes into 6 parameters into account:
n -> number of iterations

s -> sample size required for drawing a plane
t -> threshold value
x,y,z -> data points obtained from the pc1.csv/pc2.csv

- The number of iterations will run over the sample size and check for the error between the different entities.
- Once the error is lesser than threshold, the counter value is incremented and the number of points which are inliers are obtained.
- The above steps will run for as many iterations as requested and the final plane is plotted using the vector obtained for the points which are inliers based on the threshold provided.
- The cross product between the difference of 2 vectors is used to find the vector normal to the plane.
- A surface plane passing through the inlier points is plotted.



Plane estimated using RANSAC for pc2.csv



Problems encountered:

The Ransac algorithm is random and the surface plots are not constant. The number of inliers found is based on the threshold and the random values of x,y,z obtained differ from one iteration to another. Thus, the best fit curve was only able to be plotted after multiple trial and error attempts.

Inference/Interpretation:

It can be inferred that the RANSAC algorithm looks computationally expensive as it involves more calculations and runs over multiple iterations. Also the RANSAC algorithm is highly random and higher training is required for optimal results. Thus, for the given data, with lesser outliers, the least square method is computationally inexpensive and provides constant results.

Final Interpretation for Outlier Rejection: It can be inferred that the RANSAC method is the best for outlier rejection since we can provide the threshold and improve the fit for the data available and our precise need based on the application. Also for data where there are more outliers, RANSAC method/algorithm is the best fit to reject the outliers and provide an optimal surface, fitting the points.