ENPM673 Project One Report

Suriya Suresh

1. Ball Tracking

For this, a video of a red ball being thrown was given. The trajectory of the ball takes a parabola. Each frame of the video was read in an infinite loop which terminates on the lack of new frames from the video. Each frame underwent processing and a sequence that is described below.

The steps done were:
1. The part of the video which had the hand throwing the ball was masked off to reduce false positives before doing the masking operation.
2. Doing a Gaussian Blur with kernel size (11,11) to reduce noise.
3. The frame is then converted to an HSV color space which is easier to filter out specific colors.
4. The cv2.inrange function was used to filter out specific HSV values within a lower and upper limit, giving a frame of black and white pixels. The target pixels are white, which had specific colors within the defined range.
5. The image was eroded and dilated to reduce white noise and to counter the effects of erosion on the image.
6. Then the mean of the pixels detected to determine the center of the ball.
7. The center points (x,y) were then fitted using Least Squares to give the parabola equation which fits all the points as well as possible.
8. The plots of the original path of the ball and the calculated path of the ball were plotted on a graph.
9. The coordinates of the x component of where the ball landed were determined, given the y component.

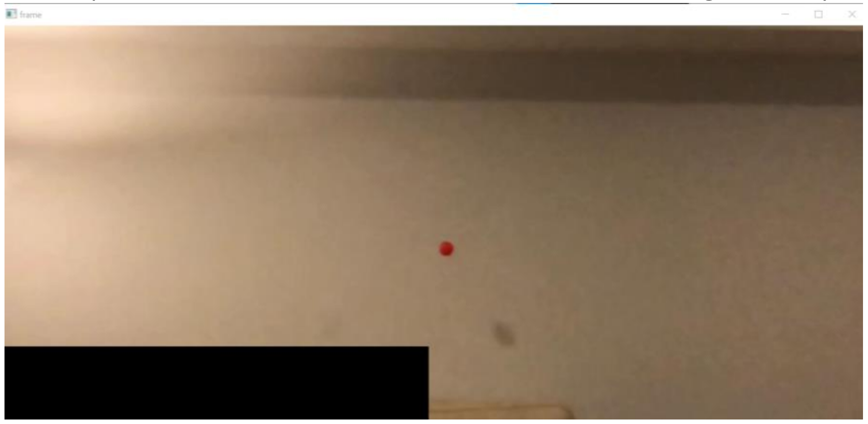


Fig 1: The raw frame with the lower corner masked off.

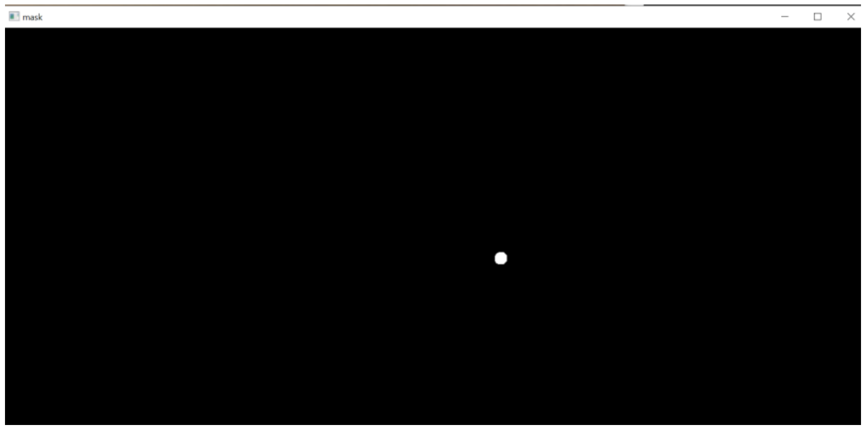


Fig 2: The output of cv2.inrang

1.1 Least Squares Fitting of Parabola

The least squares fitting works on the principle of minimizing the mean square error between the predicted points and actual points for the best fit of a line. The general equation of a parabola is taken, and the error is calculated along the Y direction only.
For least squares, the equation of the parabola was taken to be:

Equation of Parabola $= ax^2+bx+c=y$

$$y=\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad x=\begin{bmatrix} x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_1 & 1 \end{bmatrix} \qquad A=\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$A=(X^TX)^{-1}(X^TY)$$

Fig 3: Equation of parabola and Matrix form of equation

The error is found by:

$$E=\|Y-XA\|^2$$

Fig 4: The error equation

Differentiating it with respect to the coefficient matrix we get was used to determine the A Matrix value.

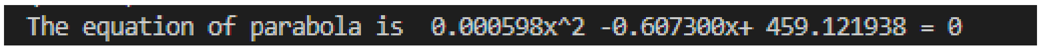
$$=(Y-XA)^T(Y-XA)$$
$$=Y^TY-2(XB)^TY+(XA)'XA)$$

Fig 5: Expanding the error equation.

$$dE/dA = 2X^TXA - 2X^TY=0$$
$$X^TXA=X^TY$$
$$A=(X^TX)^{-1}(X^TY)$$

Fig 6: Differentiating the error equation.

The final equation after fitting was:



```
The equation of parabola is  0.000598x^2 -0.607300x+ 459.121938 = 0
```

Fig 7: The equation of the parabola after fitting

1.2 The plot of the Parabola

The plot of the parabolas, both extracted and calculated is:

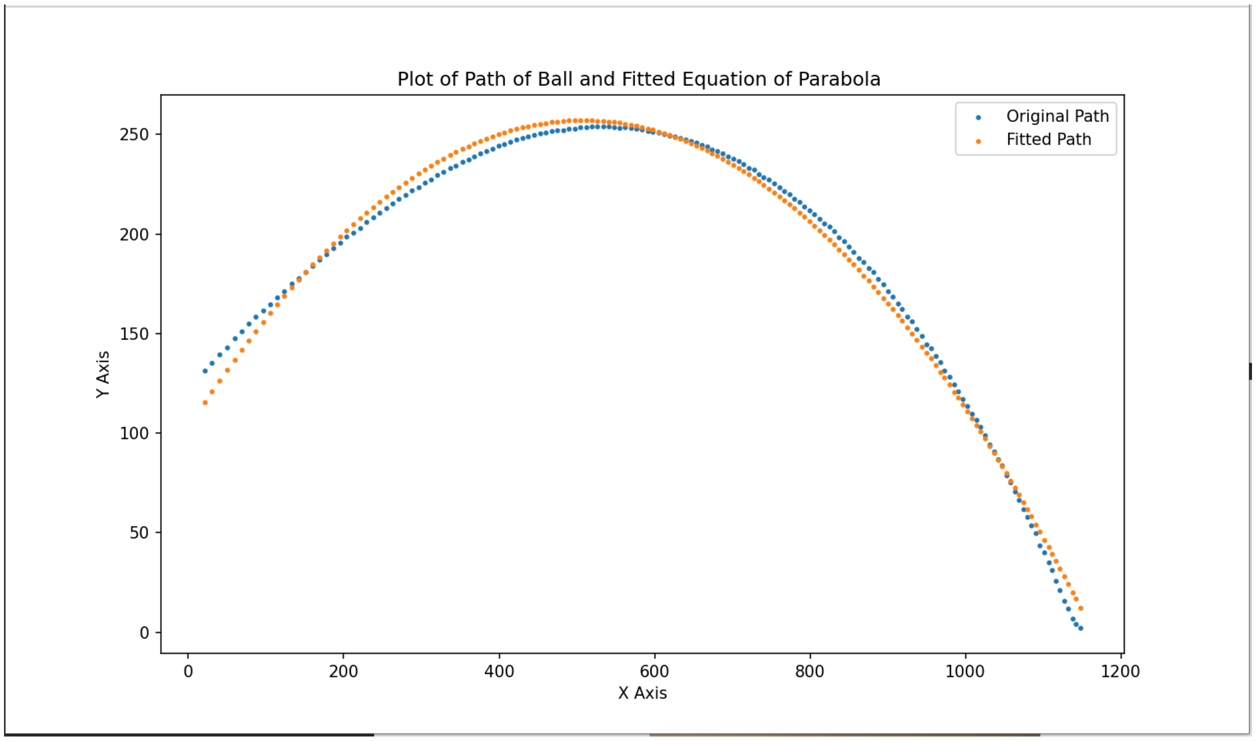


Fig 8: Equations of the Original path of the Ball and fitted path of the ball

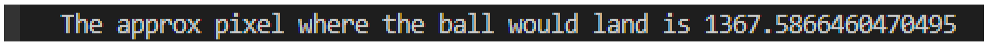
The equation of the parabola found after fitting was used for calculating the landing coordinates of the ball. The fitted y points were used, the first point was taken as the origin where the ball started its trajectory.

1.3 Results and Observations

We can see that the outline of the parabola curve is very similar to the original path of the ball. It also shows the noise has not been fully eliminated and traces of it exist, affecting the detection of pixels of the path the ball took. The fitting of the curve to the points has been relatively good. Using more computational expensive fitting methods is not desirable as the number of points is lower and the fit may not be better.

1.4 Landing Point of Ball

The landing point of the ball is



```
The approx pixel where the ball would land is 1367.5866460470495
```

Fig 9: The point of the landing of the ball

We get a positive and negative value from the solved equation. The negative value is ignored as the ball starts on the positive X-Y plane and travels in the direction of the positive X-axis.

2. Covariance Matrix, Least Square, Total Least Squares and RANSAC Fitting of Point Clouds

This problem involved two given 3-D point clouds which the calculation of covariance matrixes was done. A plane was approximated to the points using various fitting methods as described later in this section.

2.1 General Overview of Steps Done

The steps done were:
1.The points were extracted from the given CSV files using pandas.
2. Calculate the covariance matrix of Point Cloud 1 and 2.
3. Using the covariance matrix, the eigenvalues, and eigen vector were found to determine the surface normal of the plane of the point clouds.
4. Least squares, Total least squares, and RANSAC algorithms were used to fit the points to a plane and the planes plotted along with the points.

2.2 Covariance Matrix

The covariance matrix represents the variance and covariances of set of points. It showcases the relation between the variables of a set of points. We calculate the Covariance matrix by:

$$\text{Covariance matrix} = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z') \end{pmatrix}$$

Fig 10: Covariance Matrix

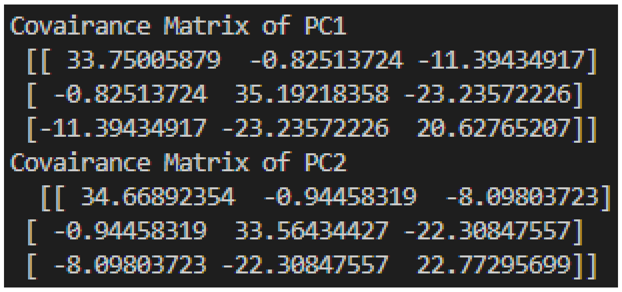
The covariance for each element of the matrix is calculated by:

$$\mathrm{cov}(x,y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{n-1}$$

Assuming the sample size of the entire dataset is not n & only n points were given.

Fig 11: The covariance formula

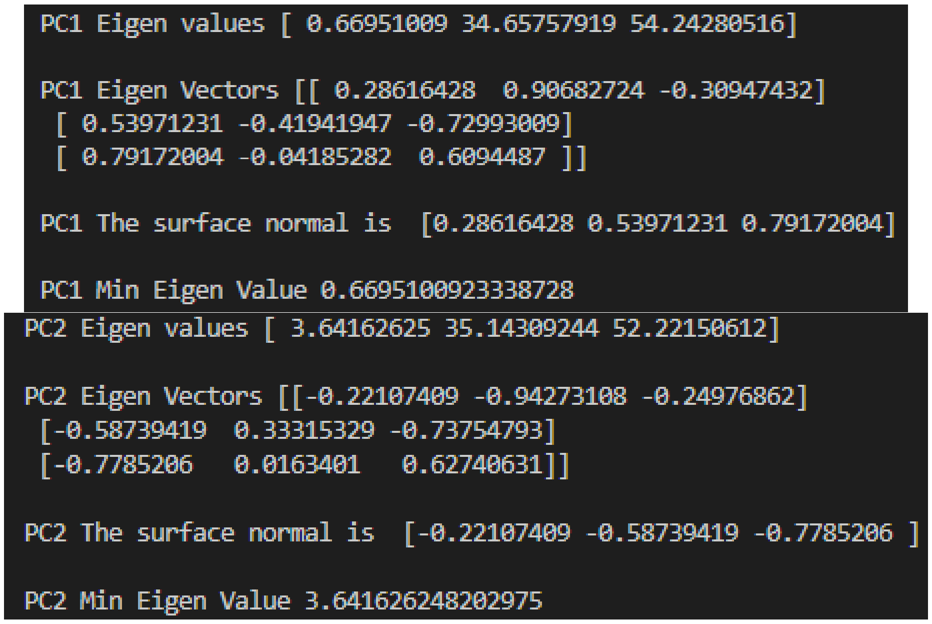
The Covariance matrixes for PC1 and PC2 are:



```
Covairance Matrix of PC1
 [[ 33.75005879  -0.82513724 -11.39434917]
 [ -0.82513724  35.19218358 -23.23572226]
 [-11.39434917 -23.23572226  20.62765207]]
Covairance Matrix of PC2
  [[ 34.66892354  -0.94458319  -8.09803723]
 [ -0.94458319  33.56434427 -22.30847557]
 [ -8.09803723 -22.30847557  22.77295699]]
```

Fig 12: Covariance matrix of PC1 and PC2

We found the eigenvalues and eigenvectors of the matrixes. The vector associated with the least eigenvalue is the surface normal direction and its magnitude is 1.



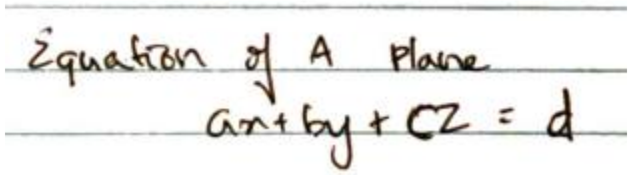
```
PC1 Eigen values [ 0.66951009 34.65757919 54.24280516]

PC1 Eigen Vectors [[ 0.28616428  0.90682724 -0.30947432]
 [ 0.53971231 -0.41941947 -0.72993009]
 [ 0.79172004 -0.04185282  0.6094487 ]]

PC1 The surface normal is  [0.28616428 0.53971231 0.79172004]

PC1 Min Eigen Value 0.6695100923338728
```

```
PC2 Eigen values [ 3.64162625 35.14309244 52.22150612]

PC2 Eigen Vectors [[-0.22107409 -0.94273108 -0.24976862]
 [-0.58739419  0.33315329 -0.73754793]
 [-0.7785206   0.0163401   0.62740631]]

PC2 The surface normal is  [-0.22107409 -0.58739419 -0.7785206 ]

PC2 Min Eigen Value 3.641626248202975
```

Fig 13: The eigen values, eigen vectors and surface normal of PC1 and PC2

2.3 Least Square Fitting

The steps and equations used for the least squares method are similar to the ones described for fitting the parabola in the previous question. Only the equation for fitting is the equation of a plane.
We take the equation of the plane:

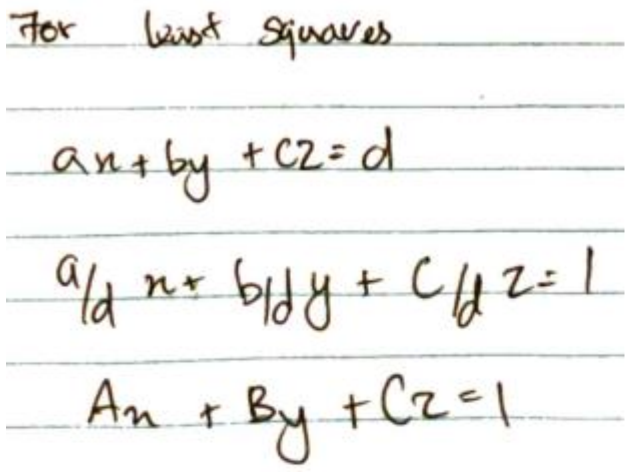


Equation of A Plane

$$ax + by + cz = d$$

Fig 14: Equation of the Plane

For least squares we take the matrixes as



For least squares

$$ax + by + cz = d$$

$$a/d\, x + b/d\, y + c/d\, z = 1$$

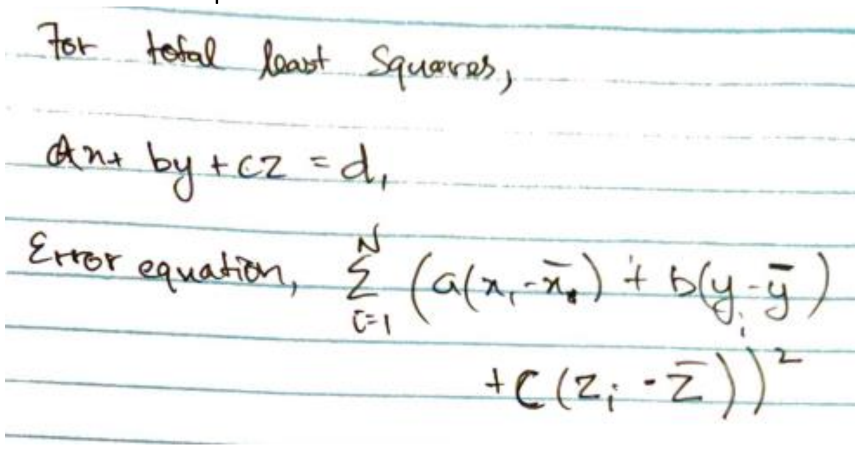
$$Ax + By + Cz = 1$$

Fig 15: The equation of plane used for the least squares method

The equation of the plane is divided by the RHS to obtain one to simplify calculation overall. After the values of A,B and C are found, the plane equation is found and used to plot the plane and the points to show how well the plane fits the data.

2.4 Total Least Squares

Total least squares operate on minimizing the error along the perpendiculars of the plane to find the best fit. The equations used for the total least squares fitting are described below.
The equation of the plane and the error equation is:



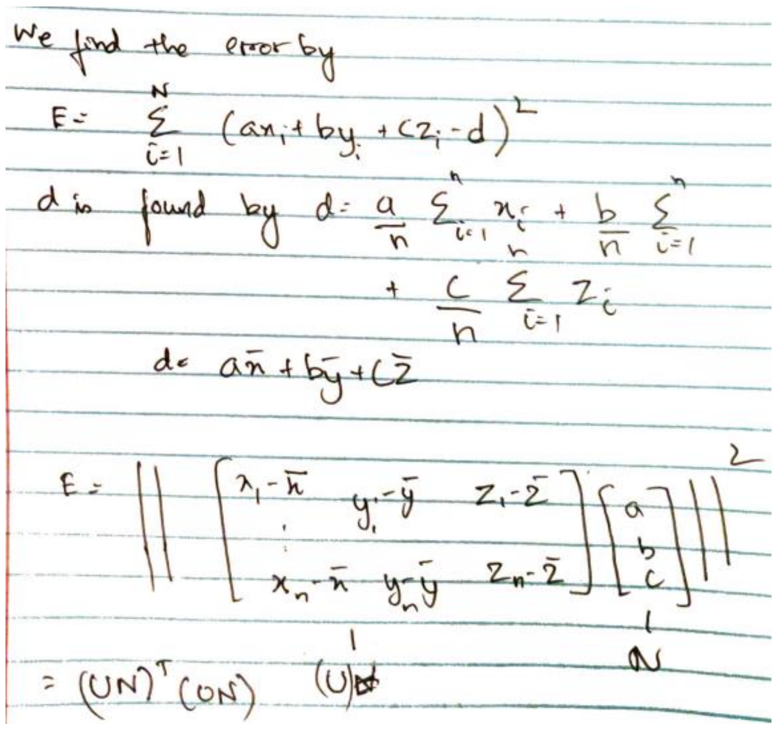
For total least Squares,

$ax + by + cz = d,$

Error equation, $\sum_{i=1}^{N} \left(a(x_i - \bar{x}) + b(y_i - \bar{y}) + c(z_i - \bar{z})\right)^2$

Fig 16: The plane equation used for TLS.

In the error equation, d is substituted to get the final equation from which the moment matrix U is determined.



We find the error by

$$E = \sum_{i=1}^{N} (ax_i + by_i + cz_i - d)^2$$

d is found by $d = \frac{a}{n}\sum_{i=1}^{n} x_i + \frac{b}{n}\sum_{i=1}^{n} + \frac{c}{n}\sum_{i=1}^{n} z_i$

$$d = a\bar{x} + b\bar{y} + c\bar{z}$$

$$E = \left\| \underset{U}{\begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ \vdots & & \\ x_n - \bar{x} & y_n - \bar{y} & z_n - \bar{z} \end{bmatrix}} \underset{N}{\begin{bmatrix} a \\ b \\ c \end{bmatrix}} \right\|^2$$

$= (UN)^T (UN)$ $(U)N$

Fig 17: Determining the equation for the N matrix

When differentiating with respect to the N matrix we get

$$\frac{dE}{dN} = 2(U^TU)N = 0$$

$$U^TU = 0$$

Fig 18: Final equation used for TLS

The solution for the equation is found by determining the eigenvalues and eigenvectors of the final error equation for which the solution cannot be zero as the norm of N is equal to one. The elements of the vector associated with the smallest eigenvalue are the solutions to the variables a,b, and c.

2.5 RANSAC Fitting

The RANSAC algorithm involves a few basic steps which are:

1. A subset of the points is chosen, usually the size which is least needed to find the values of a model.
2. A model is fit to that subset.
3. Using a certain parameter, the points that are close to the model with respect to that parameter are found and the rest is beyond a certain threshold are rejected as outliers.
4. This process is done N times and the best model with the least outliers is chosen as the model for the set of points.

Here the model was fitted using least squares and total least squares, The number of iterations chosen. The number of iterations can be found by:

$$N = \log(1- p)/ \log(1- (1- e)^s)$$

Where:

N is the total number of iterations

p is the probability of getting a good sample

s is the number of points in a subset

e is the probability a point is an outlier.

For fitting this specific point cloud, the values taken were s=3, p=0.99, and e=0.5.
RANSAC handles noises and outliers better than least squares and total least squares.PC1 and PC2 has noise and PC2 has outliers. However, there is a risk of overfitting which when used on another dataset which characteristics similar to PC1 or PC2, it may not fit well. There is a fine line between underfitting, fitting it aptly, and overfitting a plane to a set of points.

Here the step 2, model fitting was done with least squares and total least squares was done and plotted in the following section. This was done to determine the effects of changing the method used to fit the proposed on the final fit which has been captured in the plots in the next section.The number of iterations was taken as 1000 as the calculated N was very low, about 50 iterations.

2.6 Plots

Least Squares, Least Total Squares, and RANSAC Methods with Least Squares and Total Least Squares Fitting for PC1

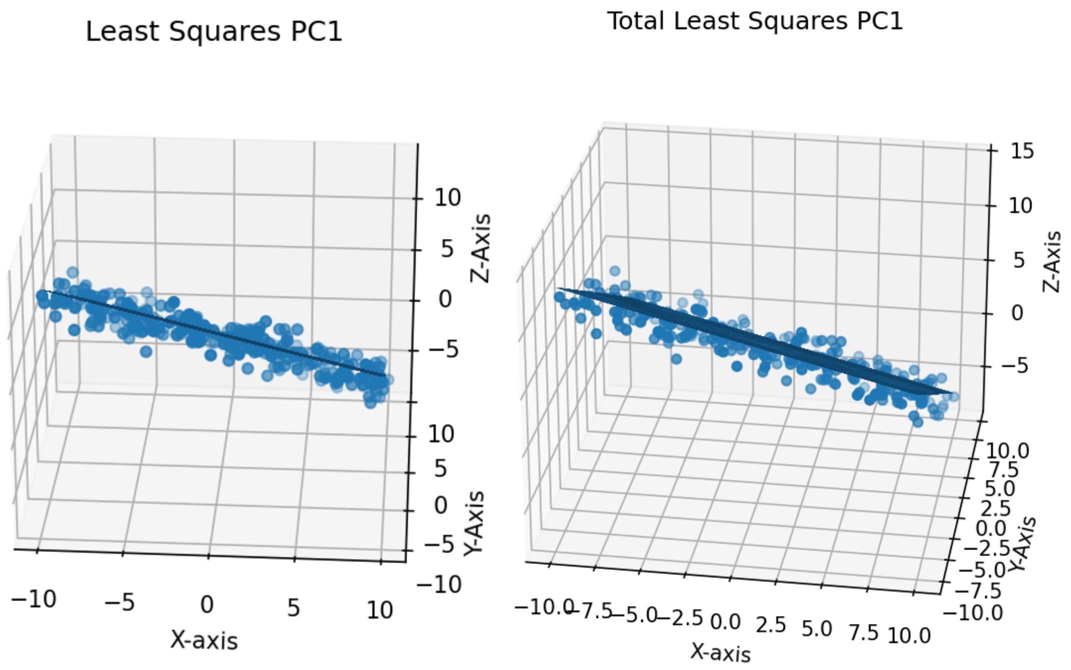


Fig 19: Least Squares and Total Least Squares Plots

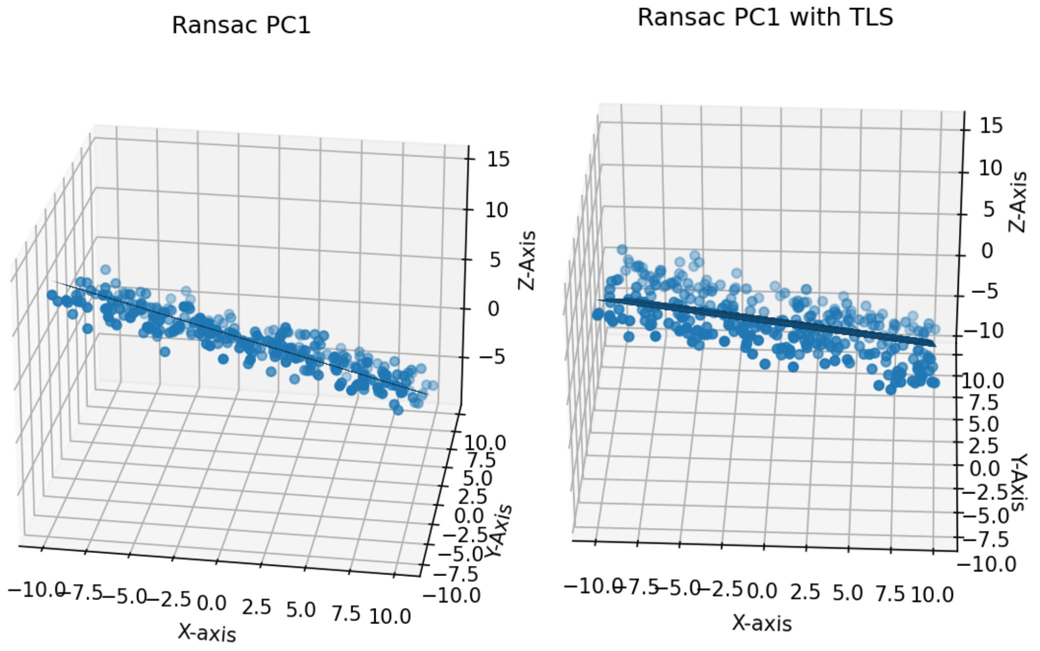


Fig 20: Plot of RANSAC with LS and TLS

Least Squares, Least Total Squares, and RANSAC Methods with Least Squares and Total Least Squares Fitting for PC2

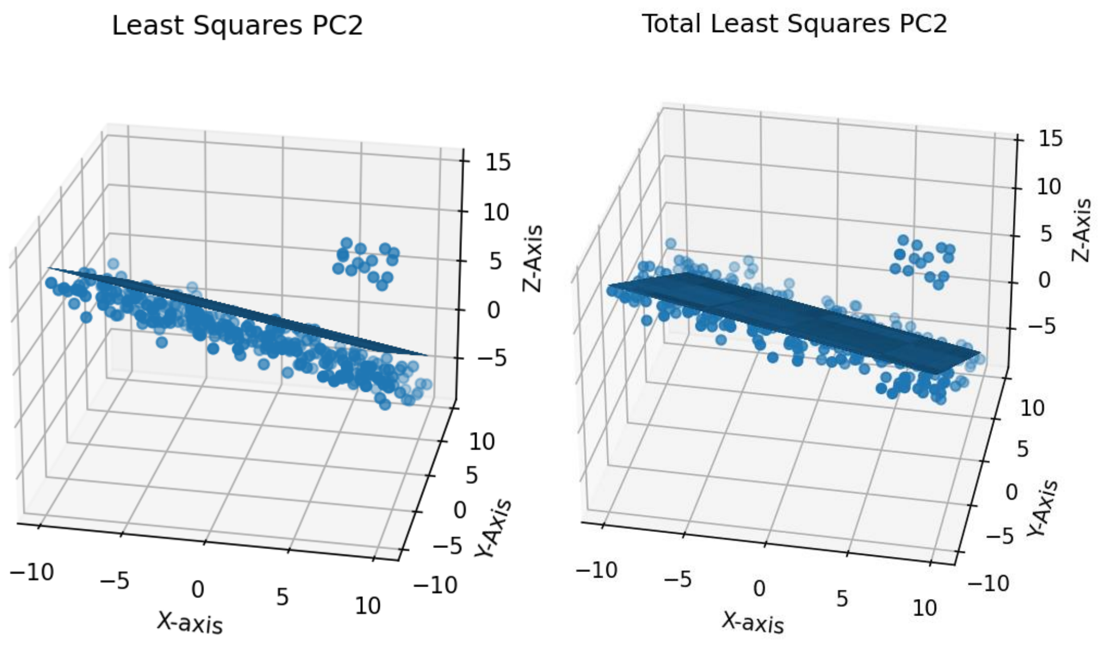


Fig 21: Least Squares and Total Least Squares Plots

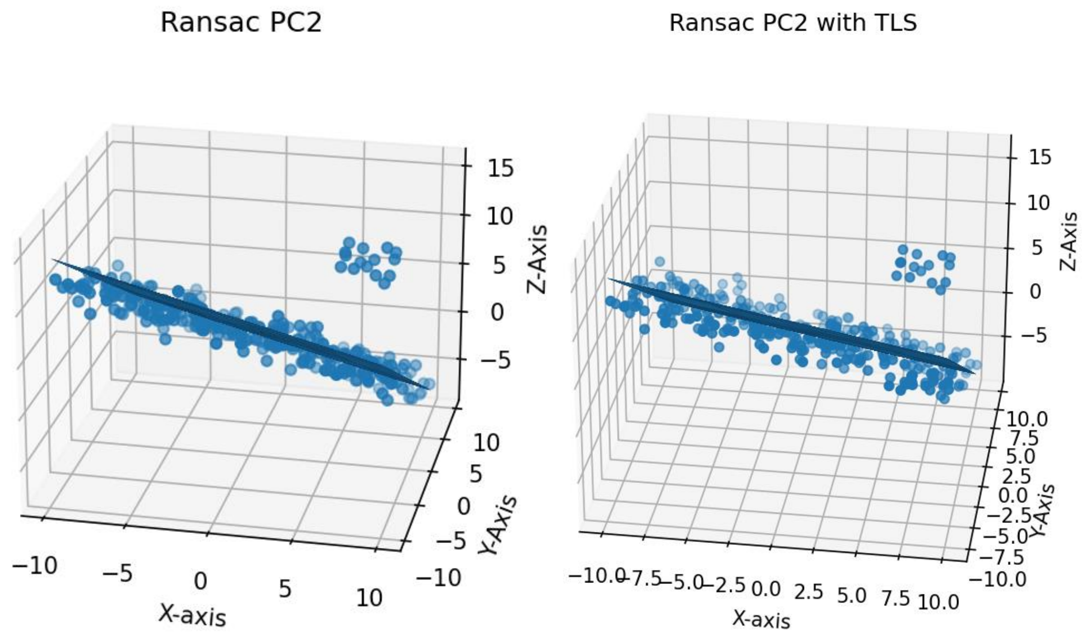


Fig 22: Plot of RANSAC with LS and TLS

2.7 Results and Observations

As we can see from the plot, RANSAC offers the best fit over other fitting methods. We can see for PC1,least squares and total squares have a similarly fitted plane with a few different points not on the plane in both the graphs. RANSAC fits the points and RANSAC with TLS fits better than RANSAC with LS , which shows the ability of total least squares to handle noise better than least squares. For PC2, we can that the least squares method did not fit the points well as the plane is over majority of the points. The influence of the outliers present in PC2 can be clearly seen affecting the fitting. Total least squares fit to the points of PC2 much better but still doesn't fit very well. RANSAC with LS and RANSAC with TLS fit the data much better but RANSAC with LS is affected more by noise and outliers. RANSAC with TLS has fitted PC2 the best with the influence of outliers to be lower. This shows the inherent weakness of least squares to noise and using RANSAC reduces the effect of noise to an extent. RANSAC in combination with TLS method would be the preferred method to fit a plane to the PC1 and PC2.

The iterations of the loop were kept at 1000 as more iterations offered better performance at a much higher computational cost in terms of time. The tradeoff in time taken for performance gains is not suited for lower number of points like PC1 and PC2 have. The values of the coefficients for the equation of the plane changed for RANSAC each time the program was run, but the change in values were within a nominal range. The number of inliers and outliers of the RANSAC algorithm were constant within the margin of error of a few points for multiple runs of the program.

3. Problems Faced and Solutions Implemented

This section describes the issues faced and the solutions implemented for them.

For question 1:

1.Noise and false positives in the video

Due to the changing lightning conditions of the video, the hand sometimes was seen in the mask and noise was causing issues. It was resolved by doing blur and eroding and dilation of the image. A rectangle covering the hand was done for the raw video which helped reduce false positives. Further tuning of the HSV lower and upper limits for cv2,inrange also helped.

2. Getting NaN values in finding center of ball

If there was not a ball in the frames at the start and end of the videos, values of NaN were returned and which were included in the calculation of the center point of the ball. A if condition to check for NaN values resolved the issue.

3.Plotting the points on a Standard X-Y Plane

The image had the origin on the top left. After calculating the points, basic subtraction was done to plot the points on the normal X-Y plane. Basically, a transformation from the image coordinate frame to the graph coordinate plane was done.

3.Mismatching Dimensions of Numpy Arrays

While doing matrix operations for Least Squares, the dimensions of the arrays did not match. The shape of the arrays was changed before the operations were done and it resolved the issue.

For question 2:

1.Number of iterations for RANSAC

When the number of iterations N was calculated using the formula provided, the resulting plane fitted the points pretty badly. As a result, a standard of 1000 iterations was done but the number of iterations was kept at 5000 initially which took a longer time with increases in performance at the cost of more resources.

2. Plotting the Plane Equations

Plotting the plane equation for total least squares was sometimes going astray due to a mistake in passing the smallest eigenvector from the functions. A check to ensure the smallest eigenvalue's equal vector was returned was implemented.

3.Implementing the RANSAC Algorithm

Initially there was confusion on which fitting method to use to fit the model to the subset of points. As a result both the methods of least squares and total least squares were used to fit the models and accordingly give the graph. The performance of both methods could be seen also.

4. References

- Class Lecture Notes
- OpenCV documentation
- Numpy Documentation
- Pandas Documentation
- Matplotlib Documentation