1)

Homework 1

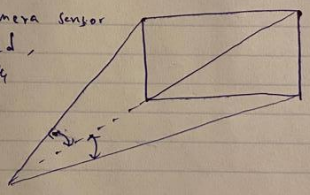
① Resolution = 5MP

Focal length (F) = 25mm.

Field of view = $2\tan^{-1}(x/2F)$

= 31°

Since the camera sensor is square shaped, the horizontal & vertical field of view have the same value.



② Object dimensions = 5×5

Camera is placed 20m away.

⇒ Size-to-distance ratio = 0.5/20 = 0.025.

Span of projected image → (0.025).25

= 0.625mm

Total area = $(14 \times 14)\ mm^2$

Area per pixel = $\frac{14 \times 14}{5 \times 10^6}$

Pixel width = $\sqrt{\frac{14 \times 14}{5 \times 10^6}} = 6.26 \times 10^{-3}\ mm$

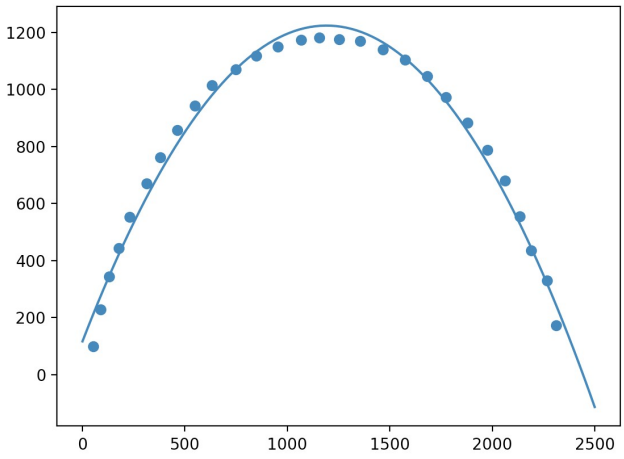
Number of pixels = $\frac{0.625}{6.26 \times 10^{-3}} = 100.$

2)

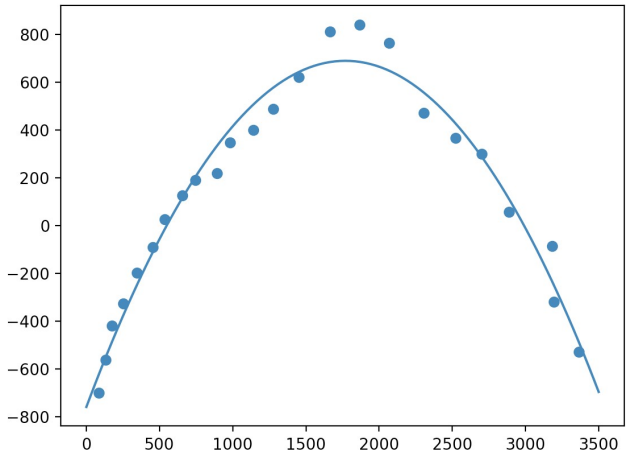
First step of solving this problem is to extract data points from individual frames. To make it easier to extract the data points , a binary image was generated. Due to this, the coordinates of the ball could be extracted as the coordinates where the image intensity is 0. (using np.where(image=0))

After extracing the data points, the linear least square method was used to fit the data. The weight vector can be computed using (Pseudoinverse(X).Y)

Following are the plots from fitting the data from the 2 videos



Data fitting from Video1



Data fitting from Video 2

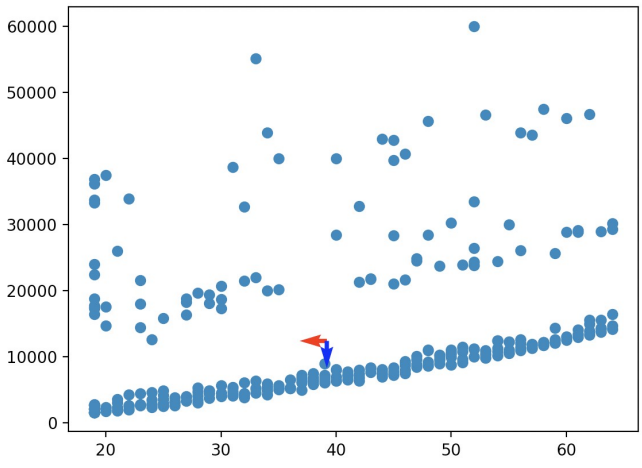
It can be observed from the above graph that even when there is noise included in the video, the least squares method is capable of generating a decent estimate of the data.

3)

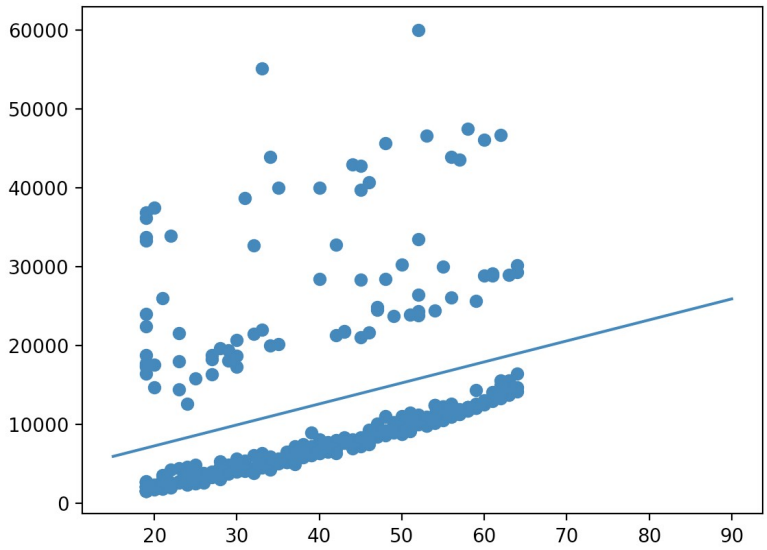
The data from the csv file can be extracted into a dataframe.

The 2X2 covariance matrix can be computed on python using the mathematical equations for varaince and covariance calculations.

Following is the graph generated by computing the covariance matrix, eigen values and eigen vectors. The two arrows represent the eigen vectors of the data

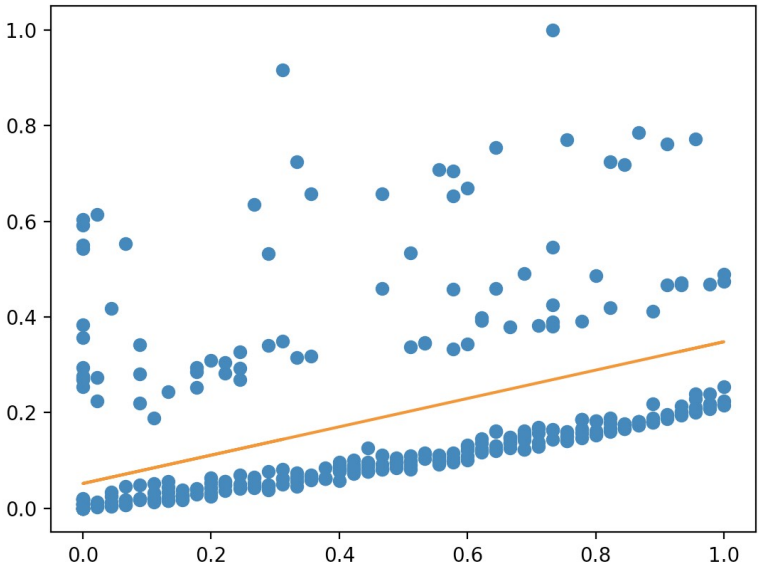


Following is the graph generated by fitting the data using linear least squares method



Standard least square function for this is the same as in question 2, only difference being the matrix X contains 3 columns of x^2 x and 1

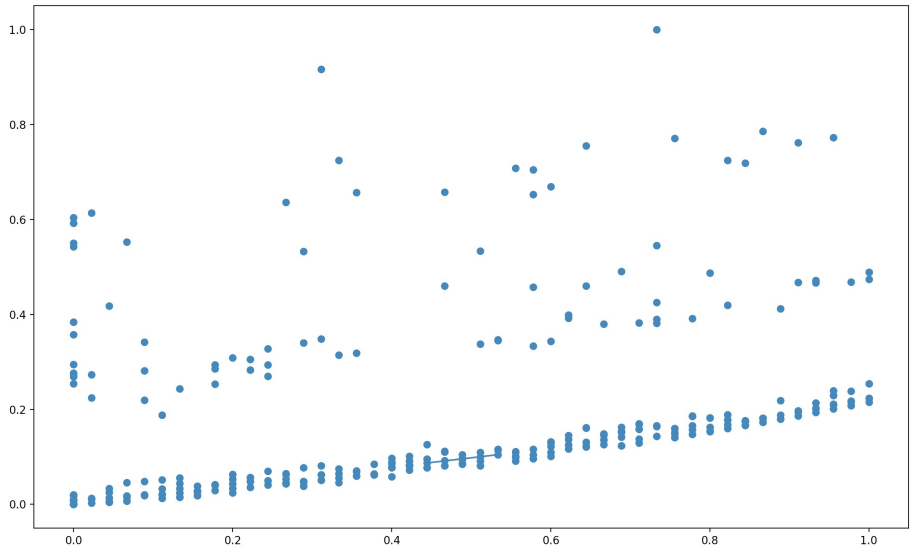
Following is the graph generated by using the total least squares method

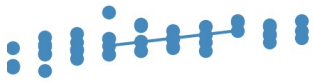


The total least squares method computes perpendicular distances of data points from the prediction line rather than vertical distances. Therfore, there is no possibility of a 0 veritcal distance from a data point to the line and leads to a slightly better accuracy.

For the RANSAC function, 2 random data points are selected and the number of outliers is computed. The points that give the least number of outliers are used to form the line.

Following graph is generated using the RANSAC method





Standard least square technique is simpler to compute as compared to RANSAC

As we can observe from the graphs above, RANSAC generates the best possible result to fit the given data

The main drawback of using standard least squares and total least squares is, if some of the data points have a very large error, they can effect the fitting of the line especially when the total number of data points is not very high. The total least squares method also has the tendency to overfit data

However, in RANSAC we can fit the data accurately even if there are outliers with a large error so long as there are sufficient itertions. Due to these reasons, RANSAC is the ideal method for data for this problem. Even though the large number of outliers shift the line in linear and total least squares method, RANSAC is not effected.

4)

Singular value decomposition of a matrix:

A matrix can be divide into 3 components

$$A = U\Sigma V^T$$

Where U and V are orthogonal matrices and Σ is a diagonal matrix with singular values

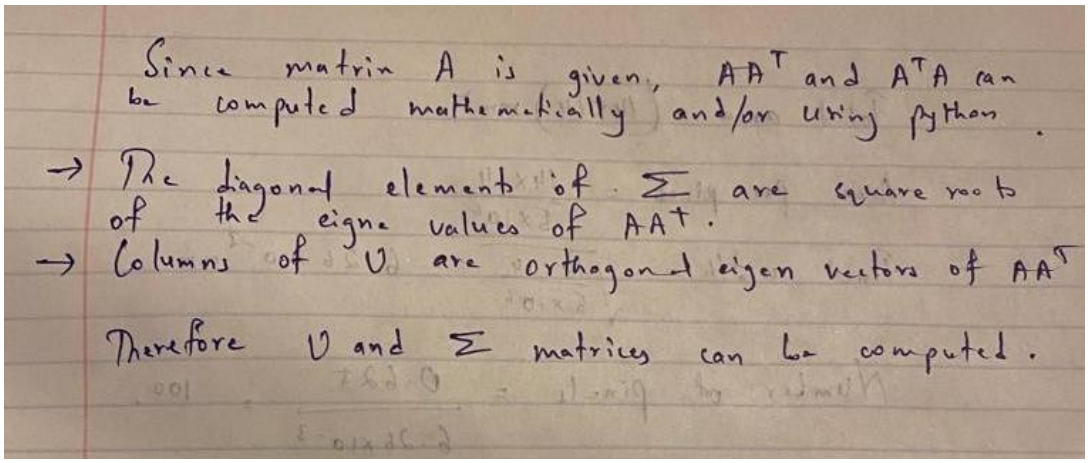
$$A^TA = V\Sigma^T U^T U\Sigma V^T$$

Since U is orthogonal, $U^TU = I$

$$\Rightarrow A^TA = V\Sigma^T\Sigma V^T$$

$$AA^T = U\Sigma V^T V\Sigma^T U^T, \quad V^TV = I$$

$$\Rightarrow AA^T = U\Sigma\Sigma^T U^T$$



U , V and H matrices respectively computed using the python code:

```
[[ 1.17519867e-02  3.44207228e-04 -5.15532162e-02 -4.66128587e-01
  -2.60345896e-01 -6.78428560e-02 -8.41087769e-01  1.08122937e-02]
 [ 1.17517760e-02  3.43641967e-04 -8.72103737e-02 -4.59351955e-01
  -2.49098952e-01 -8.85591890e-02  3.54169471e-01  7.65455993e-01]
 [ 3.58735699e-01  6.54942912e-01  1.34538659e-02 -4.65084492e-01
   1.70101644e-01  2.93617516e-01  1.82289869e-01 -2.78385485e-01]
 [ 1.43494223e-01  2.61976394e-01 -4.45383120e-01  1.36060221e-01
  -5.00795526e-01 -5.87488150e-01  1.52897236e-01 -2.73099287e-01]
 [ 7.74962678e-01  2.27117371e-02  4.08516159e-01  2.84937362e-01
   3.19642679e-02 -2.35211438e-01 -1.59658351e-01  2.62688692e-01]
 [ 2.81806634e-01  8.24745878e-03 -6.92167142e-01  3.15915567e-01
   1.14149714e-02  5.01908806e-01 -1.69560615e-01  2.46628160e-01]
 [ 1.84643411e-01 -3.16806256e-01  2.48466337e-01 -3.46544961e-02
  -6.98268275e-01  4.67261587e-01  1.81630339e-01 -2.52393736e-01]
 [ 3.69278450e-01 -6.33614920e-01 -2.88917222e-01 -3.93333286e-01
   3.18917542e-01 -1.75016528e-01  1.52633610e-01 -2.61429000e-01]]
```

```
[[ 2.84043894e-03  3.14430147e-03 -2.46384735e-01 -1.58554932e-01
  -1.75245114e-01  1.76705635e-01  9.13738625e-01 -1.20261073e-01
   5.31056350e-02]
 [ 2.42121739e-03 -1.28321626e-03 -3.77000733e-01  1.76600215e-01
   6.89508147e-01  5.90273326e-01 -5.29344506e-02 -2.23230961e-03
  -4.91718843e-03]
 [ 2.20891154e-05  1.13495064e-05 -2.37217168e-03 -3.65660431e-03
   5.19584361e-03  7.52000216e-03  6.59901594e-02  7.85970681e-01
   6.14648552e-01]
 [ 1.09109680e-03  1.17416448e-03  6.61240940e-01  3.41172744e-01
   5.01749740e-01 -2.32499365e-01  3.72052358e-01 -4.25903576e-02
   1.77018784e-02]
 [ 1.63479471e-03 -2.90636016e-03  5.74279813e-01 -7.10405325e-02
  -3.14549320e-01  7.49883366e-01 -6.19835401e-02  4.58785876e-03
  -3.93375075e-03]
 [ 1.33908907e-05 -1.14077892e-05  5.80190908e-03 -2.15181510e-03
   2.88147957e-03 -5.73504703e-03 -1.22489909e-01 -6.04930528e-01
   7.86750146e-01]
 [-6.96053715e-01 -7.17961695e-01 -7.57487349e-05 -3.79564118e-03
   2.50334194e-03 -1.50223526e-04  4.37766998e-03 -5.55196293e-04
   2.36025045e-04]
 [-7.17950893e-01  6.96067270e-01  1.62813209e-03 -3.77529395e-03
   2.49754245e-03  3.65599795e-03 -6.00114914e-04  3.48250733e-05
  -4.91718843e-05]
 [-6.16016024e-03  2.29933343e-05 -1.73453679e-01  9.06741660e-01
  -3.78319687e-01  6.21986452e-02  2.52338778e-02 -2.47794677e-03
   7.62164204e-03]]
```

```
[[ 5.31056350e-02 -4.91718843e-03  6.14648552e-01]
 [ 1.77018784e-02 -3.93375075e-03  7.86750146e-01]
 [ 2.36025045e-04 -4.91718843e-05  7.62164204e-03]]
```

Report by: Vishwanath Datla

UID: 117705928