ENPM 673

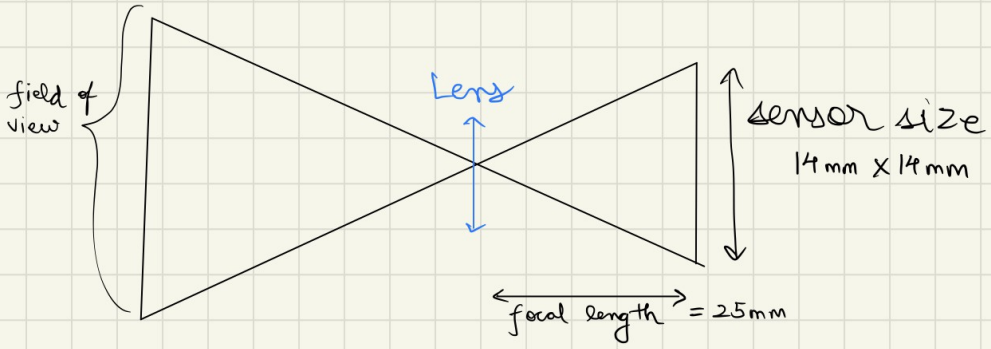
HOMEWORK 1

DIVYANSH AGRAWAL
UID: 117730692

Problem 1

Q1

(1)

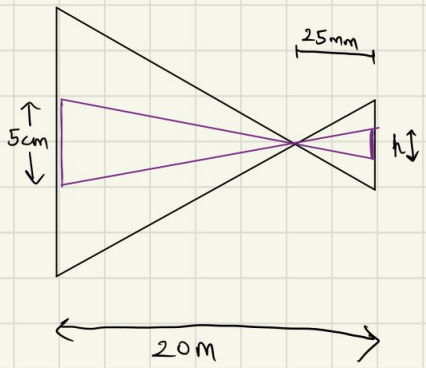


$$\text{angle of view} = 2\tan^{-1}\left(\frac{\text{sensor width}}{2 \times \text{focal length}}\right)$$

$$= 2\tan^{-1}\left(\frac{14}{2\times 25}\right) = 2\tan^{-1}\left(\frac{14}{50}\right) = 0.546 \text{ rad} = 31.28^{\circ}$$

since the camera sensor is square shaped, both vertical & horizontal FOV will be 0.546 rad.

(2)



h → height of object in image

using similar triangle properties

$$\frac{5\text{cm}}{20\text{m}} = \frac{h}{25\text{mm}}$$

$$\frac{5\times 10^{-2}}{20} = \frac{h}{25\text{ mm}}$$

$$\frac{125}{20} \times 10^{-2}\ mm = h$$

$$h = 0.0625\ mm$$

since object is square, area covered by object in image will be equal to h^2

$$\frac{\text{Camera resolution}}{\text{sensor area}} = \frac{\text{object's pixels in image (?)}}{\text{image area}}$$

$$\frac{5 \times 10^6}{14\,mm \times 14\,mm} = \frac{x}{(62.5 \times 10^{-3})^2\,mm^2}$$

$$\frac{5 \times (62.5)^2}{196} = x$$

$$x \approx 99.649$$

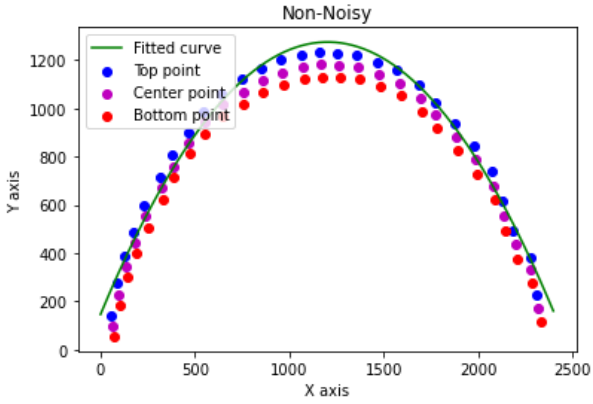
pixels are integers

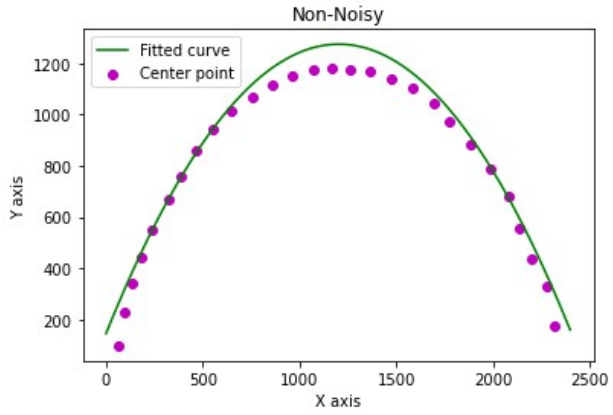
so $x = 100$ pixels

or object will occupy 100 pixels in image

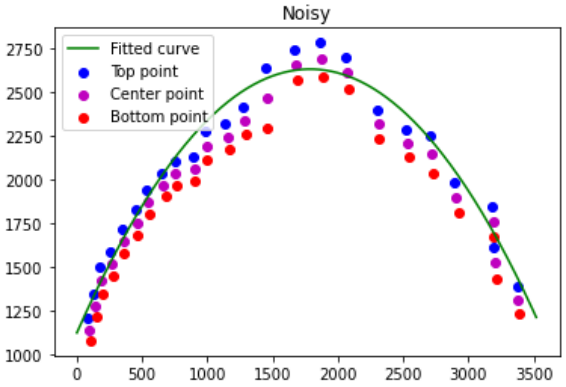
Problem 2: Extracting Pixels and Fitting Curve

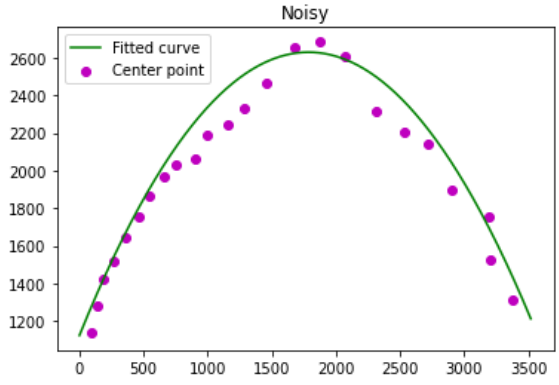
1. Read the video file frame by frame, using a while loop (operations in the further steps are given foe each frame and will be repeated because of the use of while loop)
2. Extract the red channel from each frame
3. Subtract 255 from each pixel to normalize it
4. Set a threshold value (this was done by checking the intensity values around the edge of the ball)
5. Using numpy.where, find the indices where intensity values are greater than threshold values
6. numpy.where stores the indices of required points in a 2D array, so extract the values at these points (for both top point and bottom point)
7. To find the center point, take the mean of top and bottom point
8. Use these points to fit a curve, a parabola here. Using pseudo inverse of the a matrix. This will give us the coefficients, a, b and, c of the parabolic equation, a*x^2 + b*x + c = 0. An image on the next page shows the linear least square math.





Video 1: No noise in the video





Video 2: Noise in the video which causes some points to go out of the curve

Q2 Least Square Method

parabola eqn $\Rightarrow ax^2 + bx + c = y$

$\therefore$ error $= E = \sum_n (y - ax^2 - bx - c)$

Let $X = [x^2 \quad x \quad 1]$

$$A = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$Y = y$

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

$$E = (Y - XA)(Y - XA)^T$$

$$E = YY^T - 2(XA^T)Y + (XA)^T XA$$

since we want to minimize the error, differentiate E w.r.t A $\left(\frac{\partial E}{\partial A}\right)$ & equate it to zero

$$\therefore \frac{\partial E}{\partial A} = -2X^T Y + 2X^T XA = 0$$

$$\Rightarrow A = (X^T X)^{-1} X^T Y$$

$(X^T X)^{-1} X^T = X^+$ (pseudo inverse of X)

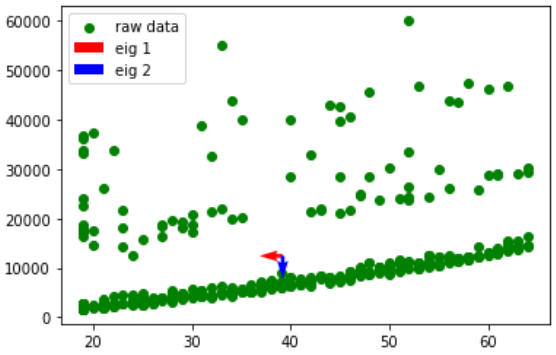
[here we assume that there is no error in the output so we minimize the offset]

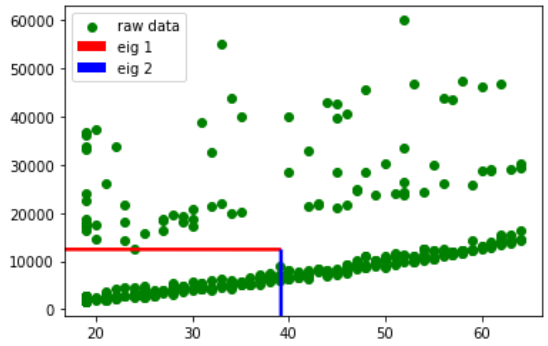
Problem 3: Reading data, finding co-variance matrix, plotting eigen vectors, using Least Square, Total Lease Square and RANSAC to fit curves

```
[[1.96841221e+02 5.23717982e+04]
 [5.23717982e+04 1.22218100e+08]]
```
Co-variance Matrix

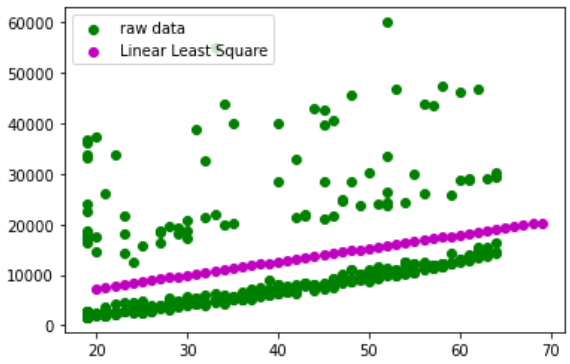
```
[1.74399299e+02 1.22218122e+08]
```
Eigen values of the co-variance matrix

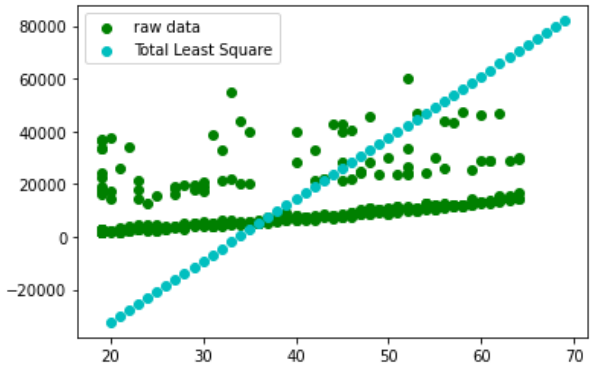
```
[[-9.99999908e-01 -4.28511555e-04]
 [ 4.28511555e-04 -9.99999908e-01]]
```
Eigen vectors of the co-variance matrix





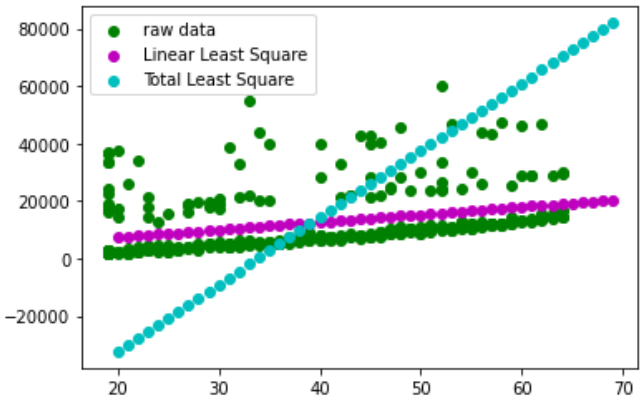
First image shows the Eigen Vectors plotted along with the data, origin in the mean of the data. Second image shows scaled eigen vectors that is eigen values time the respective eigen vector.



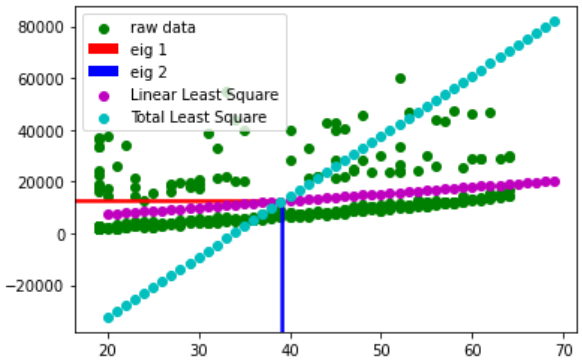


Linear Least Square (LLS) method

Total Lease Square (TLS) method



Comparing TLS and LLS. It shows that LLS does not consider the outliers that is we find the offset distance in the Y values. We consider that there is no error in Y in LLS. Whereas in TLS, we consider that there is error in Y values and we minimize the orthogonal distance of the data points from the fitted curve. Since the Y values are much more in range as compared to the X values, we are getting a line which is tilted towards Y and not only along X.



Plot showing everything that is, eigen vectors, TLS, LLS and raw data.

LLS

Advantages:

- Simplicity: It is very easy to explain and to understand
- Applicability: There are hardly any applications where least squares doesn't make sense
- Theoretical Underpinning: It is the maximum-likelihood solution and, if the Gauss-Markov conditions apply, the best linear unbiased estimator

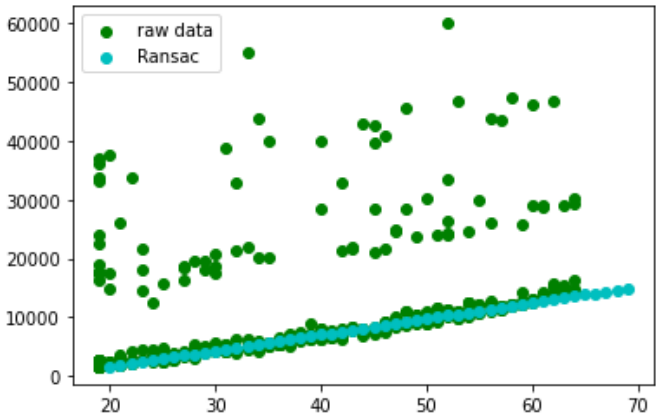
Disadvantages:

- Sensitivity to outliers
- Not rotation-invariant
- Fails completely for vertical lines
- Test statistics might be unreliable when the data is not normally distributed (but with many datapoints that problem gets mitigated)
- Tendency to overfit data (LASSO or Ridge Regression might be advantageous)

TLS

The advantage of this method is that it considers outliers too. Since we are minimizing the orthogonal distance of the data points from the curve, it can be used in the cases where there are a lot of outliers.

RANSAC



Since RANSAC doesn't consider outliers, the fitted curve is exactly in the direction of data. This was plotted considering the threshold value of distance equal to 30 units, that is, if a certain point is more than 30 units away (orthogonal distance) from thee curve then it will be considered as an outlier. Also, the accuracy here was considered more than 60%, that is if the number of inlier points are more than 60% of total points then the graph will be plotted.

Advantages

- simple and easy to implements
- successful in different contexts
- Often works well in practice

Disadvantages

- many parameters to tune
- trade-off accuracy vs time
- cannot be used if ratio of inliers to outliers in too small

(c) if there are a lot of outliers, then its better to use Total Lease Square method as the outliers are also considered while fitting the curve. But if the inliers are much more than outliers then RANSAC should be used as it gives a better fitted curve (compared to LLS and TLS).

Problem 4: (a)

Q4 (a)

A ⇒ given above

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

U → columns of this are orthogonal eigenvectors of AA^T

SVD → singular value decomposition

$$A = U \Sigma V^T \qquad [A \text{ is a } m \times n \text{ matrix}]$$

- U is an orthogonal matrix of size $m \times m$ $[U^T U = I]$
  columns of U are orthogonal eigen vectors of $A^T A$.
- Σ is a diagonal matrix of the eigen values of AA^T
- V is an orthogonal matrix of size $n \times n$.
  columns of V are orthogonal eigen vectors of $A^T A$

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

$$AA^T = U \Sigma V^T (U \Sigma V^T)^T$$

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

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

Columns of U are orthogonal eigen vectors of AA^T
Σ is the diagonal matrix of eigen values of AA^T
Columns of V are orthogonal eigen vectors of $A^T A$
$(V^T = V^{-1})$

$$\therefore AV = U \Sigma$$