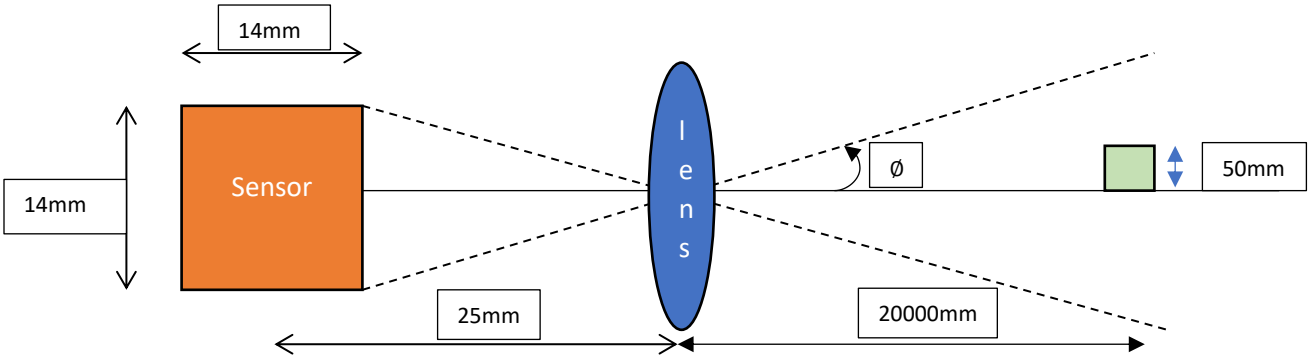
ENPM 673 - Perception for Autonomous Robots

Homework 1

Problem 1:



1. Field of view

Field of view is given by: $\varphi = \tan^{-1}\left(\frac{d}{2f}\right)$

Substituting our values, we get: $\varphi = \tan^{-1}\left(\frac{14}{2*25}\right)$

Therefore, $\varphi = 15.64^{\circ}$

The total FOV will be, $2 * \varphi = 31.28^{\circ}$

Since the sensor is a square the FOV will be same vertically and horizontally.

2. Number of pixels of the object

From similar triangles, we get: $\frac{\grave{y}}{y} = \frac{\grave{D}}{D}$ -> equation (1)

From thin lens formula, we get: $\frac{1}{D} + \frac{1}{\grave{D}} = \frac{1}{f}$ -> equation (2)

Substituting values in equation (2), we get: $\frac{1}{20000} + \frac{1}{\grave{D}} = \frac{1}{25}$

Solving for $\grave{D}$, we get: $\grave{D} = 25.03mm.$

Substituting this value in equation (1), we get: $\frac{\grave{y}}{50} = \frac{25.03}{20000}$

Solving for $\grave{y}$, we get: $\grave{y} = 0.063mm$

Total number of pixels are: 5MP

For $14 \times 14\ mm^2$, we have 5MP

For $0.063 \times 0.063mm^2$, we will have $number\ of\ pixels = \frac{5000000 \times 0.003969}{196}$

Therefore for $0.063 \times 0.063mm^2$, we will have $\sim = 100\ pixels$

Problem 2:

Check the code attached for the completed implementation. For this problem we filter the red channel and apply threshold to covert each frame into a black and white image. Now using the highest and the lowest point of the ball we use Standard least squares method to fit the curve.

The curves for both the videos are as follows:

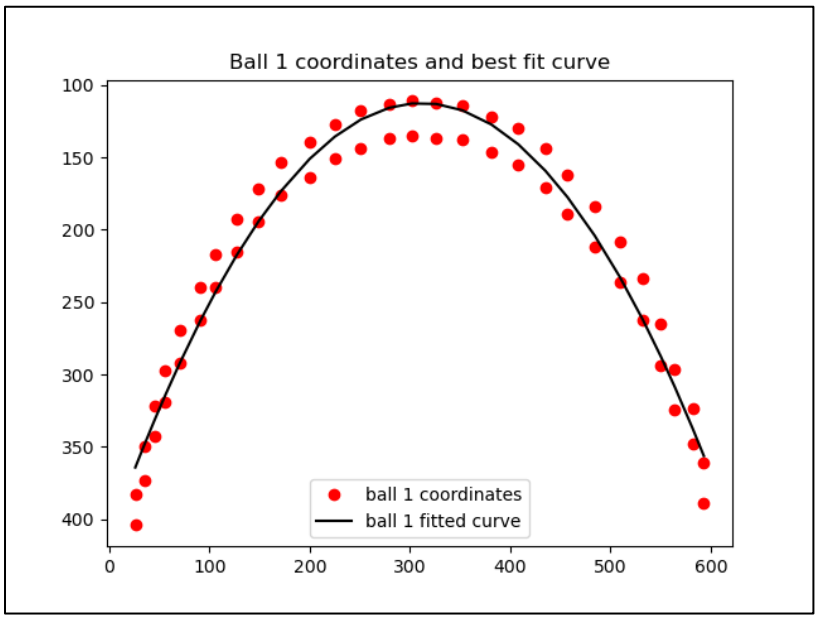


Figure 1: Trajectory of Normal ball

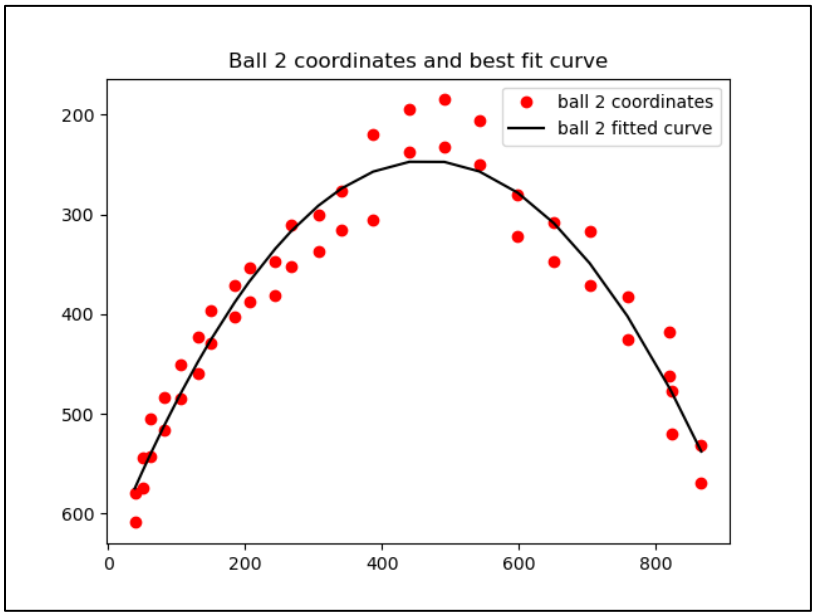


Figure 2: Trajectory of the noisy ball

Problem 3:

1. Refer the code attached for the complete implementation. After computing the covariance matrix and finding its eigen values and eigen vectors, the values we get are as follows:

```
The Covariance Matrix:
 [[0.09750556 0.01998987]
 [0.01998987 0.03594573]]

Eigen Values:
 [0.10342712 0.03002417]

Eigen Vectors:
 [0.95881596 0.28402809] [-0.28402809  0.95881596]
```

Figure 3: The Covariance Matrix, its eigen values and vectors

Plotting the eigen vectors we get:

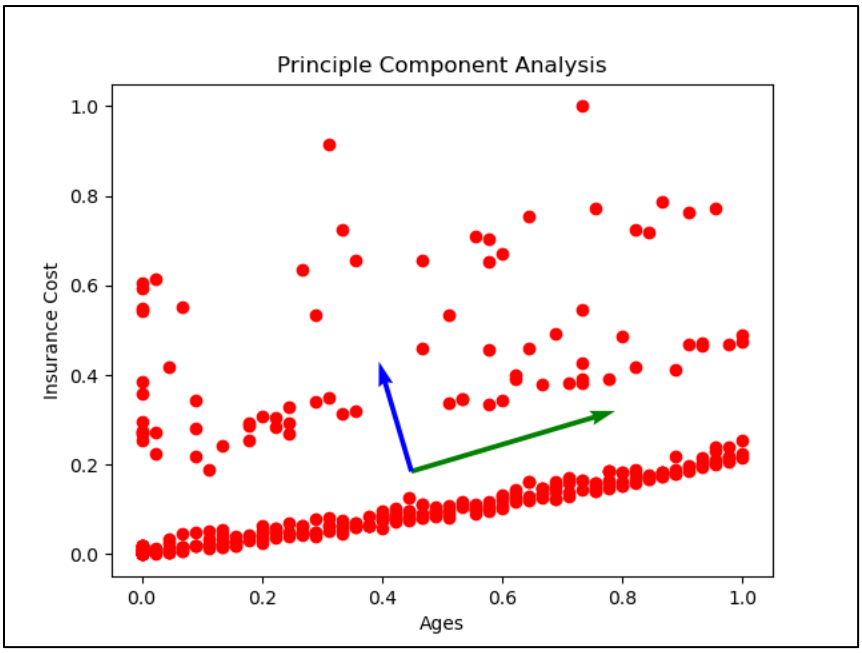


Figure 4: Plot of the eigen vectors

The green arrow shows the dominant direction of the data.

2. Line fitting using:

Refer the code attached for the complete implementation.

a. Linear least square method:

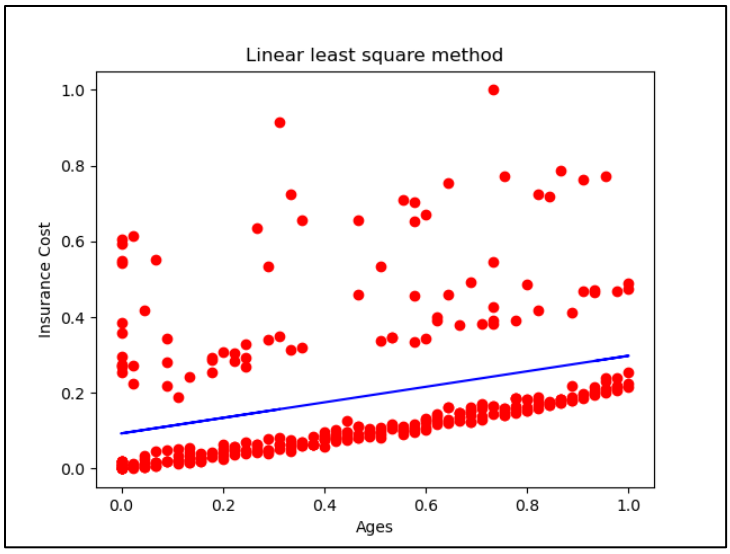


Figure 5: Line fitting using Linear least squares method

In this method we use the line equation $y = mx + c$, for finding the best line for the data. We do this by minimizing the slope m and offset c based on the distance between all the points and the line.

The biggest advantage is that it is very easy to compute the line.

The noticeable drawback for this problem, as seen the plot above, is that it tries to accommodate the outliers and hence there is an offset from the actual data.

b. Total least square method:

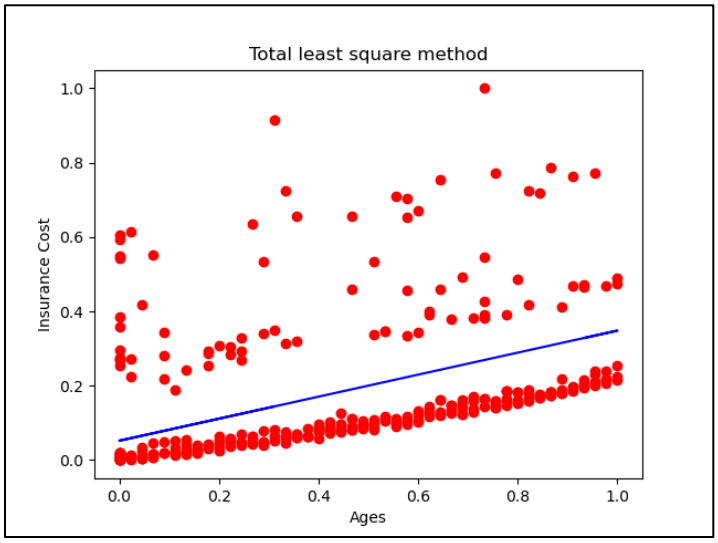


Figure 6: Line fitting using Total least squares method

Like linear least square method, the total least square method also tries to fit the line by using $ax + by = d$. Here $a, b\ and\ d$ are the perpendicular distances from the line.

In this method the line gets closer to the mass of the data and unlike linear least square method, this method accounts for vertical lines and rotation as well.

Since this method also tries to accommodate the outliers, the line has an offset from the mass of the data towards the outliers.

c. RANSAC

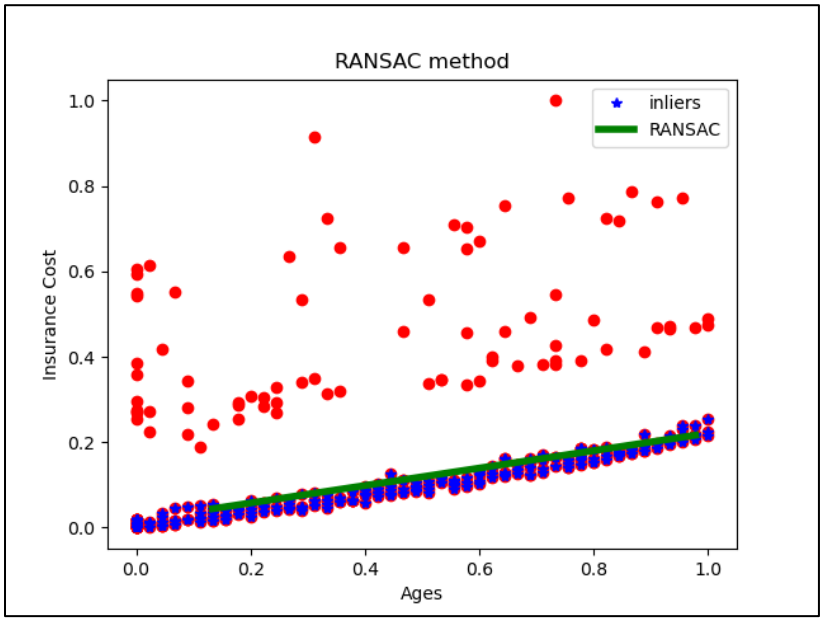


Figure 7: Line fitting using RANSAC method

By using this method, we find a line between two random points, compute the distance between this line and the points around the line based on a set threshold. Only the points that have a distance lesser than the threshold are the inliers. Repeat this process until you find a line that has the maximum number of inliers.

From the plot we can clearly see the ability of RANSAC to completely reject the outliers and provide the best fit for the data.

The drawback was to tune the threshold and wait for multiple iterations for the algorithm to find the best line.

3. For this problem we normalize the provided data and then find the dominant direction of the data using principal component analysis. Once we get an understanding of the direction of the data, we try three different methods (Linear least square method, Total least square method and RANSAC) to fit a line in the direction of the data. After plotting the data using Linear least square method and Total least square method, we can understand that although it is easier and faster to fit a line for the data, the outlier rejection is very minimal. Clearly RANSAC rejects the outliers better than the other two methods, therefore, RANSAC would be the better choice to reject outliers. The least square methods can be used when time is a constraint with a trade-off with the accuracy. When high precision is needed, it is better to use RANSAC.

Problem 4:

1. For an $m \times n$ matrix A, there exist a factorization, Singular Value Decomposition, as follows:

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

Where, U is an $m \times m$ matrix given by the orthogonal eigenvectors of AA^T, V is an $m \times m$ matrix given by the orthogonal eigenvectors of A^TA and $\Sigma = diagonal(\sigma_1, \dots\dots, \sigma_n)$ where $\sigma_1 = \sqrt{\lambda}$ and λ are the eigen values of AA^T or A^TA.

For our problem,

Finding U:

a. Compute AA^T

b. Find the eigen values and eigen vectors of AA^T

c. Combine the eigen vectors orthogonally to find the U matrix

Finding V:

a. Compute A^TA

b. Find the eigen values and eigen vectors of A^TA

c. Combine the eigen vectors orthogonally to find the V matrix

Finding Σ:

a. Method - 1:

i. Find the square root of the eigen values found using AA^T and form a diagonal matrix

ii. Add a zeros column at the end

b. Method – 2:

i. Find the square root of the eigen values found using A^TA and form a diagonal matrix

ii. Remove the last row from this matrix

Since we have a homogeneous linear equation, $Ax = 0$ the homography matrix is the last column of V^T. In our case V is 9×9, therefore the last column will be 9×1, we can resize this column to a 3×3 matrix to find our homography matrix H.

2. Refer the code attached for the complete implementation. The resulting H matrix from the code is as follows:

```
Homography matrix H is :

[[ 0.0062 -0.      -0.1735]
 [-0.9067 -0.3783 -0.0622]
 [ 0.0252 -0.0025  0.    ]]
```

Figure 8: Homography matrix