ENPM673 – Perception for Autonomous Robots

Homework 1

Pratik Acharya

117513615

Feb 14, 2022

Problem 1[20 Points]

0.1 Compute the Field of View of the camera in the horizontal and vertical direction.

Ans: Given: Resolution = 5 MP, Sensor = 14 mm x 14 mm, Focal Length = 25 mm

Field of view(FoV) of a camera is given by :

$$FoV = 2 * \arctan{(x/2f)} \tag{1}$$

where x is the lens dimension.

Hence, in vertical dimension the FoV is :

$$\begin{aligned} FoV(Vertical) &= 2 * \arctan{(14/2(25))} \\ &= 2 * \arctan{(0.28)} \\ &= 2 * 0.273 \\ &= 0.56\, Radians \end{aligned}$$

As the horizontal dimension of the sensor is same as vertical dimension,

$$FoV(Vertical) = FoV(Horizontal) = 0.56\, Radians$$

0.2 Assuming you are detecting a square shaped object with width 5cm, placed at a distance of 20 meters from the camera, compute the minimum number of pixels that the object will occupy in the image.

Ans: Given: Distance = 20 m = 20000 mm, Width = 5 cm = 50 mm

For a camera, the image dimension to focal length ratio is equal to the ratio of object dimension to distance. Hence the formula would be :

$$\frac{Image\ Dimension}{Focal\ Length} = \frac{Object\ Dimension}{Object\ Distance}$$

$$Image\ Dimension = \frac{Focal\ Length * Object\ Dimension}{Object\ Distance} \tag{2}$$

Substituting the given values,

$$\begin{aligned} Image\ Dimension &= \frac{25 * 50}{20000} \\ &= 0.0625\, mm \end{aligned}$$

The image area for a square image would be:

$$\begin{aligned} Image\ Area &= 0.0625 * 0.0625 \\ &= 0.00390625\, mm^2 \end{aligned}$$

Similarly, the ratio of Image Area to Sensor Area is equal to Image in Pixels to Sensor Area in Pixels.

The camera resolution is given as 5 MP, hence the area in pixels is given as,

$$Area = 5 \times 10^6$$

The sensor is given to be $14\, mm \times 14\, mm$, which is $196\, mm^2$.

Using this, the Image Area in pixels is calculated as

$$\begin{aligned} \frac{Image\ Area}{Sensor\ Area} &= \frac{Image\, in\, pixels}{Sensor\, width\, in\, pixels} \\ Image\, in\, pixels &= \frac{Image\ Area\ *\ Sensor\, width\, in\, pixels}{Sensor\ Area} \end{aligned}$$

Substituting the values,

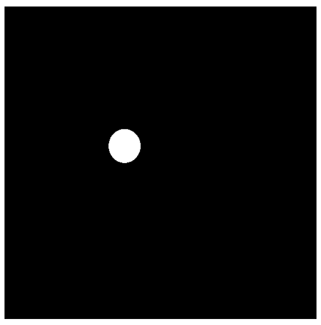
$$\begin{aligned} Image\, in\, pixels &= \frac{0.00390625 \times 5 \times 10^6}{196} \\ &= 99.65 \end{aligned}$$

Hence, the image will occupy **minimum 99 pixels**.

Problem 2[30 Points]

For this problem, first the video was read frame by frame and it was converted from its original BGR(Blue, Green, Red) colorspace to HSV(Hue, Saturation, Value) colorspace. Next, a red color filter was applied on the image to convert it into a binary black and white image, where the black part was the background and the white part was the ball[Fig 1(a)].

To find the position of the ball in the frame, an opencv library function called as 'moments' was used. Moments takes weighted average of the pixels in the image, which gives the centroid of any white blob in the frame. For each frame, the center of the ball was calculated and the coordinates of the center point were stored in a array[Fig 1(b)].


(a) Ball frame with mask

(b) Ball marked at the center

Figure 1: OpenCV Images

Same was repeated for the second video where noise was introduced in the video.

The calculated center-points were then plotted on a graph to show the trajectory of the ball. A function was written to calculated a curve that would fit the trajectory of the ball. Standard Least Square was used to fit a parabolic curve to the sample points. The x and y coordinates of the ball were made into a matrix as such :

$$A = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_i^2 & x_i & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{bmatrix}$$

The above matrices were solved as $Ax = B$, using eigen value decomposition. The results were coefficients, which were used to fit a curve using quadratic equation. The fitted curves were as follows:

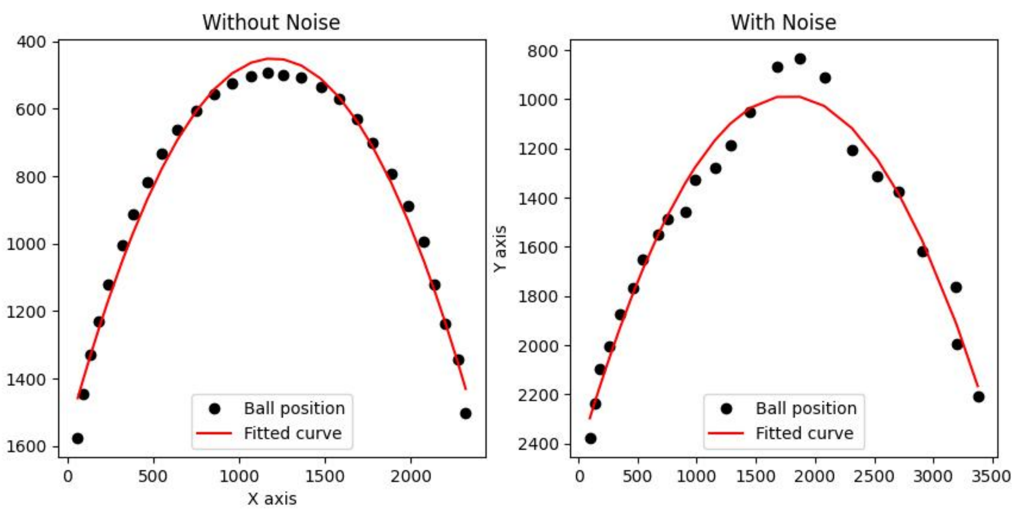


Figure 2: Fitted trajectory for ball video with and without noise

Problem 3[30 Points]

As the data was in a '.csv' file, pandas library was used to convert the data into arrays in python. Later the data was used into find the covariance matrix, and line fitting through Standard Least Square, Total Least Square and RANSAC methods.

Part 1

To calculate the covariance matrix, first the variance of the data was calculated by substractind individual data points from its mean. This variance was used to calculate the elements of the covariance matrix as such:

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

Eigen values and eigen vectors were calculated for the covariance matrix using the built in 'eig' function in the numpy library. The eigen vectors were then plotted on a graph along with the data to show the direction of variance in the data.

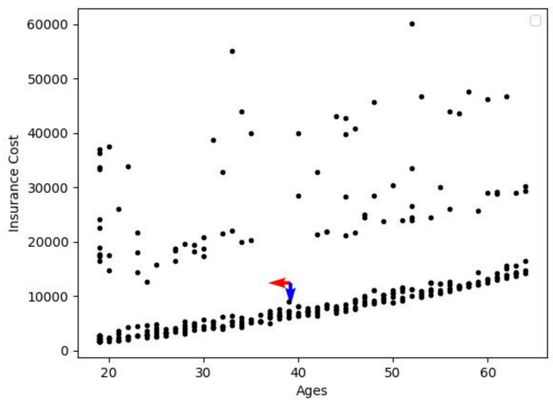


Figure 3: Eigen vectors of the covariance matrix

Part 2

Standard Least Square(SLS)

In Standard Least Square method, the data is fitted to a line by using the vertical distance between the line and the data points as a measurement criteria. This is the simplest form of line fitting and can be simplified to a matrix form. The problem is to minimize the error which is :

$$E(a,b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2$$

Which can be represented in matrix form as:

$$E = \parallel Y - XB \parallel^2$$

Differentiating the above with respect to B and simplifying will give:

$$B = (X^T X)^{-1}(X^T Y)$$

Which can be solved to get the equation for the line.

To fit a line to the data using Standard Least Square method, the data was first formatted into a matrix as such:

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_i & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \end{bmatrix}$$

These matrices were used to calculate the solution of equation $AX = B$, which was solved as follow, to get the x matrix.

$$X = (A^T A)^{-1}(A^T B)$$

The result was a 2×1 matrix that contained the coefficients for the line that was the best fit for the data. This line was then plotted on a graph along with the data points[fig.4].

Advantages

- It is faster to compute

Disadvantages

- It performs poorly when the data is distributed vertically
- It is prone to undesired results in the presence of outliers

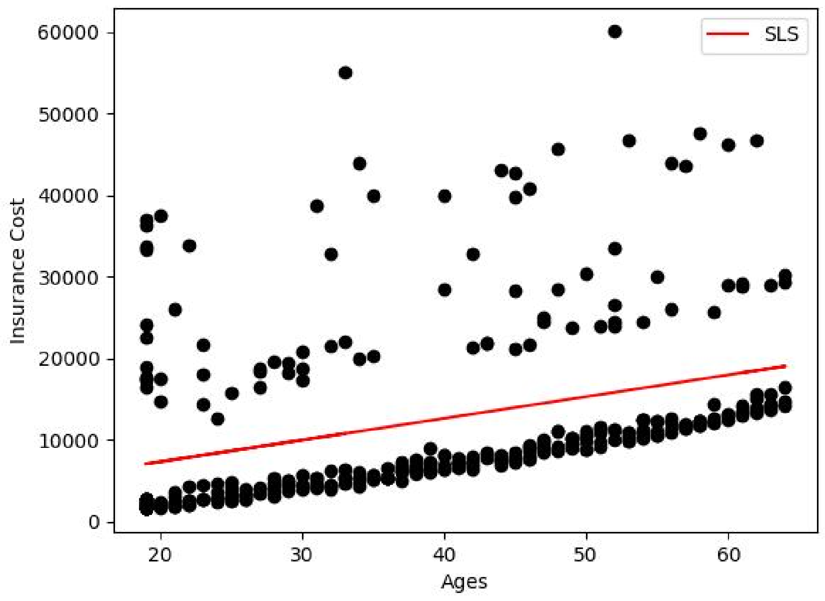


Figure 4: Fitted line using Standard Least Square

Total Least Square(TLS)

For total least square, the perpendicular distance between the data points and the line are used as a measurement metric. Hence, the minimization problem changes to:

$$E(a,b) = \sum_{n=1}^{N}(ax_n + by_i - d))^2$$

where,

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

Hence the equation can be written as:

$$E(a,b) = \sum_{n=1}^{N}(a(x_n - \bar{x}) + b(y_i - \bar{y}))^2$$

It can be written in matrix form as :

$$E = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_i - \bar{x} & y_1 - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2$$

It can be written as :

$$E = (UN)^T(UN)$$

where,

$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_i - \bar{x} & y_1 - \bar{y} \end{bmatrix}$$

Now taking derivative with respect to N, we get

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

Which can be solved to get the coefficients for the best fitting line.

In the code, the mean was calculated for the data and was subtracted from the data points and was made into a matrix. This matrix was then used as given in the above formula to calculate the coefficient of the line, by calculating eigen vector corresponding to the smallest eigen value for U^TU. The following line was found to be the best fit using TLS.

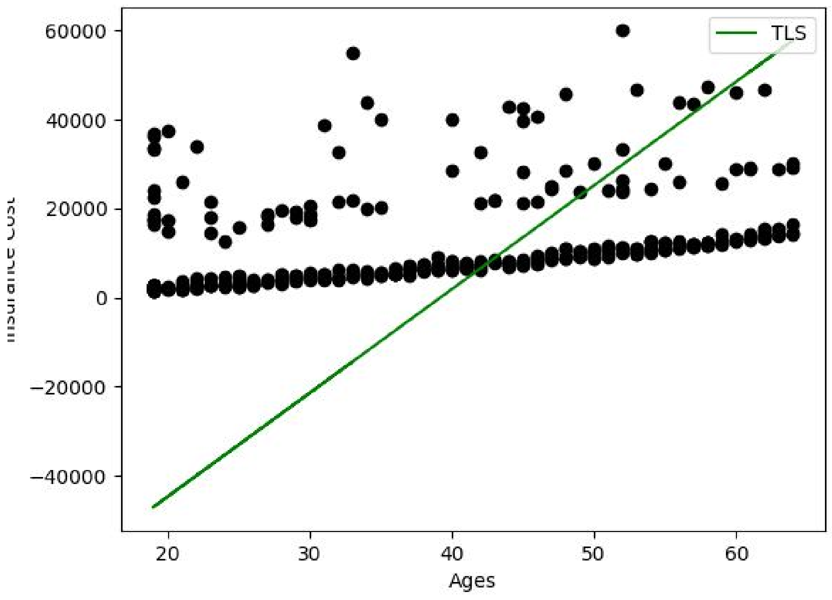


Figure 5: Fitted line using Total Least Square

Advantages

- It performs well even for data that is distributed vertically

Disadvantages

- It is prone to undesired results in the presence of outliers

RANSAC

To fit a line using RANSAC method, first two random points were chosen from the data set and an equation of line was generated from the same. The line was then used to find the points from the data set that lie between a set threshold, which was selected by observing the data set and identifying the inliers in the data set. Such points were counted and stored to be compared. This process was repeated over a sample size which was determined to ensure that there is a high probability of getting a line that passes through the inliers. At each iteration, the amount of points inside the threshold were compared and the line with the highest points in its threshold was chosen. This line was then selected and plotted as the best fit for the data using RANSAC [Fig.6].

Advantages

- It performs well even for data that is distributed vertically
- Outliers can be handled better with RANSAC than with SLS ans TLS

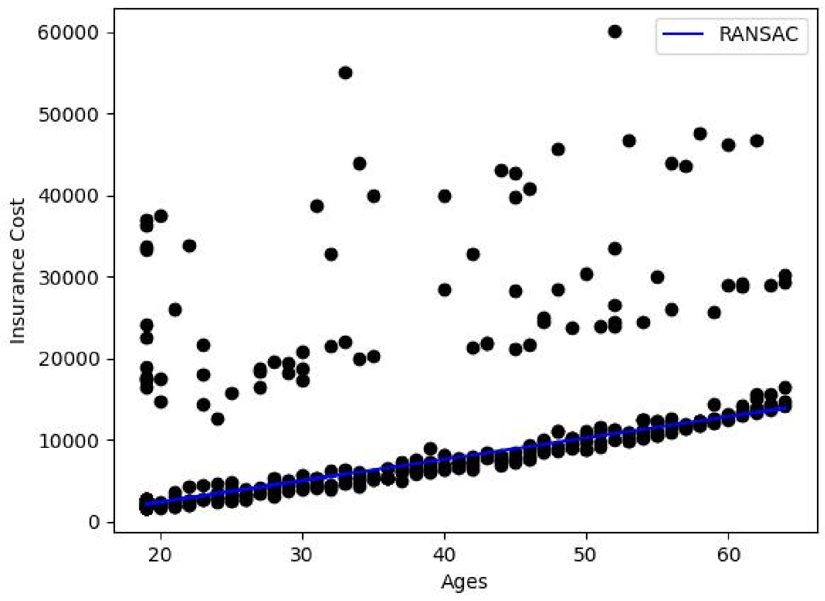


Figure 6: Fitted line using RANSAC

Disadvantages

- Data needs to be manually examined to decide threshold

Part 3

Comparing the results for the fitted line from SLS, TLS and RANSAC, we can observe than RANSAC provides a line than agrees with the trend in the data. While SLS and TLS are both prone to outliers, SLS performs better in fitting the line. SLS outputs a line that does not pass through any of the data points, hence it would be a bad fitting for predictions. TLS generated a line, which has its slope highly altered due to the outliers. This alteration renders it useless as it would not be able to follow the trend of the data. RANSAC follows the data according to the trend and provides the most accurate trend line, as it rejects the outliers from the dataset. Hence, RANSAC would be the best choice for this dataset.

Problem 4[20 Points]

Singular value decomposition is used to factorise a non-square matrix of shape $m \times n$ as follows:

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

Where U is a $m \times m$, V is a $n \times n$ and Σ is a $m \times n$ matrix.

Each of the matrices are formed as follows:
U is a matrix which has the orthogonal eigen vectors of the AA^T matrix as its columns.
V is a matrix which has the orthogonal eigen vectors of the A^TA matrix as its columns.
Σ is a diagonal matrix which has the square root of the eigen values of the U or V matrix as its diagonal elements.

Singular value decomposition can be used to solve a system of equation which has non-square matrix. The system of equations can be written as $Ax = 0$, and singular value decomposition can be used on A to find the solution to the equation. The eigen vector corresponding to the smallest eigen value is the answer to the system of equation.

To solve the problem using python, first the matrix was initialized. Using the above formulas, the U, V and Σ matrices were calculated. As for a singular value decomposition, the eigen values are arranged in a descending order, the eigen values and eigen vectors were sorted to follow a descending order. The Σ matrix was a square matrix as it was formulated from the eigen values, hence a column of 8×1 zero matrix is added to it to make it a 8×9 matrix.

Using the above method, the following result was calculated for the homography matrix.

$$\begin{bmatrix} 0.05310 & -0.04917 & 0.06146 \\ 0.01770 & -0.03933 & 0.07867 \\ 0.02360 & -0.04917 & 0.07621 \end{bmatrix}$$

Code

The code is submitted in the ELMS portal and is also available in the following GitHub repository

Perception-for-Autonomous-Robots.

References

`https://cmsc426.github.io/math-tutorial/#ransac`