ENPM 673 - Perception for Autonomous Robots

Assignment - 1

Kumara Ritvik Oruganti

117368963

okritvik@umd.edu

A. James Clark School of Engineering

University of Maryland

College Park - 20742

February 14, 2022

Contents

List of Figures

Chapter 1

Problem 1

1.1 Problem Statement

Assume that you have a camera with a resolution of 5MP where the camera sensor is square shaped with a width of 14mm. It is also given that the focal length of the camera is 25mm.

1. Compute the Field of View of the Camera in the horizontal and vertical direction.
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.

1.2 Solution

1.2.1 Part A

Given that the width of the sensor is 14mm and the focal length(f) is 25mm.

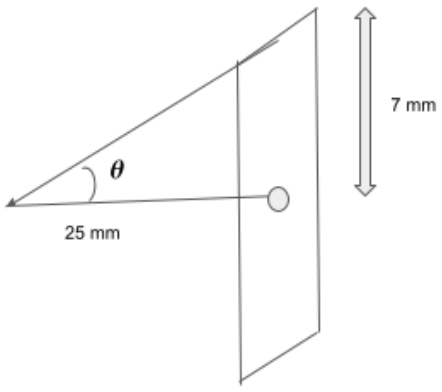

Figure 1.1: Ray Diagram to calculate Field of View

From the figure 1.1 the θ is the half of the field of view angle. Since the camera sensor is square shaped, we can say that the Vertical Field of View (VFV) is equal to the Horizontal Field of View (HFV).

$$HFV = VFV$$

By using the Pythagorean theorem,

$$\frac{VFV}{2} = \arctan(\frac{opposite}{adjacent}) \tag{1.1}$$

$$\implies VFV = 2 \times \arctan(\frac{7mm}{25mm}) \tag{1.2}$$

$$\implies VFV = 2 \times 15.642^o \tag{1.3}$$

$$\implies VFV = HFV = 31.284^o \tag{1.4}$$

1.2.2 Part B

Given that the square shaped object of width 5cm is placed at a distance of 20 meters from the lens. By using the thin lens formula [1],

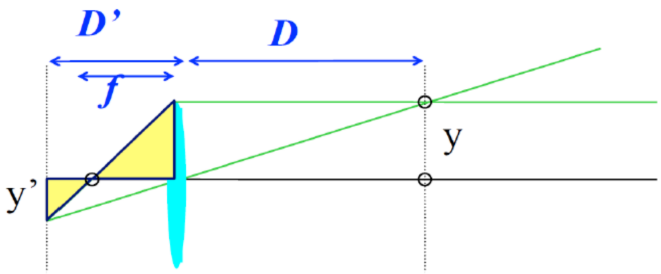

Figure 1.2: Figure depicting the similar triangles for Thin Lens Formula

From the above figure,

$$\frac{1}{D} + \frac{1}{D'} = \frac{1}{f} \tag{1.5}$$

$$\implies D' = \frac{Df}{D-f} = \frac{10000}{3995} cm \tag{1.6}$$

$$\frac{y'}{y} = \frac{D'}{D} \tag{1.7}$$

$$\implies y^{'} = y \times \frac{D^{'}}{D} = \frac{25}{3995}cm = \frac{250}{3995}mm \tag{1.8}$$

The sensor size is given as 14mm x 14mm. The resolution of the camera is $5{\times}10^6$ pixels.

Hence, 1mm^2 has $\frac{5\times10^6}{14\times14}$ pixels.

The area of the image formed is $\frac{250}{3995} \times \frac{250}{3995}mm^2$.

The number of pixels occupied by the object in the image $= \frac{250}{3995} \times \frac{250}{3995} \times \frac{5\times10^6}{14\times14}$.

$\implies Pixels = 99.9 \simeq 100$ pixels.

Chapter 2

Problem 2

2.1 Problem Statement

A ball is thrown against a white background and a camera sensor is used to track its trajectory. We have a near perfect sensor tracking the ball in video 1 and the second sensor is faulty and tracks the ball as shown in video 2. Clearly, there is no noise added to the first video whereas there is significant noise in the second video. Assuming that the trajectory of the ball follows the equation of a parabola:

1. Use Standard Least Squares to fit curves to the given videos in each case. You have to plot the data and your best fit curve for each case. Submit your code along with the instructions to run it.

2.2 Solution

2.2.1 Extracting Data

1. Open the video using OpenCV.
2. Read the each frame.
3. Extract the red channel.
4. Extract the pixel indices where there is low intensity (presence of ball).
5. Find the middle point index of the ball by taking the average of top and bottom pixels indices.
6. The indices found will be the points used to fit a parabola.

2.2.2 Least Square Method

Let us assume the curve to fit as,

$$f(x) = ax^2 + bx + c \tag{2.1}$$

$$y = ax^2 + bx + c \tag{2.2}$$

Let (x_i,y_i) be the points which are the points out of the curve. In the standard least squares method, we try to minimize the square of the distance between the actual position and the required position.

$$e_i = y_i - f(x_i) \tag{2.3}$$

$$\implies e_i = y_i - (a{x_i}^2 + bx_i + c) \tag{2.4}$$

$$\implies e_i = y_i - a{x_i}^2 - bx_i - c \tag{2.5}$$

$$E = \sum_{i=1}^{n} e_i^2 \tag{2.6}$$

$$E = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{2.7}$$

The parameters that we want to find are a,b,c. To find the constants, we will find the partial derivative of the equation (2.7) with respect to a,b and c and equate them to zero. We will get three equations in three unknowns which can be solved by using AX=B method.

$$\frac{\partial E}{\partial a} = 2 \times \sum_{i=1}^{n} (y_i - a{x_i}^2 - bx_i - c)(-x_i^2) \tag{2.8}$$

$$\implies \sum_{i=1}^{n} y_i x_i^2 = a \sum_{i=1}^{n} x_i^4 + b \sum_{i=1}^{n} x_i^3 + c \sum_{i=1}^{n} x_i^2 \tag{2.9}$$

$$\frac{\partial E}{\partial b} = 2 \times \sum_{i=1}^{n} (y_i - a{x_i}^2 - bx_i - c)(-x_i^) \tag{2.10}$$

$$\implies \sum_{i=1}^{n} y_i x_i = a \sum_{i=1}^{n} x_i^3 + b \sum_{i=1}^{n} x_i^2 + c \sum_{i=1}^{n} x_i \tag{2.11}$$

$$\frac{\partial E}{\partial c} = 2 \times \sum_{i=1}^{n} (y_i - a{x_i}^2 - bx_i - c)(-1) \tag{2.12}$$

$$\implies \sum_{i=1}^{n} y_i = a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i + cn \tag{2.13}$$

Writing (2.10) (2.12) (2.13) in AX=B form,

$$\text{A}= \begin{bmatrix} \sum_{i=1}^{n} x_i^4 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i & n \end{bmatrix} \text{X} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\text{B} = \begin{bmatrix} \sum_{i=1}^{n} y_i x_i^2 \\ \sum_{i=1}^{n} y_i x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

$$\implies X = A^{-1}B \tag{2.14}$$

2.2.3 Curve Fitting

Curve Fitting for Video 1

The indices of the frames start from the top left corner. But the plotting starts from the bottom left corner. Hence a transformation is applied to convert the coordinates of image to the coordinates of the curve. The result is shown in the below figures.

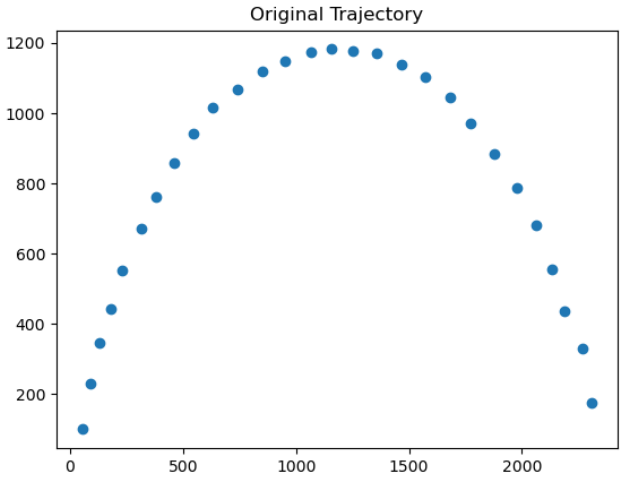

Figure 2.1: Trajectory of ball without noise

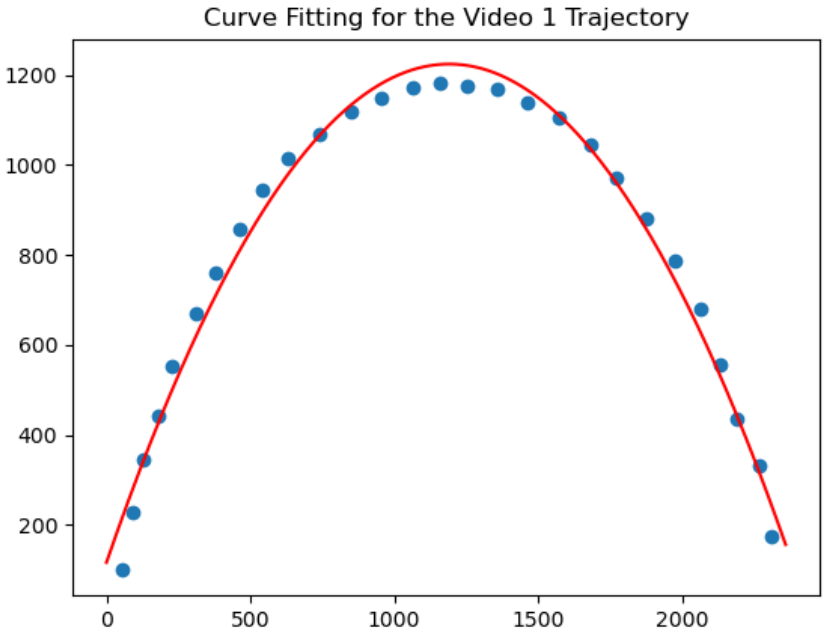

Figure 2.2: Trajectory curve fitting of ball without noise using least squares

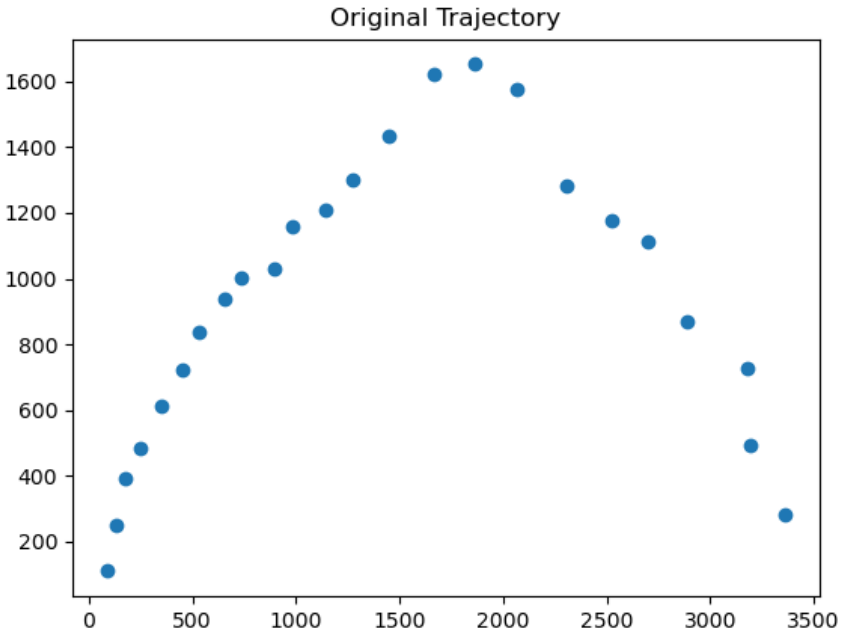

Figure 2.3: Trajectory of ball with noise

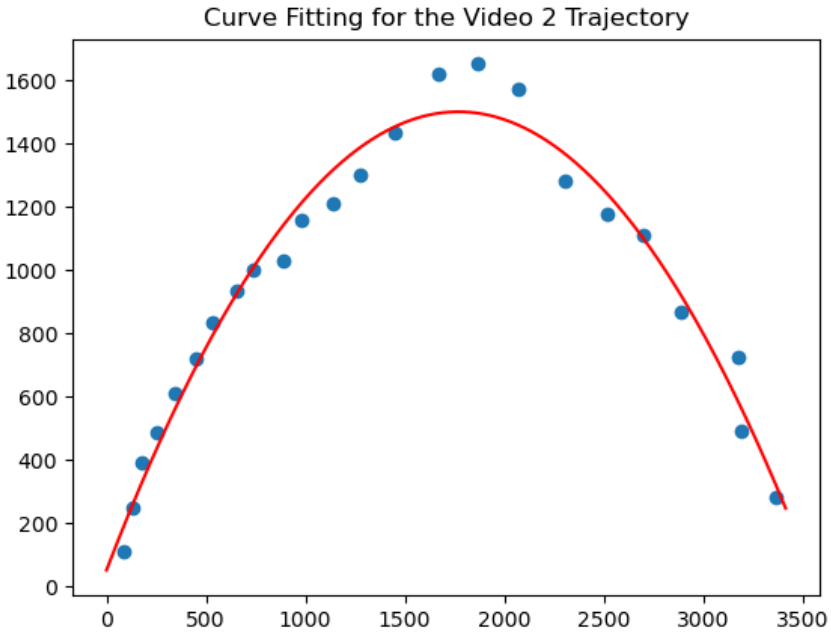

Figure 2.4: Trajectory curve of ball with noise using least squares

The code is provided in the github.

Chapter 3

Problem 3

3.1 Problem Statement

In the above problem, we used the least squares method to fit a curve. However, if the data is scattered, this might not be the best choice for curve fitting. In this problem, you are given data for health insurance costs based on the person's age. There are other fields as well, but you have to fit a line only for age and insurance cost data. The data is given in .csv file format and can be downloaded from here.

1. Compute the co-variance matrix (from scratch) and find its eigenvalues and eigenvectors. Plot the eigenvectors on the same graph as the data.
2. Fit a line to the data using linear least square method, total least square method and RANSAC. Plot the result for each method and explain drawbacks/advantages for each.
3. Briefly explain all the steps of your solution and discuss which would be a better choice of outlier rejection technique for each case.

3.2 Solution

3.2.1 Calculation of Co-variance Matrix

The generalized co-variance of two random variables X and Y is given by [2],

$$Cov(X,Y) = E[\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})] \tag{3.1}$$

(3.2)

where, E is the expectation.

$$\implies Cov(X,Y) = \frac{1}{n} \times \sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y}) \tag{3.3}$$

where, n is the number of samples. We know that the variance of a random variable X is given by,

$$\implies Var(X) = \frac{1}{n} \times \sum_{i=1}^{n} (X_i - \bar{X})^2 \tag{3.4}$$

The Co-variance Matrix is also called as the variance-covariance matrix [2]. The co-variance matrix of two random variables is given by

$$\text{C} = \begin{bmatrix} Var(X) & Cov(X,Y) \\ Cov(Y,X) & Var(Y) \end{bmatrix} \implies C = \frac{1}{n} \times \begin{bmatrix} \sum_{i=1}^{n}(X_i - \bar{X})^2 & \sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y}) \\ \sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X}) & \sum_{i=1}^{n}(Y_i - \bar{Y})^2 \end{bmatrix}$$

For the given CSV data, the parameters are age and charges. Since the units are different for the two random samples, the data will be normalized using the min-max normalization technique [7]. The min-max normalization of data X with each element of it as x_i is given by,

$$\hat{x_i} = \frac{x_i - min(X)}{max(X) - min(X)} \tag{3.5}$$

The eigen values of the co-variance matrix C represents the co-variances. The eigen vectors represent the linearly independent directions of variation in the data. Below is the figure that shows the directions of variation for the given data.

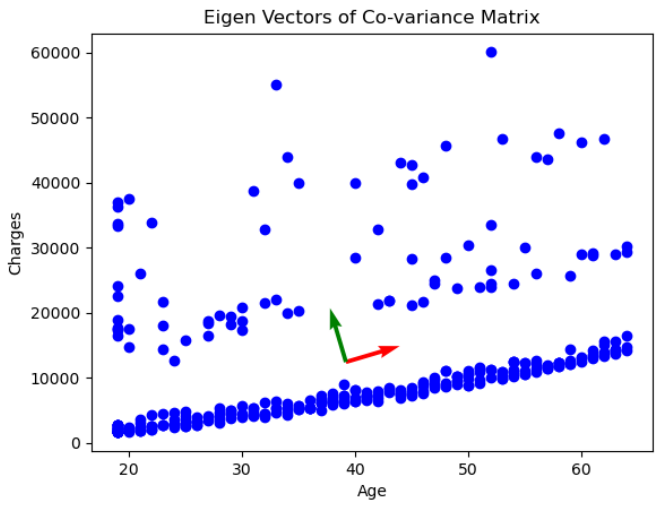

Figure 3.1: Eigen Vectors of the co-variance matrix for the given data

In the figure 3.1 the red arrow shows the principal component where there is more variance in the data. Since the plot normalized, the eigen vectors are plotted with scale = 10 for better visibility.

3.2.2 Least Squares Curve Fitting

Let us assume the curve to fit as,

$$f(x) = ax + b \tag{3.6}$$

$$y = ax + b \tag{3.7}$$

Let (x_i,y_i) be the points which are the points out of the curve. In the standard least squares method, we try to minimize the square of the distance between the actual position and the required position.

$$e_i = y_i - f(x_i) \tag{3.8}$$

$$\implies e_i = y_i - (ax + b) \tag{3.9}$$

$$\implies e_i = y_i - ax - b \tag{3.10}$$

$$E = \sum_{i=1}^{n} e_i^2 \tag{3.11}$$

$$E = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{3.12}$$

The parameters that we want to find are a and b. To find the constants, we will find the partial derivative of the equation (2.7) with respect to a, b and equate them to zero. We will get two equations in two unknowns which can be solved by using AX=B method [3].

$$\frac{\partial E}{\partial a} = 2 \times \sum_{i=1}^{n} (y_i - ax_i - b)(-x_i) \tag{3.13}$$

$$\implies \sum_{i=1}^{n} y_i x_i = a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i \tag{3.14}$$

$$\frac{\partial E}{\partial b} = 2 \times \sum_{i=1}^{n} (y_i - ax_i - b)(-1) \tag{3.15}$$

$$\implies \sum_{i=1}^{n} y_i = a\sum_{i=1}^{n} x_i + bn \tag{3.16}$$

Writing (3.15) (3.16) in AX=B form,

$$\text{A}= \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix} \text{X} = \begin{bmatrix} a \\ b \end{bmatrix} \text{B} = \begin{bmatrix} \sum_{i=1}^{n} y_i x_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

$$\implies X = A^{-1}B \tag{3.17}$$

Below is the figure that shows the Least Squares method to fit a straight line for the given data.

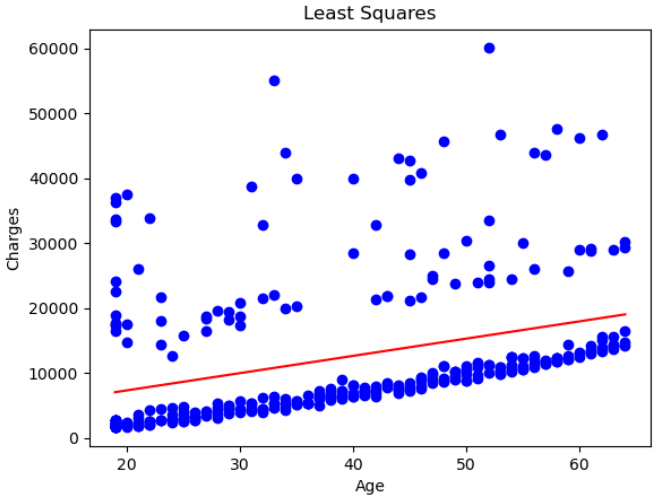

Figure 3.2: Least Squares Line fitting for the given data

3.2.3 Total Least Squares Curve Fitting

Let us assume the curve to fit as,

$$ax + by = d \tag{3.18}$$

$$where, a^2 + b^2 = 1 \tag{3.19}$$

Let (x_i,y_i) be the points which are the points out of the curve. In the total least squares method, we try to minimize the sum of the square of the perpendicular distance between the actual position and the required position.

$$d = a\bar{x} + b\bar{y} \tag{3.20}$$

$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2 \tag{3.21}$$

Taking the distance as in (3.20) will reduce the number of unknown variables to two (a,b). Now, the equation (3.21) can be partially differentiated with a, b to find the required equations. Let us consider X,Y as the random variables and x_i and y_i are the samples of the respective random variables. Substituting (3.20) in (3.21)

$$E = \sum_{i=1}^{n} (ax_i + by_i - a\bar{x} + b\bar{y})^2 \tag{3.22}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y})^2 \tag{3.23}$$

The equation (3.23) can be written in the matrix form as

$$E = (UN)^T (UN) \tag{3.24}$$

where,

$$\mathrm{U} = \begin{bmatrix} X - \bar{x} & Y - \bar{y} \end{bmatrix} \; \mathrm{N} = \begin{bmatrix} a \\ b \end{bmatrix}$$

The Shape of U matrix is (n,2). Differentiating the equation (3.24) with N and equating it to 0,

$$\frac{\partial E}{\partial N} = 2 \times (U^T U)N = 0 \tag{3.25}$$

The solution to the (U^TU)N=0 is the eigen vector of (U^TU)N=0 associated with the smallest eigen value of the (U^TU)N=0.

Below is the straight line curve fitting for the given data using Total Least Squares technique.

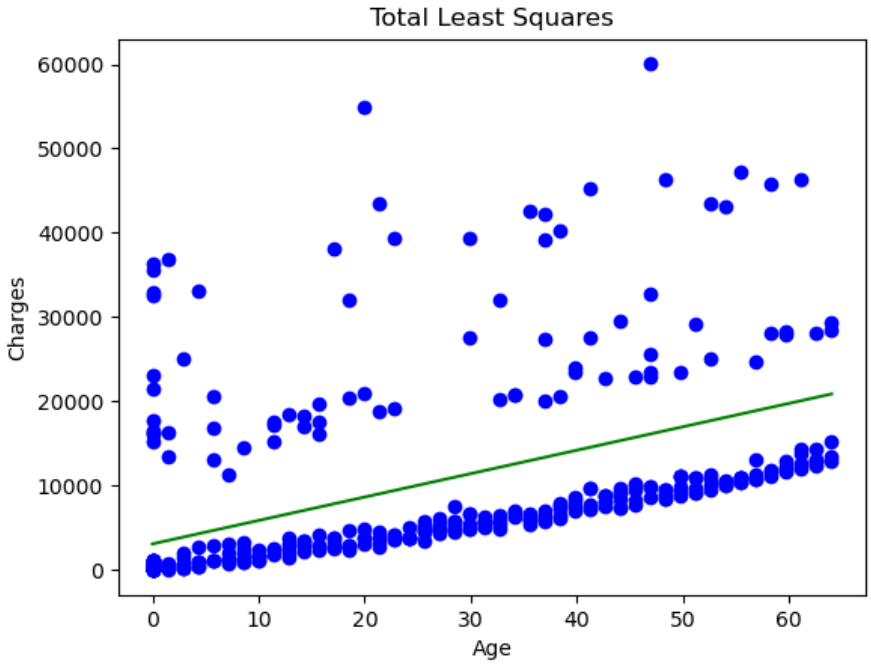

Figure 3.3: Total Least Squares Line fitting for the given data

3.2.4 RANSAC Curve Fitting

RANSAC is an acronym for Random Sample Consensus. It is a framework for model fitting in the presence of large number of outliers.

Steps for model fitting

1. Randomly select the minimal number of points required to fit the model. For example, a line fitting requires two points.
2. Hypothesize a model.
3. Compute the error function
4. Select points that are consistent with the model.
5. Repeat the model till the best fit is found for the hypothesized model. Otherwise, tune the required parameters and re-run the algorithm.

RANSAC requires four parameters.

- Number of points (s)

- Threshold distance (t)
- Probability of inlier (p) (desired accuracy)
- Number of iterations (N)

The number of iterations N can be updated adaptively using the equation

$$N = \frac{log(1-p)}{log(1-(1-e)^s)} \tag{3.26}$$

where, $e = 1 - \frac{number\ of\ inliers}{total\ number\ of\ points}$. The inliers are calculated using the Total Least Squares method as it involves calculating the perpendicular distance between the model and the points. The then found error in distance is compared with the threshold. If it satisfies the limit, then that point is considered as an inlier. Below is the figure that depicts the straight line curve fitting model for the given data using RANSAC.

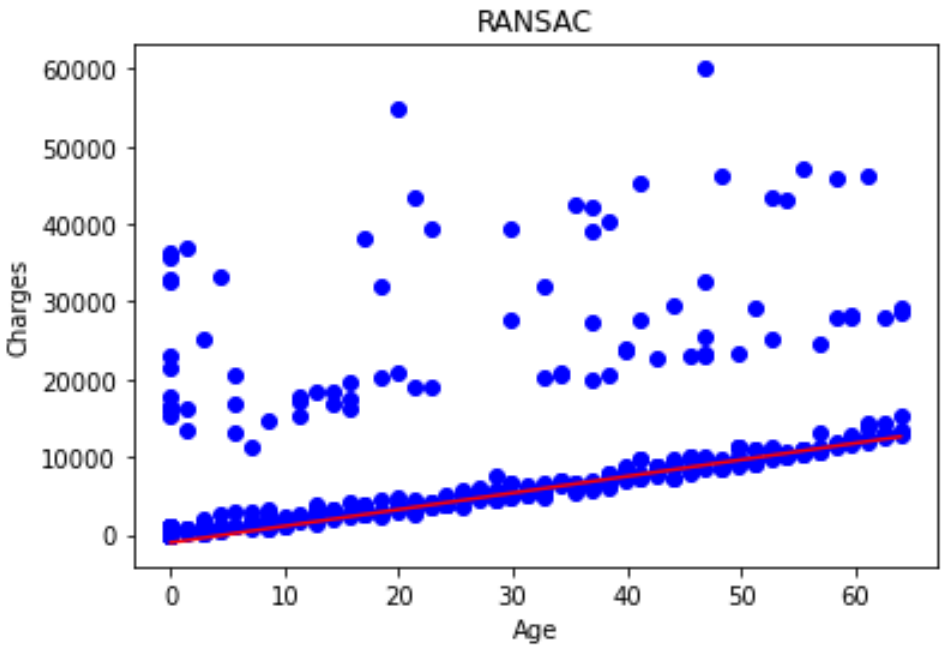

Figure 3.4: RANSAC Line fitting for the given data

The analysis of the curves is provided in the next subsection.

3.2.5 Analysis of LS, TLS and RANSAC

Least Squares Technique

From the figure 3.2, the red line is almost in the middle of the distribution. This is because one random variable is taken and minimizing the error in sum of squares of the distance. If we take another variable to fit the model, then the least squares method fail as it is not rotation invariant. The distance measurement is also with only one random variable which makes the technique not suitable for the vertical lines (in case if the random variable is X). Normalizing the data is not required as only one random variable is taken for calculating the distance.

Total Least Squares Technique

Total Least Squares is equivalent to calculating scaled co-variance matrix of the data. The data should be normalized as the perpendicular distance is calculated from all the random variables and the samples can be of different units. I encountered the problem when the data is not normalized. The min max normalization is used to normalize the data between 0 and 1. From the figure 3.3 the green line is slightly tilted towards the outliers (noise). The squared error in distance heavily penalizes outliers.

Random Sample Consensus

RANSAC gave the best output among the curve fitting techniques implemented. It greatly eliminated the outliers. The problem with the RANSAC is tuning the four parameters to get the perfect model fit for the data. RANSAC also requires multiple iterations refer (3.26) and sometimes it may not give output at all as the parameters tuned are high for the model. Updating of the iterations adaptively makes the algorithm more efficient than iterating for fixed number of time. RANSAC algorithm doesn't guarantee the same output as it is probabilistic in nature.

Chapter 4

Problem 4

4.1 Problem Statement

The concept of homography n Computer Vision is used to understand, explain and study visual perspective, and specifically, the difference in appearance of two plane objects viewed from different points of view. This concept will be taught in more detail in the coming lectures. For now, you just need to know that given 4 corresponding points on the two different planes, the homography between them is computed using the following system of equations Ax = 0, where A is given by

$$A = \begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1 * xp1 & y1 * xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1 * yp1 & y1 * yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2 * xp2 & y2 * xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2 * yp2 & y2 * yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3 * xp3 & y3 * xp3 & xp3 \\ 0 & 0 & 0 & -x3 & -y3 & -1 & x3 * yp3 & y3 * yp3 & yp3 \\ -x4 & -y4 & -1 & 0 & 0 & 0 & x4 * xp4 & y4 * xp4 & xp4 \\ 0 & 0 & 0 & -x4 & -y4 & -1 & x4 * yp4 & y4 * yp4 & yp4 \end{bmatrix} \tag{4.1}$$

$$x = \begin{bmatrix} H11 \\ H12 \\ H13 \\ H21 \\ H22 \\ H23 \\ H31 \\ H32 \\ H33 \end{bmatrix} \tag{4.2}$$

$$H = \begin{bmatrix} H11 & H12 & H13 \\ H21 & H22 & H21 \\ H31 & H32 & H33 \end{bmatrix} \tag{4.3}$$

Find the homography matrix H for the given point correspondences

	x	y	xp	yp
1	5	5	100	100
2	150	5	200	80
3	150	150	220	80
4	5	150	100	200

1. Show mathematically how you will compute the Singular Value Decomposition(SVD) for the matrix A.
2. Write python code to compute the SVD.

4.2 Solution

4.2.1 Singular Value Decomposition

SVD is a factorization of real or complex matrix. If a matrix A is given, then the singular value decomposition of the matrix A is given by

$$A = U\Sigma V^T \tag{4.4}$$

Where U is an mxm complex unitary matrix with columns as orthogonal eigen vectors of AA^T. Σ is an mxn rectangular diagonal matrix with diagonal elements as the square root of the eigen values (singular values) of AA^T. V is nxn complex unitary matrix with columns as orthogonal eigen vectors of A^TA [4] [5] [6]. Let us see what the decomposition and definitions of UΣV^T means.

$$AA^T = U\Sigma V^T(U\Sigma V^T)^T \tag{4.5}$$

$$= U\Sigma V^T V\Sigma^T U^T \qquad but \;\; [V^TV = I] \tag{4.6}$$

$$\implies AA^T = U\Sigma\Sigma^T U^T \tag{4.7}$$

The equation 4.10, is similar to the eigen decomposition. Here the $\Sigma\Sigma^T$ is the diagonal matrix with elements as the eigen values of the matrix AA^T. The columns of the U are orthogonal eigen vectors of AA^T. Similarly,

$$A^TA = (U\Sigma V^T)^T U\Sigma V^T \tag{4.8}$$

$$= V\Sigma^T U^T U\Sigma V^T \qquad but \;\; [U^TU = I] \tag{4.9}$$

$$\implies A^TA = V\Sigma^T\Sigma V^T \tag{4.10}$$

Since, V is orthogonal matrix, $V^T = V^{-1}$. Hence,

$$AV = U\Sigma \tag{4.11}$$

Important point to note is to arrange the columns in the descending order of the corresponding eigen values.

4.2.2 Computation of Homography matrix using SVD

$$\text{The matrix A} = \begin{bmatrix} -5 & -5 & -1 & 0 & 0 & 0 & 500 & 500 & 100 \\ 0 & 0 & 0 & -5 & -5 & -1 & 500 & 500 & 100 \\ -150 & -5 & -1 & 0 & 0 & 0 & 30000 & 1000 & 200 \\ 0 & 0 & 0 & -150 & -5 & -1 & 12000 & 400 & 80 \\ -150 & -150 & -1 & 0 & 0 & 0 & 33000 & 33000 & 220 \\ 0 & 0 & 0 & -150 & -150 & -1 & 12000 & 12000 & 80 \\ -5 & -150 & -1 & 0 & 0 & 0 & 500 & 15000 & 100 \\ 0 & 0 & 0 & -5 & -150 & -1 & 1000 & 30000 & 200 \end{bmatrix}$$

The matrix Σ =

```
S Matrix
[[6.02148954e+04 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 3.18245207e+04 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 2.60893068e+02 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.86219278e+02
  0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  1.45606434e+02 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 6.08809411e+01 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 3.89873638e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00 0.00000000e+00 8.10241308e-01
  0.00000000e+00]]
```

Figure 4.1: Σ matrix

The matrix U =

```
U Matrix
[[ 1.17519867e-02  3.44207228e-04 -5.15532162e-02 -4.66128587e-01
  -2.60345896e-01 -6.78428560e-02  1.08122912e-02 -8.41087769e-01]
 [ 1.17517760e-02  3.43641967e-04 -8.72103737e-02 -4.59351955e-01
  -2.49098952e-01 -8.85591890e-02  7.65455994e-01  3.54169473e-01]
 [ 3.58735699e-01  6.54942912e-01  1.34538659e-02 -4.65084492e-01
   1.70101644e-01  2.93617516e-01 -2.78385484e-01  1.82289868e-01]
 [ 1.43494223e-01  2.61976394e-01 -4.45383120e-01  1.36060221e-01
  -5.00795526e-01 -5.87488150e-01 -2.73099286e-01  1.52897235e-01]
 [ 7.74962678e-01  2.27117371e-02  4.08516159e-01  2.84937362e-01
   3.19642679e-02 -2.35211438e-01  2.62688692e-01 -1.59658350e-01]
 [ 2.81806634e-01  8.24745878e-03 -6.92167142e-01  3.15915567e-01
   1.14149714e-02  5.01908806e-01  2.46628160e-01 -1.69560615e-01]
 [ 1.84643411e-01 -3.16806256e-01  2.48466337e-01 -3.46544961e-02
  -6.98268275e-01  4.67261587e-01 -2.52393736e-01  1.81630339e-01]
 [ 3.69278450e-01 -6.33614920e-01 -2.88917222e-01 -3.93333286e-01
   3.18917542e-01 -1.75016528e-01 -2.61429000e-01  1.52633609e-01]]
```

Figure 4.2: U matrix

The matrix V^T =

```
Vt Matrix
[[ 2.84043894e-03  2.42121739e-03  2.20891154e-05  1.09109680e-03
   1.63479471e-03  1.33908907e-05 -6.96053715e-01 -7.17950893e-01
  -6.16016024e-03]
 [ 3.14430147e-03 -1.28321626e-03  1.13495064e-05  1.17416448e-03
  -2.90636016e-03 -1.14077892e-05 -7.17961695e-01  6.96067270e-01
   2.29933343e-05]
 [-2.46384735e-01 -3.77000733e-01 -2.37217168e-03  6.61240940e-01
   5.74279813e-01  5.80190908e-03 -7.57487349e-05  1.62813209e-03
  -1.73453679e-01]
 [-1.58554932e-01  1.76600215e-01 -3.65660431e-03  3.41172744e-01
  -7.10405325e-02 -2.15181510e-03 -3.79564118e-03 -3.77529395e-03
   9.06741660e-01]
 [-1.75245114e-01  6.89508147e-01  5.19584361e-03  5.01749740e-01
  -3.14549320e-01  2.88147957e-03  2.50334194e-03  2.49754245e-03
  -3.78319687e-01]
 [ 1.76705635e-01  5.90273326e-01  7.52000216e-03 -2.32499365e-01
   7.49883366e-01 -5.73504703e-03 -1.50223526e-04  3.65599795e-03
   6.21986452e-02]
 [ 9.13738625e-01 -5.29344506e-02  6.59901593e-02  3.72052358e-01
  -6.19835401e-02 -1.22489909e-01  4.37766998e-03 -6.00114914e-04
   2.52338778e-02]
 [-1.20261073e-01 -2.23230961e-03  7.85970681e-01 -4.25903576e-02
   4.58785877e-03 -6.04930528e-01 -5.55196293e-04  3.48250734e-05
  -2.47794677e-03]
 [ 5.31056349e-02 -4.91718842e-03  6.14648552e-01  1.77018783e-02
  -3.93375074e-03  7.86750146e-01  2.36025044e-04 -4.91718842e-05
   7.62164204e-03]]
```

Figure 4.3: V^T matrix

Note that the numpy.linalg library doesn't give eigen vectors that can recompute the matrix A back. This is because multiplying the eigen vector with a scalar gives the same

vector. Some of the columns can be different from the actual decomposition using the inbuilt library. In the U matrix 3^{rd} column and 8^{th} column has opposite signs when compared. In the V^T matrix, 5^{th} and 7^{th} rows have the same signs when compared with the matrix returned by the inbult function svd.

The homography matrix can be calulated by AX=0. The solution to the system of homogeneous equations is the eigen vector corresponding to the least eigen value of $A^T A$ if zero is not one of the eigen values.

$$\text{Homography Matrix H} = \begin{bmatrix} 5.31056349e-02 & -4.91718842e-03 & 6.14648552e-01 \\ 1.77018783e-02 & -3.93375074e-03 & 7.86750146e-01 \\ 2.36025044e-04 & -4.91718842e-05 & 7.62164204e-03 \end{bmatrix}$$

The codes for the assignment are available in the ***github link***.

Bibliography

[1] Lecture Notes, ENPM 673.

[2] https://stattrek.com/matrix-algebra/covariance-matrix.aspx

[3] https://www.youtube.com/watch?v=GPEE8JviDGU

[4] https://www.youtube.com/watch?v=gXbThCXjZFM

[5] https://www.youtube.com/watch?v=nbBvuuNVfco

[6] https://en.wikipedia.org/wiki/Singular_value_decomposition

[7] https://www.codecademy.com/article/normalization