ENPM673: Perception for Autonomous Robots
Homework 1

Vivek Sood – 117504279

Problem 1:

Assume that you have a camera with a resolution of 9MP where the camera sensor is square shaped with a width of 14mm. It is also given that the focal length of the camera is 15mm.

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

Ans: Field of View depends on the focal length of the lens. The size of Field of View is governed by the size of the camera retina.

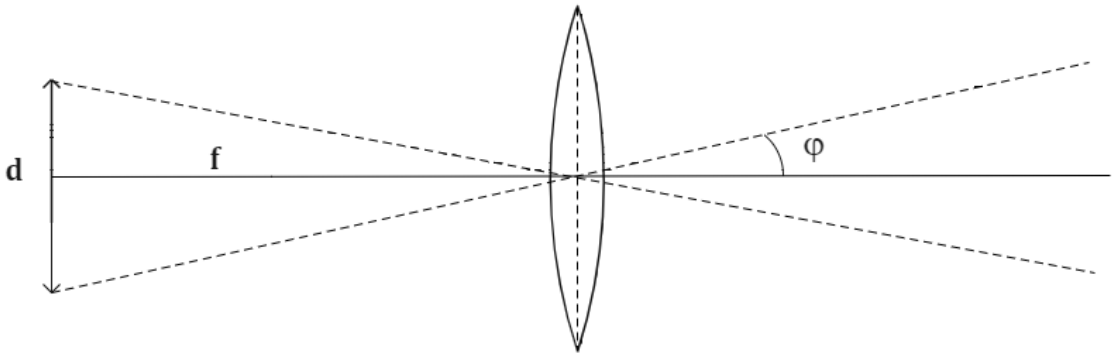

Figure 1: Field of View & Focal Length Diagram.

Mathematically field of view can be calculated using the equation where d is the sensor width and f is the focal length of the lens and the relationship is given below.

$$\varphi = \tan^{-1}(\frac{d}{2f})$$

In the given equation we substitute the value of d = 14mm and f = 15mm.

$$\varphi = \tan^{-1}(\frac{14}{30})$$

$$\varphi = \tan^{-1}(0.467)$$

$$\varphi = 25.03°$$

Hence the Field of View in horizontal and vertical direction is two times φ which is 50.06°.

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: In order to find the minimum number of pixels the given square object occupies we need to find the number of pixels per millimeter^2 of the field of view.

To find that first we need the field of view of the camera.

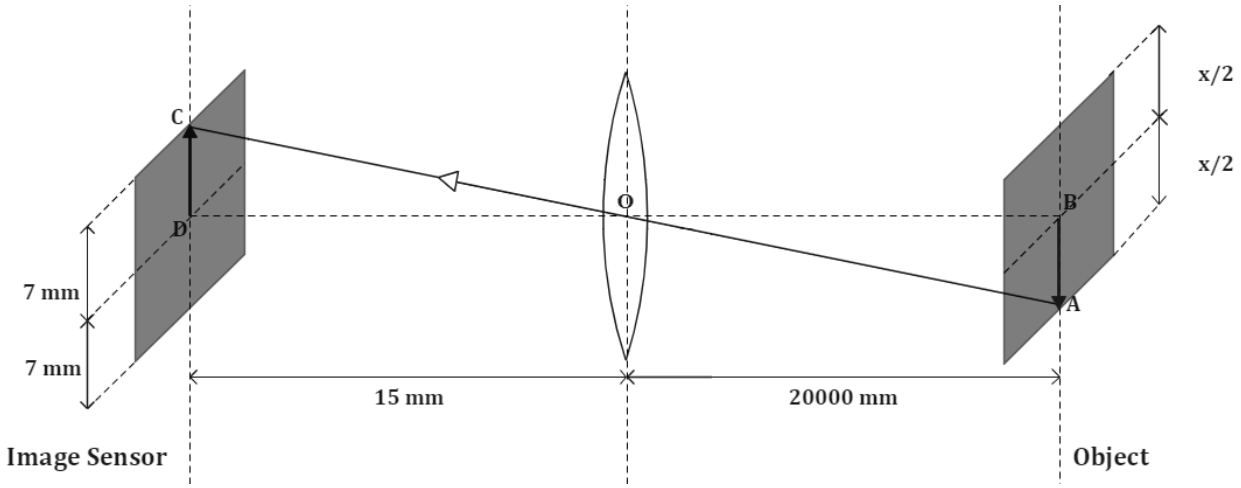

Figure 2: Ray diagram of the given problem.

We can see that triangle ΔCDO is similar to ΔABO, hence using properties of similar triangles we get.

$$\frac{AB}{BO} = \frac{CD}{DO}$$

$$\frac{\frac{x}{2}}{20000\ mm} = \frac{7\ mm}{15\ mm}$$

$$x = 18666.67\ mm$$

Now the field of view represents the area that camera can capture hence the FoV is:

$$x^2 = (18.667\ mm)^2 = 348{,}481{,}903.2289\ mm^2$$

Let the number of pixels per mm^2 *be* $'n'$. Now to obtain the number of pixels per mm^2 (let it be n) we divide the resolution of the image sensor with the FoV area.

$$n = \frac{9000000\ pixels}{348{,}481{,}903.2289\ mm^2}$$

$$n = \frac{9000000\ pixels}{348{,}481{,}903.2289\ mm^2}$$

$$n = 0.02583\ \frac{pixels}{mm^2}$$

Now the minimum numbers of pixel that the object occupies on the image sensor is the area of object times 'n'.

$$Area\ of\ object = 50 \times 50\ mm^2$$

$$Area\ of\ object = 2500\ mm^2$$

$$Minimum\ number\ of\ pixels = Area\ of\ object \times n$$

$$Minimum\ number\ of\ pixels = 2500\ mm^2 \times 0.02583\ \frac{pixels}{mm^2}$$

$$Minimum\ number\ of\ pixels = 64.465 = 65\ pixels$$

Hence the given object will occupy 65 pixels on the image sensor.

Interesting Problems/Observations:

- It was interesting to learn that ratio of the object is maintained on the sensor i.e., the object formed on the sensor will maintain the aspect ratio.
- The minimum number of pixels comes out to be 64.465 and according to general logic the approximation comes out to be 64 pixels but that would be wrong as we should take the least integer value because decimal value will take at least 1 pixel hence the answer would be 65 pixels.

Problem 2:

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 video1 and the second sensor is faulty and tracks the ball as shown in video2. Clearly, there is no noise added to the first video whereas there is significant noise in video 2. Assuming that the trajectory of the ball follows the equation of a parabola:

1. **Use Standard Least Squares, TLS and RANSAC methods 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. (Hint: Read the video frame by frame using OpenCV's inbuilt function. For each frame, filter the red channel for the ball and**

Ans: The implementation of all three algorithms i.e., Least Square Method (LS), Total Least Square Method (TLS) and Random Sample Consensus Method (RANSAC) for non-noisy and noisy input is shown in Figure 3, Figure 4, Figure 5, respectively.

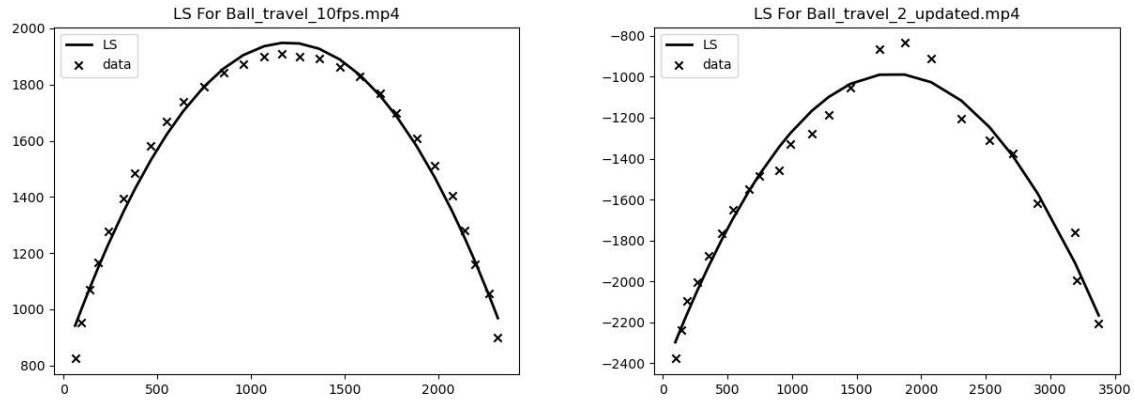

Figure 3: Least Square Method for Curve Fitting

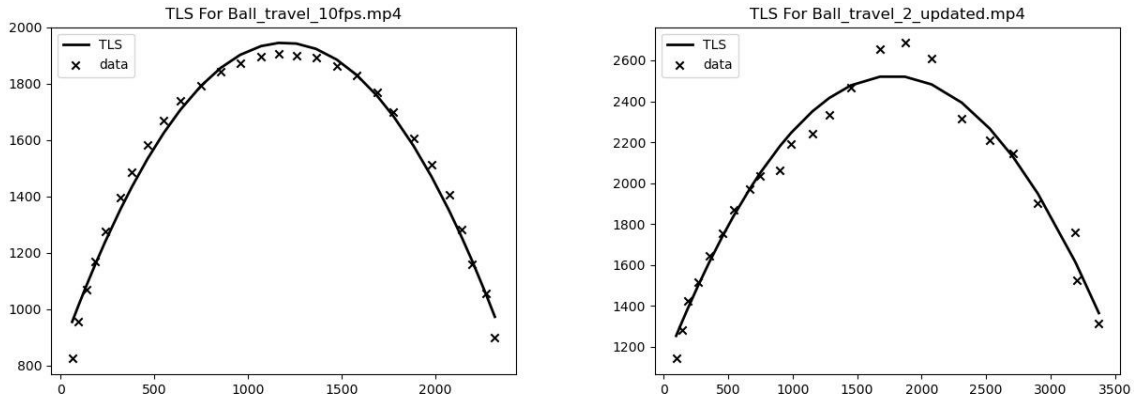

Figure 4: Total Least Square Method for Curve Fitting

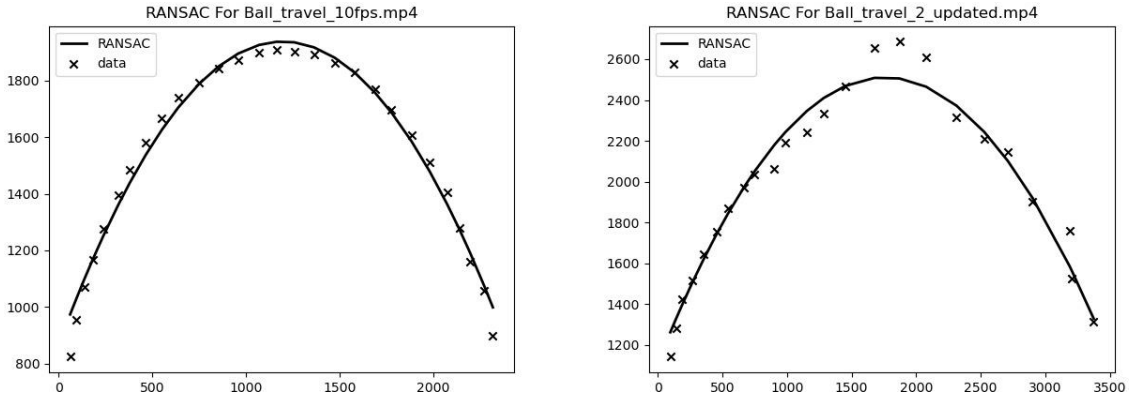

Figure 5: Random Consensus Method for Curve Fitting

2. **Briefly explain all the steps of your solution and discuss which would be a better choice of outlier rejection technique for each case.**

Ans: Before plotting it is essential to capture the dataset and in order to do that, we need to apply image processing concepts. Essentially, we need to capture the ball in the given input video.

To detect the ball in the video I first convert the video to grayscale and then separate the specific-colored object (red in our case) using **threshold** function in OpenCV.

```
1.  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
2.  ret, thresh = cv2.threshold(blurred, 120, 255, cv2.THRESH_BINARY_INV)
```

After that first we need to find the object's contour in the frame and then we need to approximate the position of the object and for that I use the **findCountours** method in OpenCV and then using moments method I find the center of the object. Before storing the values, I normalize the values of the y-coordinate by subtracting y- coordinate from the height of the frame so that the curves resemble the actual trajectory of the object.

```
1.  contours, h = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
2.  for c in contours:
3.      M = cv2.moments(c)
4.      cX = int(M["m10"] / M["m00"])
5.      cY = int(M["m01"] / M["m00"])
6.
7.      inputCorr.append((cX, thresh.shape[1]-cY))
```

Now we have received our input data set and we can use this data to generate a curve using the given curve fitting algorithms.

Implementation of curve fitting algorithms:

1. Least Square Method (LS): Least square method is a curve fitting algorithm that finds the best fitting curve for a set of given points by minimizing the sum of squares of the error generated for each point. The object in the given input follows projectile motion hence the motion can be fitted in a parabolic equation. Let the equation of the curve be.

$$y = ax^2 + bx + c$$

For a given set of points (x_1, y_1) , (x_2, y_2), …….(x_n, y_n) we can generate a system of linear equations.

$$Y = AX$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad A = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \quad X = \begin{bmatrix} {x_1}^2 & x_1 & 1 \\ {x_2}^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ {x_n}^2 & x_n & 1 \end{bmatrix}$$

Now error that we need to minimize is given by.

$$E = |Y - AX|^2$$

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

$$E = Y^T Y + (AX)^T (AX) - 2(AX)^T Y$$

Now to minimize error we differentiate the error and equate it to zero.

$$\frac{dE}{dx} = 2X^T XA - 2X^T Y = 0$$

Hence, the equation of parabola corresponding to the best-fitting model is given by.

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

2. Total Least Square Method (TLS): Total least squares method is a type of least squares data modeling technique in which observational errors are associated with both dependent and independent variables. As explained in least square method the given problem can be represented as.

$$Y = AX$$

The order of X is $m \times n$ and order of Y is $m \times k$. Now let E be the error in X and F be the error in Y. Then the equation above becomes

$$Y + F = A(X + E)$$

In the equation above we need to minimize matrices E and F. The above equation can be re-written as.

$$(Y + F)(X + E)\begin{bmatrix} A \\ -I \end{bmatrix} = 0$$

where I of the order k×k is the identity matrix. Now in order to minimize E and F matrices we need to perform single value decomposition. Using the Eckart–Young theorem, the approximation minimizing the norm of the error is such that matrices U and V are unchanged, while the smallest k singular values are replaced with zeroes.

$$[(Y + F)(X + E)] = [U_x \quad U_y]\begin{bmatrix} \sum X & 0 \\ 0 & 0_{\mathrm{k\times k}} \end{bmatrix}\begin{bmatrix} V_{XX} & V_{XY} \\ V_{YX} & V_{YY} \end{bmatrix}$$

So, by linearity,

$$[(Y + F)(X + E)] = -[U_x \quad U_y]\begin{bmatrix} 0_{\mathrm{n\times n}} & 0 \\ 0 & \sum Y \end{bmatrix}\begin{bmatrix} V_{XX} & V_{XY} \\ V_{YX} & V_{YY} \end{bmatrix}^T$$

After simplifying we get.

$$[(Y + F)(X + E)]\begin{bmatrix} V_{XY} \\ V_{YY} \end{bmatrix} = 0$$

Now if V_{YY} is nonsingular, which is not always the case, we can then right multiply both sides by $-{V_{YY}}^{-1}$ to bring the bottom block of the right matrix to the negative identity, giving.

$$[(Y + F)(X + E)]\begin{bmatrix} -V_{XY}{V_{YY}}^{-1} \\ -V_{YY}{V_{YY}}^{-1} \end{bmatrix} = (Y + F)(X + E)\begin{bmatrix} A \\ -I \end{bmatrix} = 0$$

Hence, the equation of parabola corresponding to the best-fitting model is given by.

$$A = -V_{XY}{V_{YY}}^{-1}$$

3. Random Sample Consensus Method (RANSAC): Random Sample Consensus (RANSAC) is a data modelling or a curve fitting technique that is very effective in curves where the dataset contains outliers.

My algorithm follows the following steps:

1. Select three points from the given dataset as the minimum number of equations required to find three variables is three.
2. Generate a model (coefficients in the equation of parabola) that fits the selected points.
3. For the remaining points calculate the error. If the error is greater than the user specified threshold, add it to outliers otherwise add it to list of inliers.
4. If number of inliers is lesser than the minimum inliers specified by the user reject the model and go back to step 1.
5. If number of inliers is greater than the minimum inliers specified by the user and the error using the current model is less than the previous error update the model.
6. Continue till the number of iterations specified by the user are completed.

Results

The number of iterations is set to 1000, the value of threshold is 150, and as the data set is very small the minimum number of Inliers is set to 70% of the number of inputs.

	LS	TLS	RANSAC
Video 1	34.0666	33.3854	32.5587
Video 2	66.3484	65.5709	61.1714

Table 1: Errors for different algorithms

As we can see RANSAC outperformed both LS and TLS in both scenarios but in case of video1 the results in case of RANSAC are very slightly better than LS and TLS but RANSAC is a computationally expensive algorithm hence for video 1 RANSAC is not the ideal algorithm to use. The results of least square and total least square are quite similar as the error in x-direction is negligible in both scenarios.

When it comes to video2 the results are better for RANSAC when compared to other two algorithms. This can be visually (as curves of both LS and TLS fitted curves are shifted up due to noise in Figures 3, 4, 5) as well as qualitatively (as the error is significantly low in case of RANSAC). RANSAC performs better than LS and TLS due to its superior outlier detection and rejection.

Interesting Problems/Observations:

- When we try to scan frame by frame my code gave an error just as the input video frames added hence, I added the analysing logic in *except* block in python.
- The curve generated after simply plotting the received input points was an inverted parabola hence, I normalized the value in y direction by subtracting it from the height of the frame.
- Another challenge I was to compare the performance of the algorithms quantitatively. I overcame that by creating a mean Error function that calculates the mean error of the model with all the points.

Problem 3:

The concept of homography in 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:

$$A = \begin{bmatrix} -x1 & -y1 & -1 & 0 & 0 & 0 & x1*xp1 & y1*xp1 & xp1 \\ 0 & 0 & 0 & -x1 & -y1 & -1 & x1*yp1 & y1*yp1 & yp1 \\ -x2 & -y2 & -1 & 0 & 0 & 0 & x2*xp2 & y2*xp2 & xp2 \\ 0 & 0 & 0 & -x2 & -y2 & -1 & x2*yp2 & y2*yp2 & yp2 \\ -x3 & -y3 & -1 & 0 & 0 & 0 & x3*xp3 & y3*xp3 & xp3 \\ 0 & 0 & 0 & -x3 & -y3 & -1 & x3*yp3 & y3*yp3 & yp3 \\ -x4 & -y4 & -1 & 0 & 0 & 0 & x4*xp4 & y4*xp4 & xp4 \\ 0 & 0 & 0 & -x4 & -y4 & -1 & x4*yp4 & y4*yp4 & yp4 \end{bmatrix}, x = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix}$$

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

Find the homography matrix:

$$H = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix}$$

Ans. The equation presented is of the form $AX = 0$ which is a homogeneous linear equation. As we know the best way to solve a homogeneous linear equation is by using Singular Value Decomposition on the Matrix A. The SVD method factors the matrix into two orthogonal matrix (U and V) and a diagonal matrix or sigma matrix (S) such that.

$$A = USV^T$$

From the given problem the A matrix after substituting the specified values is.

$$A = \begin{bmatrix} -5 & -5 & -1 & 0 & 0 & 0 & 500 & 500 & 100 \\ 0 & 0 & 0 & -5 & -5 & -1 & 500 & 500 & 100 \\ -150 & -5 & -1 & 0 & 0 & 0 & 30000 & 1000 & 200 \\ 0 & 0 & 0 & -150 & -5 & -1 & 12000 & 400 & 800 \\ -150 & -150 & -1 & 0 & 0 & 0 & 33000 & 33000 & 220 \\ 0 & 0 & 0 & -150 & -150 & -1 & 12000 & 12000 & 80 \\ -5 & -150 & -1 & 0 & 0 & 0 & 500 & 15000 & 100 \\ 0 & 0 & 0 & -5 & -150 & -1 & 1000 & 30000 & 200 \end{bmatrix}$$

After applying SVD to A we get the following matrices

The first matrix is U, an orthogonal matrix with values from eigen vector of AA^T

$$U = \begin{bmatrix} 0.0118 & 0.0003 & -0.0516 & -0.4661 & -0.2603 & -0.0678 & 0.0108 & -0.8411 \\ 0.0118 & 0.0003 & -0.0872 & -0.4594 & -0.2491 & -0.0886 & 0.7655 & 0.3542 \\ 0.3587 & 0.6549 & 0.0135 & -0.4651 & 0.1701 & 0.2936 & -0.2784 & 0.1823 \\ 0.1435 & 0.2620 & -0.4454 & 0.1361 & -0.5008 & -0.5875 & -0.2731 & 0.1529 \\ 0.7750 & 0.0227 & 0.4085 & 0.2849 & 0.0320 & -0.2352 & 0.2627 & -0.1597 \\ 0.2818 & 0.0082 & 0.6922 & 0.3159 & 0.0114 & 0.5019 & 0.2466 & -0.1696 \\ 0.1846 & -0.3168 & 0.2485 & -0.0347 & -0.6983 & 0.4673 & -0.2524 & 0.1816 \\ 0.3693 & -0.6336 & -0.2889 & -0.3933 & 0.3189 & -0.1750 & -0.2614 & 0.1526 \end{bmatrix}$$

The second matrix is V, an orthogonal matrix with values from eigen vector of $A^T A$

$$V = \begin{bmatrix} 0.0028 & 0.0031 & -0.2464 & -0.1586 & -0.1752 & 0.1767 & 0.9137 & -0.1203 & 0.0531 \\ 0.0024 & -0.0013 & -0.3770 & 0.1766 & 0.6895 & 0.5903 & -0.0529 & -0.0022 & 0.0049 \\ 0.0000 & 0.0000 & -0.0024 & -0.0037 & 0.0052 & 0.0075 & 0.0660 & 0.7860 & 0.6146 \\ 0.0011 & 0.0012 & 0.6612 & 0.3412 & 0.5017 & -0.2325 & 0.3721 & 0.0426 & 0.0177 \\ 0.0016 & -0.0029 & 0.5743 & -0.0710 & -0.3145 & 0.7499 & -0.0620 & 0.0046 & -0.0039 \\ 0.0000 & -0.0000 & 0.0058 & -0.0022 & 0.0029 & 0.0057 & -0.1225 & -0.6049 & 0.7868 \\ -0.6961 & -0.7180 & -0.0001 & -0.0038 & 0.0025 & -0.0002 & 0.0044 & -0.0006 & 0.0002 \\ -0.7180 & 0.6961 & 0.0016 & -0.0038 & 0.0025 & 0.0037 & -0.0006 & 0.0000 & 0.0000 \\ -0.0062 & 0.0000 & -0.1735 & 0.9067 & -0.3783 & 0.0622 & 0.0252 & -0.0025 & 0.0076 \end{bmatrix}$$

Eigen values of AA^T and A^TA are equal. Here, we get 8 eigen values, that is r = 8. Then the singular values of A, σ_i is given by.

$$\sigma_i = \sqrt{\lambda_i}$$

The third matrix is the S matrix, the first eight elements are $\sigma_1, \sigma_2 \ldots\ldots, \sigma_8$ and then a column of zeroes is appended.

$$S = \begin{bmatrix} 60214.895 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 31824.520 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 260.8931 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 186.2193 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 145.6064 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 60.8809 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 3.8987 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.8102 & 0.0000 \end{bmatrix}$$

As we can see the given question the number of unknowns i.e., nine is greater than the number of linear equations hence we cannot find the exact solution, but we can approximate by finding the solution that minimizes the sum of squared errors.

The least-squares solution of the given problem is given by the column of V, which is associated with the smallest eigen value of ATA.

The eigen values of ATA are:

[3625833626.6986, 1012800114.9102, 68065.1927,34677.6193,21201.2335,3706.4890,15.2001,0.6565,0.0000]

The smallest eigen value is 0. So, the solution is given by the column number associated with the least eigen value.

$$x = \begin{bmatrix} 0.0531 \\ -0.0049 \\ 0.6146 \\ 0.0177 \\ -0.0039 \\ 0.7868 \\ 0.0002 \\ -0.0000 \\ 0.0076 \end{bmatrix}$$

Reshaping the solution matrix, we get the homography matrix.

$$H = \begin{bmatrix} 0.0531 & -0.0049 & 0.6146 \\ 0.0177 & -0.0039 & 0.7868 \\ 0.0002 & -0.0000 & 0.0076 \end{bmatrix}$$

Interesting Problems/Observations:

- The challenge was to basically get the solution for matrix x when the number of unknowns were nine and the number of equations was only eight. I overcame that challenge by calculating the least squares to get the most optimized solution.
- An interesting observation was that the products of SVD when recombined do not give the original matrix. I realised this was because the matrix was not square.