Homework 1

ENPM-673

Perception for Autonomous Robots

Harika Pendli

M.Eng. Robotics

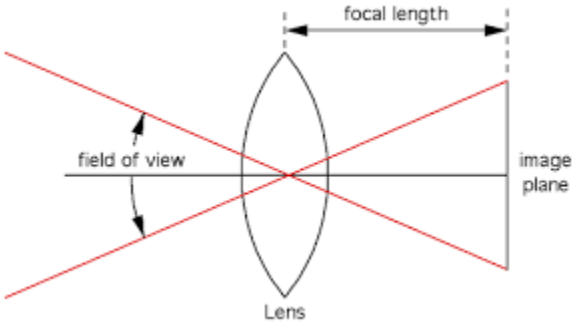
University of Maryland, College Park

Email: hpendli@umd.edu

I. Problem 1:

Given, camera with a resolution of 5mp, and width of the square sensor 14mm. Focal length of the camera is 25mm.

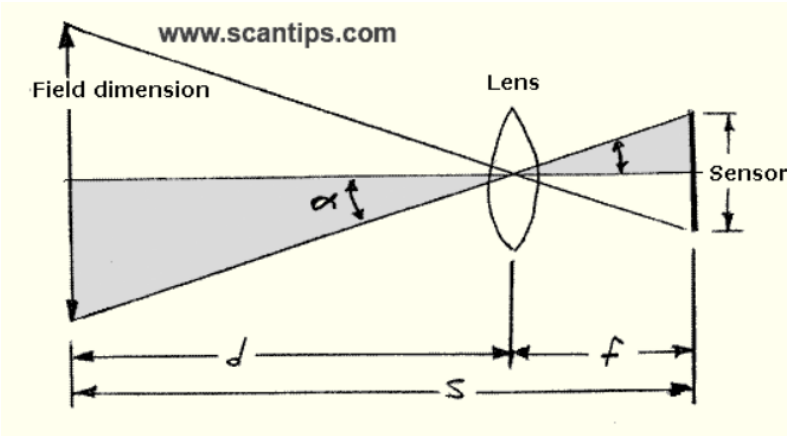
a) To compute the field of view (FoV) of the camera:



$$FoV = 2 \cdot \arctan\left(\frac{W}{2 \cdot S}\right)$$

Here, W=width of the camera sensor, S= focal length of the camera
Substituting corresponding values in the above equation, we get the Field of view as ***31.28degrees***. Since we have a square camera sensor, the horizontal and vertical field of view is equal.

b) Given, detect a square object of size 5cm at 20m from the camera. To compute the minimum number of pixels that the object will occupy in the image:



$$\frac{\text{Object height on sensor (mm)}}{\text{Focal length (mm)}} = \frac{\text{Real Object size *}}{\text{Distance to Object *}}$$

Substituting the values in the above equation, we get height of object on camera sensor as 0.0625mm.
We know that the square sensor area of 14mmx14mm is occupied by 5Mp, therefore by direct proportion rule, we have:

$$\frac{R}{Areaofsensor} = \frac{P}{Areaofimage}$$

Number of pixels occupied by the square object is approximately 99.65 pixels.

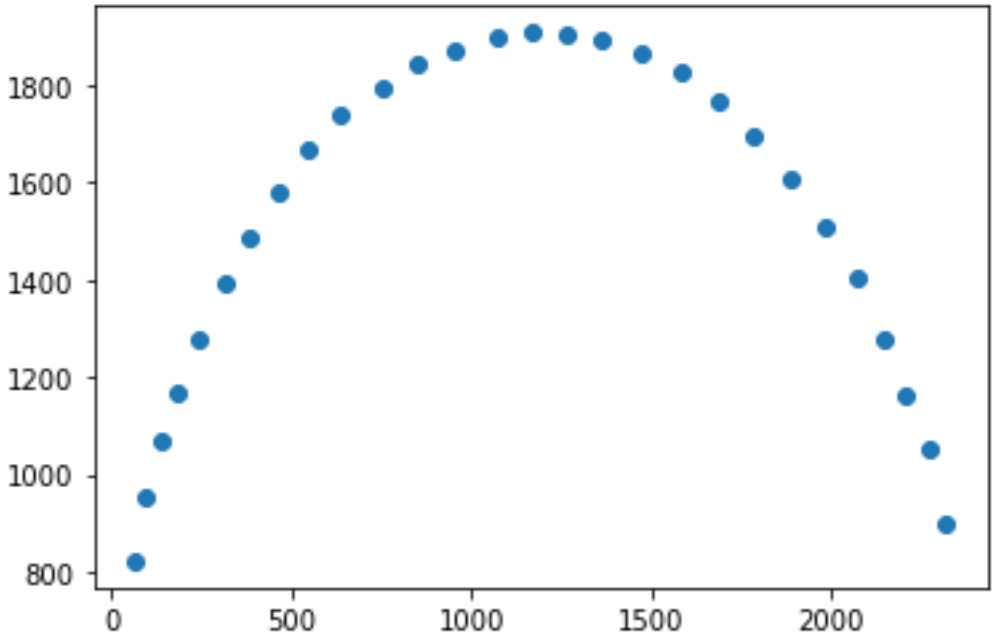
II. Problem 2:

Given that, a ball is thrown against a white background and a camera is used to track its trajectory. We are also given two videos corresponding to a no-noise-perfect sensor and a faulty sensor with noise. Assuming trajectory of ball follows equation of ball. We consider the following:

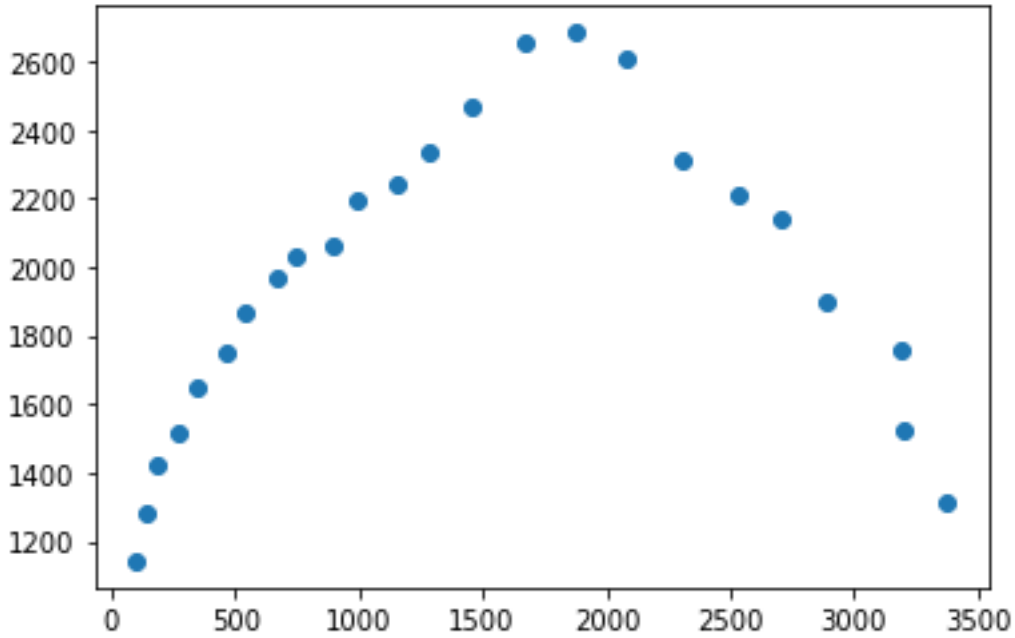
$$y = ax^2 + bx + c$$

a) To plot the points of the ball against the white background, we first convert the given videos into a series of images by reading the video from the folder.
After reading the videos we process the frames of the video such that we filter the red channel. We make use of masking filter to remove out the background and only detect the ball.
After detecting the ball, we find its topmost and bottommost points and thus calculate its centre over all the frames and form a set of datapoints.
We then perform transformation of the image coordinates into cartesian coordinates to plot the data.
To this data, we fit the Standard least-square parabola.

Plot of video-1



Plot of video-2



b) Fitting the standard least square curve to the parabolic trajectory of the ball

The error function can be written as,

$$E = \Sigma_n (y - ax^2 - bx - c)^2$$

Let $X = [x^2, x, 1]$, $Y = y$ and $D = [a, b, c]^T$. Then,

$$E = \|(Y - XD)\|^2$$

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

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

Differentiating wrt D,

$$\frac{\delta E}{\delta D} = -2X^T Y + 2X^T XD$$

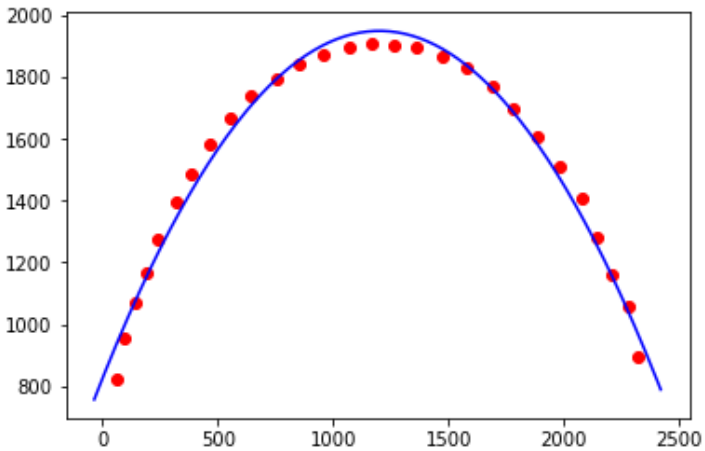
Since $\frac{\delta E}{\delta D} = 0$,

$$-2X^T Y + 2X^T XD = 0$$

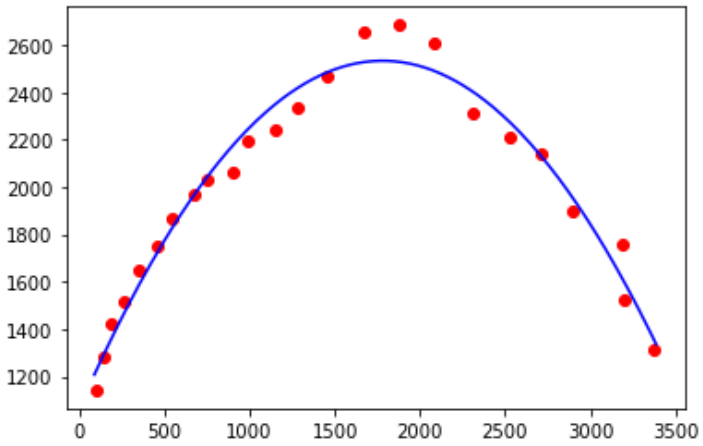
$$D = (X^T X)^{-1} X^T Y$$

The above calculation has been taken from the reference article attached to the third question.[article]

Standard Least square plot for video-1



Standard Least square plot for video 2



III. Problem 3:

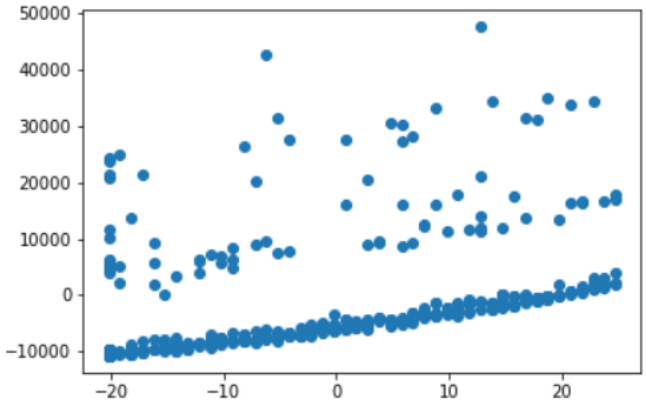
Given, data in a .csv file regarding the survey of health insurance charges based on the age of the person.

a) To compute the covariance matrix of the data. From the lecture slides of week 2, we are familiar with the following equation for some A, covariance matrix S is given by:

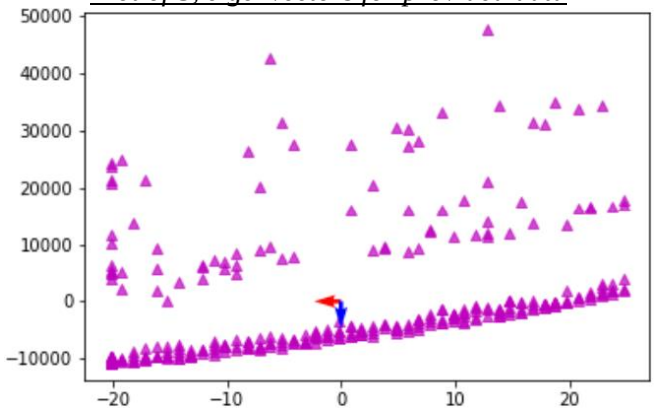
$$S = 1/n\ A\ A^T$$

Hence, we find the covariance matrix using the above equation where A is the data matrix of age, charges stacked together to create data points.

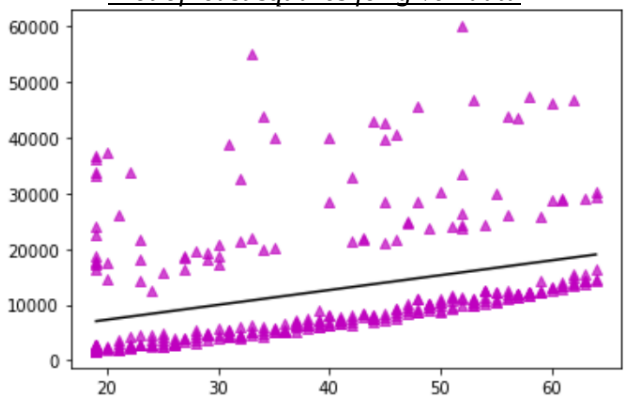
Normalized scattered data



Plot of S, eigenvectors for provided data



Plot of least squares for given data



To fit a least-squares line to a data, we try to find the distances between each data point and the line along the y- axis and try to find a line such that the sum of the distance squared values of the difference between the y coordinate of the data point is minimum. We consider the line to be: **y=mx+c**

Splitting it into matrices, like problem 2 we get the coefficients matrix D as the following, where E is the error given by square of the distance **|y-mx-c|, D=[m c]^T**

$$E = \|(Y - XD)\|^2$$

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

$$E = Y^TY - 2(XD)^TY + (XD)^TXD$$

Differentiating wrt D,

$$\frac{\delta E}{\delta D} = -2X^TY + 2X^TXD$$

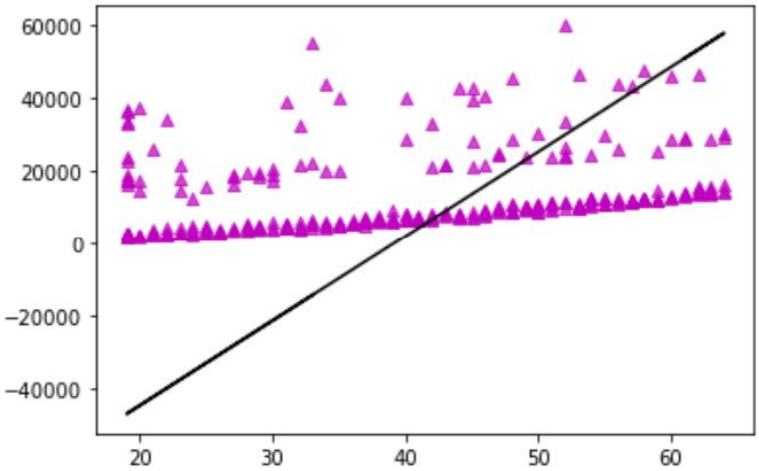
Since $\frac{\delta E}{\delta D} = 0$,

$$-2X^TY + 2X^TXD = 0$$

$$D = (X^TX)^{-1}X^TY$$

Thus, plotting the line with D coefficients matrix to the scattered plots, we get the above figure of least squares plot.

Plot of total least squares for given data



To fit total least square cure, we consider the following line equation: **ax+by+c=0**

We have, ax+by=-c

Error is given as: **E=** $\sum(ax+by+c)^2$

To minimize E, we set the derivative of E with respect to D to 0. Solving this condition, we get:

$$c = \frac{ax^2 + by}{n}$$

Therefore, $E=\sum(ax\_mean + by\_mean)^2$

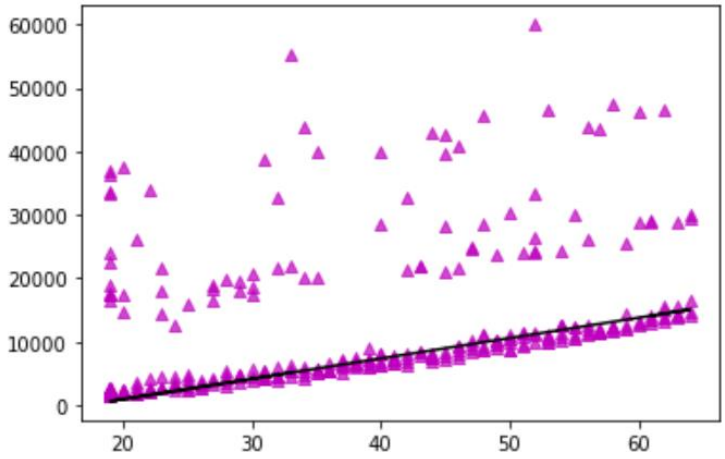
Let $U=[\ x-x\_mean \quad y-y\_mean]$, $N=[a\ b]^T$

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

Since E' is 0, we get $U^TUN=0$

We solve this by considering U^TU to be A and computing AN=0 by using SVD.

Plot of RANSAC for given data



Made use of the following pseudocode to design RANSAC function for curve fitting purposes.

```
Given:
    data – A set of observations.
    model – A model to explain observed data points.
    n – Minimum number of data points required to estimate model parameters.
    k – Maximum number of iterations allowed in the algorithm.
    t – Threshold value to determine data points that are fit well by model.
    d – Number of close data points required to assert that a model fits well to data.

Return:
    bestFit – model parameters which best fit the data (or null if no good model is found)

iterations = 0
bestFit = null
bestErr = something really large

while iterations < k do
    maybeInliers := n randomly selected values from data
    maybeModel := model parameters fitted to maybeInliers
    alsoInliers := empty set
    for every point in data not in maybeInliers do
        if point fits maybeModel with an error smaller than t
             add point to alsoInliers
        end if
    end for
    if the number of elements in alsoInliers is > d then
        // This implies that we may have found a good model
        // now test how good it is.
        betterModel := model parameters fitted to all points in maybeInliers and alsoInliers
        thisErr := a measure of how well betterModel fits these points
        if thisErr < bestErr then
            bestFit := betterModel
            bestErr := thisErr
        end if
    end if
    increment iterations
end while

return bestFit
```

Essentially, RANSAC is used to perform excellent outliners rejection while plotting a line for given dataset.

We start of by choosing two random points in the data set, draw a line between them. We calculate the perpendicular distance of the neighboring points to this line and determine the number of inliners based on the threshold value which is user specific. We iterate and repeat this process many times and consider the lines which have inliners greater than best number as best fits. We finally take the best of the best fit and end up with parameters to fit the model.

Useful formulas would be:

The recommended value for minimum distance threshold is square root of (3.84 * standard deviation)

Number of iterations given by N, p is desired accuracy, e is outliners ratio

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

$$e = 1 - \frac{no.of.inliers}{totalnumbers}$$

Analysis of the curve fitting methods:

i) Least squares method:
-A robust technique that performs well when fitted to nonoutlier data.
-Doesn't work well with too noisy data and outliers.
-Not recommended if data is aligned vertically because only y-distance is considered.

ii) Total least squares method:
-A robust technique and performs well exceptionally well for noise less data.
- Gives too much weight to outliers due to which the line doesn't give the actual information about the alignment of the inliers.

iii) RANSAC:
-Best technique to fit noisy data or also for data with outliers.
-Capable of taking user specific inputs while considering thresholds and number of iterations.
-Can take some time to process, if given data is too vast.

IV. Problem 4:

Given matrix A,x. To compute Homography matrix H by solving Ax=0. Here x is the flattened H matrix which corresponds to the last column of V vector of Svd(A), that is to the smallest eigenvalue of AA^T. We normalize the Homography matrix by making the last element of matrix 1 and changing the rest of the elements relatively.

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

Also given that,

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

Compute SVD by hand for A: [$A=U*sigma*V^T$]
Substituting the values in the above table into A matrix gives,

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

Calculate AA^T and its eigenvalues and vectors. Sort eigenvalues in descending order and place the eigenvectors corresponding to this newer order (U).

$$\begin{pmatrix} 1.17519867e-02 & 3.44207228e-04 & -5.15532162e-02 & -4.66128587e-01 & -2.60345896e-01 & -6.78428560e-02 & 1.08122929e-02 & -8.41087769e-01 \\ 1.17517760e-02 & 3.43641967e-04 & -8.72103737e-02 & -4.59351955e-01 & -2.49098952e-01 & -8.85591890e-02 & 7.65455993e-01 & 3.54169470e-01 \\ 3.58735699e-01 & 6.54942912e-01 & 1.34538659e-02 & -4.65084492e-01 & 1.70101644e-01 & 2.93617516e-01 & -2.78385484e-01 & 1.82289869e-01 \\ 1.43494223e-01 & 2.61976394e-01 & -4.45383120e-01 & 1.36060221e-01 & -5.00795526e-01 & -5.87488150e-01 & -2.73099287e-01 & 1.52897236e-01 \\ 7.74962678e-01 & 2.27117371e-02 & 4.08516159e-01 & 2.84937362e-01 & 3.19642679e-02 & -2.35211438e-01 & 2.62688692e-01 & -1.59658351e-01 \\ 2.81806634e-01 & 8.24745878e-03 & -6.92167142e-01 & 3.15915567e-01 & 1.14149714e-02 & 5.01908806e-01 & 2.46628160e-01 & -1.69560615e-01 \\ 1.84643411e-01 & -3.16806256e-01 & 2.48466337e-01 & -3.46544961e-02 & -6.98268275e-01 & 4.67261587e-01 & -2.52393736e-01 & 1.81630339e-01 \\ 3.69278450e-01 & -6.33614920e-01 & -2.88917222e-01 & -3.93333286e-01 & 3.18917542e-01 & -1.75016528e-01 & -2.61429000e-01 & 1.52633610e-01 \end{pmatrix}$$

Similarly, calculate A^TA and its eigenvalues and vectors. Sort eigenvalues in descending order and place the eigenvectors corresponding to this newer order (V).

$$\begin{pmatrix} 2.84043894e-03 & 3.14430147e-03 & -2.46384735e-01 & -1.58554932e-01 & -1.75245114e-01 & 1.76705635e-01 & 9.13738625e-01 & -1.20261073e-01 \\ 5.31056350e-02 & & & & & & & \\ 2.42121739e-03 & -1.28321626e-03 & -3.77000733e-01 & 1.76600215e-01 & 6.89508147e-01 & 5.90273326e-01 & -5.29344506e-02 & -2.23230964e-03 \\ -4.91718844e-03 & & & & & & & \\ 2.20891154e-05 & 1.13495064e-05 & -2.37217168e-03 & -3.65660431e-03 & 5.19584361e-03 & 7.52000216e-03 & 6.59901592e-02 & 7.85970680e-01 \\ 6.14648552e-01 & & & & & & & \\ 1.09109680e-03 & 1.17416448e-03 & 6.61240940e-01 & 3.41172744e-01 & 5.01749740e-01 & -2.32499365e-01 & 3.72052359e-01 & -4.25903575e-02 \\ 1.77018784e-02 & & & & & & & \\ 1.63479471e-03 & -2.90636016e-03 & 5.74279813e-01 & -7.10405325e-02 & -3.14549320e-01 & 7.49883366e-01 & -6.19835401e-02 & 4.58785873e-03 \\ -3.93375075e-03 & & & & & & & \\ 1.33908907e-05 & -1.14077892e-05 & 5.80190908e-03 & -2.15181510e-03 & 2.88147957e-03 & -5.73504703e-03 & -1.22489909e-01 & -6.04930528e-01 \\ 7.86750146e-01 & & & & & & & \\ -6.96053715e-01 & -7.17961695e-01 & -7.57487350e-05 & -3.79564118e-03 & 2.50334194e-03 & -1.50223526e-04 & 4.37766998e-03 & -5.55196291e-04 \\ 2.36025045e-04 & & & & & & & \\ -7.17950893e-01 & 6.96067270e-01 & 1.62813209e-03 & -3.77529395e-03 & 2.49754245e-03 & 3.65599795e-03 & -6.00114914e-04 & 3.48250731e-05 \\ -4.91718843e-05 & & & & & & & \\ -6.16016024e-03 & 2.29933343e-05 & -1.73453679e-01 & 9.06741660e-01 & -3.78319687e-01 & 6.21986452e-02 & 2.52338778e-02 & -2.47794676e-03 \\ 7.62164205e-03 & & & & & & & \end{pmatrix}$$

Find the sigma matrix which is given by the diagonal matrix of the size of the A matrix with diagonal elements filled with the square root of eigenvalues of U.

$$\begin{pmatrix} 6.021490e+04 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 3.182452e+04 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 2.608900e+02 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 1.862200e+02 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 1.456100e+02 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 6.088000e+01 & 0.000000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 3.900000e+00 & 0.000000e+00 \\ 0.000000e+00 & & & & & & & \\ 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 0.000000e+00 & 8.100000e-01 \\ 0.000000e+00 & & & & & & & \end{pmatrix}$$

Thus, we have the three matrices of the svd matrix decomposition: U, sigma, V

Following the first two lines of this solution we get the x matrix as:
Therefore, the homography matrix is given by:

$$H = \begin{bmatrix} 6.96774195e+00 & -6.45161293e-01 & 8.06451613e+0 \\ 2.32258065e+00 & -5.16129035e-01 & 1.03225806e+02 \\ 3.09677420e-02 & -6.45161292e-03 & 1.00000000e+00 \end{bmatrix}$$

One of the problems faced during the calculation of the Svd matrix is, due to some error in eigenvalues, they were negative which mathematically not possible. Due to this, I had to absolute negative values to make the error go to zero.
Also, while calculating homography one of the constraints to implement, was to check for four points in each point set. Another issue is the calculation of nine unknowns with eight equations, which is resolved by considering the last term of the Homography matrix to be 1.