ENPM673 #HW1

Problem1:
Resolution = 9MP, Focal length = 15mm, camera sensor width = 14mm
Part 1:
To compute the field of view in the horizontal and vertical direction
The field of view in this case would be the same in both the directions since it is a square shaped sensor, and the dimension would be same in both the directions.

The field of view can be computed as
FOV=2 X arctan (sensor dimension/2f)

FOV = 2 x 24.98
=> **49.96 degrees**

Part 2:
To compute the minimum number of pixels an object being detected by the sensor will occupy in the image.

Width of square shaped object = 5cm = 50mm
Distance of object from the camera = 20m = 20000mm

The ratio of the size of the object on the sensor(mm) and the size of the object in reality (mm) is the same as the ratio of focal length and distance to the object (similar triangles on both sides of the lens).

$$\frac{\text{Object height on sensor (mm)}}{\text{Real height of object (mm)}} = \frac{\text{focal length (mm)}}{\text{Distance of sensor from object (mm)}} \qquad (1)$$

The size of object on the sensor can be computed by dividing the image height in pixels and multiply the height of the sensor.

$$\text{Object height on sensor (mm)} = \frac{\text{sensor height (mm) x object height on sensor in pixels}}{\text{image height (pixels)}} \qquad (2)$$

The goal is to find the object height in pixels.

Substituting (2) in (1)

$$\text{object height (pixels)} = \frac{\text{(focal length) x (real height of object) x (image height in pixels)}}{\text{distance of sensor from object x sensor height (mm)}}$$

image height in pixels = 3000 pixels since this is a square shaped sensor

objects real height = 50mm

Object height on sensor in pixels needs to be computed.

Substituting the respective values, we get

$$\text{object height (pixels)} = \frac{15 \times 3000 \times 50}{20000 \times 14}$$

$$\Rightarrow 8.03 \text{ pixels}$$

Since the object is square shaped the width would be similar to the height

Therefore, the minimum number of pixels occupied by the object is $8.03^2 = 64.48$ which is approximately around **64 pixels**.

Problem2:

The three methods used to fit curves for the videos are least squares, total least squares and RANSAC methods.

Before running the different methods. The following steps are to be taken:

1. Read the video files frame by frame.
2. Filter out the red ball from the rest of the image. Here I convert the image into the HSV color space and set a mask to find the red values.
3. I use a bitwise operator where I multiply the original image with the black value of zero which would black out all the non-red regions.
4. I then resize the image by dividing the rows and columns by 5.
5. The top and bottom red pixel of the ball is found.
6. These coordinates are averaged to find the centroid of the ball in the image.
7. The x and y coordinates are populated into an array and the respective method functions are called.

Least squares method

The steps for undertaking the least squares method is as follows:

1. Obtain A for the equation Ax=B
2. The A matrix is formed by creating a M x 3 matrix of the form $[x^2 \;\; x \;\; 1]$ consisting of the X coordinate values.
3. The B matrix is a N x 1 matrix which consists of just the Y pixel coordinate values.
4. Based on this I call another function to compute the x values in Ax=B
5. I obtain x values by performing $(A^TA)^{-1}.(A^TB)$. The x values would consist of 3 elements a, b, c of the polynomial equation $y=ax^2 + bx + c$
6. The 3 x 1 matrix is then multiplied with A to obtain the new B matrix values.
7. The x coordinate values are then plotted with the new estimated values. This would give a parabolic curve fitting the different positions of the ball in each frame

Total least squares method

1. In this method we need to similarly find out the x values in Ax=B.
2. But in this case we need to compute the sum of x^2, x, y, x^2y, x^3, x^4.
3. Based on these values the A matrix and B matrix would be formed. The A matrix would be of the form:

$$\begin{bmatrix} sum(x^2) & sum(x) & n \\ sum(x^3) & sum(x^2) & sum(x) \\ sum(x^4) & sum(x^3) & sum(x^2) \end{bmatrix}$$

The Y matrix would be of the form $[sum(y) \quad sum(y^2) \quad sum(x^2y)]$

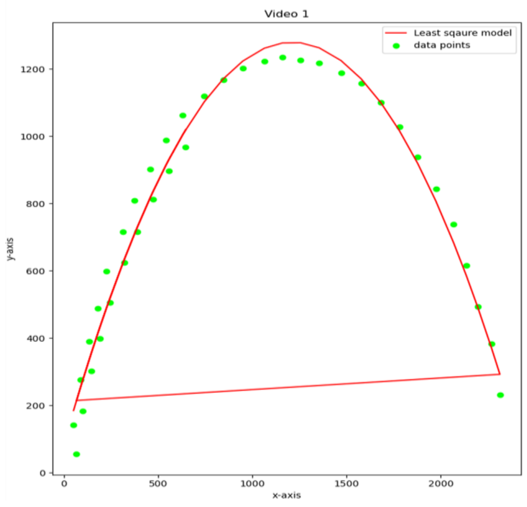
4. The a, b, c values is obtained by solving these equations using SVD.
5. Since Ax= B where x= [a b c], to solve for x we need to solve for x = A^{-1}. B
6. The eigen values and vectors for U can be obtained by computing A.A^T.
7. The singular matrix sigma can be obtained by creating a diagonal matrix with the diagonal elements consisting of the square root of the eigen values.
8. Based on this we can find V^T which would ϵ^{-1}. U^T.
9. Therefore A^{-1} would be V.ϵ^{-1}.U^T.
10. Based on this the a, b, c values can be obtained to compute the new estimated values.

RANSAC method

1. Create the A matrix of the from [x^2 x 1].
2. Set a threshold which is a standard deviation of the Y coordinate values. The SVD can be divided by 3,4, 5 for better fits to create a narrow distribution.
3. The desired probability is set to be 0.95, and 3 sample points are taken at a time since we have 3 unknowns in the quadratic equation a, b, c.
4. A while loop is run to check if the number of iterations is greater than the iterations done.
5. A matrix of 4 columns is created consisting of the A matrix of the form [x^2 x 1] and the y values in a B matrix.
6. To plot the RANSAC model I use the estimate of the least squares model based on 3 sample points.
7. The new B matrix values obtained is subtracted from the original B matrix to find the error.
8. The inlier count is checked by comparing this value with the threshold and if it is less than threshold the point is counted as an inlier.
9. The probability of outliers is verified by computing 1 – inlier count/ total number of data points.
10. The number of iterations can be found by N = log (1− p)/ log (1− (1− e) ^{s})
11. The model which has the maximum number of inlier points is chosen as the best RANSAC fit.
12. Whenever the number of iterations is less than the iterations the flow breaks out of the loop and returns the best fitting model.

Issues encountered.

1. While plotting the best fits for the different methods I considered both the top and bottom pixel coordinates to find the best fit. Even though I would be getting a parabolic curve to fit the data, the extreme end of the parabola would be connected as in the below image. Averaging the top and bottom pixel values helped me solve this issue.



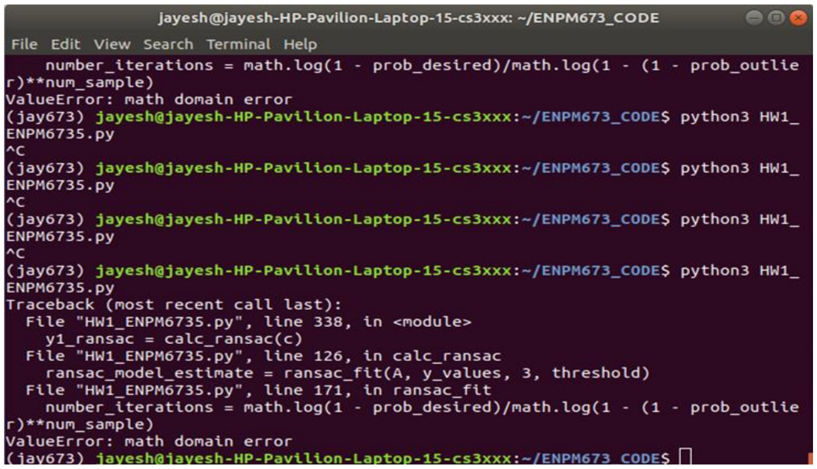
2. While computing the TLS I obtained the eigen values to be negative. To avoid this I multiplied the eigen values with -1 with the negative values which helped mt obtain a best fit for TLS.

Observations and understanding on which method is the best fit for the data

The following are some of the observations when running RANSAC under different conditions.

For RANSAC 3 points are selected as samples as we need to three equations to solve the quadratic equation ax2 +bx + c. I set the threshold to be a standard deviation of B matrix of the frame. I divide this value by 2, 3 or 5 to create a narrow boundary and check the maximum inliers in it. This increases the overall probability of the inliers.

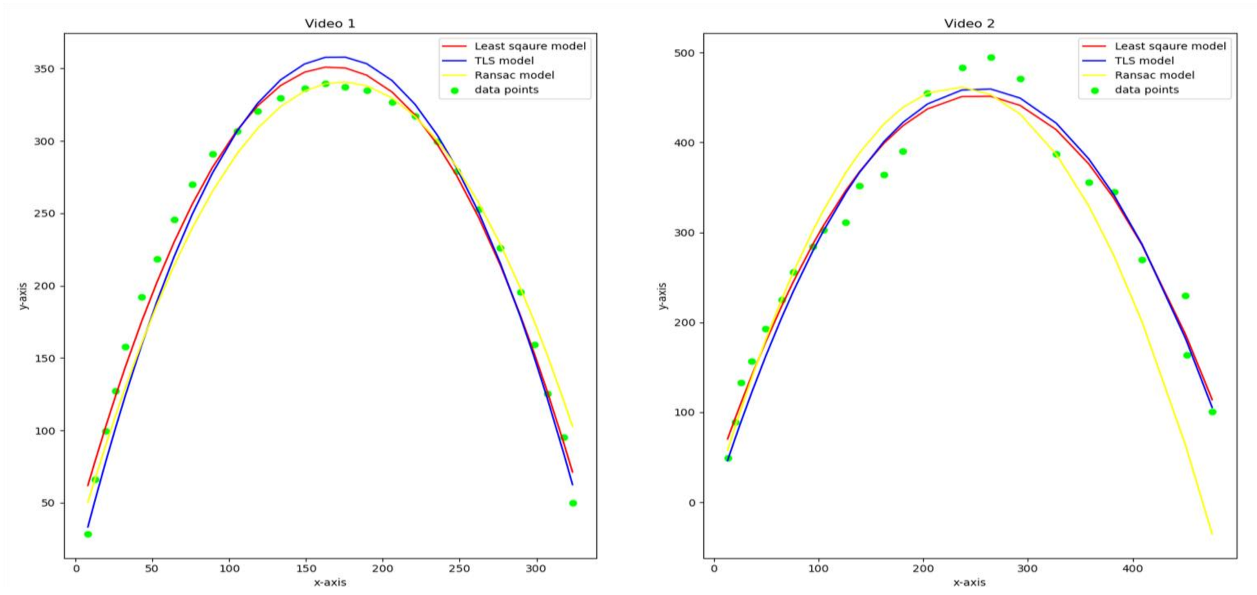
When threshold = standard deviation of y or y/2 where y refers to the Y coordinates of the pixel values in a frame, the below error is obtained.


jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx: ~/ENPM673_CODE
File Edit View Search Terminal Help

```
    number_iterations = math.log(1 - prob_desired)/math.log(1 - (1 - prob_outlie
r)**num_sample)
ValueError: math domain error
(jay673) jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx:~/ENPM673_CODE$ python3 HW1_
ENPM6735.py
^C
(jay673) jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx:~/ENPM673_CODE$ python3 HW1_
ENPM6735.py
^C
(jay673) jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx:~/ENPM673_CODE$ python3 HW1_
ENPM6735.py
^C
(jay673) jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx:~/ENPM673_CODE$ python3 HW1_
ENPM6735.py
Traceback (most recent call last):
  File "HW1_ENPM6735.py", line 338, in <module>
    y1_ransac = calc_ransac(c)
  File "HW1_ENPM6735.py", line 126, in calc_ransac
    ransac_model_estimate = ransac_fit(A, y_values, 3, threshold)
  File "HW1_ENPM6735.py", line 171, in ransac_fit
    number_iterations = math.log(1 - prob_desired)/math.log(1 - (1 - prob_outlie
r)**num_sample)
ValueError: math domain error
(jay673) jayesh@jayesh-HP-Pavilion-Laptop-15-cs3xxx:~/ENPM673_CODE$ 
```

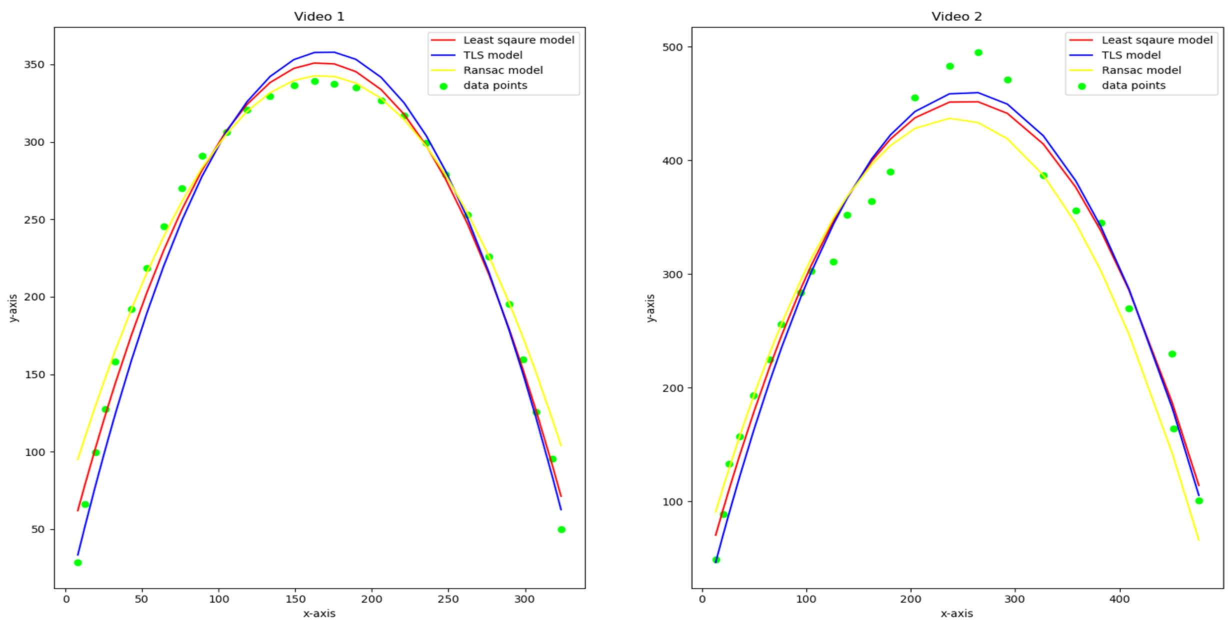
The error is due to the inlier count being equal to the total number of points as a result of which the probability of outliers is zero and log (0) would be undefined when computing the number of iterations.

For certain runs where threshold = standard deviation (y/2) the below graph is obtained

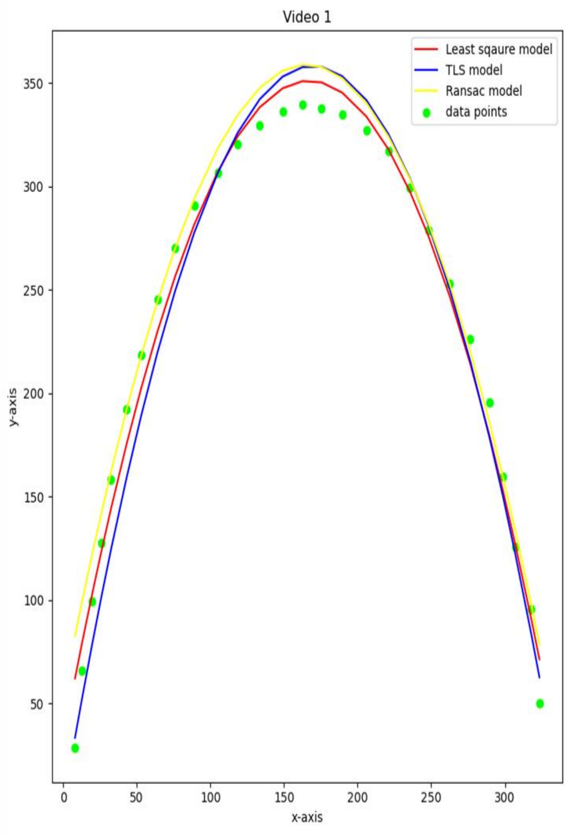


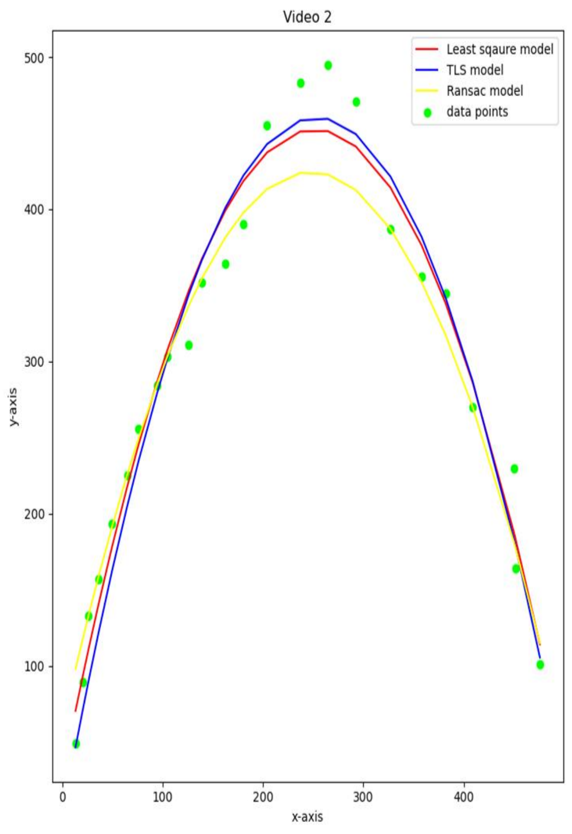
RANSAC has a larger distribution to fit the points and therefore looks to connect even the far points in video 2.

When threshold is set to be the standard deviation of all the Y values/3. The following curves are obtained for the two videos.

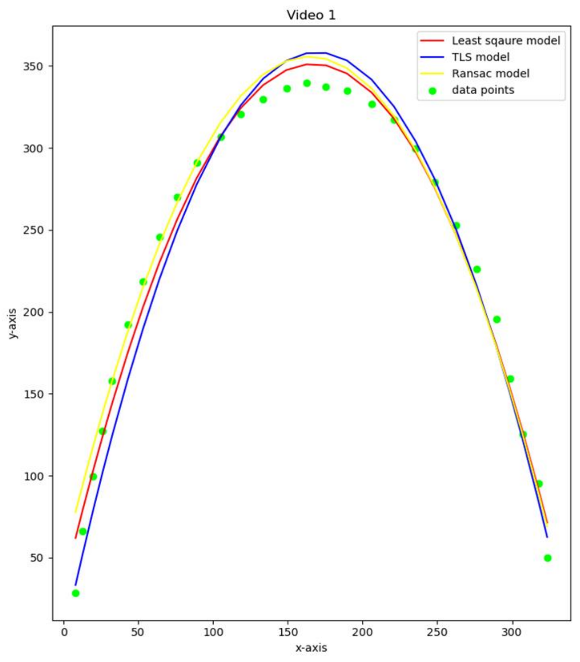


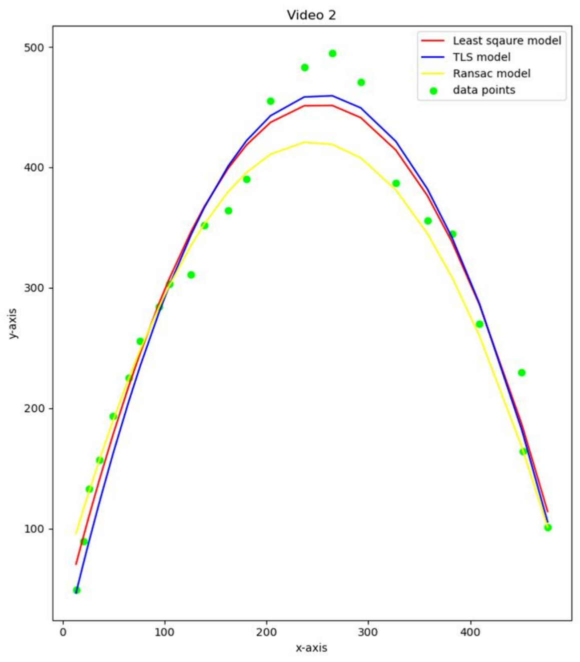
When threshold = y/4





When threshold = y/5

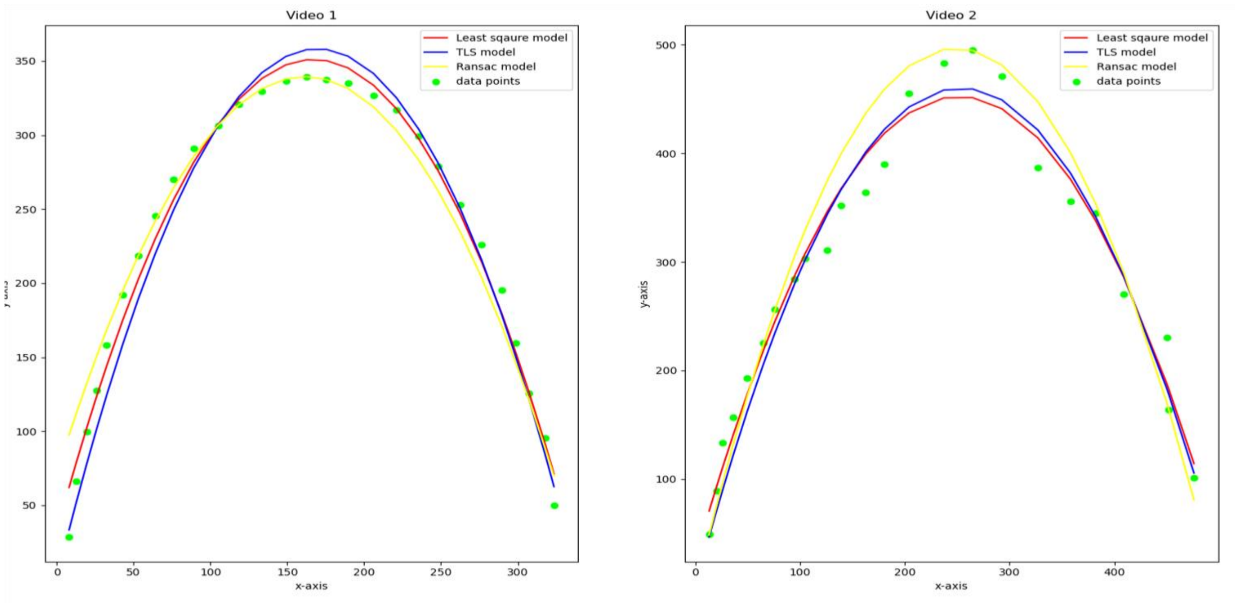




In case of video 1 all the 3 methods are fitting the points nicely since all the points are evenly spaced out. Whereas for video 2 LS and TLS are looking to cover all the points which are not in proximity with the other points and the curve lies more towards these points. RANSAC on the other hand tries to cover only those points which lie within the threshold. The threshold is obtained as the standard deviation of y values which as we divide by 3,4,5 creates a narrower boundary and thus the curve is more downward compared to LS and TLS.

Therefore, based on the above observations the following conclusions can be made.

1. The LS and TLS plots try to cover almost all the points in the image for video 2 whereas RANSAC ignores some points as outliers and covers only those which are inliers.



2. The other advantage of using RANSAC is the function runs a certain number of times till it finds the best fit which are consistent with an estimated model (in this case the least squares model). The model with the maximum number of inlier points is chosen as the best fit model.

Therefore, I feel RANSAC is a better method to fit the data because there is a pattern based on which the outliers are rejected and does not connect points which are far away from each other as in video 2.

Problem3:
To show mathematically how SVD would be computed for matrix A

The goal is to find the U, Sigma and V^T matrices
The following are the steps to mathematically compute the SVD.

Step1: Compute the U matrix.
1.1 Find A^T and calculate $A.A^T$.
1.2 Find the eigen values (λ) of this matrix by performing $(A.A^T – \lambda I) = 0$.
1.3 Substitute the λ values in $(A.A^T – \lambda I)$ and obtain the respective eigen vectors. Normalizing values in the eigen vectors we can get the U matrix.

Step 2: Compute the ϵ matrix.
The sigma matrix is a diagonal matrix which consists of the square root of the eigen values as its diagonal elements.

Step 3: Compute the V matrix.
1.2 Find A^T and calculate $A^{T.A}$.
1.2 Find the eigen values (λ) of this matrix by performing $(A^T.A – \lambda I) = 0$.
1.3 Substitute the λ values in $(A^T.A– \lambda I)$ and obtain the respective eigen vectors. Normalizing values in the eigen vectors we can get the V matrix.
1.4 Obtain V^T from V.

Step4: SVD of A would be given by $(U.\epsilon.V^T)$