HW#1

ENPM673

Rishabh Singh

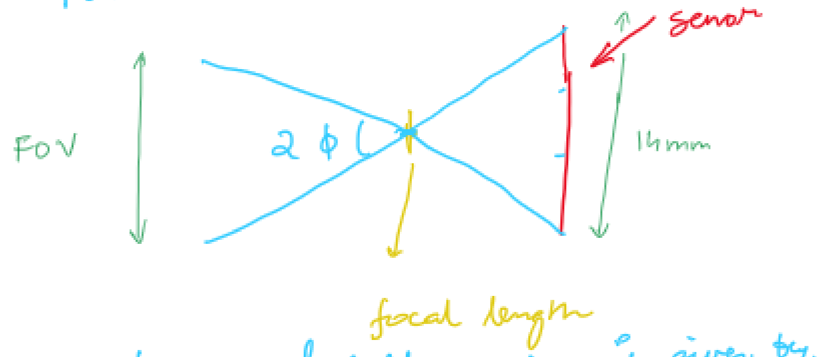
117511208

U of Maryland, CP

Problem 1:

Part (a)

FOV in vertical direction



we know that field of view is given by,

$$\phi = \tan^{-1}\left(\frac{d}{2f}\right) = \tan^{-1}\left(\frac{14}{50}\right)$$

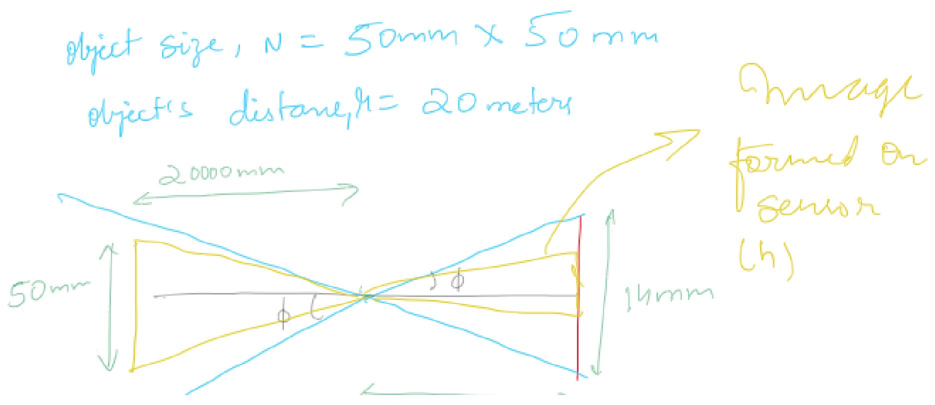
The above formula shows the relation between field of view, focal length, and sensor height/width. Here, Φ is the field of view, f (25 mm) is the focal length and d (14 mm) is the sensor dimension. Now by putting the values in the formula we got the value of a field of view = 31.68° or 0.546 rad. Since the sensor is square both the horizontal and vertical FOV's are the same.

Part (b)

In this part, we have been asked to find the number of pixels covered by the object kept at a certain distance from the sensor.

The resolution of sensor is 5 MP. Now, we have solved this problem using similar triangles geometry.

(Height of object/distance from focus) = (height of image/focal length)


object size, N = 50mm x 50mm
object's distance, H = 20 meters
20000mm
50mm
ϕ
ϕ
14mm
Image formed on sensor (h)

Also,

(Pixels covered by image/resolution) = (image area / sensor area)

Using these two ratios, we got the minimum pixels covered by the image as approximately 100 pixels.

Problems faced: I was first considering that the ratio of objects' height and FOV at that point should be equal to the ratio of pixels covered by the image and resolution of the camera.

Problem 2:

In this section, we had to find the trajectory of a ball in a noisy and not noisy video. The steps followed were as follows"

Read the video

Found the grayscale image using red color filtering

Divided the video into the frames

At each frame I studied what is the average value I am getting at the top and bottom-most point of the ball to decide a threshold value.

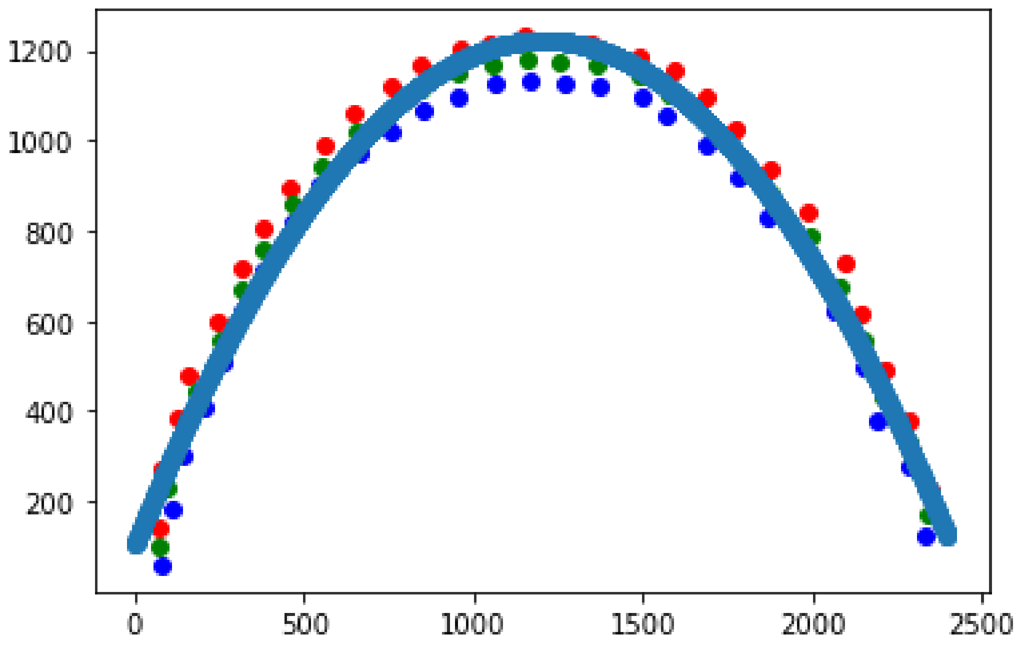
Then I used the NumPy where function to find the index of both the top and bottom pixels.

Plotted the curve for top pixel, bottom pixel and the mean of those values.

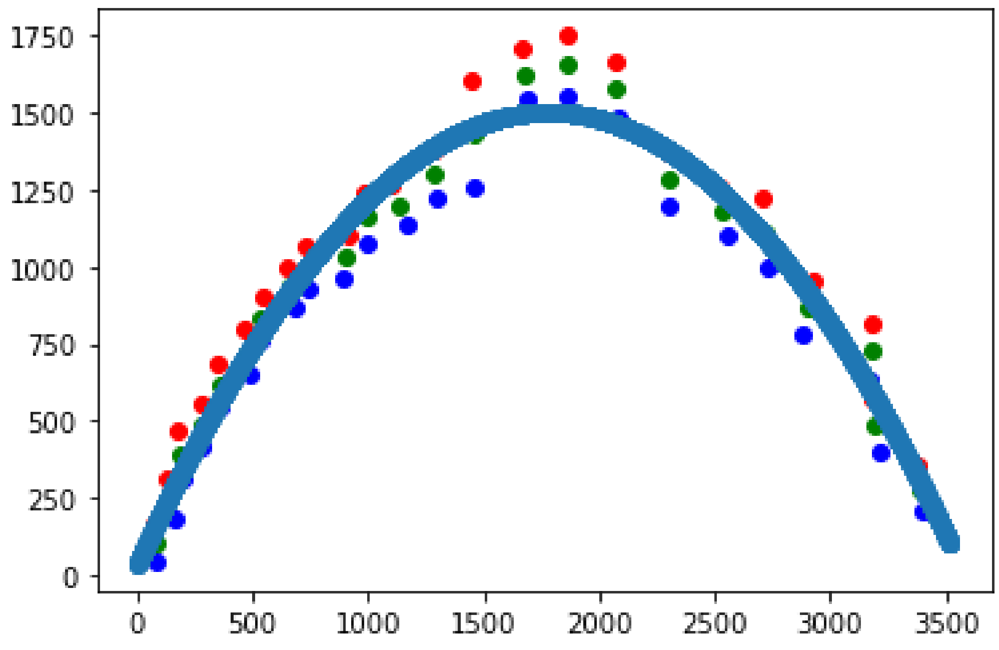
Since, this was a curved plot, I used the quadratic equation to fit the curve using least squares method.

$$\hat{x} = (A^T A)^{-1} A^T b.$$

Where, x bar gave the value of the coefficients. Which was used to plot the curve below:



Ball Video curve fit with no noise



Ball video curve fit with noise

Problems faced:

1) I had no idea of how to use OpenCV library
2) I had no idea about how to find the pixels

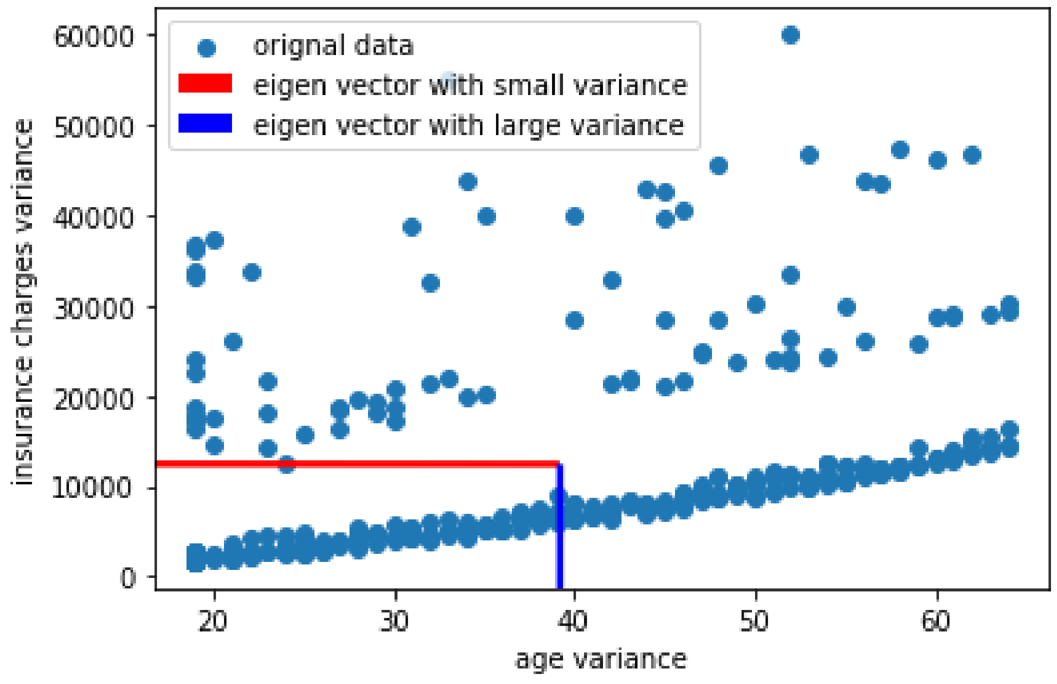
3) I was using for loops to find the top and bottom pixel, which was taking 6 minutes. But then I utilized the where() from the NumPy library which optimized the code to just 3 seconds.

Problem 3:

In this question, we were given the data to be potted by using Standard Least Squares, Total Least-squares and RANSAC method. We also had to find the covariance matrix and pot the eigenvectors.

Part (1)

Data vs. eigen vectors:



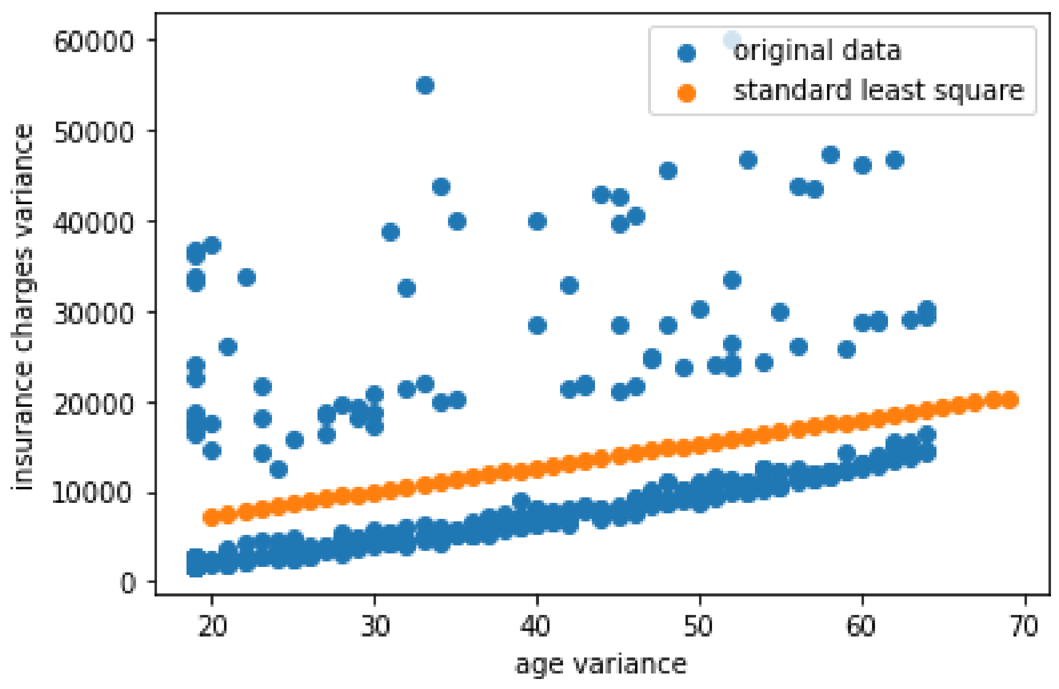
The spread in the y data is way more than x.

Problem: The eigenvectors are not aligned according to the data spread.

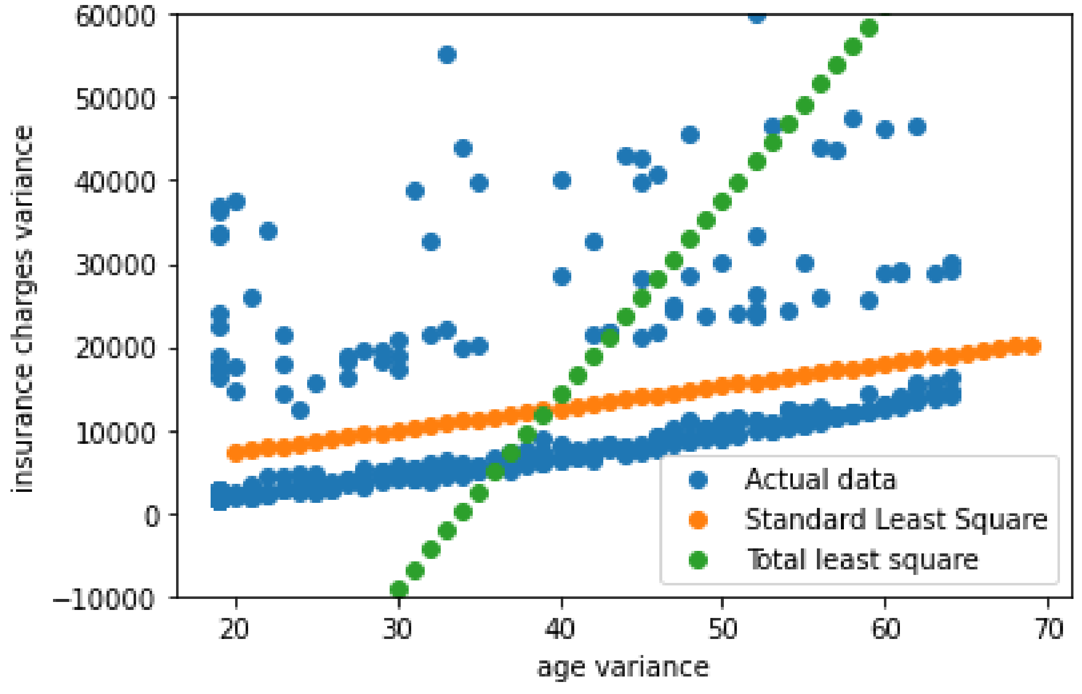
Part (2)

Standard Least Square method:

This used the same method as in question 2 except the fact that here we just had a line to fit.

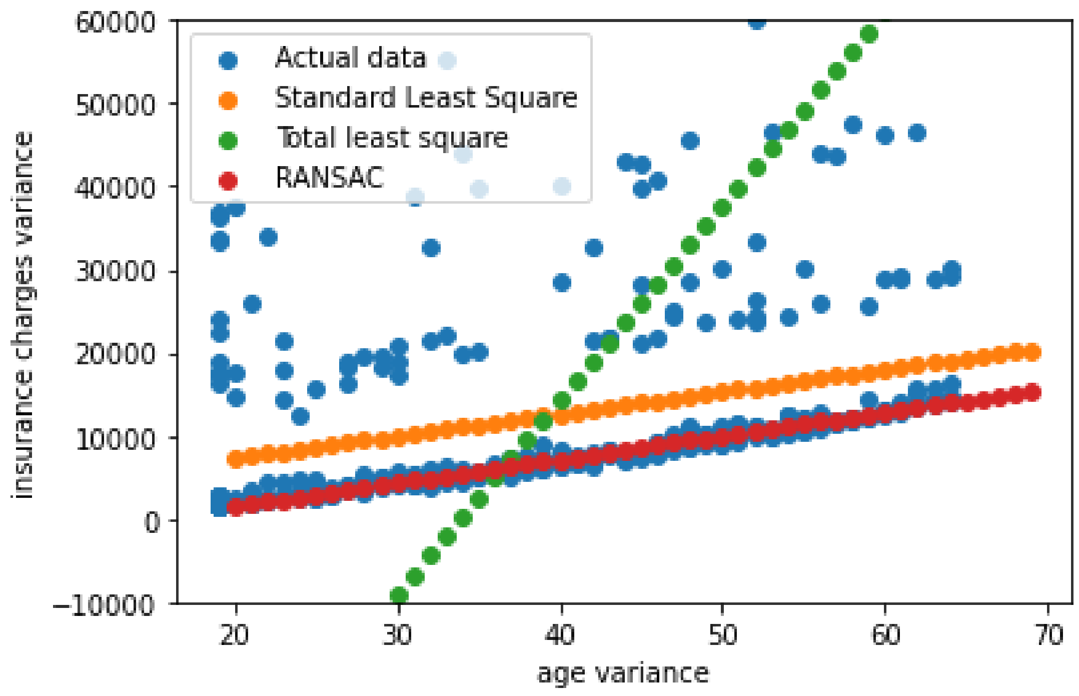


Total Square method fit:



Problem: I couldn't find a better idea of how to code this method. I used a thorough method to do this.

RANSAC data fit:



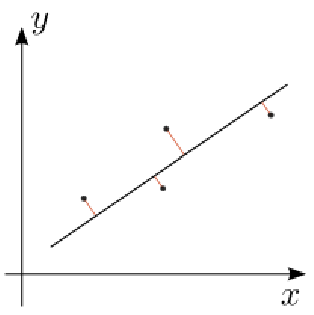
Part (3)

Overall, since we had a good amount of noise, RANSAC gave us the best fit. RANSAC utilized the total least square function to optimize the curve fit. So, we always must utilize the TLS method to code RANSAC. I ended up making a function of Total Least Square. But understanding and coding the RANSAC method was the toughest.

Steps utilized:

Total Least Square:

In total least square we found the error in both x and y values.



a)

This makes it better than the standard least square. We found the squared error in both x and y data using the mean of the data and formed a vector as below:

$$U = [x^2 - \bar{x^2}, x - \bar{x}, y - \bar{y}] \text{ and } N = [a, b, c],$$

After minimizing the error we get the following homogenous equation:

$$U^T U N = 0$$

This is then solved using SVD where the a, b, and c coefficients are the column of V transpose corresponding to minimum eigenvalue or singular value in Sigma. Then, the line was plotted using the optimized a, b and c value.

RANSAC method:

This method utilized the total least square (TLS) function.

a) We first selected two random points from the data, passed it through the TLS function and returned the coefficient.
b) Using the coefficient we found the outliers by defining a threshold.
c) We continue selecting the points unless we got the accuracy we want.

Problem 4:

All the steps for calculating the SVD are explained in the pdf and the jupyter notebook.