REPORT

ENPM 673 PERCEPTION OF AUTONOMOUS SYSTEMS

QUESTION 1

For the problem 1, a red ball thrown across a room needs to be detected.

PART 1

- The First step involves reading the video file ball.mov using the inbuilt feature in opencv called videoCapture.

- After following the syntax to successfully display the video, including adding waitkey, and reading the frames, the video needs to be converted to HSV(Hue, saturation, value) from the default format, BGR(Blue, Green, Red).

- Next step involves setting the lower and upper threshold values of the HSV in the red region so that the ball can be localized from the image. Using these threshold values, a mask function with the help of a BITWISE AND function is introduced that will not only localize the red ball but also show its trajectory in the video.

- After the trajectory of the ball is obtained, we need to obtain the coordinates of the ball in the video at different frames. The np.where() function is used to store the coordinates of the ball along the various frames in the video. Wherever the mask value is not 0 or black i.e. the locations where it locates the ball, the x and y coordinates are stored in separate arrays.

- The mean values of the is found and then appended into these empty lists so that we can obtain the center coordinates of the ball in each frame.

PART 2

- The following step involves plotting these center points of the ball on a graph. The matplotlib library is being used for this problem to plot the obtained results. First, we define a least square function that takes 3 arguments(coefficients of the equation) and code the formula obtained from the powerpoint given In class.

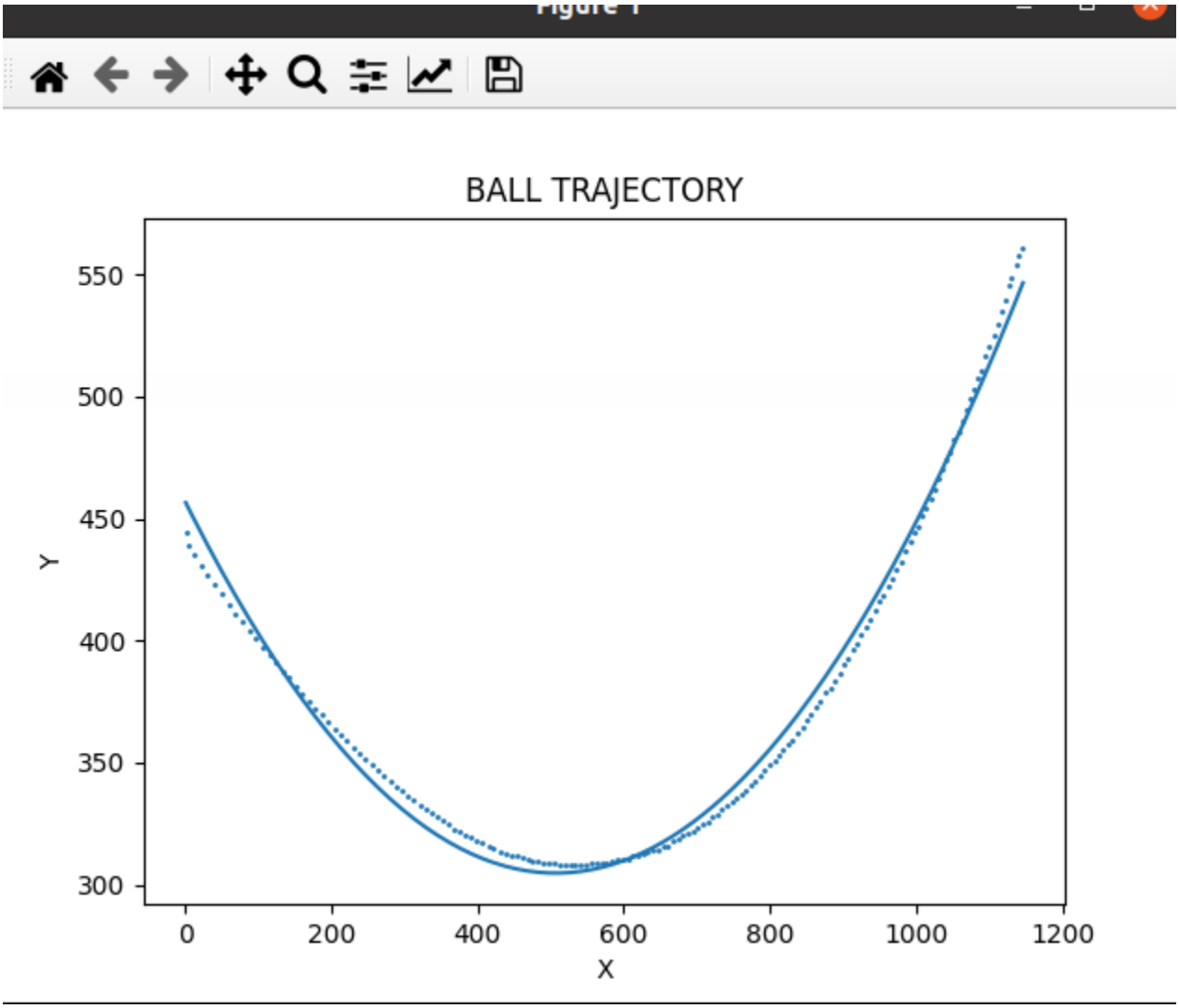
PART 3

- The third part of the question requires us to find the value of X coordinate from a value of Y coordinate that is not placed in the current frame. From the trajectory, we obtain the first value of Y coordinate that pops up at the instance of red ball appearing in the video.

- The value is then added with 300 and from the coefficients obtained in the previous step, we find the value of the new X coordinate by fitting the updated Y value and the coordinates.

Value of the new Y coordinate=1380.8455588590514

OUTPUTS



PROBLEMS ENCOUNTERED

- *Getting used to OpenCV functions and getting to know the fact that we need to include a mask to localize the ball took a considerable amount of time.*
- *Getting to know the numpy functions and cognizance of functions like np.where() consumed a lot of time.*

PROBLEM 2

For the problem 2, two datasets are provided containing noisy LIDAR points data in the form of x,y,z coordinates

PART 1(a):

This part of the problem requires us to find the covariance matrix of the three coordinates given in data set pc1.csv respectively.

- First, we use the pandas data frame to load the data set into the visual studio code. Next we store the 3 coordinates in separate arrays so that it would be easy to use in the formulas.

- The covariance matrix contains elements that requires the computation of the variance and covariance of each columns and in between columns.

- Therefore, 2 separate functions called covariance and variance is created and the respective formulas are coded in these blocks so that the next time these values are used, we will only have to call these functions. The variance and covariance values of individual columns are then substituted in order to obtain the covariance matrix

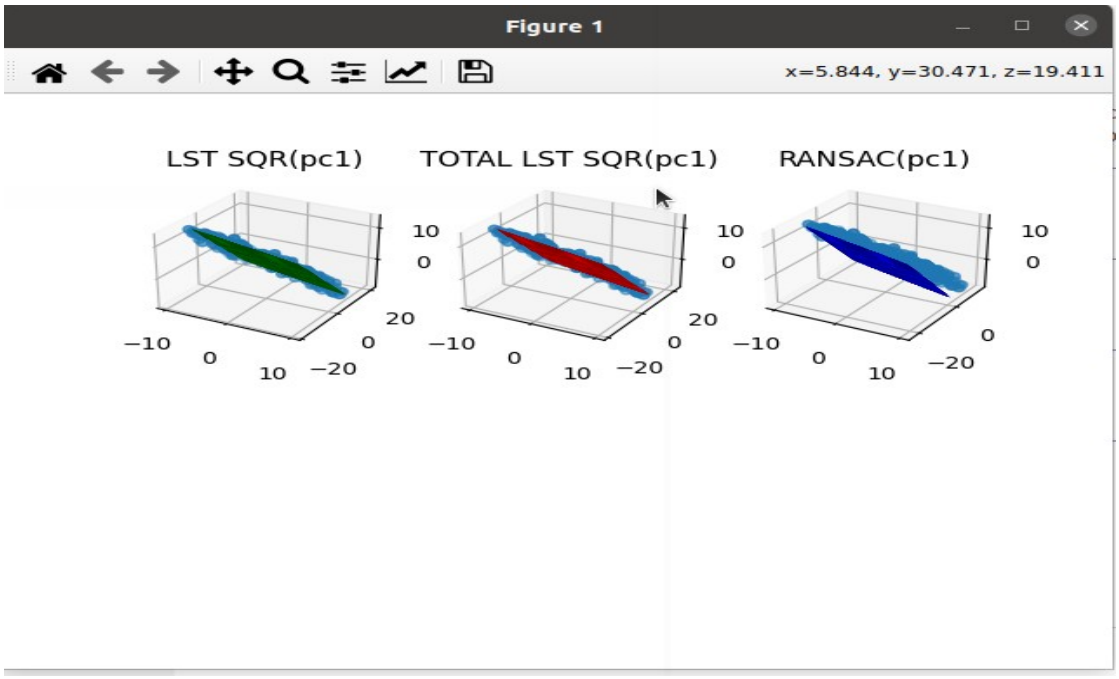
PART 1(b):

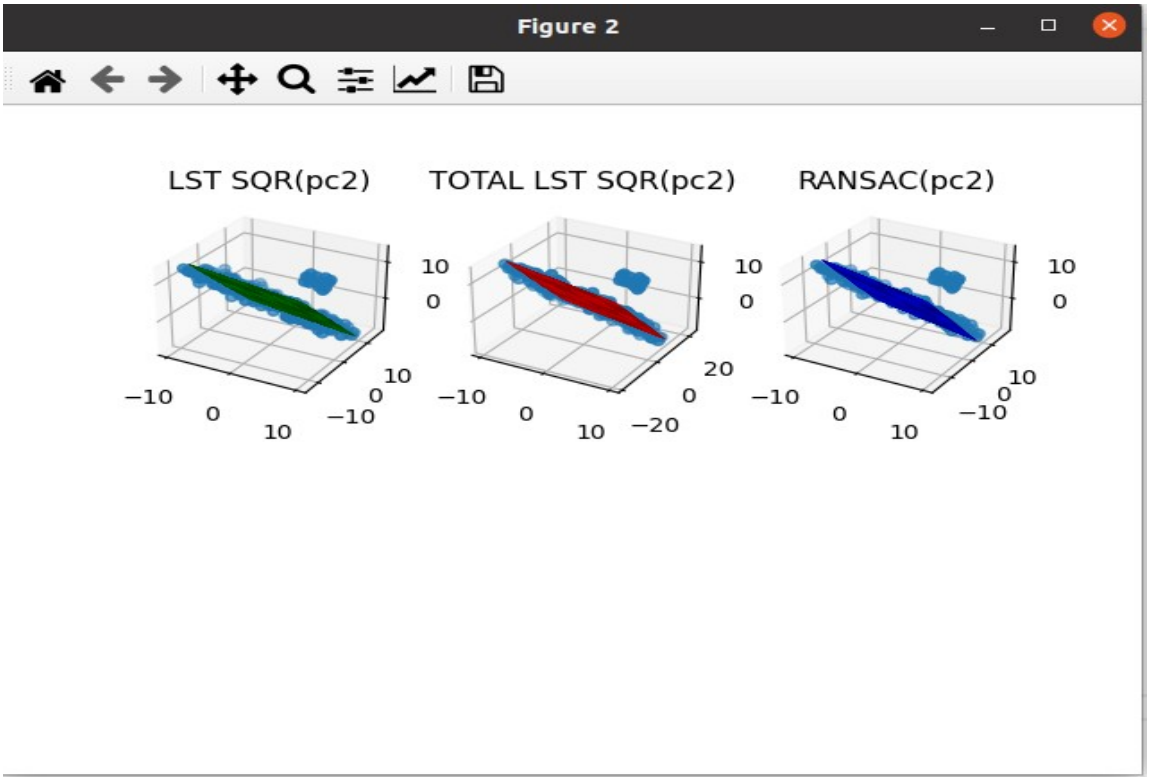
- From the covariance matrix obtained, we then calculate eigen value and eigen vector. The eigen value is the magnitude of the surface normal whereas the eigen vector is the direction of the plane.

PROBLEMS ENCOUNTERED

- *This section of the project seemed doable and hence did not find it difficult when compared to all the other problems.*

PROBLEM 2(PART 2(a)):





INTERPRETATION

From the graphs obtained above for data sets 1 and 2, we can conclude that the least square method fits a bit better than total least square method(although both graphs look analogous to one another). We can see that the plane fits very well with all of the data points along the surface. There are a few offset points which are considered as outlayers in the graph.

PROBLEMS ENCOUNTERED

- *Working with matrices were very confusing and encountered errors like singular matrix value error and inconsistent columns.*
- *Getting to know the syntax was difficult and maximum errors encountered during this section*

PROBLEM 2(PART 2(b)

- For the Ransac function, we are running a limited number of iterations for samples containing data from the data sets given above.
- While running the iteration, we specify a maximum and a minimum threshold limit for the data points.
- If the error function or cost function of the points is within the threshold, we consider that to be inliers and the out of scope points are called outliers.
  These inliers are then used to plot the points to fit best to the given data sets.

INTERPRETATION

The RANSAC function will give different results every time we execute the code because we are using randnt function to obtain random values for the sample and every iteration is encountered with new combinations of samples.

FINAL INTERPRETATION

From the 3 methods: Least square, Total least square and RANSAC, we can conclude that Ransac is the best choice for outlier detection. With the number of iterations increased, we can enhance the quality of detecting outliers and there is scope of improvement with every iterations when compared to least and total least square, which seems to depend on a given set of data.