Homework - 1

ENPM673: Perception for Autonomous Robots

Rishabh Mukund (UID: 117556124)

Problem 1)

Given: f = 25mm (focal length)
w = 14mm (width of sensor)
Res = 5MP (Resolution of sensor)

1. FOV = 2 * ϕ = 2 * $\tan^{-1}\left(\frac{w}{2*f}\right)$

   FOV = 2 * $\tan^{-1}\left(\frac{14}{2*25}\right)$

   FOV = 2 * 15.64° Degrees

   FOV = 31.28° Degrees

As the sensor is a square of width 14mm, both the horizontal and vertical FOV will be 31.28 degrees

2. Given: d = 20m (Distance of object from sensor)
   l = 5cm = 0.05m (length of square object)

   At 20m the sensor will cover an area of,

   area_covered = $\left(2 * tan\left(\frac{FOV}{2}\right) * d\right)^2$

   area_covered = $\left(2 * tan\left(\frac{31.28}{2}\right) * 20\right)^2$

   area_covered = $(0.28 * 40)^2$

   area_covered = 11.2 m^2

   area_covered = 125.44 m^2

   area_of_ball = l^2

   area_of_ball = 0.05 * 0.05

   area_of_ball = 0.0025 m^2

   Min number of pixels occupied by the object at 20m is

   N = $\left(\frac{Res * area\ of\ ball}{area\ covered}\right)$

   N = $\left(\frac{5000000 * 0.0025}{125.44}\right)$

   N = 99.64 pixels

   N = 100 pixels

The Object will occupy 100 pixels

Problem 2)

To solve the question in problem 2, I used openCV to perform necessary actions on the video, numpy and matplotlib libraries.

I made a method, which will open the given video file and extract each frame using opencv.
From each frame, I extracted the red matrix and performed inverse binary thresholding to convert it to a binary image. The thresholding value was set based on trial and error method. I increased the thresholding value if there were white spots seen inside the ball and decreased the thresholding value to get sharp edges.

After thresholding, I searched the 2D matrix for all values of 255 and stored them in a list. So the list has the x and y coordinates of all the pixels occupied by the ball in each frame. Ideally the top most point is the first searched value of the ball and the bottom most point is the last point of the last found value of the ball. But as it is not perfectly thresholded, the edges are not sharp and hence you might get points prone to error.

To minimize this error, I found out the average of x values and min and max of the stored y values. And the
Top = avg(x), min(y)
Bottom = avg(x), max(y)

I stored each of these values and plotted them on the frame in green color to check if the results are satisfactory.

After obtaining all the points of x and y, I performed Original Least Square to generate a second order solution.
I computed the A matrix as $[x^2\ x\ 1]$
As we know AX = y
So the solution, $X = A^{-1} y$
As A is a non square, tall matrix we perform a pseudo inverse of A. i.e $[(A.T * A)^{-1} * A.T]$

So we get the best fit solution X for given points x, y. To calculate the new y points based on the solution, y_new = A * X

I plotted the x and y values, draw a line for the x, A*X values (ideal solution).

Problem 3)

1. To compute the covariance matrix, I made a method which will take the input matrix and find covariance between each row and place them in the matrix appropriately.

   cov_mat(x) = cov(x[0], x[0]), cov(x[0], x[1])
   cov(x[1], x[0]) , cov(x[1], x[1])

   cov(x, y) = (x - x_mean) * (y - y_mean).T / N

2. ***Least Square -*** I used the equations mentioned above, the only change is that A matrix will be [x 1], as we want to fit a line and line has maximum degree as 1.

   Total Least Square - As TLS is scale variant, we have to normalize the data to make sure we have the same scales on both the axes I, I used min-max normalization and then used the normalized data to calculate everything else.

I calculated the x_mean, y_mean, using this I calculated the difference in second order deviation of x and y.

w = $\Sigma(y - y\,mean)^2$ - $\Sigma(x - x\,mean)^2$

Then, I found the covariance between x and y columns as,
r = 2 * (x - x_mean) * (y - y_mean).T

In the equation y - m * x + c,
We know

$$m = \left(w + \sqrt{w^2 + r^2}\right) / r$$

And,

$$c = y\,mean - (m * x\,mean)$$

From this, we get the necessary values for the best solution, and we plot the respected equation.

Ransac - For ransac, I first normalized the data, then made a function to randomly pick 2 points from the data, With these random two points, I calculated the distance between each point in the data and the random line, if the distance was less than a certain threshold value, I added the point to the inliers list of points for that line. I made an inliers list for 2 random points and selected the best fit for the solution which has 99% of the points. I had to fine tune the threshold and the accuracy of the RANSAC.

3. After Implementing OLS, TLS and Ransac on the data, I felt that the **RANSAC was the most efficient** solution as It was less variant to noise (outliers) as compared to OLS and TLS, as in the data, most points were following a generic pattern, and had a few points outside the generic pattern.

   OLS and TLS were very sensitive to these outliers and the solution was not satisfactory, with **ransac**, it was less sensitive towards outliers and we were able to achieve 99% of the points lying within a certain threshold and hence this is the best solution for the given data.

Problem 4)

To compute SVD of a matrix, we have many methods,
The first method I tried was to find the eigenvalues and eigenvectors of (A * A.T) for U and sigma values, and then find eigenvalues and eigenvectors of A.T * A for V matrix.

The first problem I faced was that the eigenvalues and eigenvectors returned by the eig function are not ordered, and I had to first sort the eigenvalues and the corresponding eigenvectors.

The second problem I faced in this method was that the calculations of U and V were independent, and hence the eigenvectors might have different signs and it was difficult to rectify this.

So I decided to use a method in which the calculations for U and V are dependent.
The method I used was.

For a given matrix A.

I found the eigenvalues of A * A.T for the S matrix.

Then I found the eigenvectors of A.T * A, through which I generated the V matrix.

I calculated U with the help of S and V, as:

$Ui = A * Vi / \sigma_i$

In this method of calculation, U and V are dependent so we will get the same signs for corresponding eigenvectors.