Spring 2023

Project 1

Perception for autonomous robots

Februray 22 2023

Student:
Nishant Awdeshkumar Pandey

Instructors:
Dr. Samer Charifa

Course code:
ENPM 673

Contents

1 Question1

1.1 First part

In the given question the video shows the trajectory of a ball thrown by a person. The trajectory of the ball has to be plotted to further compute it's equation. The following method is used to get the desired results:

1. The video is captured and read frame by frame.
2. The color channel is changed from BGR to HSV.
3. As the ball is of red color, thresholding has to be done to filter out red channel by setting the correct upper and lower limit. Multiple iterations had to be made to come to this value
4. Pixel coordinates are extracted and their mean is computed to find the ball's centre of the ball.
5. After finding these coordinates, graph to show the trajectory of the ball is plotted.

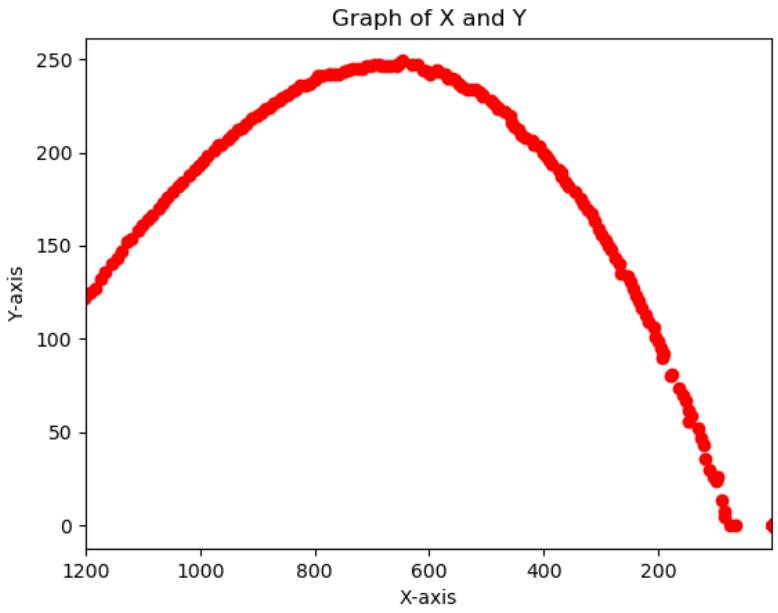

Figure 1: Ball tracking

1.2 Second part

To obtain the best fit curve using the Standard Least Square method. The following steps are used:

1. In linear algebra, the Standard Least Squares method can be represented as a system of linear equations. Let's consider a set of n data points (x1, y1), (x2, y2), ..., (xn, yn). We can represent this data as a matrix equation:
   Xb = y
   where X is an n x 2 matrix, b is a 2 x 1 column vector representing the coefficients of the line (intercept and slope), and y is an n x 1 column vector representing the observed y-values.

2. We can rewrite this equation as:
$$X^T X b = X^T y$$
where X^T is the transpose of the matrix X. This equation is known as the normal equation and it gives us the values of b that minimize the sum of the squared distances between the observed values and the predicted values. To solve for b, we can use matrix algebra to find the inverse of X^T X and multiply it by $X^T y$:
$$b = (X^T X)^{-1} X^T y$$
This gives us the coefficients of the best fitting curve for the given set of data points.

3. To compute the equation of the curve the coefficients of the equation $ax^2 + bx + c$ of the curve are computed and printed on the screen.

4. The obtained equation for this curve is:
$$y = -0.000585x^2 + 0.828943x - 44.530622$$

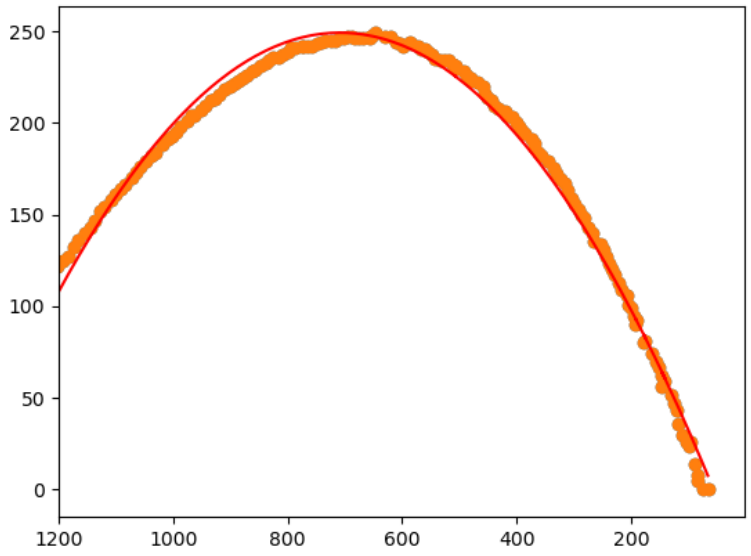

Figure 2: Curve fitting

5. The curve above fits the scatter point well, however the accuracy is a not very good at the peak of the curve, this could be due to some noise.

1.3 Third part

1. To obtain the landing point's x-coordinate for the given video, we need to use the obtained equation. After substituting the sum of the initial y coordinate with 300 and solving the quadratic equation of the landing point's x coordinate is obtained.

2. However to take the origin from the top left corner the height of the frame is substracted from the initial y value.
3. The answer obtained is :

1369

2 Question2

2.1 Part 1

The question asks us to find the covariance for the given data and further use the covariance matrix to compute the magnitude and direction of the surface normal. To execute this the following method is used:

1. The covariance matrix is a 3x3 matrix that represents the variances and covariance of the x, y, and z coordinates.
2. The formula to calculate the covariance matrix for 3D data is:

$$covariance = \frac{1}{(n-1)}(X - \bar{X})(X - \bar{X})^T$$

here n is the number of samples, X is the input data matrix of shape (n, 3), and mean(X) is the mean vector of the data matrix along the columns axis, of shape (1, 3).
The obtained value is as follows:
covariance is:

$$\begin{bmatrix} \begin{bmatrix} 33.7500586 & -0.82513692 & -11.39434956 \end{bmatrix} \\ \begin{bmatrix} -0.82513692 & 35.19218154 & -23.23572298 \end{bmatrix} \\ \begin{bmatrix} -11.39434956 & -23.23572298 & 20.62765365 \end{bmatrix} \end{bmatrix}$$

3. The surface normal is the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. The magnitude of the surface normal is equal to the square root of the smallest eigenvalue.
The obtained value of the direction vector is:

$$\begin{bmatrix} 0.28616428 & 0.53971234 & 0.79172003 \end{bmatrix}$$

Magnitude is:

0.8182357727310519

2.2 Part 2

The question asks to obtain the total least square and Standard Least squares for the given two point clouds. The method to obtain is illustrated below:

1. To find the total least square:
   - Given a 3D matrix A with dimensions m x n x p, we first reshape it into a 2D matrix B with dimensions mp x n by concatenating the p 2D matrices along the rows.
   - Compute the SVD of the matrix B: B = $U \sum V^T$, where U and V are orthogonal matrices and $\sum$ is a diagonal matrix of singular values.
   - Let r be the rank of the matrix B, which can be determined by counting the number of non-zero singular values in $\sum$.

- Let D be a diagonal matrix of size n x n, with D(i,i) = 1 if $i \leq r$, and D(i,i) = 0 if $i < r$.
- Compute the TLS solution by setting the matrix X = VDV^T, where V is the transpose of the V matrix from the SVD. X is the matrix of coefficients that defines the hyperplane that best fits the data points.
- Finally, reshape the matrix X into a 3D matrix of dimensions m x n x p to get the TLS solution for the original 3D matrix A.

2. To find the standard least square method:

- We use $X = ((A^TA)^{-1}A^T)Y$ where A is the design matrix, Y is the target variable, and X is the vector of coefficients that minimize the sum of squared errors between the predicted values and the actual values.
- To apply this equation to a 3D matrix A with dimensions m x n x p, we need to first reshape it into a 2D matrix B with dimensions mp x n, by concatenating the p 2D matrices along the rows.
- Then, we can define the design matrix A as B, and the target variable Y as a vector with dimensions mp x 1, containing the values we want to predict.
- Finally, we can use the matrix equation to calculate the LS solution for the 3D matrix:

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

- After calculating X, we can reshape it into a 3D matrix with dimensions m x n x p to obtain the LS solution for the original 3D matrix A.

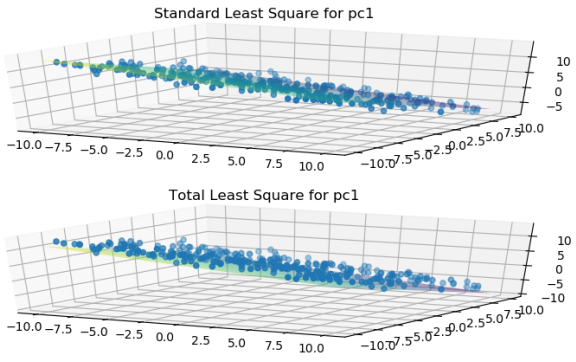

Figure 3: For data set 1

3. To, fit a surface to the data using RANSAC (using total least square):

- Define the mathematical model: First, we need to define a mathematical model that describes the relationship between the dependent and independent variables in the data. For example, in this case we have 3d points that need to be fit in a plane. Eqaution of a plane is

$$ax + by + cz + d = 0$$

where a,b,c are the components of the normal vector which is perpendicular to plane.

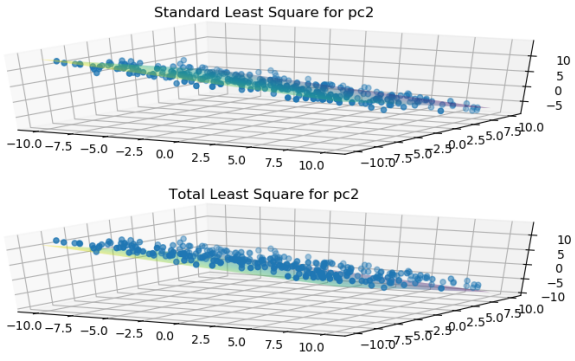

Figure 4: For data set 2

- Randomly sample a subset of data points: Randomly sample a subset of the data points, typically a small fraction of the total data points, and fit the mathematical model to this subset of points using TLS.
- Compute the error for each data point: Once we have fitted the mathematical model to the subset of data points, we can use it to compute the error for each data point. The error is simply the distance between the observed data point and the predicted value of the dependent variable using the mathematical model. In this case to compute the error of the data points from the plane we use formula:
  Distance of plane from point =

$$\frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

  Where x_0, y_0, z_0 are the point from which the distance needs to be computed.
- Identify the inliers: We can then identify the inliers as the data points whose error is less than a pre-defined threshold value (in our case 0.1). The threshold value is typically set based on the noise level of the data and the desired level of confidence in the result.
- Refit the model to the inliers: Once we have identified the inliers, we refit the mathematical model to all of the inliers using TLS. This step is important because the initial subset of points may not have been representative of the entire dataset, and refitting to the inliers helps to ensure that we obtain a more accurate estimate of the parameters of the mathematical model.
- Repeat the process: Finally, we repeat the process for a pre-defined number of iterations or until we have achieved a satisfactory level of confidence in the result. At each iteration(in our case 1100 iterations), we randomly sample a new subset of points, fit the mathematical model to this subset using TLS, identify the inliers, and refit the model to the inliers.

4. The threshold value is selected appropriately so that we get the desired number of in-liners. The number of iterations is calculated using the below formula,

$$N \text{ (No of iterations)} = \frac{log(1-p)}{log(1-(1-e)^s)}$$

   where, e = probability that a point is an outlier (can be estimated visually)
   s = number of points in a sample, p = desired probability that we get a good sample.

5. However, as the number of iterations are very small the above formula is not used and the iterations are directly feeded into the loop.

6. For our implementation the threshold is kept 0.1 (Max distance between points and plane), number of iterations is 1100.

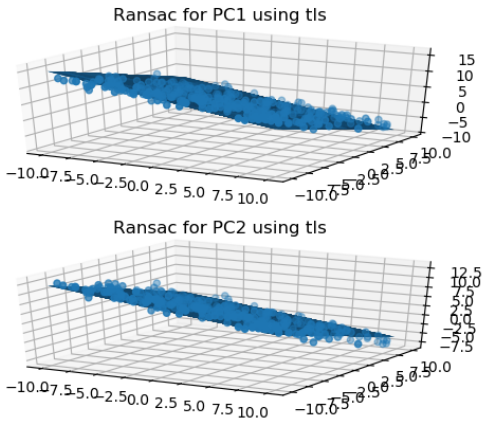

Figure 5: RANSAC using total least square method for point cloud 1 and 2

2.3 Observations and Interpretation

7. After analizing all the obtained results we can infer that:

   - RANSAC method is the better method out of all three fitting methods for noisy data or data with multiple outliers.
   - As the number of iterations increase the accuracy also increases, however, after a certain value the error will start increasing.
   - Total least square method is better than standard least square method to obtain more accurate fitting. TLS takes into account errors in both the dependent and independent variables, so it is better suited for handling outliers than the standard LS algorithm.
   - Increasing the number of iterations for RANSAC will slow down the process and reduce the efficiency thus only limited iterations are computed.

3 Problems Encountered and solutions

Following road blocks were found during the entire process of completing the project:

1. 1.1 - In the first question finding the threshold value for red channel was a bit tough, after multiple iterations I reached an accurate value.

2. 1.2 - To understand various algorithms and convert them in the python code format was bit tough as I am not very good with the language. I had to read thorough the numpy documentations a lot of time.

3. 1.3 - To shift the origin of the start location to the top left corner from the original bottom left corner was a bit tricky I subtracted the height of the frame from the obtained y.
4. 2.1(a)- No problems encountered.
5. 2.1(b)- No problems encountered.
6. 2.2(a)- Righting the SVD function from scratch for calculating the total least square was a bit tricky, I had to sort the eigen values in decreasing order to get the right result.
7. 2.2(b)In the second question executing the RANSAC algorithm was relatively harder than the rest of the methods as it is long and involves various steps. I had to look at various examples to understand it and then execute it. I organised everything using functions, to make things simpler. Also the graph obtained after every run of the code is a bit different, to reduce the chages I had to increase the number of iterations.
8. Taking care of the shape of the various arrays while computations was difficult.