Project-1

Date-2/22/2023

Course: Perception for Autonomous Robots

Course Code: ENPM-673

Student: Rishikesh Jadhav

UID: 119256534

Instructor: Prof.Samer Charifa

1 Ball Tracking

In this problem, a red ball is thrown against a wall, and using the OpenCV library we track the ball and plot the pixel coordinates of the red ball center, and also find a best-fit curve using least squares.
Steps in the Execution:

1. The video is opened using the cv2.VideoCapture function, and a while loop is implemented to read each frame of the video using the read() function.
2. The video's color channel is converted from BGR to HSV using cv2.cvtColor.
3. cv2.inRange is employed to filter the red color channel, where the upper and lower threshold values are specified.
4. After filtering the red channel, the pixel coordinates are obtained and the center of the ball is determined by finding the mean of x and y coordinates.
5. Using the pixel coordinates gathered from each video frame, the best-fit curve for the coordinates is determined via the least squares method.

1.1 Least squares

In the least squares method, we find the best-fit curve by reducing the mean square error. The general equation for a parabola is $y = ax^2+bx+c$. The error is calculated in the vertical direction only and the equation is $E = \sum_{i=1}^{N}(y_i-ax_i{}^2-bx_i-c)$. The error equation in the matrix form is $E = \|Y - XB\|^2$, where

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \; X = \begin{pmatrix} x_1{}^2 & x_1 & 1 \\ x_2{}^2 & x_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n{}^2 & x_n & 1 \end{pmatrix} \; B = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

After differentiating the error equation w.r.t B, equating it to zero, and after solving we get
$B = (X^TX)^{-1}(X^TY)$.
Coefficients a, b, and c of the parabola equation are calculated using the above equation, and the x, and y pixel coordinates of the center of the ball and the curve is plotted using those coefficients.
The equation of the curve is $y = -0.000599089x^2 + 0.777464315x - 4.955751932$
After finding the equation of the curve, the x coordinate of the ball's landing spot is fetched by giving the sum of the first y coordinate and 300 as input and then solving the quadratic equation. There are 2 possible points from the curve equation, since the pixel coordinate cannot be negative, it is neglected and the x coordinate of the ball's landing spot is the positive value which is 1271.548412854277.

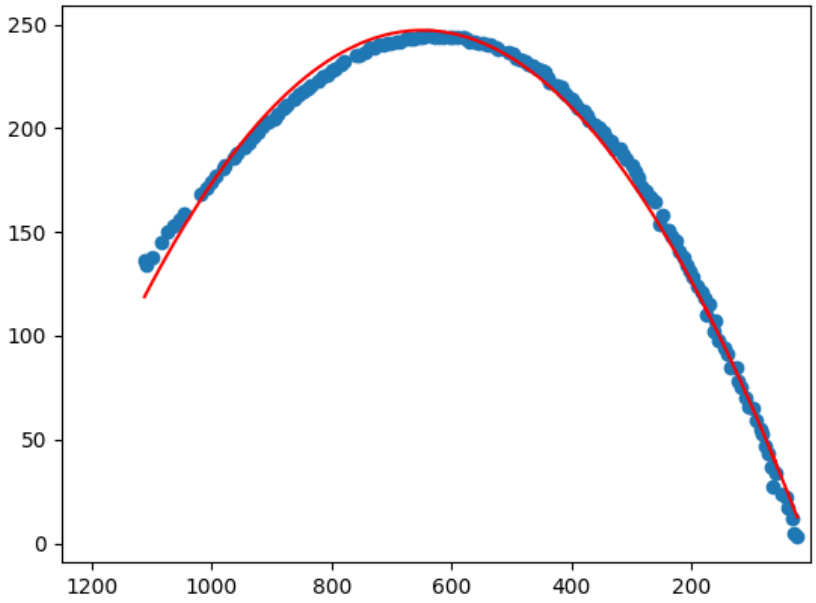

Figure 1: Pixel coordinates and best-fit curve to predict trajectory

2 Covariance matrix, LS, TLS and RANSAC

2.1 Covariance matrix and Surface normal

Covariance is a measure of the joint variability of two random variables. Covariance between two same variables is called variance. The covariance of two variables (x,y) is found by using the below equation -

$cov(x, y) = \frac{1}{n}(\sum(x - \bar{x})(y - \bar{y}))$, where $\bar{x}$ and $\bar{y}$ are the mean values of x and y respectively

The covariance matrix for 3D data is given by: $\begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$

The covariance matrix for the pc1 data set is : $\begin{pmatrix} 33.7500586 & -0.82513692 & -11.39434956 \\ -0.82513692 & 35.19218154 & -23.23572298 \\ -11.39434956 & -23.23572298 & 20.62765365 \end{pmatrix}$

The surface normal's direction and magnitude were determined by analyzing the eigenvalues and eigenvectors of the covariance matrix. Specifically, the eigen vector corresponding to the smallest eigenvalue of the covariance matrix was used to determine the direction of the surface normal. In this case, the direction of the surface normal is (0.28616428, 0.53971234, 0.79172003), while its magnitude is 0.8182357727.

2.2 Standard Least Squares Method for 3D point cloud

As mentioned above in the least squares method, we find the best-fit surface plane by reducing the mean square error. The general equation for a surface is $z = ax + by + c$. The error is calculated as $E = \sum_{i=1}^{N}(z_i - ax_i - by_i - c)^2$.
The error equation in the matrix form is $E = \|Z - XB\|^2$, where

$$Z = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} X = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ {z_n}^2 & y_n & 1 \end{pmatrix} B = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

The error is found only in the z-axis direction in this method. After differentiating the error equation w.r.t B, equating it to zero, and after solving we get,
$B = (X^T X)^{-1}(X^T Y)$
Using the above equation and the x, y, and z points of the data set, coefficients a, b, and c of the surface equation are found and the surface plane is plotted.

2.3 Total Least Squares Method for 3D point cloud

In the standard least squares method, the error is only found in one direction and if the points lie in the same direction as the error-calculated direction, then the least squares might not produce good results. Whereas in the total least squares method, the error is found orthogonally to the plane, and the error is minimized to find the best-fit plane.
The equation of the plane in the total least square method is $d = ax + by + cz$ and the squared error function is given as $E = \sum_{i=1}^{N}(ax_i + by_i + cz_i - d)^2$
Coefficients a, b, and c are calculated by minimizing the error. The error function is differentiated w.r.t 'd'
$\frac{\partial f}{\partial x} = \sum_{i=1}^{N} -2(ax_i + by_i + cz_i - d)$
d is calculated using the formula $d = a(meanofx) + b(meanofy) + c(meanofz)$
Substituting the d in error equation -
$E = \sum_{i=1}^{N}(a(x_i - \bar{x}) + b(y_i - \bar{y}) + c(z_i - \bar{z}))^2$
The error equation in matrix form is $E = (UN)^T(UN)$,

where $U = \begin{pmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ \vdots & \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} & z_n - \bar{z} \end{pmatrix} N = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$

Differentiating the error equation w.r.t N and equating it to zero, and after solving the equation we get $2(U^T U)N = 0$.
The solution of this equation is given by the eigen vector corresponding to the smallest eigen value of the U matrix.
The three coordinates of the smallest eigen vector are (a, b, c) and d is calculated using a, b, c, and mean values of x, y, z.
After finding the coefficients a, b, c, and d of the equation the surface is plotted.

2.4 RANSAC Method

Random sample consensus or RANSAC is an iterative method to find the best model which contains outliers. If there are more outliers in the data, the least squares method might not produce the best model. Whereas, the RANSAC method estimates the best model with fewer outliers, by selecting random subsets and calculating the number of in-liners and outliers. The steps involved in RANSAC are:

1. The first step is to select a subset of random points from the given data points. For finding the best-fit surface with less number of outliers, the min no of sample points to be given are three.
2. To fit a model to the subset, the standard least squares method is used and a model is hypothesized.
3. Then after finding the model, the error is computed and if the error value of each point is less than the threshold value, it is labeled as an in-liner and if the error is more than the threshold it is labeled as an outlier.
4. The process is repeated for the prescribed number of iterations until the desired count of in-liners is achieved.
5. After finding the desired model, the surface is plotted.

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

3 Observations and Interpretation of results

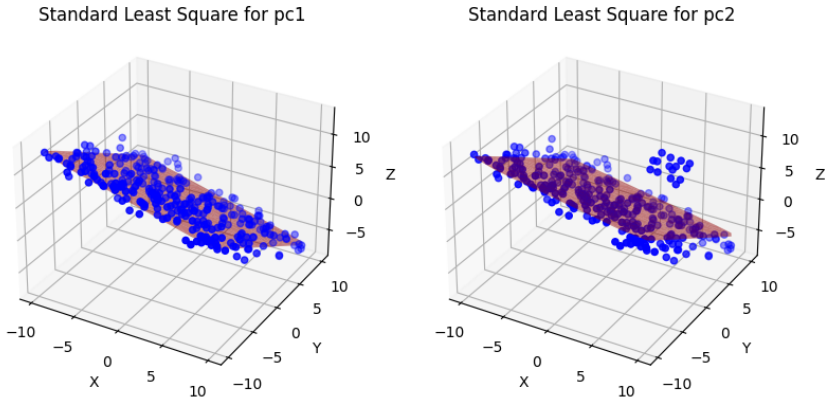

Figure 2: Standard Least Square Fit

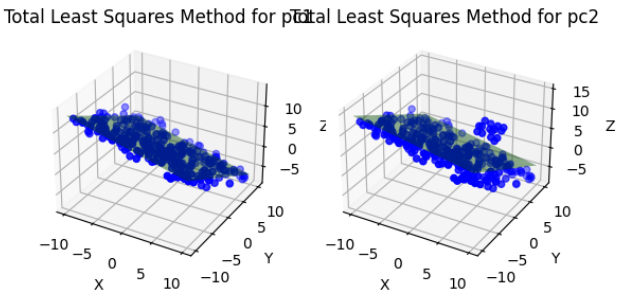

Figure 3: Total Least Square Fit

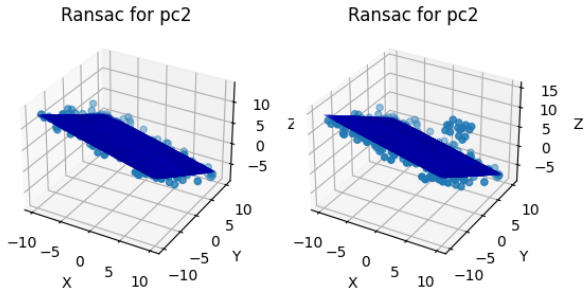

Figure 4: RANSAC fit

1. When working with data that contains noise, the total least squares method typically outperforms the least squares method. Although the plots generated by both methods can look similar and be difficult to distinguish, the RANSAC method is generally considered to produce the best model when dealing with noisy data.

2. Upon examining the plots for the pc2 dataset, it is evident that the RANSAC method generated the most accurate model. It's worth noting that RANSAC produces different sets of values for a, b, and c with each run, but the variation between these sets is typically small. By adjusting the probability values and increasing the number of iterations, we can obtain the desired inlier values without significantly slowing down the program.

3. While all methods produced similar results for the pc1 dataset, there was a noticeable difference in performance for the second dataset, with RANSAC outperforming the other methods. In general, RANSAC is the preferred choice for outlier rejection and produces the most accurate models in such cases.

4 Problems Encountered

1. One of the primary challenges encountered during ball tracking is determining the upper and lower threshold limits for the mask used to filter the red channel. After multiple iterations and inputs, the correct limits were ultimately discovered to successfully extract the red channel.

2. During ball tracking, there were instances in which the red channel was not filtered in certain frames, resulting in zero pixel coordinates and causing an error when attempting to calculate the mean. This issue was addressed by implementing an if condition to leave empty sets.

3. In the total least squares method, a problem arose while assigning the eigen vector corresponding to the smallest eigenvalue to the variables a, b, and c. The array that stored the values caused issues during the calculation of new z values. A minor adjustment was required to convert the 1D array to float.

4. An error was encountered during RANSAC when calculating the number of samples, as the probability values given as input resulted in a denominator of zero. The issue was resolved by changing the probability values.

5. The question, which involved implementing the RANSAC algorithm, proved to be more challenging than the other methods due to its length and multi-step nature. To fully comprehend and execute it, I had to reference multiple examples. In order to simplify the process, I organized the code into functions. Additionally, the resulting graph after each run of the algorithm varied slightly, so I increased the number of iterations to reduce these fluctuations.