ENPM673 Project 1

Vineet Kumar Singh (ID: vsingh03)

February 2023

Contents

1 Problem 1

1.1 Trajectory Plotting

In this section, the trajectory of the object is tracked from the given video. To achieve this, the approach taken is as described below:

1. The video is read frame by frame and each frame is first changed to BGR to HSV and then masked to filter the red channel. This is done in function *filter()*.
2. HSV image is then masked with Lower limit of [0,100,100] and upper limit of [3,250,250] of HSV values which was chosen after multiple iterations on the HSV colour scale.
3. The masked image is then dilated to highlight the features, converted to grayscale and finally thresholded to give a binary image.
4. The binary image is then passed to the function *locate_center* which uses *np.where()* to locate while points in the image.
5. The set of located points is separated into x and y coordinate and filtered for the ball using the following logic $(max(x) - min(x) < 100) and (max(y) - min(y) < 150)$
6. The complete set of center coordinates is then processed to remove outliers. The distance of each point is checked with respect to its neighbours, and if it exceeds certain value then the point is removed from the final set.
7. This is implemented as: $((abs(pts[i-1][0] - pts[i][0]) > 20) or abs(pts[i-1][1] - pts[i][1]) > 15)$.
8. Once the final set of points is available, the data is plotted using matplotlib and further processed in the next section for curve fitting.

The final trajectory plot is as shown in Fig 1.

1.2 Curve fitting using Least Squares

In this section, a parabola equation is fitted to the plotted trajectory. Least squares method is used for the curve fitting.

The equation of parabola is given as:

$$y = ax^2 + bx + c$$

The error function for above equation is given as:

$$E = \Sigma_n (y - ax^2 - bx - c)^2$$

To write above equation in vector form, let $X = [x^2, x, 1]$, $Y = y$, and $D = [a, b, c]^T$ then

$$E = \|Y - XD\|$$

. On expanding and solving we get:

$$E = Y^T Y - 2(XD)^T Y + (XD)^T (XD)$$

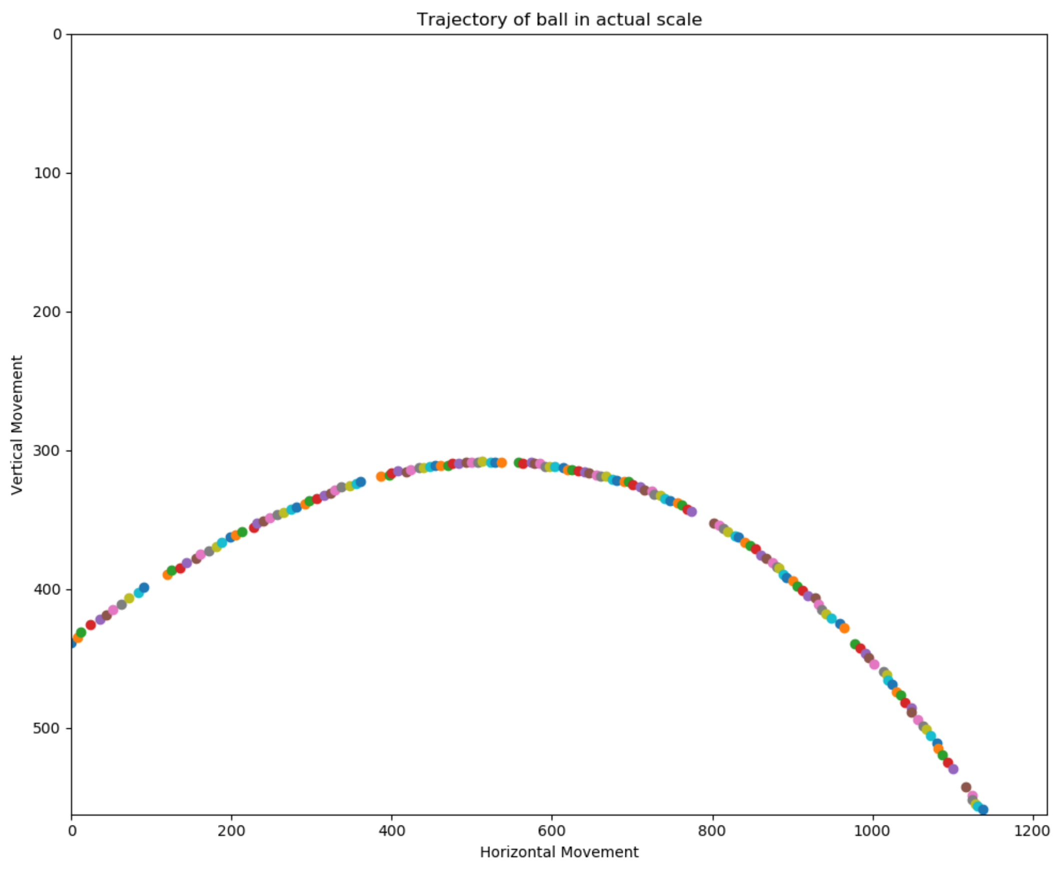


Figure 1: Scatter plot of tracked trajectory

Differentiating w.r.t. D we get:

$$\frac{\partial E}{\partial D} = -2X^T Y + 2X^T X D$$

As, $\frac{\partial E}{\partial D} = 0$

$$\therefore D = (X^T X)_{-1}(X^T Y)$$

The final parabola equation after LS curve fitting is:

$$y = [0.00059506] * x^2 + [-0.59374862] * x + [452.82377122]$$

The curve plot along with trajectory is as shown in Fig 2.

1.3 Object pixel calculation

The initial and final position of the center coordinates are known. The difference in y coordinates is equal to 300 pixels as given. So to calculate the equivalent x pixel we use below formula. $pixel\_x\_final = 300 * (x[-1] - x[0])/(y[-1] - y[0])$
Starting point of ball in video frame [0. 438.5]
Final point of ball in video frame [1138. 558.5]
So the X pixel of landing position of ball is approximately: **2845 pixel**

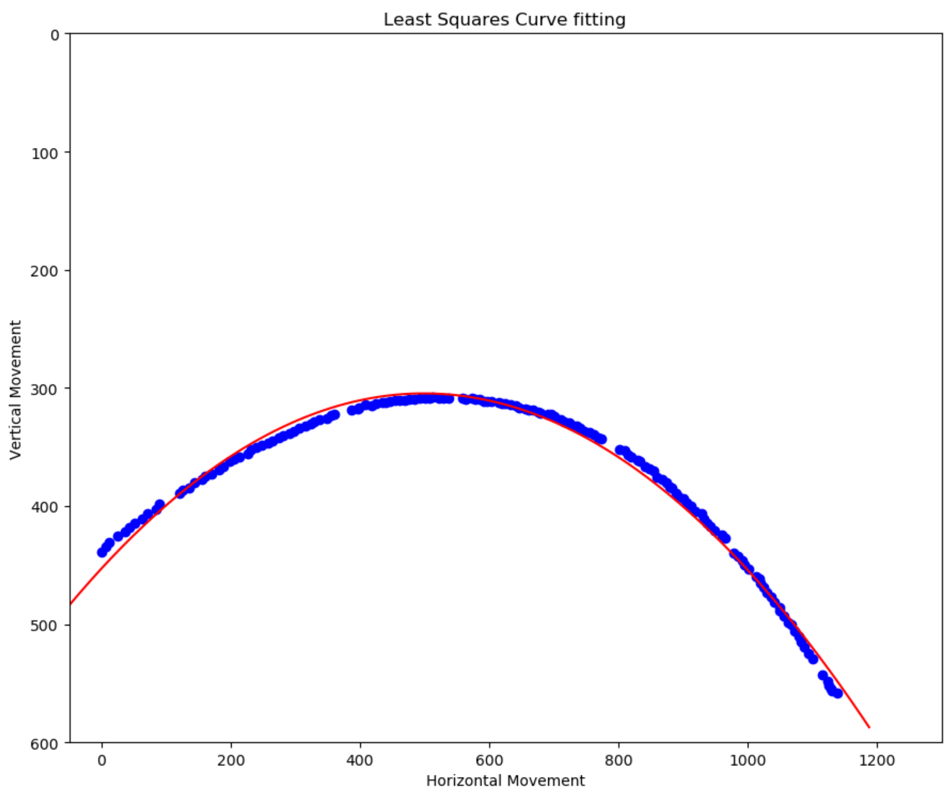


Figure 2: Parabola fit for the trajectory using LS

1.4 Problems Faced

1. The major challanges faced in this section was to filter out the ball properly from the given video. The hand in the video is has a high content of red channel content which makes it difficult to filter out. Also as the ball travels, its colour as detected by the camera changes due to changing angle of light reflection. So having the proper threshold to filter out the ball was difficult. I used the HSV colour chart and then manually tried to figure out the range of colours that the ball should be in.

2. The second challenge was the rejection of noise from the binary image, as there is a significant amount of white spots throughout the video. To address this, I calculated the maximum and minimum pixels that was being detected in the frame. Then I checked for every frame if the detected pixels are in the neighbourhood of each other. If not, then the particular frame is rejected from the final trajectory points, and this helped to remove noise. There are gaps in Figure 1 which corresponds to points which were rejected because of noise.

3. For calculation of the LS fit, the problem faced in representing entire equations as vectors so that processing can be faster as compared to using two for loops for calculation.

4. For calculation of the final x pixel of the ball, the problem faced was that the final and initial frames were not detected properly, so fine tuning of the parameters in video processing was done to ensure that the first and last points were proper.

2 Problem 2

2.1 Covariance matrix calculation

The dataset pc1.csv along with surface normal is plotted as shown in Fig 3.

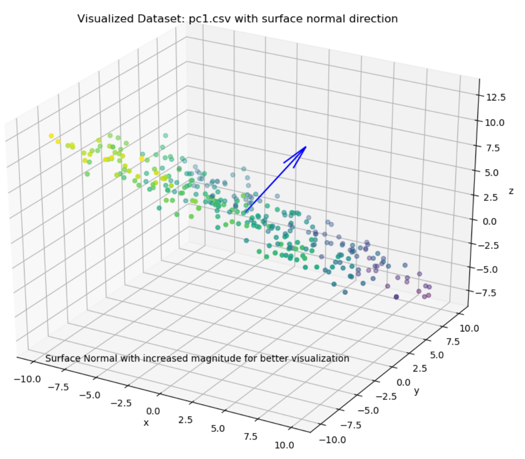


Figure 3: 3D data pc1.csv

Covariance matrix is given as:

$$\begin{bmatrix} Var(x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Var(y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Var(z) \end{bmatrix}$$

where

$$Var(x) = \frac{1}{n}\Sigma(x_i - \bar{x})(x_i - \bar{x})^T$$

$$Cov(x,y) = \frac{1}{n}\Sigma(x_i - \bar{x})(y_i - \bar{y})$$

Result of Covariance matrix is:

$$\begin{bmatrix} 33.6375584 & -0.82238647 & -11.3563684 \\ -0.82238647 & 35.07487427 & -23.15827057 \\ -11.3563684 & -23.15827057 & 20.5588948 \end{bmatrix}$$

To find the surface normal to the dataset, the eigen values of the dataset is calculated. The eigen vector corresponding to the smallest eigen value is the vector normal to the surface of the plot.

The eigen values for the dataset is:

Eig Values: $\begin{bmatrix} 0.66727808 \\ 34.54205318 \\ 54.06199622 \end{bmatrix}$

The eigen vector corresponding to the smallest eigen value is:

Smallest Eigen Vector: $\begin{bmatrix} 0.28616428 \\ 0.53971234 \\ 0.79172003 \end{bmatrix}$

The magnitude of the smallest eigen vector is:

Magnitude = 1.0

Direction of surface normal is along vector :$73.37154447\hat{x} + 57.33594138\hat{y} + 37.65345767\hat{z}$

2.2 RANSAC, TLS and LS plane fitting

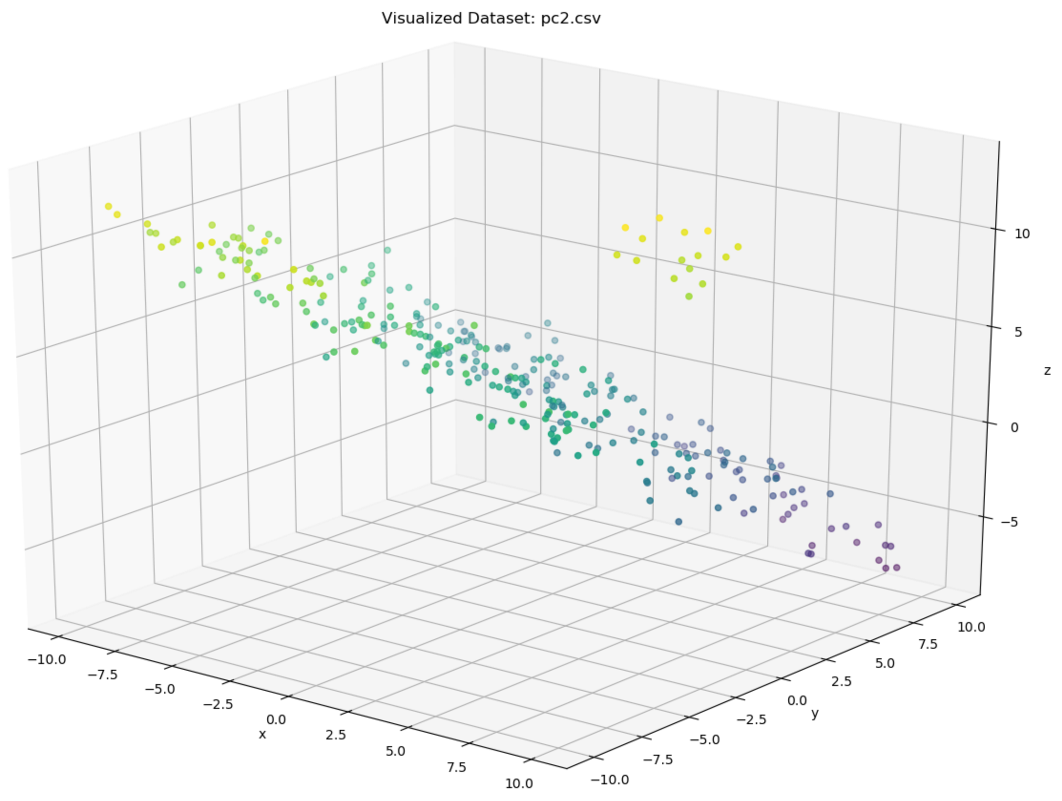


Figure 4: 3D data pc2.csv

Visualized dataset pc1.csv is already shown in Fig 3 and pc2.csv is shown in Fig 4. This section tries to fit a plane to the datasets using Least Squares, Total Least Squares, and RANSAC method.

Equation of Least Squares for planes:

Analyzing the data, we see that a plane can fit the data. So taking equation of plane as:

$$z = ax + by + c$$

Error is given as:

$$E = ax_i + by_i + c - z_i$$

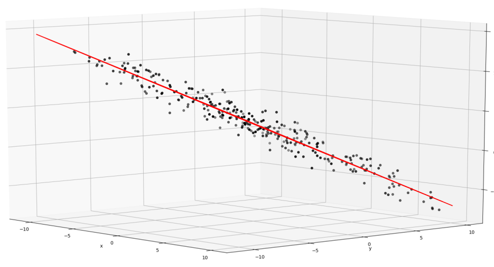
Let $X = [x, y, 1], D = [a, b, c]^T, Z = z$
we get

$$E = \|Z - XD\|^2$$

Simplifying as in section 1 we get:

$$D = (X^T X)^{-1}(X^T Z)$$

The obtained curves for pc1.csv and pc2.csv using LS plane fitting are as given in Figure 5:
Problems Faced: The implementation of Equation of plane in vector form was a challenge.


(a) LS fit pc1.csv

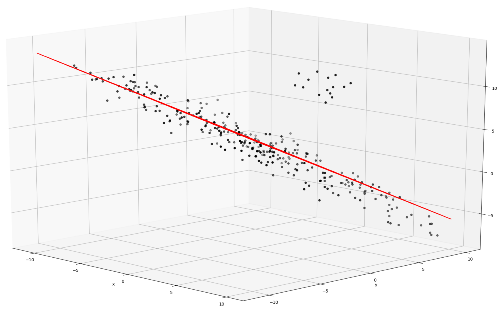

(b) LS fit pc2.csv

Figure 5: Least Squares Plane fit

Potting the curve using calculated coefficients and in the same plot as the dataset was a challenge. To address this, a single object of pyplot was passed as arguments to all the plots so that all the planes will be plotted on the same plot.

Method of Total Least Squares:

The Equation of the plane can be written as:

$$ax + by + cz = d$$

The error function can be written as:

$$E = \Sigma_n(ax + by + cz - d)$$

Since to minimize the Error $= 0$

$$2\Sigma_n(ax + by + cz - d) = 0$$

$$nd = \Sigma_n(ax + by + cz$$

$$d = \frac{\Sigma_n(ax + by + cz)}{n}$$

$$d = a\bar{x} + b\bar{y} + c\bar{z}$$

where $\bar{x} + \bar{y} + \bar{z}$ represent the mean over n data points. Substituting this value of d, we get the error function as:

$$E = \Sigma_n(a(x - \bar{x}) + b(y - \bar{y}) + c(z - \bar{z}))^2$$

Let $U = [(x - \bar{x}), (y - \bar{y}), (z - \bar{z})]$ and $N = [a, b, c]$, then

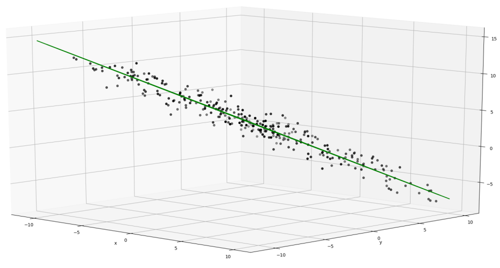
$$E = (UN)^T(UN)$$

$$= 2U^TUN$$

Therefore,

$$U^TUN = 0$$

To solve this equation, SVD can be used as this is a homogeneous equation. Let $U^TU = A$, Then above equation becomes $AN = 0$. The obtained curves for pc1.csv and pc2.csv using TLS plane fitting are as given in Figure 6


(a) TLS fit pc1.csv

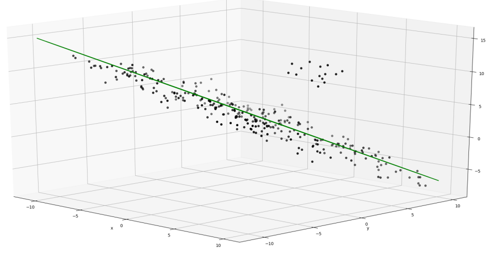

(b) TLS fit pc2.csv

Figure 6: Total Least Squares Plane fit

Problems faced in implementing TLS was the calculation of Eigen Values and Vectors, and properly storing the values in the matrix form for processing. Also parsing the smallest eigen vector from the matrix was a problem. This was solved using matrix slicing techinque and using a separate variable to first store the index of smallest eigen value and then using it to extract corresponding eigen vectors.

Method of Random Sample Consensus(RANSAC)

This method uses a combination of Random Sampling and a curve fit method to fit the randomly selected samples. In this report, the method of Total Least Squares method is used for plane fitting. The steps used are listed as below:

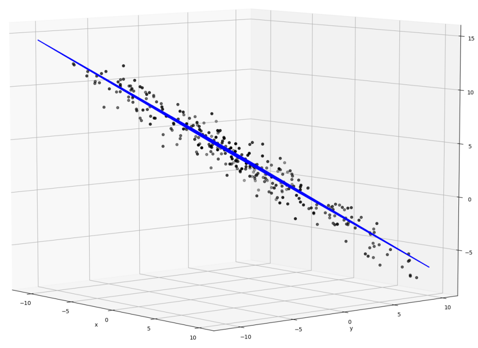
1. Minimum point selection to represent the plane. 3 points for the given datasets.
2. Fit the plane using TLS for the randomly selected 3 points.
3. Calculated the error of all the datapoints w.r.t. the obtained plane using error function of TLS given as:
$$E = \Sigma_n(ax + by + cz - d)$$
4. Choose a threshold so that probability for inlier is p (e.g. 90
5. Take a count of points within the threshold value.
6. This is continued for N iterations where N is given as:
$$N = \frac{log(1-p)}{log(1-(1-e)^s)}$$

where p is the desired accuracy, e is the outlier ratio for the dataset counted by visual inspection, and s is the minimum number of data to fit the curve.
For pc1.csv parameters are: $p = 0.99$, $t = 0.95$, $s = 3$, *outliers* $= 1$
For pc2.csv parameters are: $p = 0.99$, $t = 0.95$, $s = 3$, *outliers* $= 15$

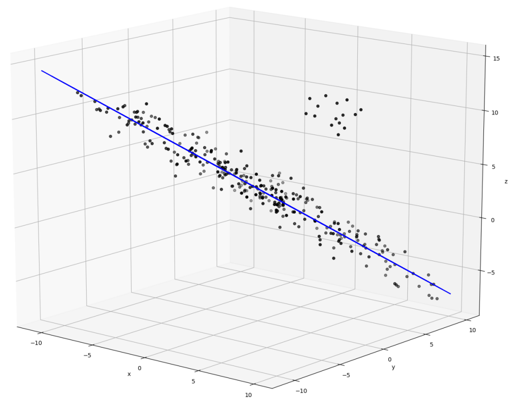
The final graphs for the RANSAC plane fit is shown in Figure 7.

The comparison of all the Plane fitting methods is shown in Figure 8 for the datasets pc1.csv and pc2.csv

Problems Faced: Selection of the proper values for threshold, and probability was a challenge. This was addressed by doing multiple iterations for these two values. Also for the dataset pc1.csv, as there are no major outliers, so taking the number of outliers = 0 gave a division y zero error during the calculation of N. So a value of 1 was chosen just to avoid this error.

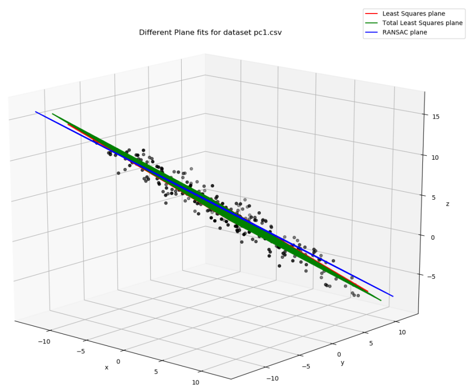


(a) RANSAC fit pc1.csv

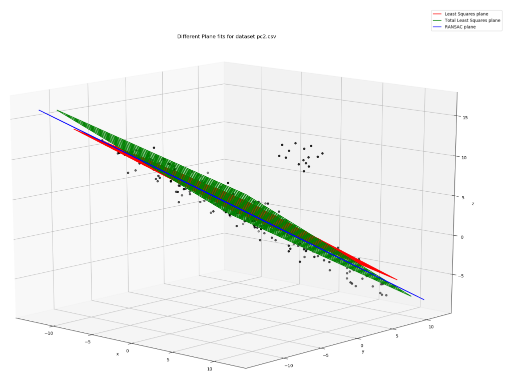


(b) RANSAC fit pc2.csv

Figure 7: Total Least Squares Plane fit



(a) Comparison pc1.csv



(b) Comparison pc2.csv

Figure 8: Total Least Squares Plane fit

3 Results and Conclusions

Problem 1:

The trajectory of centre of ball in the video is shown in Figure 1.
The final parabola equation after LS curve fitting is:

$$y = [0.00059506]x^2 + [-0.59374862]x + [452.82377122]$$

The X pixel of landing position of ball is approximately: **2845.0 pixel**

Problem 2:

The Covariance matrix for given dataset is:

$$\begin{bmatrix} 33.6375584 & -0.82238647 & -11.3563684 \\ -0.82238647 & 35.07487427 & -23.15827057 \\ -11.3563684 & -23.15827057 & 20.5588948 \end{bmatrix}$$

The magnitude of surface normal is: 1.0
The direction of the surface normal is : $73.37154447\hat{x} + 57.33594138\hat{y} + 37.65345767\hat{z}$

The Equations of plane fits for dataset pc1.csv are:
LS Plane fit: $z = -0.353955x + -0.668551y + 3.202554$
TLS Plane fit: $z = -0.361446x + -0.681696y + 3.201213$
RANSAC Plane fit: $z = -0.382434x + -0.726258y + 2.797926$

The Equations of plane fits for dataset pc2.csv are:
LS Plane fit: $z = -0.251884x + -0.671737y + 3.660257$
TLS Plane fit: $z = -0.283967x + -0.754501y + 3.660076$
RANSAC Plane fit: $z = -0.317876x + -0.658338y + 3.525898$

Conclusions for plane fitting methods:
Least Squares plane fitting performed well for the pc1.csv dataset which is noiseless data. But the performance on dataset with outliers was not good. The output depends on the distribution of data and presence of outliers affected the result.

Total Least Squares plane fitting performed well for the pc1.csv dataset which is noiseless data. But the performance on dataset with outliers was not good. The error for outliers was much more and it affected the final output. The final plane was biased towards noisy data.

RANSAC output performs good for data with outliers, and almost similar in performance (not as good) as LS and TLS for noiseless data. The comparison is as shown in Figure 8.

From the method of implementation, it can be concluded that RANSAC is a better method of outlier rejection. The process of sampling ensures that the outliers in the dataset are not included while fitting a curve to the dataset.