Curve Fitting

A. JAMES CLARK
SCHOOL OF ENGINEERING

Pooja Kabra

February 2020

Table of Contents

A ball is thrown against a white background and a camera sensor is used to track its trajectory. We have a near perfect sensor tracking the ball in video1 and the second sensor is faulty and tracks the ball as shown in video2. Clearly, there is no noise added to the first video whereas there is significant noise in video 2. Assuming that the trajectory of the ball follows the equation of a parabola:

- We will use Standard Least Squares, TLS and RANSAC methods to fit curves to the given videos in each case and plot the data and best fit curve for each case.
- Briefly discuss which would be a better choice of outlier rejection technique for each case.

Preprocessing:

We first resize our video using *imutils.resize* to a width of 500 pixels in order to reduce processing time. The height gets scaled automatically to around 393 pixels.
For every frame, we extract the coordinates of non-white pixels (which will fetch us red pixels)

For this set of coordinates, we find lowermost and uppermost y coordinates to find lowest and highest pixel on the blob.

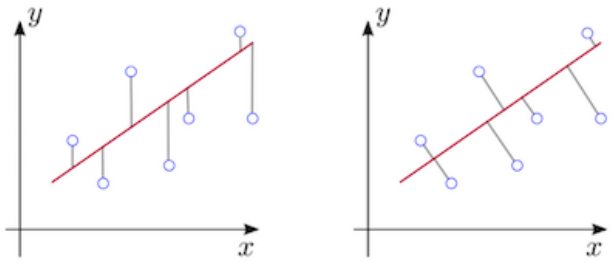

Figure 1: Visual comparison between SLS (left) and TLS (right)

Blue points are our data points and red line is the regression line.

Standard Least Squares:

SLS minimizes the sum of the distances squared only in the y direction. It assumes no error in the x direction i. e. it expects all data to be sampled exactly. Geometrically, this is the *parallel* distance to the regression line.

Consider video2. We have 24 datapoints.
y is a column vector of all y co-ordinates [24 x 1]
b is the column vector of curve parameters we are trying to solve for [3 x 1]
a is a [24 x 3] matrix. Its 1st, 2nd and 3rd columns are all x co-ordinates raised to 0, 1 and 2 powers respectively.

ab = y

$S(b) = \sum_{i=1}^{24} |y_i - \sum_{j=1}^{3} a_{ij}\, b_j|^2$

This quadratic minimization is solved by:

$$\hat{b} = (a^T a)^{-1} a^T y$$

TLS:

Unlike SLS, TLS considers observational errors and minimizes errors for independent variable and dependent variable. Geometrically, this is the *perpendicular* distance to the regression line.

$$S(b) = \frac{\sum_{i=1}^{24} |y_i - \sum_{j=1}^{3} a_{ij} b_j|^2}{\sum_{j=1}^{3} |b_j|^2 + 1}$$

This quadratic minimization is solved by:

$$\hat{b} = (a^T a - \sigma^2 I)^{-1} a^T y$$

Where σ is the least of the singular values of [a y] and I is an identity matrix (3 x 3 for the above case).

RANSAC (Random Sample Consensus Algorithm):

When our dataset has many outliers, RANSAC delivers robust results.

The algorithm is simple:

1. We select *s* datapoints at random from our entire dataset and use it to estimate a curve Here, *s* is the minimum number needed to fit our model.
   s = 3 for our case as we are trying to estimate a second-degree curve. *(SLS has been used for this step. TLS can be used too.)*

2. Next, we count the number of all the datapoints that lie within a threshold *th* from the curve. This count *inliers_iter* will be appended to the list *inliers*. We also store parameters *b* for the curve to the list *all_b*.

3. We repeat steps 1 and 2 again and again for *t* trials.

4. Our final curve is that one out of all trials, which gave maximum inliers. We run argmax on the *inliers* list and use the obtained index to find curve parameters from *all_b*.

The number of trials t can be obtained as:

$$T = \frac{log\ (1-p)}{\log\ (1-(1-e)^s)}$$

Here,
s= samples = 3 for our case
p= desired probability that we get a good sample = 0.999, this is something we get to decide.

e= outlier ratio e = 25% by simply observing the dataset, we can see that around 6/24 points are outliers in video2.

Plots:

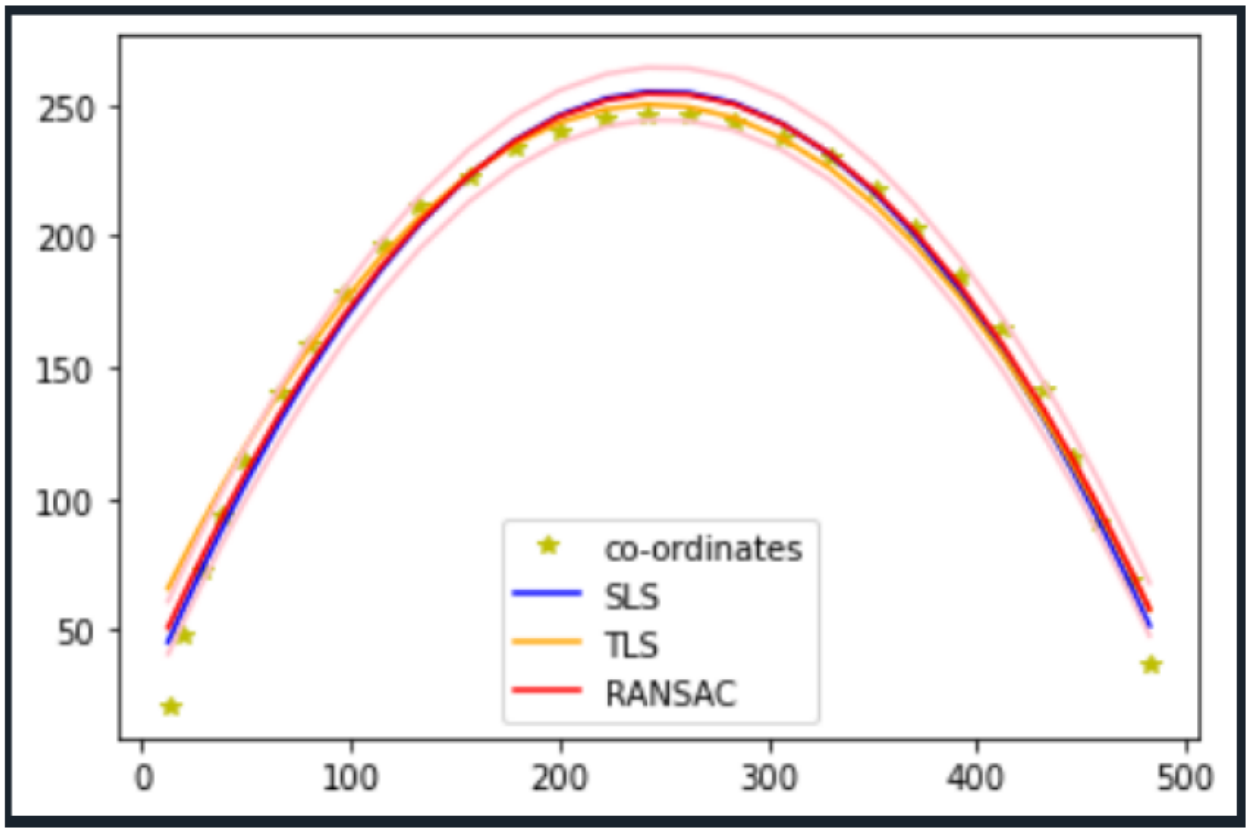

Figure 2: Clean Video

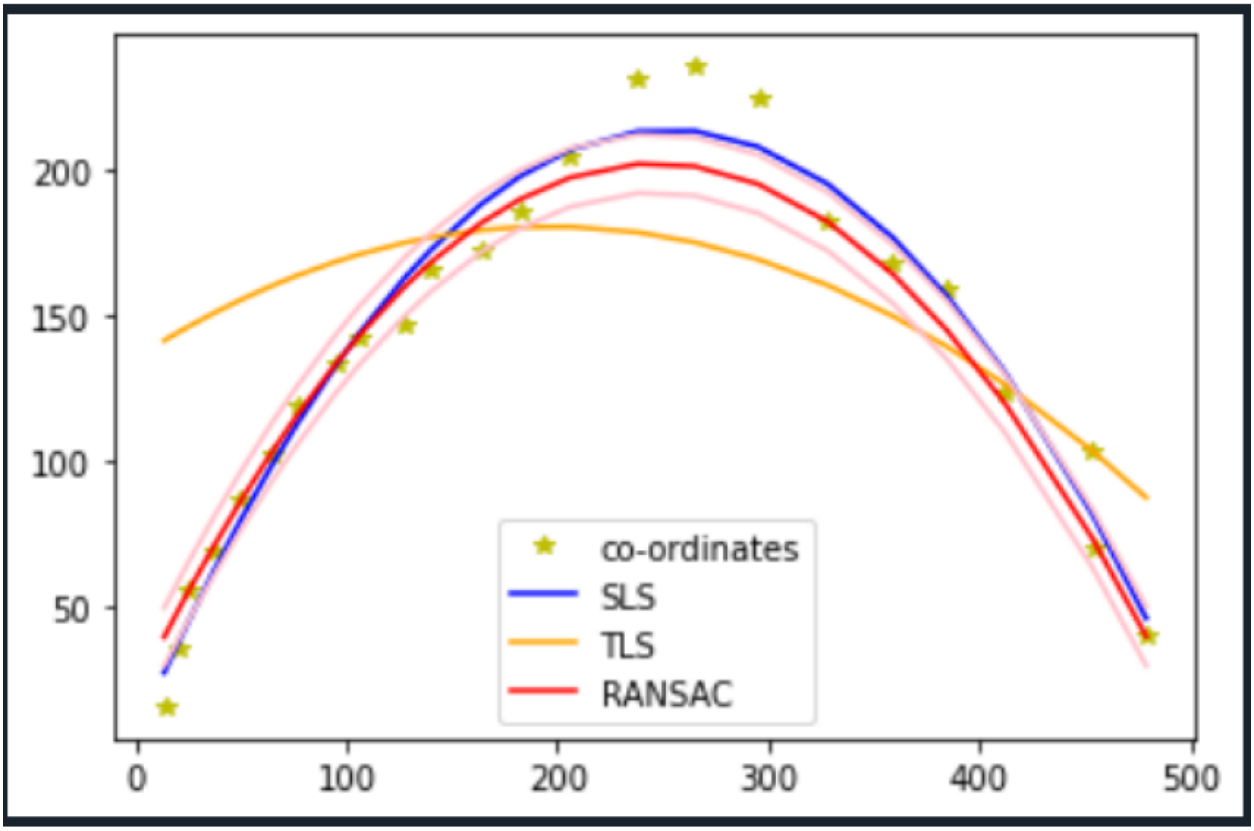

Figure 3: Noisy Video

References:

1. Total Least Squares in comparison with OLS and ODR
https://towardsdatascience.com/total-least-squares-in-comparison-with-ols-and-odr-f050ffc1a86a

2. Real Statistics using Excel
https://www.real-statistics.com/regression/total-least-squares/