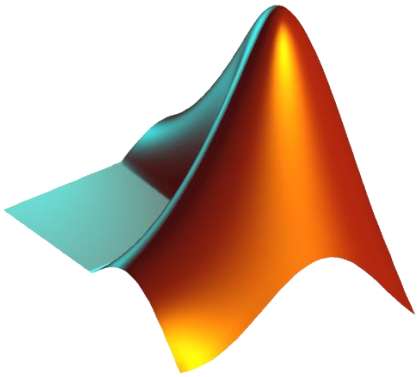
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Electrical and Electronic Engineering



EEE 212 Project

SIMPLIFIED CURVE FITTING

Submitted By,

Md Liton Ali
ID: 1906080

Anindya Kishore Choudhury
ID: 1906081

Submitted To,

Mr. Hamidur Rahman
Associate Professor,
Department of Electrical and Electronic Engineering
Bangladesh University of Engineering Technology.

Shahed Ahmed
Lecturer,
Department of Electrical and Electronic Engineering.
Bangladesh University of Engineering Technology

Shaimur Salehin Akash
Lecturer (PT),
Department of Electrical and Electronics Engineering
Bangladesh University of Engineering Technology

INDEX

List of Figures:

Introduction:

In our daily life, we have several mathematical figures that equations we need to know for various using purposes. Our project problem was to write a code that will segment a particular curve and find the easy equation for that portion. And we had to create a simple interactive GUI design with MATLAB app designer.

Theory:

1. Image Read:

To read the image, the theory is simple. We have taken an image of 256 RGB format image which has three-dimensional pixel data. Then we have converted that image to grayscale making the three-dimensional pixel data a single-dimensional one. Then using the image read intensity slider in the GUI, we have adjusted the intensity of pixels where the unwanted pixels will be made white by setting the value 1, and the desired pixels will be set to 0 which will constitute the black points.

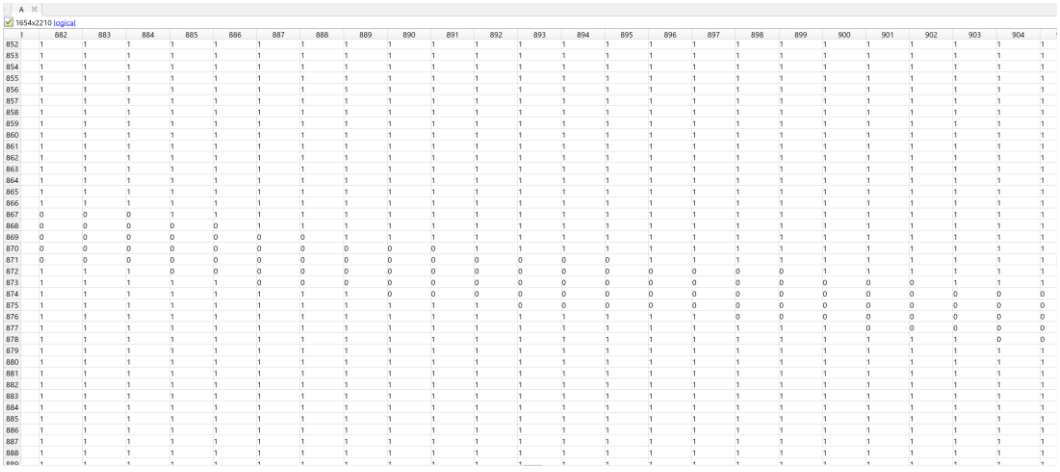


Figure 1: O corresponding to the position of the black points while 1 is the unwanted white background pixel

2. Smoothing:

As the curves drawn in hand, the thickness of the pencil or pen often results in a picture where the pixels are denser in the middle and lighter and faded out on both the sides. Also, the thickness of the curve results in pixels along the vertical or horizontal axis which is unnecessary. So, smoothing is necessary to remove the pixels which is scanned as noise on both the sides and redundant in constituting the graph. So, the among all the pixels, smoothing is basically taking just one pixel in the average region to rebuild the curve for the further analysis.

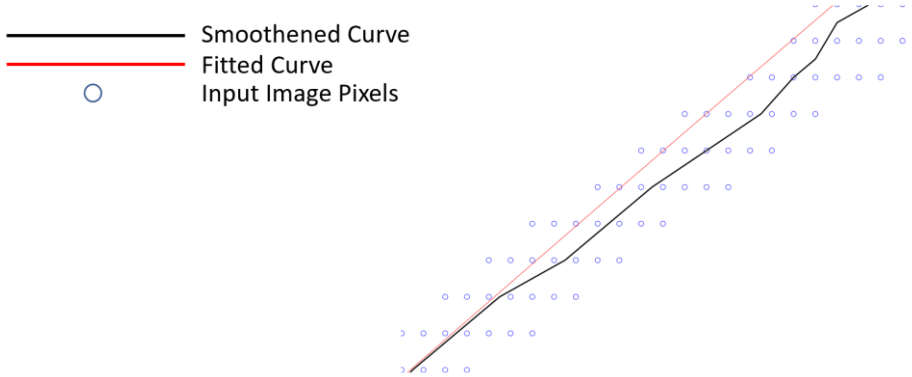


Figure 2: A zoomed-in version of the output

3. Find peaks:

A peak is the point whose ordinate is greater than that of the adjacent points, mathematically $f(x)>f(x-h)$ & $f(x)>f(x+h)$. A crest is the point whose ordinate is smaller than that of the adjacent points, mathematically $f(x)<f(x-h)$ & $f(x)<f(x+h)$. This is exactly what is translated in the code. In this way, the curve is segmented between the first peak to the next one

4. Best fit:

To find the best fit curve, we have used six equations to compare and find out which shows the minimum error. Then the equation model with minimum error one is selected for that segment of the curve to display the results and the corresponding plot. We have also used the theory of regression in fitting the equations.

5. Peak Spacer:

Peak spacer is used to select the distance between the peaks. Sometimes a peak may get selected through a noisy scanned input that is undesired at the first place, So, a user can select the minimum spacing between the peaks to remove the undesired peaks and segmentations in the output curve and the equation. Peak spacer is selected from the GUI slider.

Algorithm Flow chart:

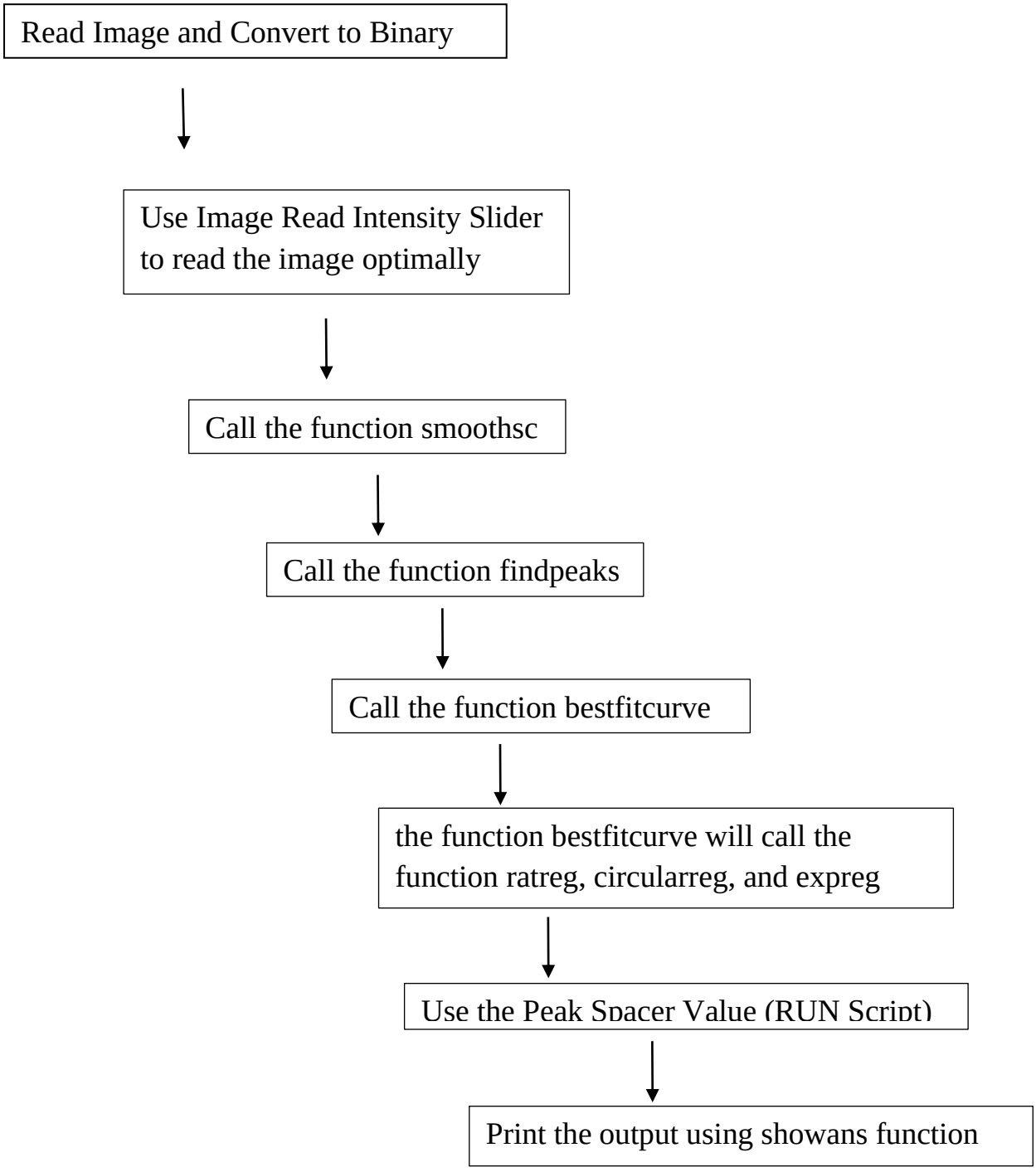


Figure :3

Main Code:

```
% A code to segment a input curve and find easy
% Equation for each particular segment.
% Input curve should be a bitmap image with
% White background and curve should be drawn in black.
% (saved in 256 color mode).

clc;
clear all;
format short;
close all;
warning off;
threshold_gap_per =2;
imageread_intesity = 80;
%Input Image.
A = imread('colorimagetry_1.jpg');

A = rgb2gray(A);
[r , c] = size (A);
threshold_gap =(r*threshold_gap_per*0.01);
A = A> 255*imageread_intesity*0.01;

% Smooth spikes, vertical and horizontal lines.
[x ,y] = smoothsc(A);

% plot the smoothened curve
plot(x,y, 'LineWidth', 1.5, 'color', 'k');
hold on;
% Find Peak and Crest points.
[peakx,peaky,px] = findpeaks(x,y);


% Number of equations is Number of peaks minus 1.
q = length(px);
% % % % dips('Fitted Equations Are:');
```

```
s = 1;
i = 1;
% plot(peakx(i), peaky(i),'ko','LineWidth',2.0);
% %for i = 1: q-1

while i <= q-1
   if ((px(i+s)-px(i)) > threshold_gap) || (i+s == q)
   m = px(i);
   n = px(i+s);
   % Segment the curve for fittng.
   [r,ym,error_for_rms] = bestfitcurve(x(m:n),y(m:n));
   if i==1
      error_col = error_for_rms;
   end
   if i>1
      error_col = [error_col error_for_rms];
   end
   % Display results.
   showans(r,ym,peakx(i),peakx(i+s)); %fitted graph
display
   i = i+s;
   plot(peakx(i),peaky(i),'ko','LineWidth',2.0); %peak crest
display
   s = 1; %setting the s again to the base to compare the
current peak with the immediate next one
   else
      s = s+1;
      continue
   end
end
rmserrorfinal = rms(error_col) %rms error display
legend ('Original Curve', 'Peaks and Crests', 'Fitted
Curve');
```

User Defined Functions:

1. bestfitcurve
2. circulareg
3. expreg
4. findpeaks
5. ratreg
6. showans
7. smoothsc

Equation Library:

1. Linear equation: y=mx+c
2. Second degree polynomial: y=ax2+bx+c
3. Third degree polynomial: y=ax3+bx2+cx+d
4. Circular equation: x2+y2+2gx+2fy+c=0
5. Rational equation: y = (x+A)/(Bx+C)
6. Exponential equation: y = exp(ax+b)

Input & Output on GUI:

Case 01:

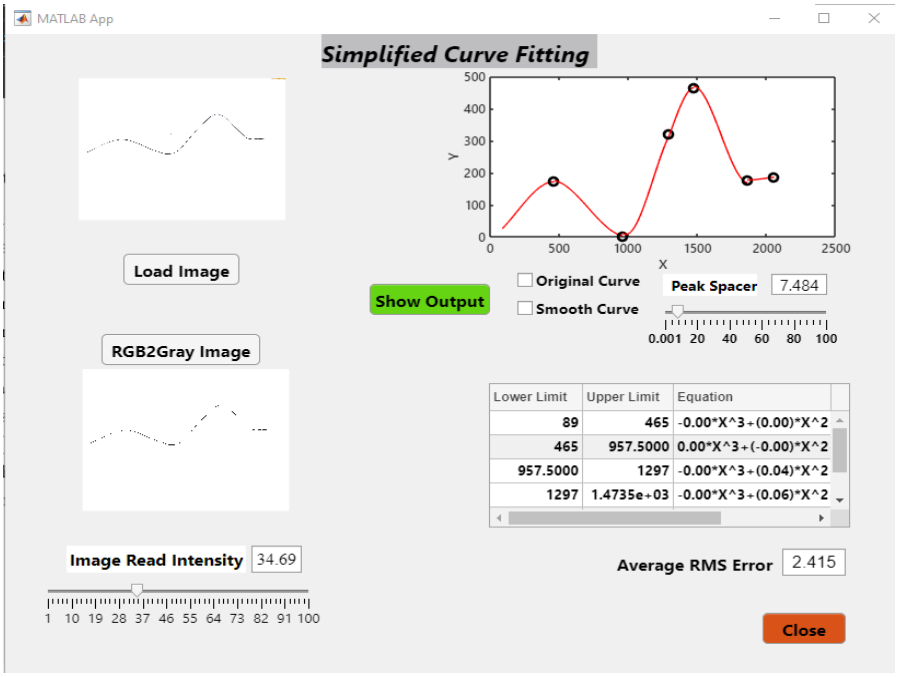


Figure 4

Case 02:

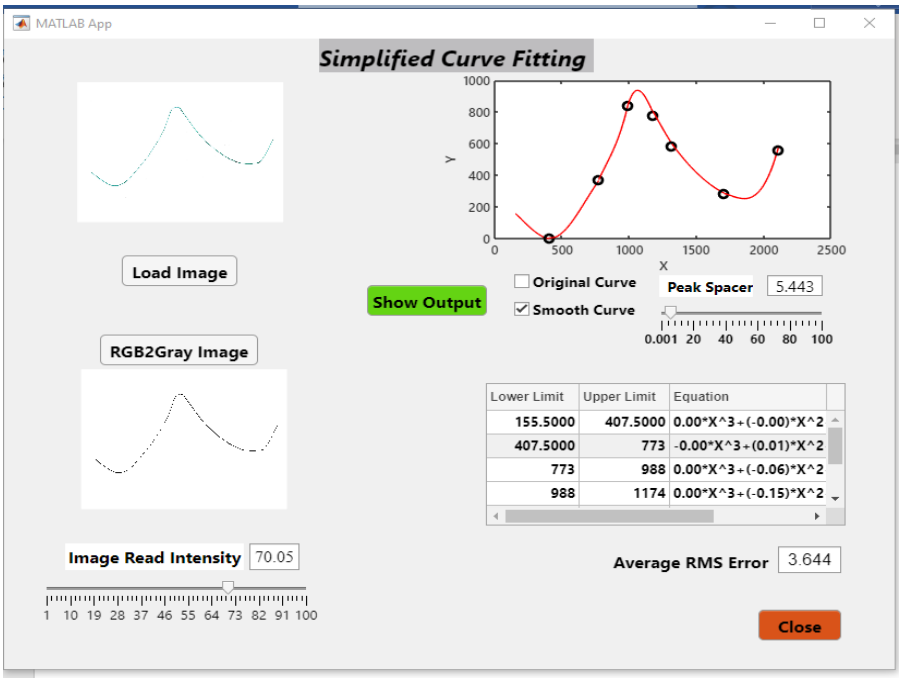


Figure 5

Case 03:

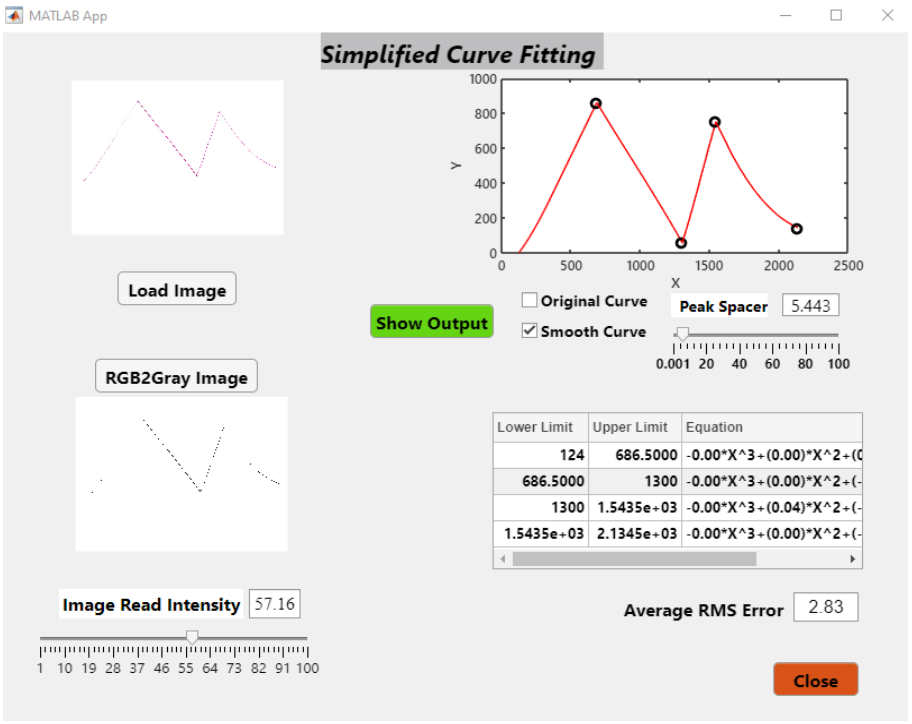


Figure 6

Case 04:

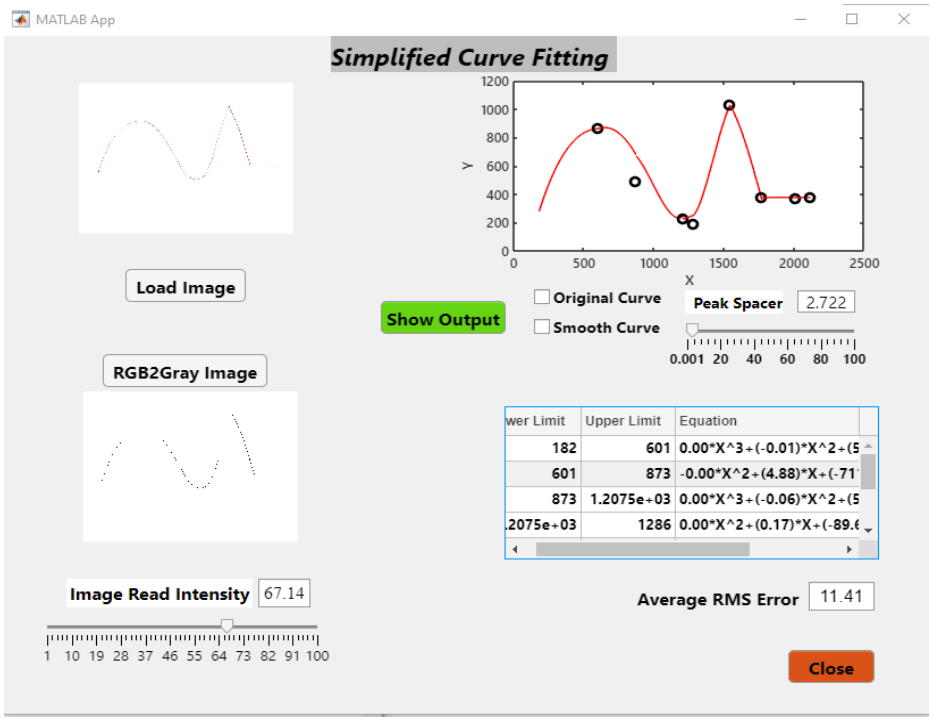


Figure 7

Case 5:

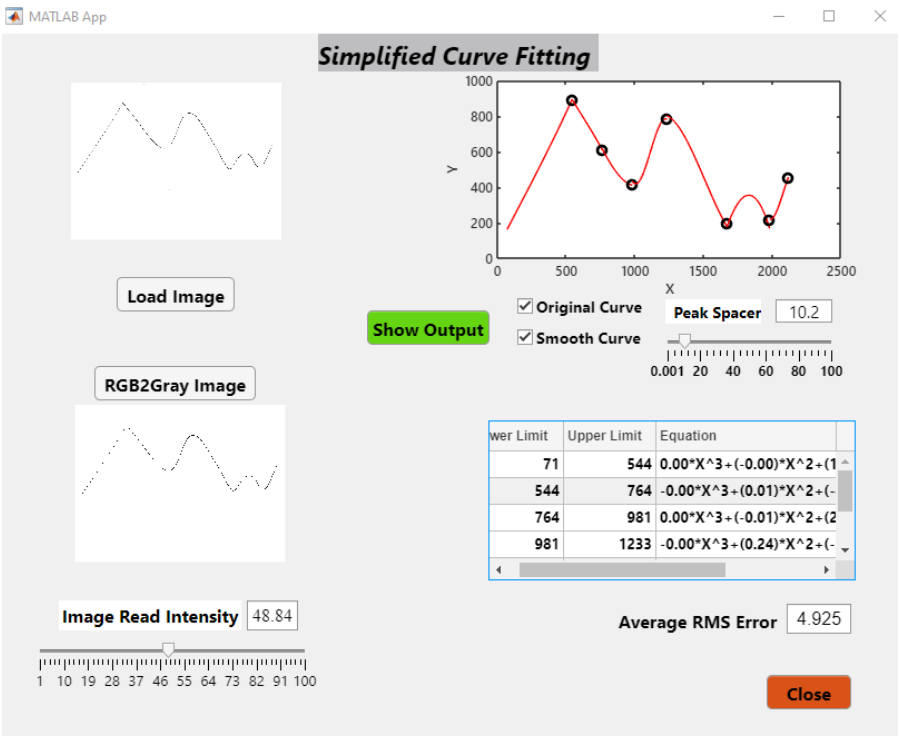


Figure 8

Application:

1. Curve fitting of an image and then output equation
2. Electrical machine-like oscilloscope that gives figure output and this program will detect equation of the figure.
3. Road curve detection from image processing on Smart vehicle.

Conclusion:

It was a simple user interactive input output project. As it was the very begging, the project has some shortcomings. Here are:

- The fitted curve is not very accurate at the peak points.
- Also, the curve is not very accurate to find out any vertical or horizontal line present in the image.
- There is no intelligent equation matching for faster equation match.
- The equation library is not too big.
- The advanced image processing is not done to remove the noise from the input picture.

Under the above circumstances, we want to go further to improve the project implementing bellows

- A more sensitive algorithm to find out the bending and flat lines in the curve accurately.
- New approach in segmentation to eliminate the peak problems.
- Comparing with more equations.
- Advanced Image Processing to remove the noise from the picture.
- Implement the project in case of autonomous driving vehicle