# License Plate Detection Algorithm

|  |
| --- |
| Name |
| Parth Doshi |
| Ansh Jain |
| Meghan Patil |

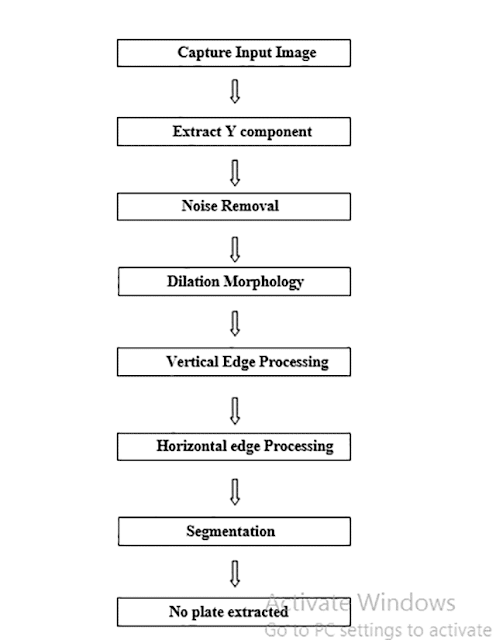
# Abstract:

According to the survey conducted in 2016, there are about 260 million registered cars in India alone. This has a major impact on the security of the cars since there can be cases when the car is stolen, owner of the car has an unpaid ticket and so on. Identification of such cars if done manually would prove to be stressful as well as time consuming. So in order to prevent that we can make use of an license plate detection algorithm which can easily detect the number plates of a vehicle and determine the details of the owner. The algorithm is based on Optical Character Recognition (OCR) which extracts the text from the input image. The algorithm can monitor the vehicles in the surrounding areas thereby enhancing the security.

# Introduction:

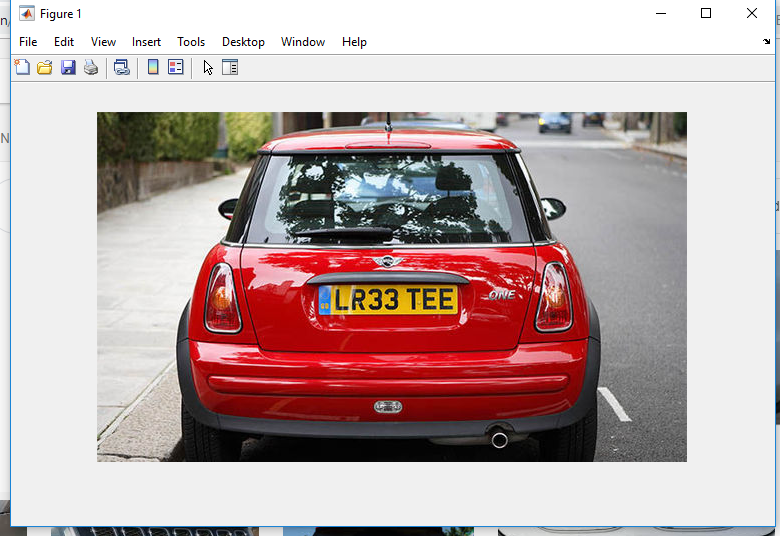
So as we all know the number of people using vehicles is increasing day by day , so it is difficult to manually enforce laws and traffic rules for smooth traffic flow. Toll-booths are constructed on free ways parking structures etc. But all the vehicles have one thing in common is the license plate. Each vehicle can be identified by its license plate and it gives more information about the vehicle and the person who owns it. So in this project we are trying to implement an algorithm which is used to detect the license plate of the vehicle and get the information about the vehicle. We are doing this using confidence related predictions. Multiple detection are doing for a single license plate , some methods are applied and then the detected regions are merged together. Tracker as in some threshold value is used to restrict the search of image to a particular region that is avoiding detection of useless region. After the regions are detected and passed through final elimination stage to get the final region having most probability of containing the license plate.

# Block diagram/Working/Flowchart

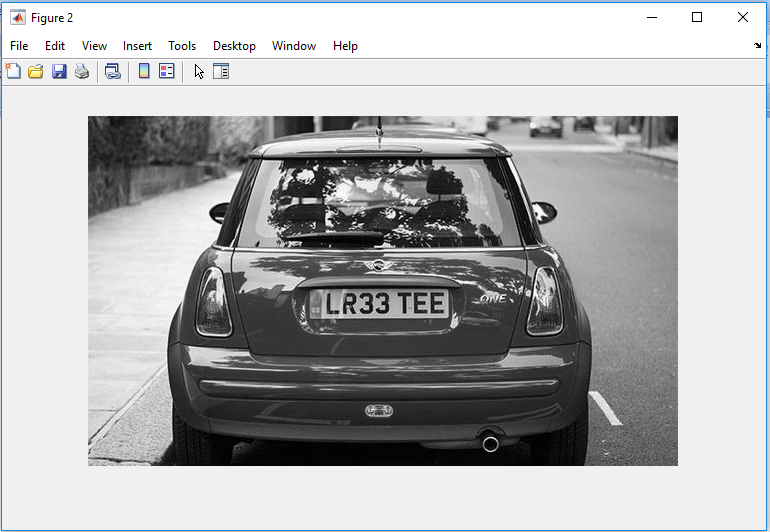


**STIMULATION RESULTS**

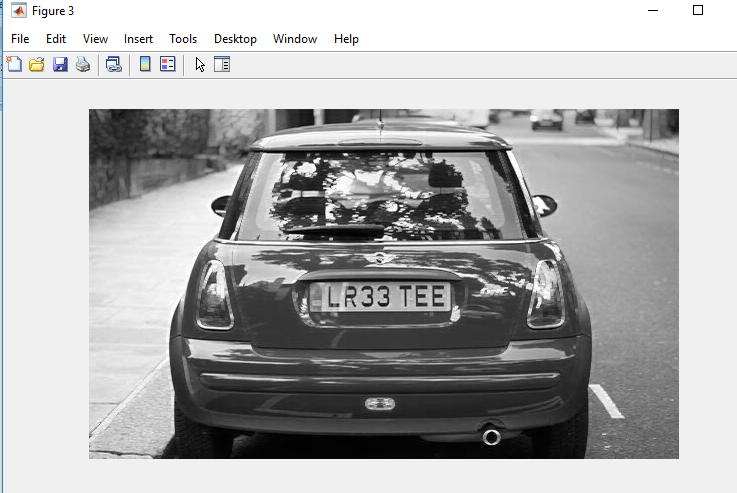
1. **INPUT IMAGE**



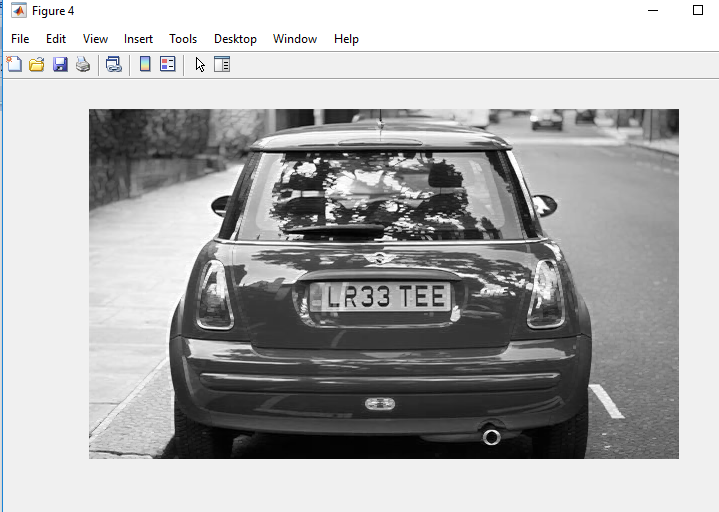
1. **EXTRACTING Y COMPONENT**



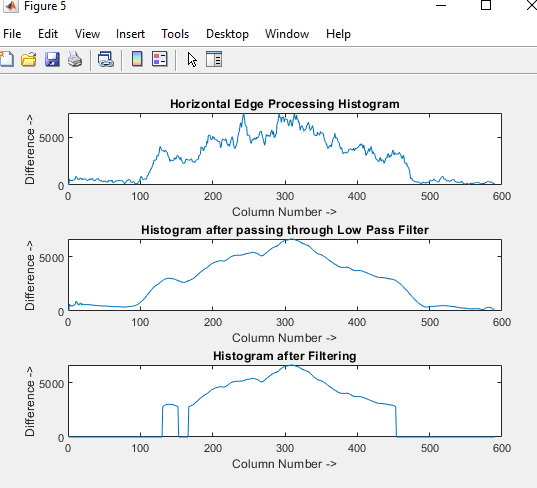
1. **AFTER REMOVING THE NOISE**



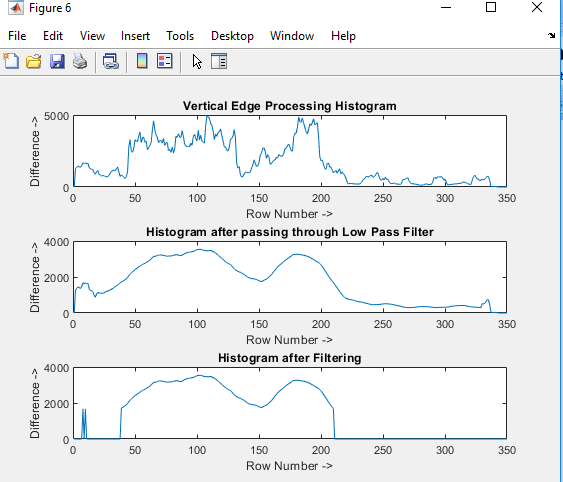
1. **AFTER DILATION MORPHOLOGY**

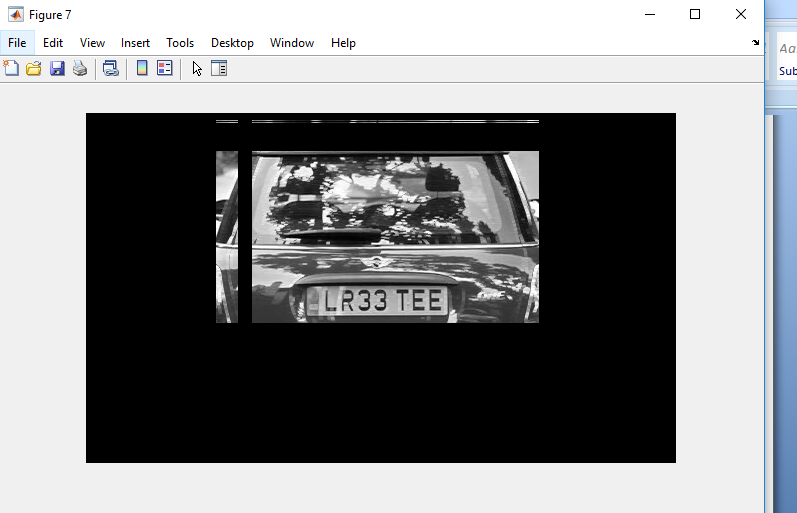


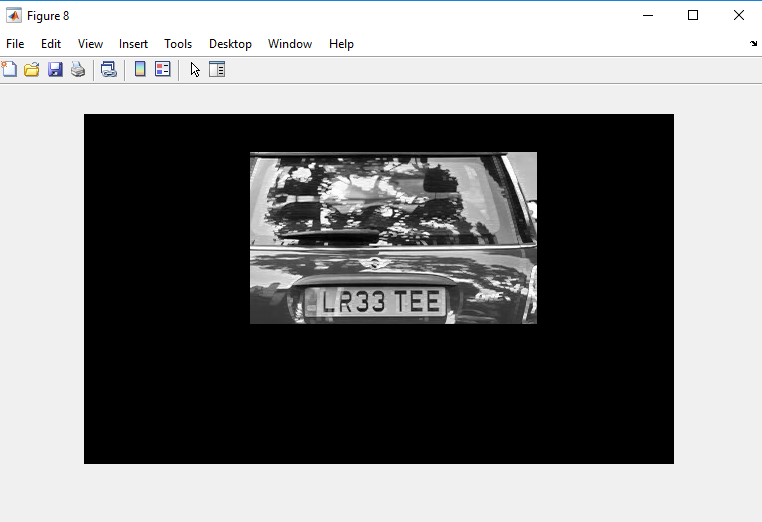
1. **VERTICAL EDGE PROCESSING**



1. **HORIZONTAL EDGE PROCESSING**



1. **SEGMENTATION**
2. **AFTER EXTRACTING THE NUMBER PLATE**



# Applications and Future work

In this project we have implemented and algorithm using image processing which can detect a vehicle’s number plate and extract the number i.e, the text from it.

While doing so, we have made certain assumption as to the portion that needs to be extracted and the part of the image that should not be included in the extracted part and so on.

We can further improve its quality by consider certain parameters which are mentioned and explained below:

·         The improvements can be made in text reorganization which would enable to extract the text from any of the written format. This improvement increases the efficiency and also increases the probability of catching the culprits who mislead the security system by changing the format of the number plate on a regular basis.

We can further connect it to internet or web allowing the updation of criminal records whenever found.

A fast algorithm for automatic license plate detection system for detecting the license number plate is being implanted that achieves a high detection rate without the need for a high quality images from expensive hardware.

A list of possible/potential applications is shown below:

* Highway Electronic Toll Collection
* Traffic Monitoring Systems
* Online Parking Systems
* Identification Of Stolen Cars
* Identification Of Cars with Unpaid Tickets
* Border Control

# Conclusion

As shown in the code, we take the image of a car with number plate LR33 TEE. First we extract the y component i.e we covert the image into grey scale. Then we remove the noise that is present in the image and it is followed by dilation. After these steps, horizontal edge processing is done which gives the maximum value from the column number 70-120 which is being extracted by passing through the filter. And similarly vertical edge processing is performed which maximum value from the column 125-175 in histogram. Then the probable candidates are identified and the candidate with the highest probability is selected and extracted. This ensures that the unwanted regions are not being included in the extracted portion.

Furthermore this technique can also prove to be useful in security systems in places such as parking spaces, mall etc. It is a cheap and effective method compared to other methods such as ultrasonic sensors and repeaters. The efficiency of this algorithm also depends upon the type of the input image that is being provided. Nevertheless it is a cheap, efficient and trustworthy method to use if the security needs to be maintained and when the budget is low.

# References

  MATLAB MathWorks, [**http://www.mathwork.com**](http://www.mathwork.com/), 10/10/18

·       Rafael C Gonzalez, Digital Image Processing 2nd Edition.

# Appendix

**CODE:**

clc; % Clear command window.

clear all; % Delete all variables.

close all; % Close all figure windows except those created by imtool.

imtool close all; % Close all figure windows created by imtool.

workspace; % Make sure the workspace panel is showing.

% Read Image

I = imread ('CAR2.jpg');

figure(1);

imshow(I);

% Extract Y component (Convert an Image to Gray)

Igray = rgb2gray(I);

[rows cols] = size(Igray);

%% Dilate and Erode Image in order to remove noise

Idilate = Igray;

for i = 1:rows

for j = 2:cols-1

temp = max(Igray(i,j-1), Igray(i,j));

Idilate(i,j) = max(temp, Igray(i,j+1));

end

end

I = Idilate;

figure(2);

imshow(Igray);

figure(3);

title('Dilated Image')

imshow(Idilate);

figure(4);

imshow(I);

difference = 0;

sum = 0;

total\_sum = 0;

difference = uint32(difference);

%% PROCESS EDGES IN HORIZONTAL DIRECTION

disp('Processing Edges Horizontally...');

max\_horz = 0;

maximum = 0;

for i = 2:cols

sum = 0;

for j = 2:rows

if(I(j, i) > I(j-1, i))

difference = uint32(I(j, i) - I(j-1, i));

else

difference = uint32(I(j-1, i) - I(j, i));

end

if(difference > 20)

sum = sum + difference;

end

end

horz1(i) = sum;

% Find Peak Value

if(sum > maximum)

max\_horz = i;

maximum = sum;

end

total\_sum = total\_sum + sum;

end

average = total\_sum / cols;

figure(5);

% Plot the Histogram for analysis

subplot(3,1,1);

plot (horz1);

title('Horizontal Edge Processing Histogram');

xlabel('Column Number ->');

ylabel('Difference ->');

%% Smoothen the Horizontal Histogram by applying Low Pass Filter

sum = 0;

horz = horz1;

for i = 21:(cols-21)

sum = 0;

for j = (i-20):(i+20)

sum = sum + horz1(j);

end

horz(i) = sum / 41;

end

subplot(3,1,2);

plot (horz);

title('Histogram after passing through Low Pass Filter');

xlabel('Column Number ->');

ylabel('Difference ->');

%% Filter out Horizontal Histogram Values by applying Dynamic Threshold

disp('Filter out Horizontal Histogram...');

for i = 1:cols

if(horz(i) < average)

horz(i) = 0;

for j = 1:rows

I(j, i) = 0;

end

end

end

subplot(3,1,3);

plot (horz);

title('Histogram after Filtering');

xlabel('Column Number ->');

ylabel('Difference ->');

%% PROCESS EDGES IN VERTICAL DIRECTION

difference = 0;

total\_sum = 0;

difference = uint32(difference);

disp('Processing Edges Vertically...');

maximum = 0;

max\_vert = 0;

for i = 2:rows

sum = 0;

for j = 2:cols %cols

if(I(i, j) > I(i, j-1))

difference = uint32(I(i, j) - I(i, j-1));

end

if(I(i, j) <= I(i, j-1))

difference = uint32(I(i, j-1) - I(i, j));

end

if(difference > 20)

sum = sum + difference;

end

end

vert1(i) = sum;

%% Find Peak in Vertical Histogram

if(sum > maximum)

max\_vert = i;

maximum = sum;

end

total\_sum = total\_sum + sum;

end

average = total\_sum / rows;

figure(6)

subplot(3,1,1);

plot (vert1);

title('Vertical Edge Processing Histogram');

xlabel('Row Number ->');

ylabel('Difference ->');

%% Smoothen the Vertical Histogram by applying Low Pass Filter

disp('Passing Vertical Histogram through Low Pass Filter...');

sum = 0;

vert = vert1;

for i = 21:(rows-21)

sum = 0;

for j = (i-20):(i+20)

sum = sum + vert1(j);

end

vert(i) = sum / 41;

end

subplot(3,1,2);

plot (vert);

title('Histogram after passing through Low Pass Filter');

xlabel('Row Number ->');

ylabel('Difference ->');

%% Filter out Vertical Histogram Values by applying Dynamic Threshold

disp('Filter out Vertical Histogram...');

for i = 1:rows

if(vert(i) < average)

vert(i) = 0;

for j = 1:cols

I(i, j) = 0;

end

end

end

subplot(3,1,3);

plot (vert);

title('Histogram after Filtering');

xlabel('Row Number ->');

ylabel('Difference ->');

figure(7), imshow(I);

%% Find Probable candidates for Number Plate

j = 1;

for i = 2:cols-2

if(horz(i) ~= 0 && horz(i-1) == 0 && horz(i+1) == 0)

column(j) = i;

column(j+1) = i;

j = j + 2;

elseif((horz(i) ~= 0 && horz(i-1) == 0) || (horz(i) ~= 0 && horz(i+1) == 0))

column(j) = i;

j = j+1;

end

end

j = 1;

for i = 2:rows-2

if(vert(i) ~= 0 && vert(i-1) == 0 && vert(i+1) == 0)

row(j) = i;

row(j+1) = i;

j = j + 2;

elseif((vert(i) ~= 0 && vert(i-1) == 0) || (vert(i) ~= 0 && vert(i+1) == 0))

row(j) = i;

j = j+1;

end

end

[temp column\_size] = size (column);

if(mod(column\_size, 2))

column(column\_size+1) = cols;

end

[temp row\_size] = size (row);

if(mod(row\_size, 2))

row(row\_size+1) = rows;

end

%% Region of Interest Extraction

%Check each probable candidate

for i = 1:2:row\_size

for j = 1:2:column\_size

% If it is not the most probable region remove it from image

if(~((max\_horz >= column(j) && max\_horz <= column(j+1)) && (max\_vert >=row(i) && max\_vert <= row(i+1))))

%This loop is only for displaying proper output to User

for m = row(i):row(i+1)

for n = column(j):column(j+1)

I(m, n) = 0;

end

end

end

end

end

figure(8), imshow(I);

imshow(I);