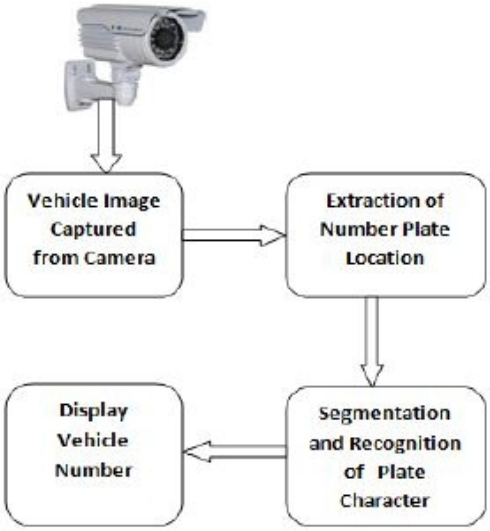
Segmenting Characters from License Plate using Image Processing Techniques

Table of Contents

INTRODUCTION

In last couple of decades, the number of vehicles has increased drastically, it is becoming difficult to keep track of each vehicle for purpose of law enforcement and traffic management. License Plate Detection is used increasingly nowadays for the same. The system performing the task of License Plate detection is known as LPR system. It is generally consists of three steps such as: Detection of License plate, Segmentation of License plate characters and Recognition of the characters of the License Plate. The scope of this project is limited to the segmentation of the License Plate characters. The various lighting conditions, camera angle, rotation degrades the accuracy of LP detection. In this project four different methods are used to detect the license plate and then **Bounding Box** method is used for the character segmentation from the detected License Plate image.

The general LPR system:



LITERATURE SURVEY

There are different methods to detect the License Plate region discussed by Christos-Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Ioannis D. Psoroulas, Vassili Loumos, Eleftherios Kayafas in [1]. Some of the methods are as follows:

1. Using the Morphological Analysis [1]
2. Using the Sobel Edge Detector and the Morphological Analysis [1][3]
3. Using the Histogram Processing [1]
4. Using the Top Hat Transform [1][2]
5. Using the Canny Edge Detector Method[Experimental]

Feng Yang, Fan Yang[2] has presented a method using Top Hat transform to detect the license plate.

INPUT IMAGE

In each of the method the input image is RGB colored image which is resized to 480x620 and is converted to gray scale image for the further processing and the Bounding Box method is used to extract the character from the detected LP region. In this project the input images are taken from http://www.medialab.ntua.gr/research/LPRdatabase.html[1] and some images are taken from the Google images.

MORPHOLOGICAL OPERATION

Morphological image processing is a collection of non linear operations related to shape or morphology of features in an image. The operation is based on the relative ordering of the pixels not on their values. The morphology is performed using the structuring element which is also a binary image. Basic operations for the image morphology are **erosion** and **dilation.**

- **Erosion**

  The erosion of a binary image f with structuring element s produces a image g with ones in all location (x,y) of a structuring element's origin at which that structuring element s fits the input image i.e.

$$g(x,y) = \begin{Bmatrix} 1, \ if \ s \ fits \ f \\ 0, \ otherwise \end{Bmatrix}$$

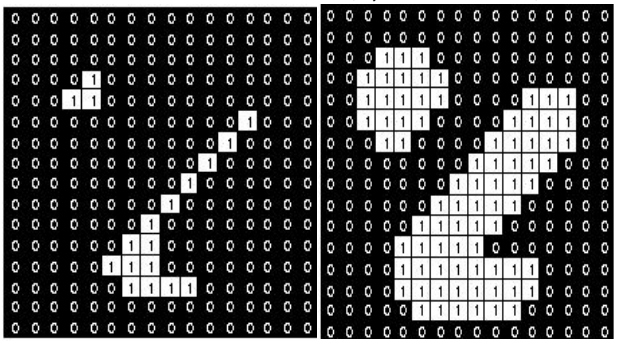


Figure: 1

Figure: 2

1	1	1
1	1	1
1	1	1

Figure: 3

Figure 2 is erode with structuring element of size 3x3 (Figure 3) to produce Figure 1.

- **Dilation**

  The dilation of a binary image f with structuring element s produces a image g with ones in all location (x,y) of a structuring element's origin at which that structuring element s hits the input image i.e.

$$g(x,y) = \begin{Bmatrix} 1, \ if \ s \ hits \ f \\ 0, \ otherwise \end{Bmatrix}$$

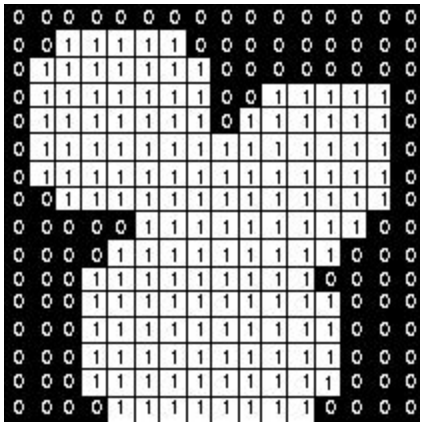


Figure: 4

Figure 4 is dilated with structuring element in Figure 3.

MATLAB IMPLEMENTATION OF LICENSE PLATE DETECTION AND CHARACTER SEGMENTATION

The License Plate detection algorithms are implemented using MATLAB. MATLAB is a powerful tool and is used to implement the tasks that require extensive computation. It provides easy and quicker implementation of algorithms as compared to C, C++ and Java. The key feature in MATLAB is that it contains a rich library functions for image processing and data analysis. This makes MATLAB ideal tool for the implementation of this project.

- **Using the Mathematical Morphology**
  The basic morphological operations (erosion and dilation) are used to find the region of interest. The image is dilated and eroded with some structuring element and **morphological gradient** is used in order to detect the edges in the image.
  Following are the steps performed:
  1. Input image

Figure: 5 (Original Input Image)

2. Converted to grayscale image and then median filtering is applied to enhance the image features.

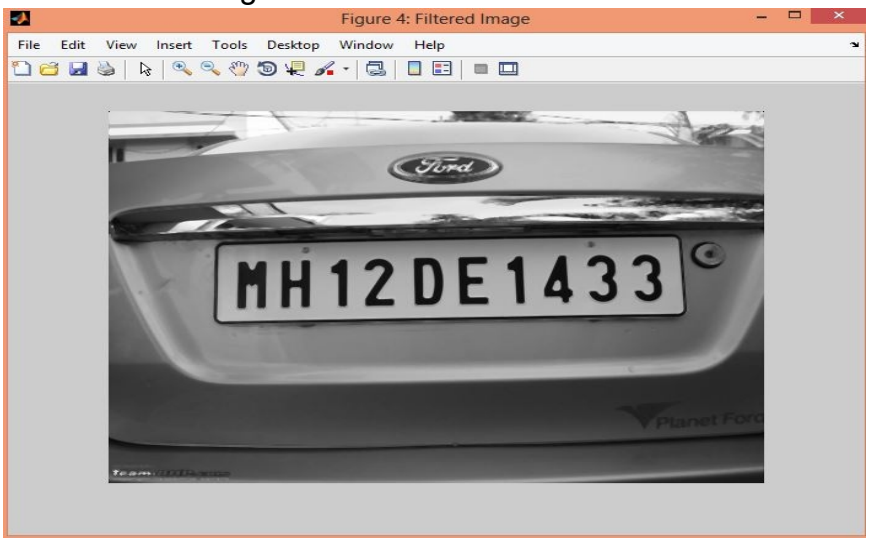


Figure :6 (Filtered gray scale image)

3. Image is dilated and and eroded with the 'disk' structuring element of radius 1.

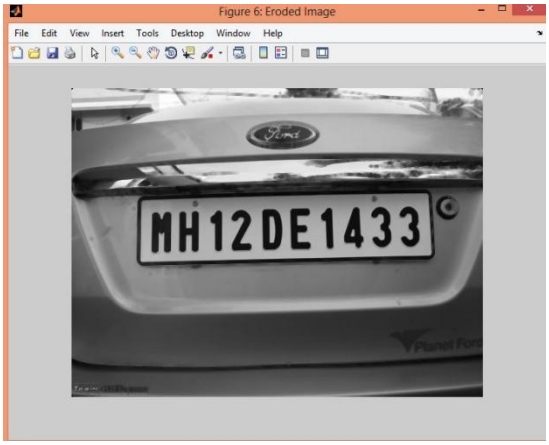


Figure:7 (Eroded Image)

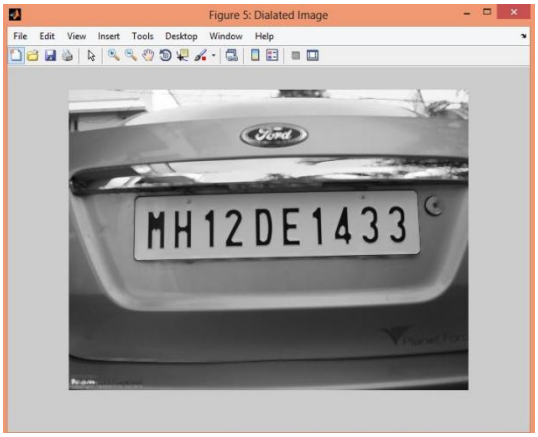


Figure: 8 (Dilated Image)

4. Morphological Gradient is applied on the previous image and the resulting image is convolved with 2x2 binary matrix to enhance the edges.

$$G(f) = f \oplus b - f \ominus b.$$

where $\oplus$ and $\ominus$ denote the dilation and the erosion, respectively.

The above formula tells about the Morphological Gradient. Here f is the input image and b is the structuring element.

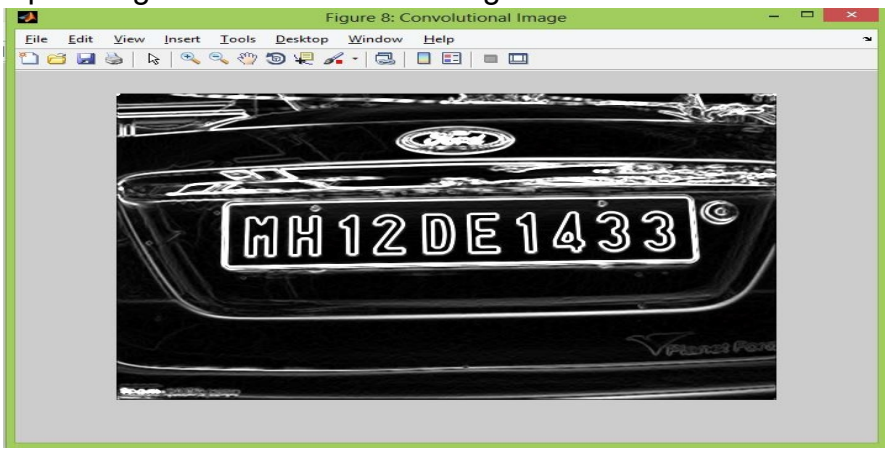


Figure: 9 (Image with enhanced edges)

5. Image is converted to binary image and all the holes are filled. After filling the holes the connected components having the area less than 100 pixels are removed.

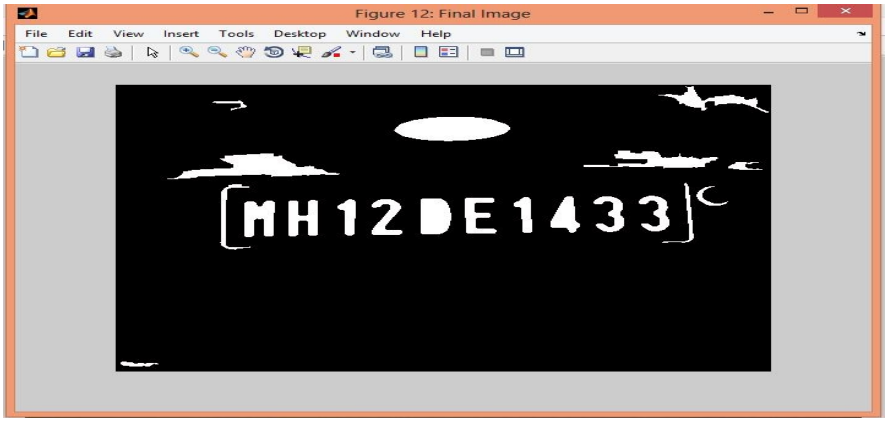
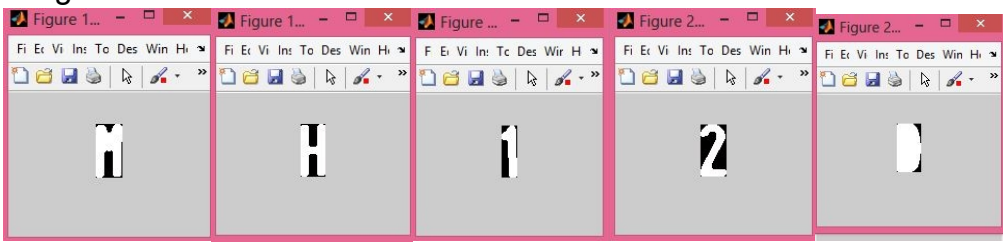


Figure: 10 (Final Image)

6. Bounding Box method is applied to segment the characters from the image



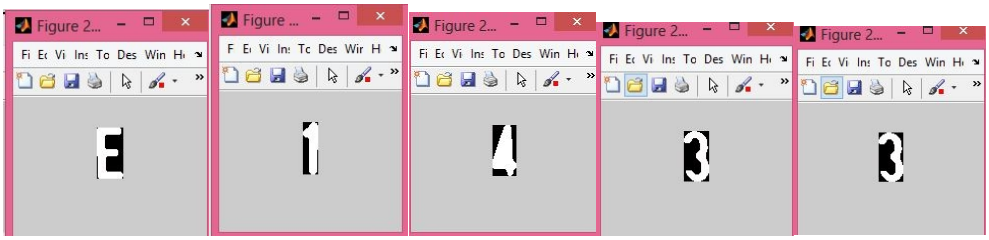


Figure: 11 (Segmented Characters)

7. Along with the segmented characters some noise is also segmented which can be removed after performing the character recognition which is beyond the scope of this project.
8. Following is the MATLAB code for this method

```
f=imread('d1.jpg'); % Reading the number plate image.
figure('name','Original Image');
imshow(f);
f=imresize(f,[400 NaN]); % Resizing the image keeping aspect ratio same.
figure('name','Resized Image');
imshow(f);
imwrite(f, 'Resized.jpg','jpg');
g=rgb2gray(f); % Converting the RGB (color) image to gray (intensity).
figure('name','Gray Image');
imshow(g);
imwrite(g, 'GrayScale.jpg','jpg');
g=medfilt2(g,[3 3]); % Median filtering to remove noise.
figure('name','Filtered Image');
imshow(g);
imwrite(g, 'Filtered.jpg','jpg');
```

```
se=strel('disk',1); % Structural element (disk of radius 1) for morphological
processing.
gi=imdilate(g,se); % Dilating the gray image with the structural element.
figure('name', 'Dialated Image');
imshow(gi);
ge=imerode(g,se); % Eroding the gray image with structural element.
figure('name', 'Eroded Image');
imshow(ge);
gdiff=imsubtract(gi,ge); % Morphological Gradient for edges enhancement.
figure('name','Morphological Image');
imshow(gdiff);
imwrite(gdiff, 'Morphedimage.jpg','jpg');
gdiff=mat2gray(gdiff); % Converting the class to double.
gdiff=conv2(gdiff,[1 1;1 1]); % Convolution of the double image for brightening the
edges.
figure('name','Convolutional Image');
imshow(gdiff);
gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1); % Intensity scaling between the range 0 to
1.
figure('name','Intensity mapped Image');
imshow(gdiff);
imwrite(gdiff, 'IntensityMApped.jpg','jpg');
B=logical(gdiff);
figure('name','Logical Image');
imshow(B);% Conversion of the class from double to binary.

er=imerode(B,strel('line',50,0));
imshow(er);
out1=imsubtract(B,er);
figure('name','Eroded Image');
imshow(out1);
%imwrite(out1, 'eroded.jpg','jpg');
 Filling all the regions of the image.
F=imfill(out1,'holes');
 Thinning the image to ensure character isolation.
H=bwmorph(F,'thin',NaN);
figure('name','Morphed Image');
%imshow(H);
H=imerode(H,strel('line',3,90));
imshow(H);
final=bwareaopen(H,100);
%figure('name','Final Image');
imshow(final);
%imwrite(final, 'Final.jpg','jpg');
Iprops=regionprops(final,'BoundingBox','Image');
for i = 1:length(Iprops)
   s = 'segment';
    s1 = 'Images\';
    s = strcat(s, int2str(i));
    s1 = strcat(s1,s);
   %figure('name', s)
    s = strcat(s,'.jpg');
   % imshow(Iprops(i).Image)
    imwrite(Iprops(i).Image, s,'jpg');
end
```

- **Using the Sobel Operator**
  Sobel operator can be used to detect vertical and horizontal edges. After detecting the edges all the non-candidate edges are removed to find the License Plate Region.

-1	0	1
-2	0	2
-1	0	1

Horizontal

-1	-2	-1
0	0	0
-1	-2	-1

Vertical

Figure: 12 (Sobel Kernels)

Following are the steps involved:

1. First three steps are common from the previous method i.e. given input color image, first the image is converted to grayscale image and then filtered using the median filter to enhance the image property.
2. Sobel edge operator is used detect the edges (vertical and Horizontal).

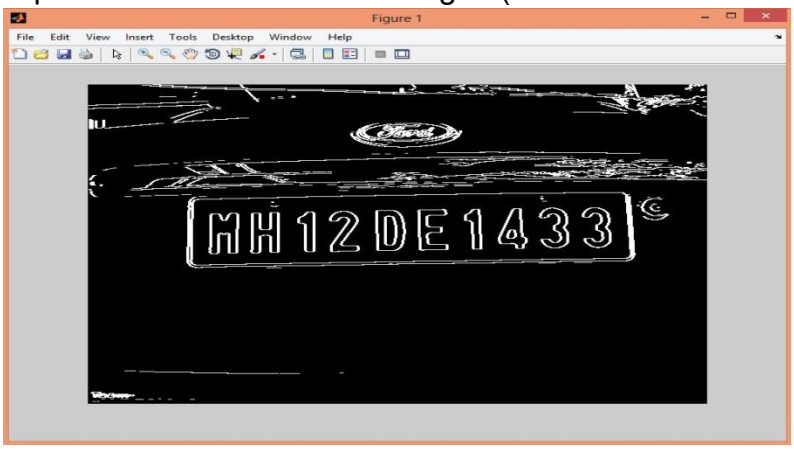


Figure: 13 (Image after application Sobel operator)

MATLAB provides edge('sobel') function to perform the above task.

3. The horizontal and vertical edges are removed by first dilating the image with the horizontal line and then dilating with vertical image.
4. Holes in the image are filled and borders along the edges are removed using the imclearborder(Image,Connectivity) function in the MATLAB.
5. The resulted image is eroded two times with disk structuring element of radius1. And again the holes are filled.

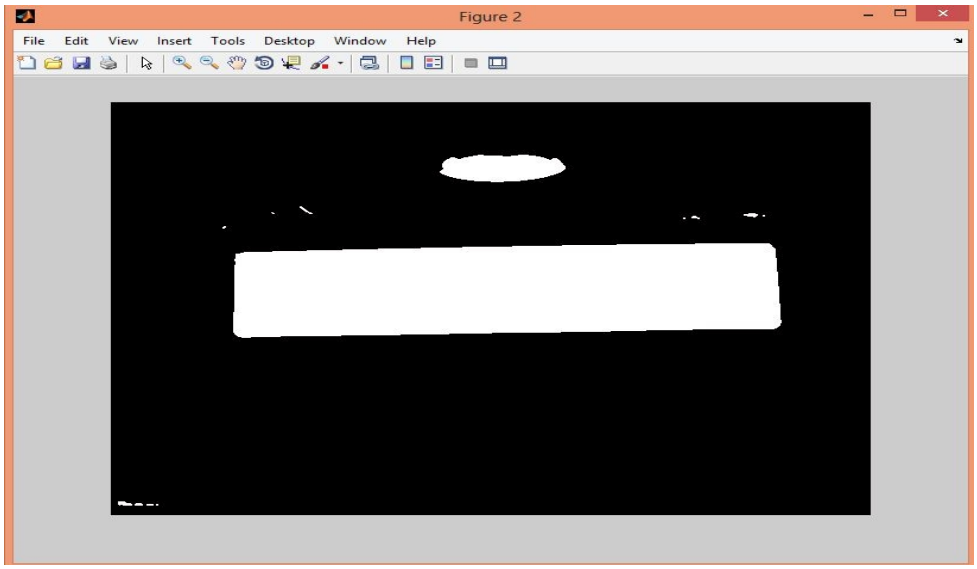


Figure: 14 (Detected LP Region)

6. The above image is multiplied with the grayscale image to find the actual LP.

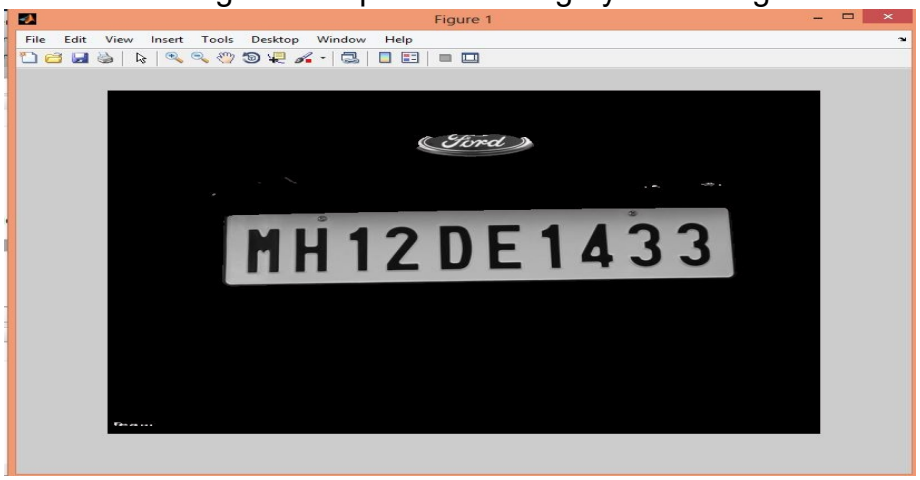


Figure: 15 (Actual LP Detected)

7. The above image is converted to the binary image and then the characters are segmented using the Bounding Box method.

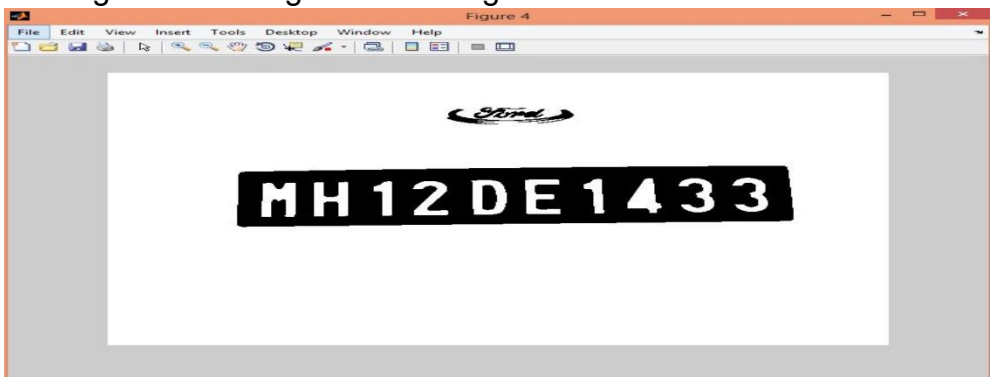


Figure: 16 (Binary Image)

8. Characters are segmented and result is same as in the previous method but with a grater accuracy.

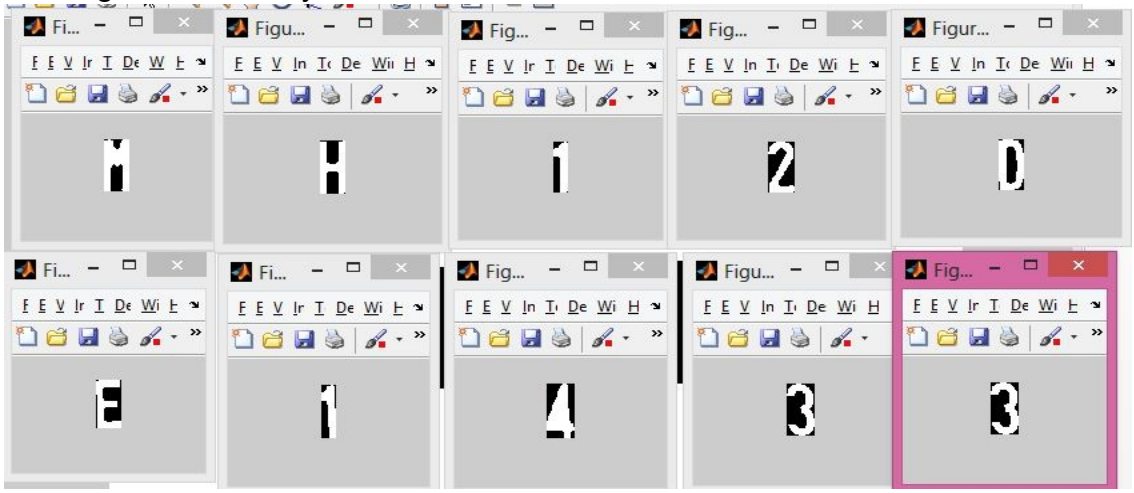


Figure: 17 (Segmented Characters)

9. MATLAB implementation.

```
I= imread('d1.jpg');
G = imresize(I, [480 640]);
H = rgb2gray(G);
%imshow(H);
%[m n] = size(G);
%H = zeros(m , n);
%n = n / 3;
%for i = 1:m
%    for j = 1:n
%        H(i,j) = 0.114 * G(i,j,1) + 0.587*G(i,j,2) + 0.299 *
G(i,j,3);
%    end
%end
%imshow(H);
H1 = edge(H, 'sobel');
H1 = double(H1);
H1 = conv2(H1, [1 1:1 1]);
%figure
%imshow(H1);
se1 = strel('line',1, 0);
sr2 = strel('line',1,90);
H2 = imdilate(H1, sr2);
H2 = imdilate(H2, se1);
H2 = imfill(H2, 'holes');
H2 = bwareaopen(H2, 100);
H2 = imclearborder(H2, 8);
se = strel('disk',1);
H3 = imerode(H2, se);
H4 = imerode(H3, se);
%size(H)
%size(H4)
H4 = H4(:,2:641);
H4 = medfilt2(H4);
%figure
%imshow(H4);
```

```
H4 = immultiply(H4, H);
figure
imshow(H4);
level = graythresh(H4);
H4 = im2bw(H4, level);
H4 = bwareaopen(H4, 100);
H4 = ~H4;
%imshow(H4);
H4 = bwareaopen(H4, 100);
%figure
%imshow(H4);
I2=regionprops(H4,'BoundingBox','Image');
for i = 1:length(I2)
   s = 'segment';
 %   s1 = 'Images\';
    s = strcat(s, int2str(i));
    %s1 = strcat(s1,s);
    figure('name', s)
    s = strcat(s,'.jpg');
    imshow(I2(i).Image)
    %imwrite(Iprops(i).Image, s,'jpg');
end
```

- **Using Histogram Processing**
  1. The input image converted to grayscale and is filtered to enhance the image features.
  2. To find a horizontal histogram, the algorithm traverses through each column of an image. In each column, the algorithm starts with the second pixel from the top. The difference between second and first pixel is calculated. If the difference exceeds certain threshold, it is added to total sum of differences. Then, algorithm will move downwards to calculate the difference between the third and second pixels.
  3. The Horizontal histogram is smoothed using the low pass filter.
  4. So on, it moves until the end of a column and calculate the total sum of differences between neighboring pixels. At the end, an array containing the column-wise sum is created. The same process is carried out to find the vertical histogram. In this case, rows are processed instead of columns.

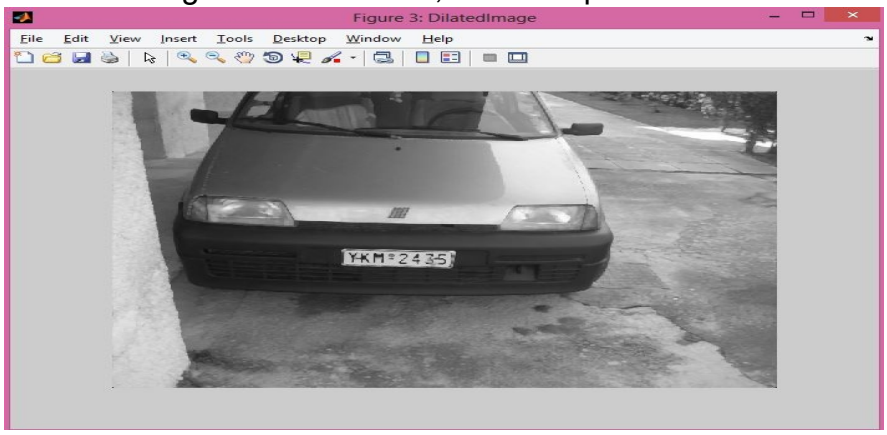


Figure: 18 (Dilated Input Image)

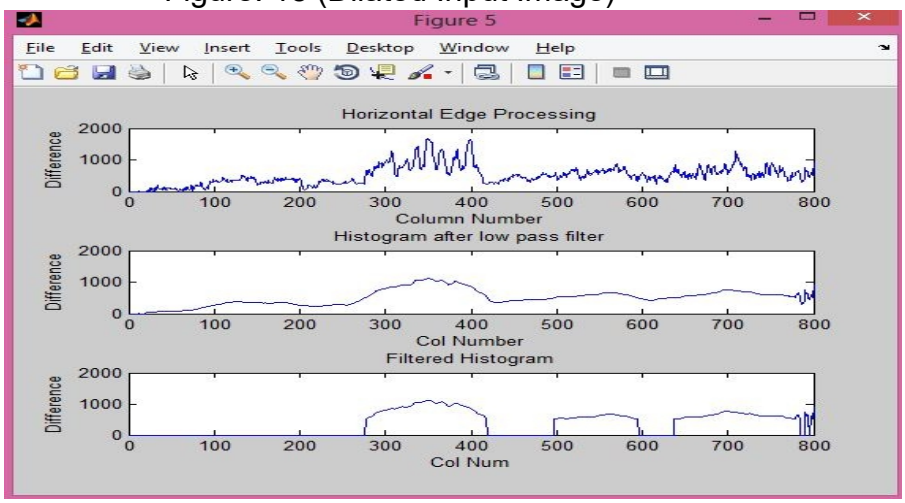


Figure: 19 (Horizontal Edge Processing)

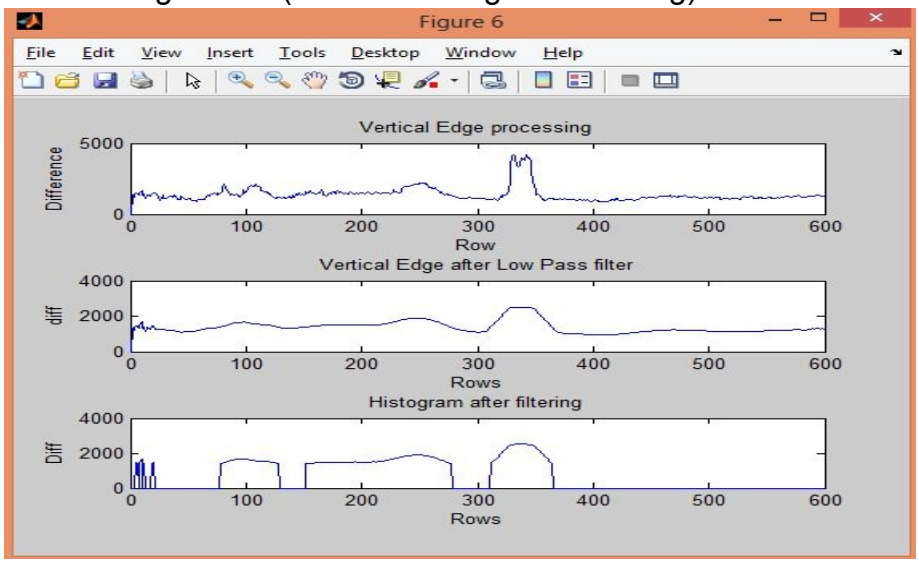


Figure: 20 (Vertical Image Processing)

5. The histogram values changes drastically between consecutive columns and rows. Therefore, to prevent loss of important information in upcoming steps, it is advisable to smooth out such drastic changes in values of histogram. For the same, the histogram is passed through a low-pass digital filter. While performing this step, each histogram value is averaged out considering the values on it right-hand side and left-hand side. This step is performed on both the horizontal histogram as well as the vertical histogram.
6. Then the unwanted regions with low histogram values and the region having high histogram values has a higher probability of containing the LP.
7. Out of the all found regions the one with the highest histogram values is considered.

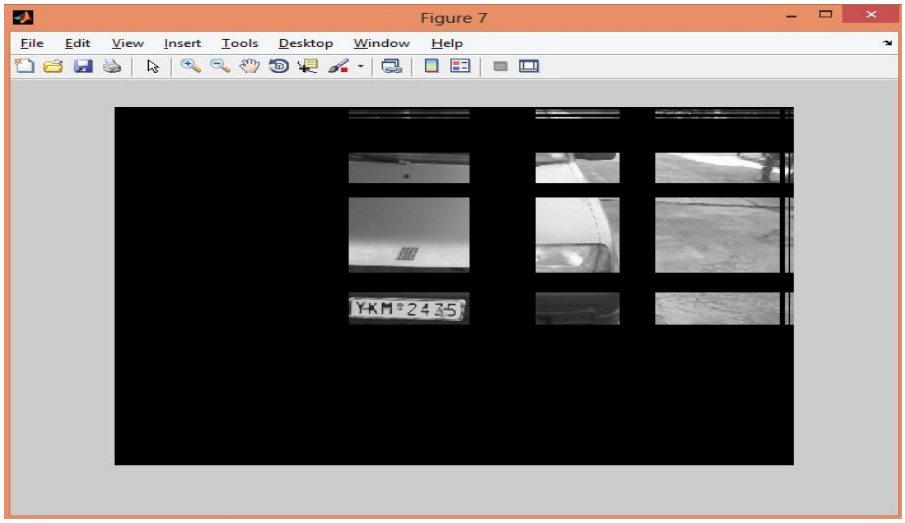


Figure: 21 (LP region candidates)

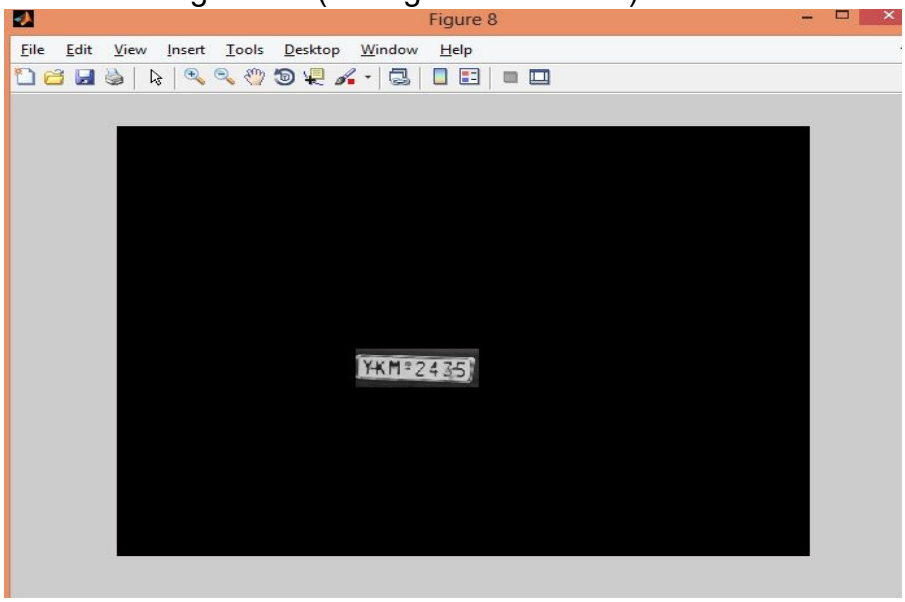


Figure: 22 (LP Detected)

8. MATLAB implementation

```
clc;
clear all;
close all;
imtool close all;
workspace;
I = imread('D:\Study\6th Sem\Image Processing\Project\Test\DSCN0410.jpg');
figure
imshow(I);
%I = imresize(I,[680 480]);
Igray = rgb2gray(I);
Igray = medfilt2(Igray);
[r,c] = size(Igray);
Idilate = Igray;
for i = 1:r
    for j = 2:c-1
        tmp = max(Igray(i,j-1),Igray(i,j));
```

```
        Idilate(i,j) = max(tmp,Igray(i,j+1));
    end
end
I = Idilate;
figure('name','Gray');
imshow(Igray);
figure('name','DilatedImage');
imshow(Idilate);
figure
imshow(I)
diff = 0;
sum = 0;
tot_sum = 0;
diff = uint32(diff);
%Horizontal Edge Processing
max_horz = 0;
maximum = 0;
for i = 2:c
    sum = 0;
    for j = 2:r
        if (I(j,i)>I(j-1,i))
            diff = uint32(I(j,i)-I(j-1,i));
        else
            diff = uint32(I(j-1,i)-I(j,i));
        end
        if (diff > 20)
            sum = sum + diff;
        end
    end
    horz1(i) = sum;
    if (sum > maximum)
        max_horz = i;
        maximum = sum;
    end
    tot_sum  = tot_sum + sum;
end
average = tot_sum / c;

figure
subplot(3,1,1);
plot(horz1);
title('Horizontal Edge Processing');
xlabel('Column Number');
ylabel('Difference');
%Smoothing horizonatal by low pass filter
sum = 0;
horz = horz1;
for i = 21:(c-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 low pass filter');
xlabel('Col Number');
ylabel('Difference');
%Filtering Horizontal Histogram
 for i = 1:c
     if (horz(i) < average)
         horz(i) = 0;
         for j = 1:r
             I(j,i) = 0;
         end
     end
 end
 subplot(3,1,3);
 plot(horz);
 title('Filtered Histogram');
```

```
xlabel('Col Num');
ylabel('Difference');
%Vertical Edge processing
diff = 0;
tot_sum = 0;
diff = uint32(diff);
maximum = 0;
max_vert = 0;
for i = 2:r
    sum = 0;
    for j = 2:c
        if (I(i,j)>I(i,j-1))
            diff = uint32(I(i,j)-I(i,j-1));
        end
        if (I(i,j)<=I(i,j-1))
            diff = uint32(I(i,j-1)-I(i,j));
        end
        if (diff > 20)
            sum = sum + diff;
        end
    end
    vert1(i) = sum;
    if (sum > maximum)
        max_vert = i;
        maximum = sum;
    end
    tot_sum = tot_sum + sum;
end
average = tot_sum / r;

figure
subplot(3,1,1);
plot(vert1);
title('Vertical Edge processing');
xlabel('Row');
ylabel('Difference');

%Smoothing Vertical Histogram
sum = 0;
vert = vert1;
for i = 21:(r-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('Vertical Edge after Low Pass filter');
xlabel('Rows');
ylabel('diff');
%Dyanamic Threshold
for i = 1:r
    if (vert(i) < average)
        vert(i) = 0;
        for j = 1:c
            I(i,j) = 0;
        end
    end
end
subplot(3,1,3)
plot(vert);
title('Histogram after filtering');
xlabel('Rows');
ylabel('Diff');
figure
imshow(I);
%Probable candidates for the LP
j = 1;
for i = 2:c-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:r-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,col_size] = size(column);
    if(mod(col_size,2))
        column(col_size+1) = c;
    end

    [tmp,row_size] = size(row);
    if(mod(row_size,2))
        row(row_size+1) = r;
    end
    %Region of interset
    for i = 1:2:row_size
        for j = 1:2:col_size
            if
   (~((max_horz>=column(j)&&max_horz<=column(j+1))&&(max_vert>=row(i)&&max_vert<=row(i+1))
   ))
                for m = row(i):row(i+1)
                    for n = column(j):column(j+1)
                        I(m,n) = 0;
                    end
                end
            end
        end
    end

    figure
    imshow(I);
```

- **Using Top Hat Transform**

$$HAT\ (A) = A - (A \circ B)$$

Where A is input image and B is the structuring element.

$$A \circ B = (A \ominus B) \oplus B$$

$$A \bullet B = (A \oplus B) \ominus B$$

1. Input image is same as the input image in the method of LP detection using the Morphological Analysis (page 5).
2. Vertical gradient of the given input image is calculated. Here f is the input image.

$$g_v(i,j) = | f(i,j+1) - f(i,j) |$$

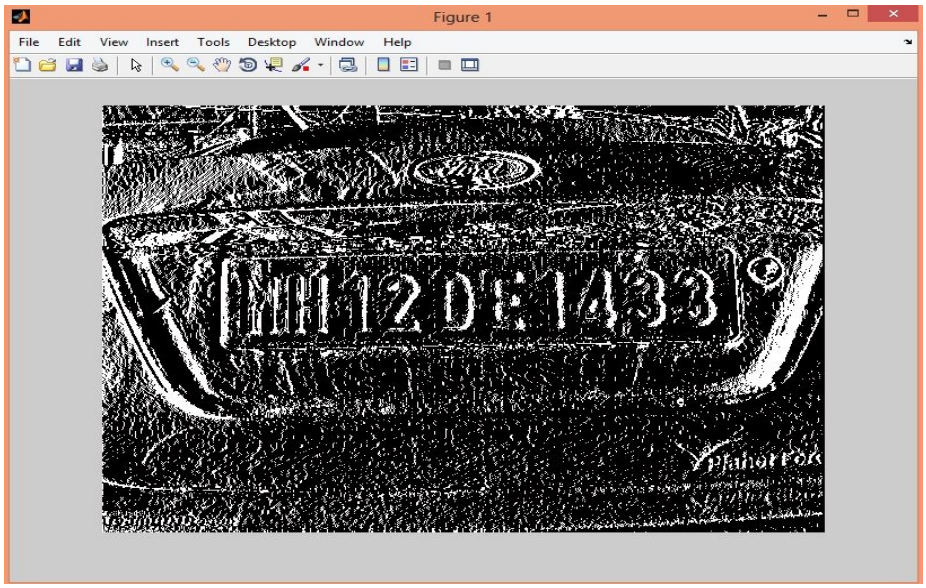


Figure: 23 (Vertical Gradient)

3. Hat transform of the above image is calculated.
4. Horizontal candidates are calculated by horizontal histogram processing.

$$T_H(i) = \sum_{j=1}^{n} g_v(i,j)$$

5. Smooth the Horizontal Candidates.

$$T'_H(i) = \frac{1}{k}\left\{ T_H(i) + \sum_{j=1}^{w} \begin{bmatrix} T_H(i-j)h(j,\sigma) + \\ T_H(i+j)h(j,\sigma) \end{bmatrix} \right\}$$

where $h(j,\sigma) = e^{-(j\sigma)^2/2}$, $k = 2\sum_{j=1}^{w} h(j,\sigma) + 1$,

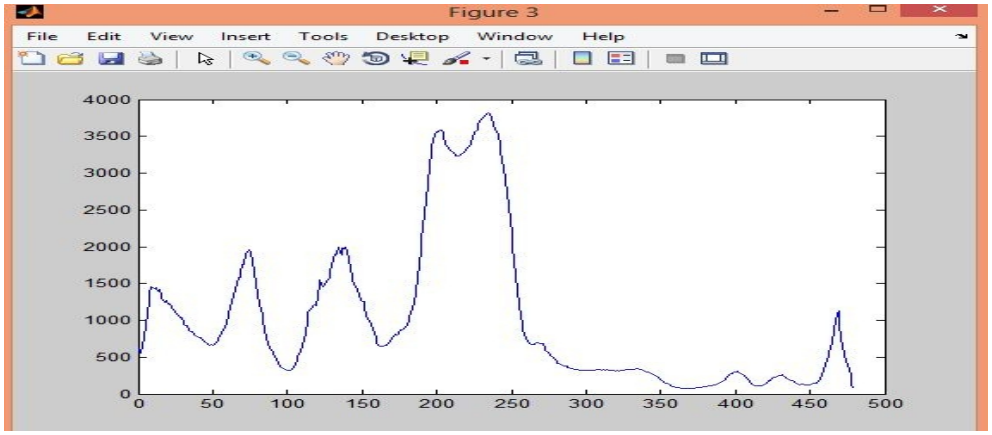


Figure: 24 (Smoothed Horizontal Candidates)

6. Find the threshold (T = threshold, aver = average, t = 3.5)

$$T = t * aver$$

7. Calculate the candidate region by examining the smoothed histogram of the horizontal candidates i.e. all those rows whose value is greater than the above threshold calculated above.

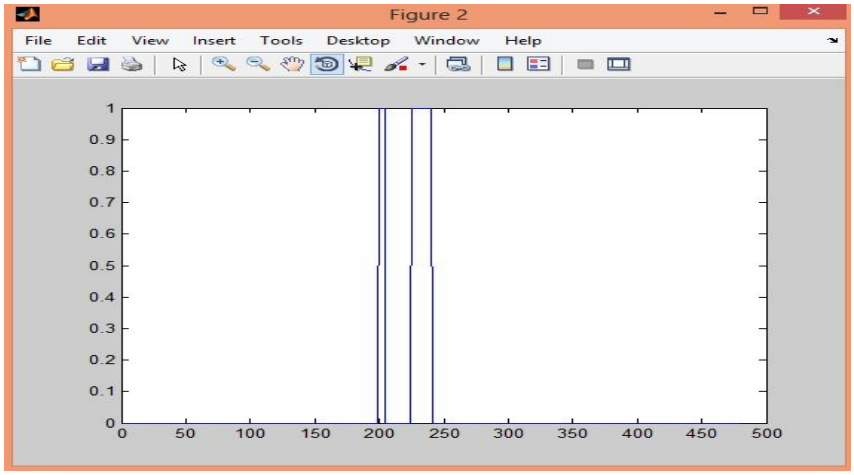


Figure: 25 (Candidate Rows)

8. Amalgamate the original image based on the candidate rows calculated in above step to find the License Plate region.

Figure: 26 (Detected License Plate)

9. The above image is converted into binary and the characters are segmented using the Bounding Box Method.
10. MATLAB implementation

```
I = imread('d1.jpg');
G = imresize(I, [480 640]);
H = rgb2gray(G);
H = medfilt2(H);
[m ,n] = size(H);
g = zeros(m,n);
for i=1:m
    for j=1:n-1
        g(i,j) = abs(H(i,j+1)-H(i,j));
    end
end
figure
imshow(g);

se = strel('disk',1);

H11 = imerode(H,se);
H12 = imdilate(H11, se);
H13 = imsubtract(H,H12);
figure
imshow(H13);

H1 = imerode(g,se);
H2 = imdilate(H1, se);
H3 = imsubtract(g,H2);%hat
T = zeros(m);
for i = 1:m
    sum = 0;
    for j = 1:n
        sum = sum + H3(i,j);
    end
    T(i,i) = sum;
end
T = diag(T);
figure
%plot(T);
imshow(H3);
w = 8;
sig = 0.05;
h = zeros(w);
for i = 1:w
    h(i,i) = exp(-((i * sig) ^2) / 2);
end
h = diag(h);
k = 0;
sum = 0;
for i = 1:w
    sum = sum + h(i,1);
end
k = 2 * sum + 1;
Th = zeros(m);
for i = 1:m
    sum = 0;
    for j = 1:w
        if (i - j > 0) && (i + j < m)
            sum  = sum + T(i-j,1) * h(j,1) + T(i+j,1) * h(j,1);
        end
        sum  = sum + T(i,1);
        Th(i,i) = sum / k;
    end
end
Th = diag(Th);
%figure
%plot(Th);
t = 3.5;
```

```
avg = 0;
[x y] = size(Th);
for i = 1:x
    avg = avg + Th(i,1);
end
avg = avg / x;
T1 = t * avg;
stack = zeros(1,x);
for i = 1:x
    if Th(i,1) >= T1
        stack(1,i) = 1;
    end
end
fst = 0;
snd = 0;
%figure
%plot(stack);
flg = 0;
i = 1;
while i <= x
    flg = 0;
    for j = i:i+100
        if j <= x
            if stack(1,i) ~= 1
                flg = 1;
            end
        end
        if flg == 1
            break;
        end
    end
    if ((j == i+100) && (flg == 0))
        fst = i;
        snd = j;
        figure
        imshow(H(i:j,:));
        imwrite(H(i:j,:),'f1.jpg','jpg');
        i = i + 50;
    end
    i = i + 1;
end

img = imread('f1.jpg');
level = graythresh(img);
img = im2bw(img,level);
%imshow(img);
img = ~img;
%imshow(img);
img = bwareaopen(img,100);
I2=regionprops(img,'BoundingBox','Image');
for i = 1:length(I2)
   s = 'segment';
 %   s1 = 'Images\';
    s = strcat(s, int2str(i));
    %s1 = strcat(s1,s);
%    figure('name', s)
    s = strcat(s,'.jpg');
%    imshow(I2(i).Image)
    %imwrite(Iprops(i).Image, s,'jpg');
end
```

- **Using the Canny Edge Detection Method[4]**

  The process of canny edge detection can be broken down to 5 different steps:

  1. Apply Gaussian filter to smooth the image in order to remove the noise

2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response the edge detection
4. Apply double threshold to determine potential edges
5. Finalize the detection of the edges by suppressing all the other edges

After the detection of the edges all the steps are same as the method involving Sobel Edge Detector Method.

6. MATLAB implementation

```
srcFiles = dir('D:\Study\6th Sem\Image
Processing\Project\Test\*.jpg');  % the folder in which ur images
exists
for i = 1 : length(srcFiles)
    filename = strcat('D:\Study\6th Sem\Image
Processing\Project\Test\',srcFiles(i).name);
    I= imread(filename);
    G = imresize(I, [480 640]);
    H = rgb2gray(G);
    %imshow(H);
    %[m n] = size(G);
    %H = zeros(m , n);
    %n = n / 3;
    %for i = 1:m
    %    for j = 1:n
    %        H(i,j) = 0.114 * G(i,j,1) + 0.587*G(i,j,2) + 0.299 *
G(i,j,3);
    %    end
    %end
    %imshow(H);
    H1 = edge(H, 'sobel');
    H1 = double(H1);
    H1 = conv2(H1, [1 1:1 1]);
    %%
    %%
    %figure
    %imshow(H1);
    se1 = strel('line',1, 0);
    sr2 = strel('line',1,90);
    H2 = imdilate(H1, sr2);
    H2 = imdilate(H2, se1);
    H2 = imfill(H2, 'holes');
    H2 = bwareaopen(H2, 100);
    H2 = imclearborder(H2, 8);
    se = strel('disk',1);
    H3 = imerode(H2, se);
    H4 = imerode(H3, se);
    %size(H)
    %size(H4)
    H4 = H4(:,2:641);
    H4 = medfilt2(H4);
    %figure
    %imshow(H4);
    H4 = immultiply(H4, H);
```

```
        figure
        imshow(H4);
        level = graythresh(H4);
        H4 = im2bw(H4, level);
        H4 = bwareaopen(H4, 100);
        H4 = ~H4;
        %imshow(H4);
        H4 = bwareaopen(H4, 100);
        %figure
        %imshow(H4);
        I2=regionprops(H4,'BoundingBox','Image');
        for j = 1:length(I2)
           s = 'segment';
         %   s1 = 'Images\';
            s = strcat(s, int2str(j));
            %s1 = strcat(s1,s);
           % figure('name', s)
            s = strcat(s,'.jpg');
            %imshow(I2(i).Image)
            %imwrite(Iprops(i).Image, s,'jpg');
        end
end
```

RESULT COMPARISON

Method Name	No. of Input Image	No. of Images With Detected LP	Accuracy (Percentage)	Error (Percentage)
Sobel Edge Detector	50	39	78%	22%
Histogram Processing	50	36	72%	28%
Mathematical Morphology	20	14	70%	30%
Canny Edge Detection Method	67	43	64%	36%
Top Hat Transform	53	35	66%	34%

CONCLUSION

Out of all the methods used for the LP detection Sobel edge detector has produced best results. The other methods especially Mathematical Morphology analysis is close to the Sobel edge detector method to detect the LP and once the LP has detected the characters are segmented using the Bounding Box method.

REFERENCES

1. License plate recognition From still images and Video sequences:A Survey (Christos-Nikolas E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Ioannis D. Psoroulas, Vassili Loumos)
2. Detecting License Plate Based on Top-hat Transform and Wavelet Transform (Feng Yang，Fan Yang)
3. Recognition of vehicle plate number in Matlab (Ragini Bhat , Bijender Mehandia)
4. Wikipedia