**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| Sr. No. | Topic | Page no |
| 1 | Introduction | 1 |
| 2 | Purpose & Tools Used | 7 |
| 3 | Fundamentals of image processing | 8 |
| 4 | RGB format and YCbCr format | 9 |
| 5 | Image types  ·         RGB images  ·         Intensity images  ·         Indexed images  ·         Binary images | 12 |
| 6 | Matlab Implementation  ·         Flowchart  ·         Code  ·         Output | 14 |

**VEHICLE NO. PLATE DETECTOR**

**Introduction:**

With increasing number of vehicles on roads, it is getting difficult to manually enforce laws and traffic rules for smooth traffic flow. Toll-booths are constructed on freeways and parking structures, where the car has to stop to pay the toll or parking fees. Also,  Traffic  Management  systems  are  installed  on  freeways  to  check  for  vehicles moving at speeds not permitted by law. All these processes have a scope of improvement. In the center of all these systems lies a vehicle. In order to automate these processes and make them more effective, a system is required to easily identify a vehicle. The important question here is how to identify a particular vehicle? The obvious answer to this question is by using the vehicle’s number plate.

Vehicles in each country have a unique license number, which is written on its license plate. This number distinguishes one vehicle from the other, which is useful especially when both are of same make and model. An automated system can be implemented to identify the license plate of a vehicle and extract the characters from the region containing a license plate. The license plate number can be used to retrieve more information about the vehicle and its owner, which can be used for further processing. Such an automated system should be small in size, and portable.

Various license plate detection algorithms have been developed in past few years. Each of these algorithms has their own advantages and disadvantages. This project described the method in which license plate is detected using confidence related predictions. As multiple detections are available for single license plate, post-processing methods are applied to merge all detected regions. In addition, trackers are used to limit the search region to certain areas in an image. It suggests a different approach of detection using binarization and elimination of unnecessary regions from an image. In this approach, initial image processing and binarization of an image is carried out based on the contrast between characters and background in license plate. After binarizing the image, it is divided into different black and white regions. These regions are passed through elimination stage to get the final region having most probability of containing a number plate.

**Purpose of this Project:**

The main purpose of this project is to detect a license plate from an image provided by a camera. An efficient algorithm is developed to detect a license plate in various luminance conditions. This algorithm extracts the license plate data from an image and provides it as an input to the stage of Car License Plate Recognition. This algorithm can be downloaded onto Texas Instrument’s TMS320DM6437 digital video development platform. The image of a vehicle is given as an input from the camera. Extracted image of the number plate can be seen on television for verification purpose.

**Tools used**

>  Windows 8.1 (operating system)

>  MATLAB**R2014a**

**FUNDAMENTALS  OF  IMAGE PROCESSING:**

An image is used to convey useful information in a visible format. An image is nothing but an arrangement of tiny elements in a two-dimensional plane. These tiny elements are called Pixels. A large number of pixels combine together to form an image, whether small or large.

Each pixel represents certain information about the image, like color, light intensity and luminance. A large number of such pixels combine together to form an image. Pixel is the basic element used to describe an image. Mostly, each pixel in an image is represented in either RGB (Red Green Blue) format or YCbCr format. In case of an RGB image, all the three components, namely R, G and B combine together to convey information about the color and brightness of a single pixel. Each component consumes certain memory space during image processing.

In case of a YCbCr image, each pixel in an image is represented as a combination of Y and Cb/Cr values. Here, Y stands for luminance, which describes light intensity, and Cb/Cr stands for chroma component, which describes color information for an image. Over the time, it has been found that YCbCr components of an image convey sufficient amount of information compared to its counter parts RGB, with less amount of memory space. This is a major advantage nowadays, as most of the applications require sufficient information at very high speed and less storage.

**RGB Format and YCbCr Format:**

These are two formats in which image can be studied. It depends on the application for which we are using images.

**RGB Format:**

In case of an RGB image, each pixel is represented by three different components R, G and B. Each of these components requires at least 8 bits for their storage. In general, a single pixel may require upto 8 \* 3 bits for its storage. An example of a representation of single pixel in RGB format is shown below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R | G | B | R | G | B |

                                    Fig. 1    ( Representation of pixels in RGB format).

The value of R, G and B each ranges from 0-255. A value of (0, 0, 0) represents a black pixel, (255, 0, 0) represents a red pixel and (0, 255, 0) represents a green pixel. So,

8 bits are required to store value for a single component.

**YCbCr Format:**

In contrast to RGB format, the YCbCr format is available with various kind of interleaving.  For  example,  a  4:2:2  YCbCr  format  suggests  that  a  single  pixel  is represented by two components, Y and C. Cb and Cr components are interleaved among the pixels. So if one pixel is represented by a combination of Y and Cb, the adjacent pixel will be represented by a combination of Y and Cr. Even if the Cb and Cr components are interleaved, its effect is not visible to human eye.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Y | Cb | Y | Cr | Y | Cb |

                                         Fig.2   (representation of pixels in YcbCr Format)

Values for Y, Cb and Cr vary from 0-255. Thus, to store a single pixel, the amount of storage required is 8 \* 2 bits, which is less compared to that required by RGB format. For this project, Texas Instrument’s EVM320DM6437 can be used for license plate detection. The kit contains internal buffers to store the incoming frames of video. The format for the type of storage is shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Cb | Y | Cr | Y |
| Cb | Y | Cr | Y |
| Cb | Y | Cr | Y |

                                  Fig.3   ( A part of frame buffer storage for input video frames).

From  the  above  image,  it  is  seen  that  the  storage of  frame  starts  with  a  C component and then a Y component. Therefore, at the 0th  location, one can find the C component while at the 1st  and alternate locations of Frame Buffer, one can find the Y component.

**IMAGE TYPES:**

**Digital Images and Basic Types:**

An image can be defined as a two-dimensional function, ***f(x,y)***where ***x***and ***y***are the spatial coordinates and the amplitude value ***f***represents the intensity (or color) of the image at that point (pixel).When ***x, y***, and ***f***are discrete values, we have a digital image. Digital images are composed of **pixels**arranged in a rectangular array with a certain height (rows) and width (columns). Each pixel may consist of one or more bits of information (8 bits being the most common), representing the intensity of the image

at that point (or color information encoded as RGB triples). The following are the four basic types of digital images:

• RGB (True Color) Images

• Intensity (Gray Scale) Images

• Binary Images

• Indexed Images

**1) RGB Images:**

It is possible to construct (almost) all visible colors by combining the three primary colors Red, Green and Blue, because the human eye has only three different color receptors, each of them sensible to one of the three colors. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately *350,000*colors. A RGB color image is a multi-spectral image with one band for each color red, green and blue, thus producing a weighted combination of the three primary colors for

**2) Intensity Images:**

A grayscale (or gray level) image is an image in which the only colors are shades of gray. In gray' color the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image. Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of grey from black to white scale image.

**3) Binary Images**

Binary images are images whose pixels have only two possible intensity values. They arenormally displayed as black and white. Numerically, the two values are often 0 for black, and either 1 or255 for white. Binary images are often produced by thresholding a grayscale or color image, in order to separate an object in the image from the background. The color of the object (usually white) is referred to as the foreground color. The rest (usually black) is referred to as the background color. However, depending on the image that is to be thresholded, this polaritymight be inverted, in such case the object is displayed with 0 and the background is with a non-zero value.

**4) Indexed Images:**

Indexed images are visually similar to RGB images but the way of representing them is different. An indexed image consists of a data matrix, X, and a color map matrix, map. The color map matrix is anm-by-3 array containing values in the range [0, 1]. Each row of map specifies the red, green, and blue components of a single color. An indexed image uses direct mapping of pixel values to color map values. The color of each image pixel is determined by using the corresponding value of X as an index into map. The value 1 points to the first row in map, the value 2 points to the second row, and so on.

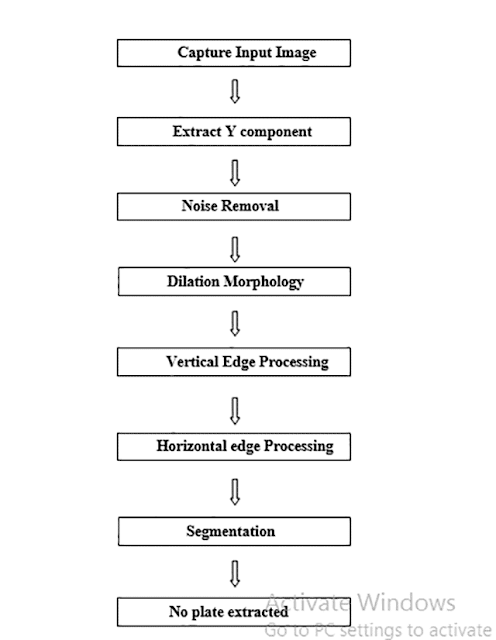
[**Watch Live Project Simulation in this Video**](https://www.youtube.com/watch?v=iRUJYOs76EU)

**MATLAB IMPLEMENTATION:**

Here we describe the implementation of License Plate Detection algorithm using MATLAB. MATLAB is a very powerful software tool used to implement the tasks that  require  extensive  computation. It provides easy and quicker implementation of algorithms compared to C and C++. The key feature in MATLAB is that it contains a rich library functions for image processing and data analysis. This makes MATLAB an ideal tool for faster implementation and verification of any algorithm before actually implementing it on a real hardware. Sometimes, debugging of errors on actual hardware turns out to be a very painful task. MATLAB provides an easy approach for debugging and correction of errors in any algorithm. Other than this, MATLAB contains many features including workspace, plot, **imread**, **imhi**st, imshow, etc. for data analysis and image processing, which makes it a better choice over other software languages like C and C++.

Considering the above advantages, the writer of this project initially implemented an algorithm for License Plate Detection using MATLAB. The algorithm initially used various inbuilt functions and implemented few user defined routines related to image processing. Once the algorithm was developed, it was verified with multiple input images containing car  number  plates.  The  input  images  contained  number plates  that  were aligned horizontally as well as at some angle from horizontal axis. Once the algorithm was completely verified, the in-built functions of MATLAB were replaced by user defined functions. A flow-chart showing the basic implementation of algorithm is shown on next page.

**FLOW CHART of ALGORITHM:**



**Convert a Colored Image into Gray Image:**

The algorithm described here is independent of the type of colors in image and relies mainly on the gray level of an image for processing and extracting the required information. Color components like Red, Green and Blue value are not used throughout this algorithm. So, if the input image is a colored image represented by 3-dimensional array in MATLAB, it is converted to a 2-dimensional gray image before further processing. The sample of original input image and a gray image is shown below:

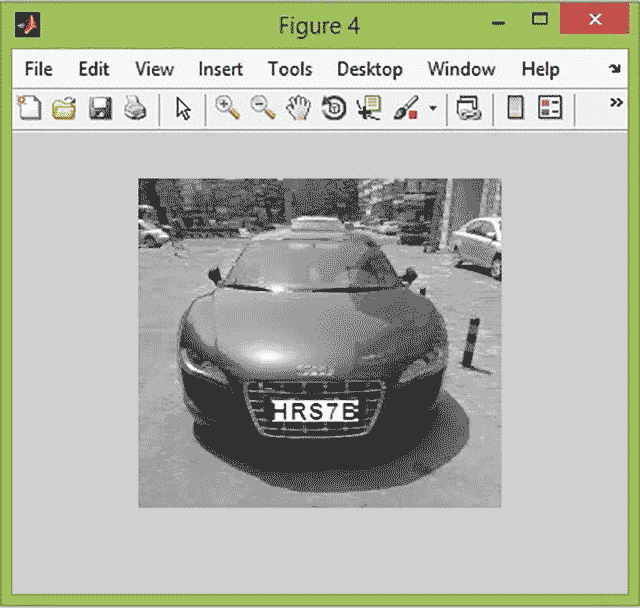
|  |
| --- |
| Matlab Image Processing Car number plate detector project code and report |
| Fig 5  RGB Image |

|  |
| --- |
| Matlab Image Processing:Car No. Plate detector Project Code+Report |
| Fig 6 Gray scale image |

**Dilate an Image:**

Dilation is a process of improvising given image by filling holes in an image, sharpen the edges of objects in an image, and join the broken lines and increase the brightness of an image. Using dilation, the noise with-in an image can also be removed. By making the edges sharper, the difference of gray value between neighboring pixels at the edge of an object can be increased. This enhances the edge detection. In Number Plate Detection, the image of a car plate may not always contain the same brightness and shades. Therefore, the given image has to be converted from RGB to gray form. However, during this conversion, certain important parameters like difference in color, lighter edges of object, etc. may get lost. The process of dilation will help  to nullify such losses.

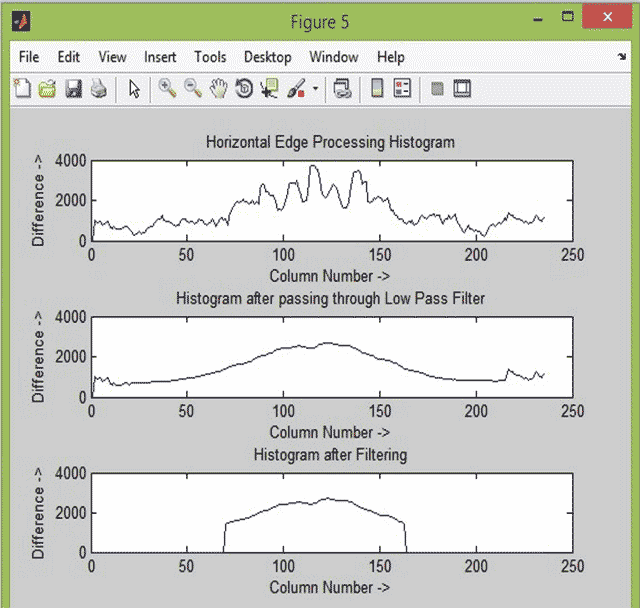
                                    Fig.7    (Image after dilation)



**Horizontal and Vertical Edge Processing of an Image:**

Histogram is a graph representing the values of a variable quantity over a given range. In this Number Plate Detection algorithm, the writer has used horizontal and vertical histogram, which represents the column-wise and row-wise histogram respectively. These histograms represent the sum of differences of gray values between neighboring pixels of an image, column-wise and row-wise. In the above step, first the horizontal histogram is calculated. 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. 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

                                    Fig.8      (Horizontal image processing graph)

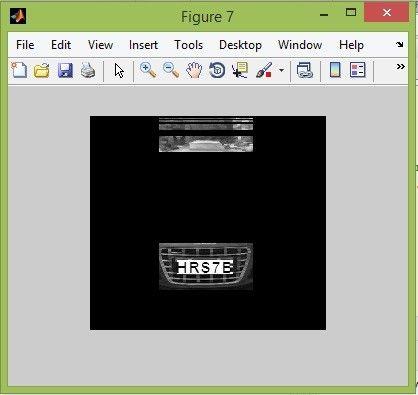


**Passing Histograms through a Low Pass Digital Filter:**

Referring to the figures shown below**,**one can see that 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. Below are the figures showing the histogram before passing through a low-pass digital filter and after passing through a low-pass digital filter.

**Filtering out Unwanted Regions in an Image**

                                                     (fig.10 image to be filtered out)



Once the histograms are passed through a low-pass digital filter, a filter is applied to remove unwanted areas from an image. In this case, the unwanted areas are the rows and columns with low histogram values. A low histogram value indicates that the part of image contains very little variations among neighboring pixels. Since a region with a license plate contains a plain background with alphanumeric characters in it, the difference in the neighboring pixels, especially at the edges of characters and number plate, will be very high. This result in a high histogram value for such part of an image Therefore, a region with probable license plate has a high horizontal and vertical histogram values. Areas with less value are thus not required anymore. Such areas are removed from an image by applying a dynamic threshold. In this algorithm, the dynamic threshold is equal to the average value of a histogram. Both horizontal and vertical histograms are passed through a filter with this dynamic threshold. The output of this process is histogram showing regions having high probability of containing a number plate. The filtered histograms are shown below here:

**Segmentation:**

The next step is to find all the regions in an image that has high probability of containing a license plate. Co-ordinates of all such probable regions are stored in an array. The output image displaying the probable license plate regions is shown below.

**Region of Interest Extraction:**

The output of segmentation process is all the regions that have maximum probability of containing a license plate. Out of these regions, the one with the maximum histogram value is considered as the most probable candidate for number plate. All the regions are processed row-wise and column-wise to find a common region having maximum horizontal and vertical histogram value. This is the region having highest probability of containing a license plate. The image detected license plate is shown below: This algorithm was verified using several input images having resolution varying from 680 \* 480 to 1600 \* 1200. The images contained vehicles of different colors and varying intensity of light. With all such images, the algorithm correctly recognized the number plate. This algorithm was also tried on images having number plate aligned at certain angle (approximately 8-10 degree) to horizontal axis. Even with such images, the number plates were detected successfully.

|  |
| --- |
| final output - Matlab Image processing project code and report |
| Final output -Car number plate detector |

**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);

**Future Implementation:**

This project can detect vehicle number plate and can extract number from it. But there are certain assumptions while doing this project. We can improve its quality based on certain parameters given below.

·         We can make improvements in terms of text reorganization, i.e we will be able to extract text in any written format. So it will help to catch culprits who misguide security system by changing format of number plate.

·         We can configure it with internet system to update the criminal records whenever found, to the next station through websites.

**Conclusion:**

Now we have take a image of car no. HRS7B. First we extracted the Y component by converting it to gray image. Then we remove the noise and performed dilation. Horizontal edge processing gives maximum value from column no 70-120 which is extracted by passing through filter. Similarly vertical image processing gives maximum value from column 125-175 in histogram. Finally region of maximum probability of no. plate is extracted and text is also extracted.

**References:**

During project work ,sources which helped me  get through this project are given below.

·         [**http://www.mathwork.com**](http://www.mathwork.com/)  - Official Matlab website.

·         Basics of Digital Image Processing using MATLAB by Rafael C. Gonzalez