CMSA3657: Project Work

Automatic License Plate Detection and Recognition

DEPARTMENT OF COMPUTER SCIENCE

ST. XAVIER'S COLLEGE (AUTONOMOUS), KOLKATA

UNDER THE GUIDANCE OF PROF. ANAL ACHARYA

Certificate of Completion

This is to certify that the following students have completed
The final year project titled
AUTOMATIC LICENSE PLATE DETECTION AND RECOGNITION

_____

Arunabha Goswami(3-15-09-0552)

_____

Abhinaba Audhya(3-15-09-0553)

_____

Sumalya Bhattacharya(3-15-09-0556)

Of the Department Of Computer Science
St. Xavier's College, Kolkata

_____ _____

(Head of Department) (Project Mentor)

Prof.A.Acharya Prof.A.Acharya

_____

(External Examiner)

Acknowledgment

We are very much grateful to the Department of Computer Science to give us opportunity to work on automatic license plate recognition. We express our heartfelt gratitude to Prof. Anal Acharya for his kind supervision and valuable advice to complete the present project work. In this context we are also grateful to all the other professors of the Dept. of Computer Science for giving us constant inspiration during the course of this work.

Last but not the least we are also thankful to all the 3rd year Computer Science Hons Students (2011-2012 batch) for their encouragement to finish this work.

Contents

INTRODUCTION

During the recent years, Intelligent Transportation Systems (ITS) are having a wide impact in people's life as their scope is to improve transportation safety and mobility and to enhance productivity through the use of advanced technologies. ITS are made up of 16 types of technology based systems. These systems are divided into intelligent infrastructure systems and intelligent vehicle systems. One of such feature is automatic license plate recognition (ALPR), which grew popular with the advancements in the field of optical character recognition (OCR).

Developing algorithms to read text automatically from an image has always been a subject of interest. Earlier character recognition technique has been used in structured processes like bank check processing. With the development of better scanners and the gain in computer processing power, the interest for optical character recognition has grown. Today there exists very advanced OCR programs that can read almost all written text. The possibility to read structured text in documents has turned the interest to the ability to read texts and symbols in other areas. One such area is automatic license plate recognition, where a digital camera is used to take photos of passing vehicles to make it possible to identify them automatically.

ALPR has gained much interest during the last decade along with the improvement of digital cameras and the gain in computational capacity. An automated LPR system can be used in many situations from speed enforcement and toll collection to management of parking lots where the ability to accurately recognise the identity of each car provides the number of cars and parking time of each car.

In some countries, LPR systems installed on country borders automatically detect and monitor border crossings. Each vehicle can be registered in a central database and compared to a black list of stolen vehicles. In traffic control, vehicles can be directed to different lanes for a better congestion control in busy urban communications during the rush hours.

In this paper a computer vision and character recognition algorithm for license plate recognition is presented to be used as a core for intelligent infrastructure like electronic payment systems (toll payment, parking fee payment), freeway and arterial management systems for traffic surveillance. Moreover, as increased security awareness has made the need for vehicle based authentication technologies extremely significant the proposed System may be employed as access control system for monitoring of unauthorized vehicles entering private areas.

In entrance gate, number plates are used to identify the vehicles. When a vehicle enters an input gate, number plate is automatically recognized and stored in database and black-listed number is not given permission. When a vehicle later exits the place through the gate, number plate is recognized again and paired with the first-one stored in the database and it is taken a count. Automatic number plate recognition systems can be used in access control. For example, this technology is used in many companies to grant access only to vehicles of authorized personnel.

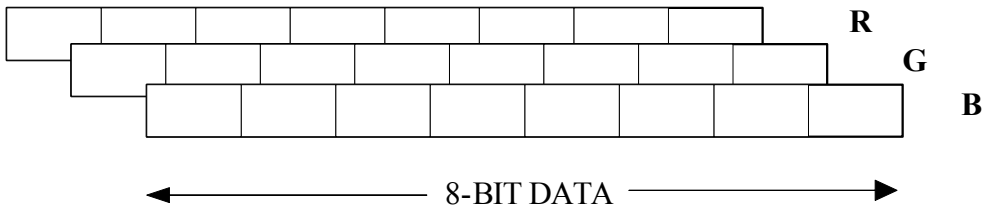
FUNDAMENTALS OF IMAGE PROCESSING

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 (Picture Elements).

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 CMYk (Cyan Magenta Yellow black) format. The RGB format is used to print an image on the screen, while the CMYk format is used to print something on paper. Therefore, we limit our discussion to only the RGB format.

RGB 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. Each of these components requires at least 8 bits for their storage. A single pixel may require up to 8 * 3 bits for its storage. An example of a of single pixel in RGB format is shown below.



ALGORITHM IN SHORT

Algorithm for ANPR system

1. Input image.
2. Convert image into binary.
3. Segmentation.
4. Number identification.
5. Save to file in given format.

Input Image from file

1. Capture image using a digital camera.
2. Store the captured image into a image file for further processing.

Convert image into binary

1. Enhance the contrast of the input image.

    If pixel value < Pre-defined small value

        set pixel value to 0

    Else If pixel value > Pre-defined large value

        set pixel value to 255

    Else

        No change.

2. Convert image into grayscale.
3. Calculate appropriate threshold value for the image and convert the image into binary image using the calculated threshold.

Segmentation

1. Apply an algorithm to find areas which can be probable candidates for the license plate.
3. Separate each candidate portion from the image and input it in the OCR module.

Number identification

1. Create the template file from the stored template images.
2. Recognise and store all the connected components found in the page in an array.
3. Remove all the connected components containing fewer pixels than a pre-defined value.
4. Select a connected component. Resize it to the size of template.
5. Compare each character with the templates. Return the best matched character.
6. After scanning the whole line, match it with the general format of license plate.
7. If match is found, save it in a string. Otherwise, pick another segment and repeat all the steps.

Save to file in given format

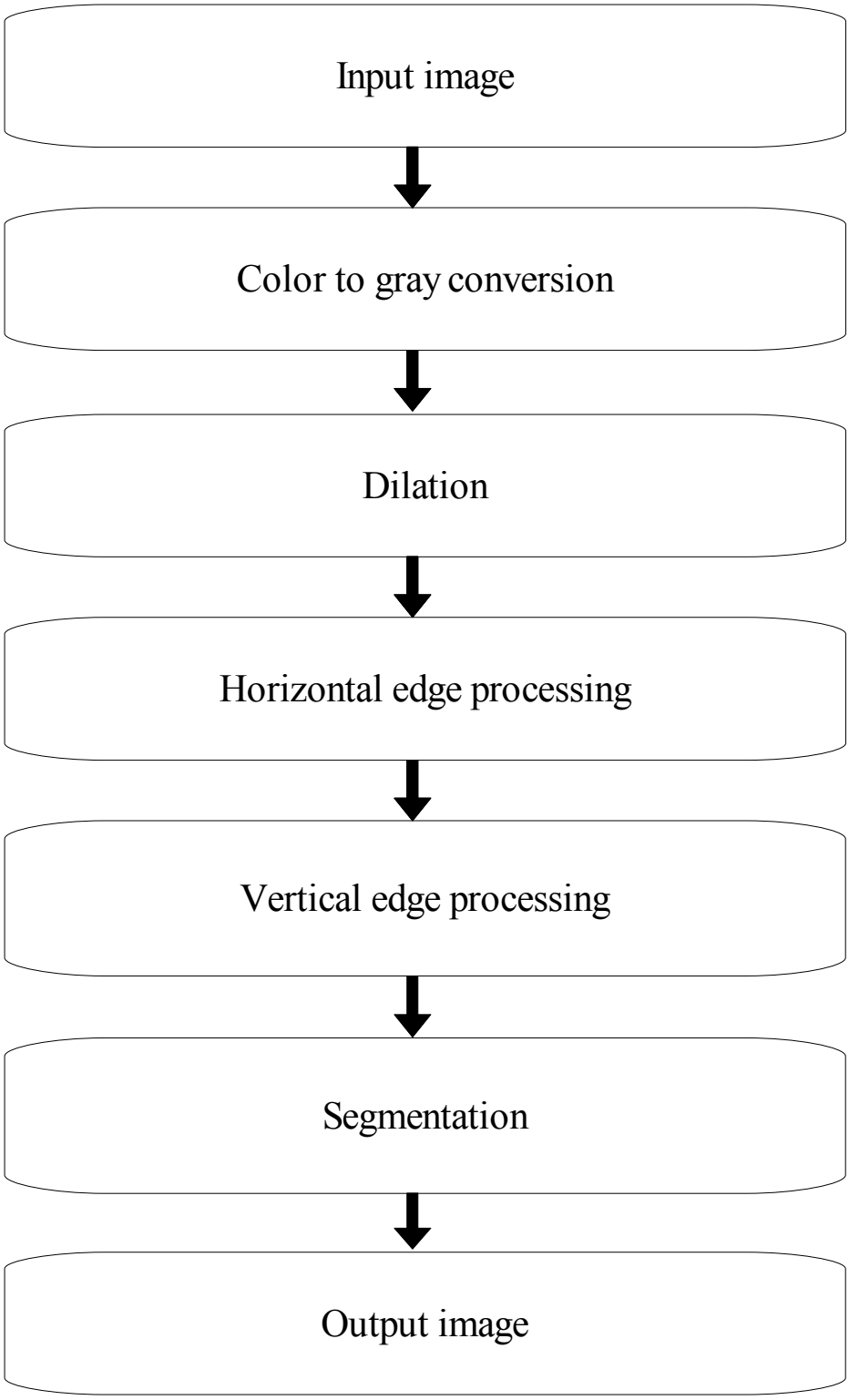
1. Open a text file in write mode.
2. Store the string obtained from the number identification process to text file in given format.
3. Close the file.

LPR - LICENSE PLATE RECOGNITION

Introduction

The LPR module of the algorithm is used to find one or more probable sub-section of the original input image, one of which is supposed to contain the number plate region. All of these regions are then fed into the Optical Character Recognition Module for character analysis and letter extraction. Therefore it is necessary that the LPR module finds the number plate region successfully in the first place. Over the years, a lot of algorithms are developed to detect the license plate area of an image – each with their own advantages and disadvantages. In this algorithm, we have used an approach that is both easy to implement and has a high success rate. The algorithm calculates the sum of differences of gray values between neighboring pixels of an image, column-wise and row-wise. After that, only the regions with larger than average sum-of-difference values are kept intact, while all the other areas are converted into black pixels – thus segmenting the input image into probable candidate region.

A flowchart showing the basic implementation of the License Plate Recognition module is shown in the next page -



Flowchart showing License Plate Recognition Using MATLAB

The steps of implementing License Plate Detection algorithm in MATLAB are described below.

Converting 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. Also, the contrast of the image is further increased, as the algorithm depends on the difference of pixel values of light and dark area. The sample of original input image and a gray image is shown below:

<Original image>

<Gray scale image>

Dilating the 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.

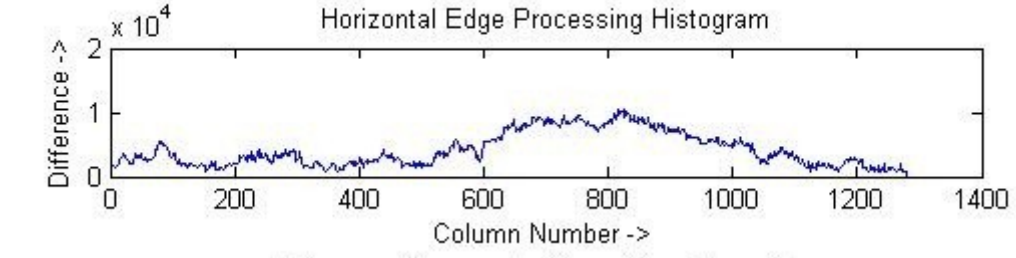
<Dilated image>

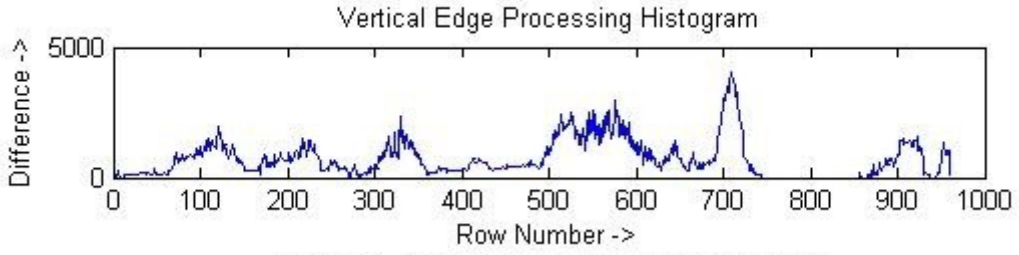
Horizontal and Vertical Edge Processing of an Image

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.

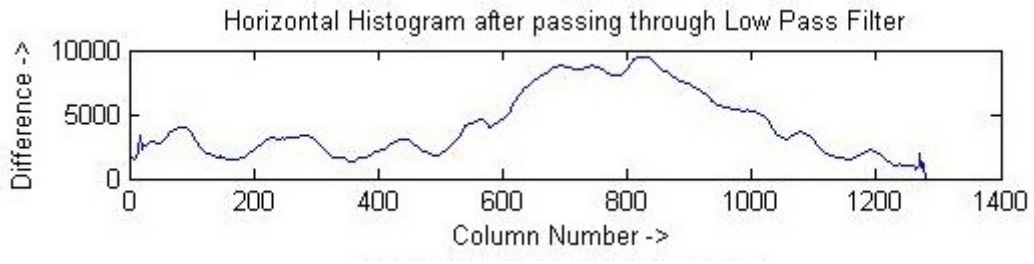


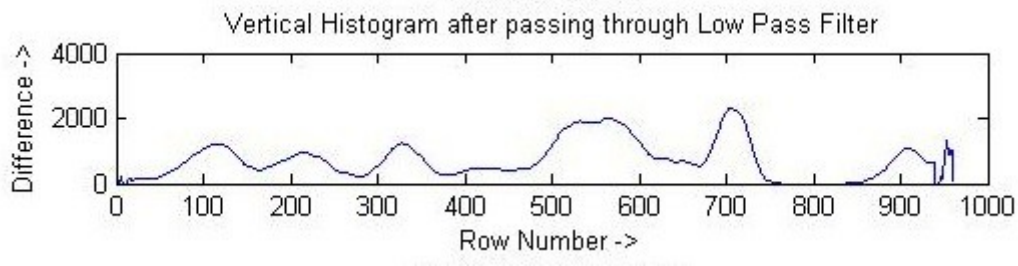


Passing Histograms through a Low Pass Digital Filter

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.



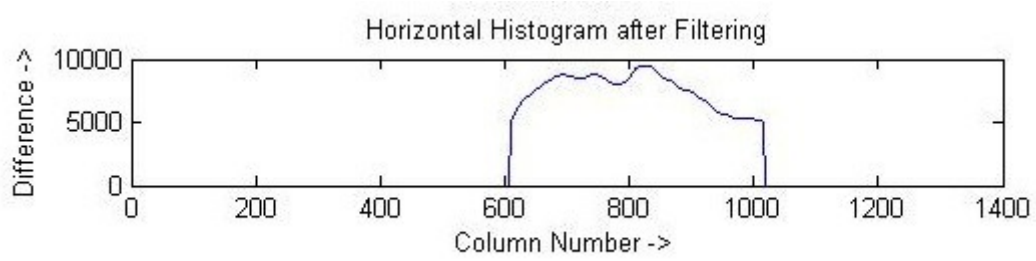


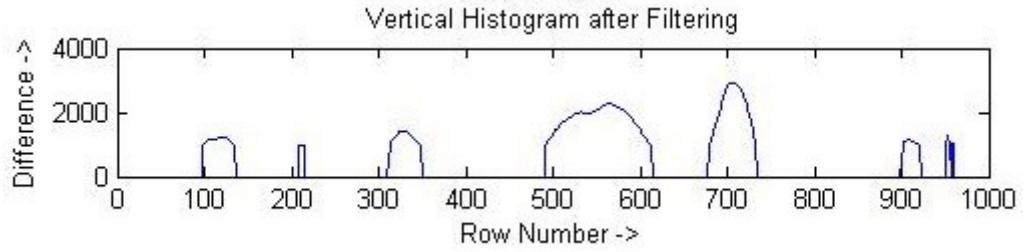
Filtering out Unwanted Regions in an Image

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 results 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:

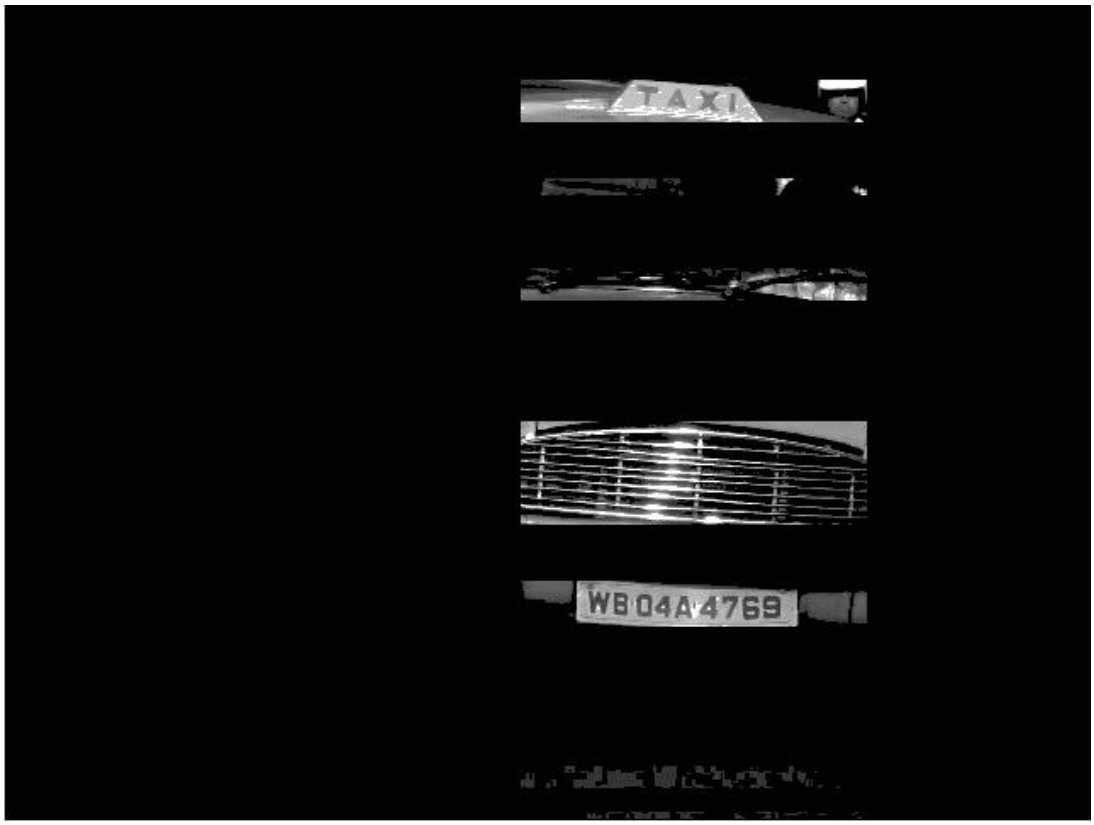




Segmentation

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.

<Image after horizontal segmentation>



<Image after vertical segmentation>

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.

OCR –OPTICAL CHARACTER RECOGNITION

Introduction

The automatic number plate recognition with OCR works by using the technology to capture the images and retrieving the license numbers on the plate. It works by simply highlighting the numbers on the image and separating them from the other objects on the screen. The automatic number plate recognition with OCR will then work to convert the data into editable, searchable and easily stored information into the database network. Business entities can make great use of the automatic number plate recognition with OCR in tracking and tracing the destinations of the company owned vehicles.

Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from some sort of original paper data source, whether documents, sales receipts, mail, or any number of printed records. It is crucial to the computerization of printed texts so that they can be electronically searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Template Matching Approach

Template matching is one of the Character Recognition techniques. It is the process of finding the location of a sub image called a template inside an image. Once a number of corresponding templates is found, their centers are

used as corresponding points to determine the registration parameters. Template matching involves determining similarities between a given template and windows of the same size in an image and identifying the window that produces the highest similarity measure. It works by comparing derived image features of the image and the template for each possible displacement of the template. This process involves the use of a database of characters or templates. There exists a template for all possible input characters. For recognition to occur, the current input character is compared to each template to find either an exact match, or the template with the closest representation of the input character. If I(x, y) is the input character,Tn(x,y) is the template n, then the matching function s(I,Tn) will return a value indicating how well template n matches the input character. Some of the more common matching functions are based on the following formulas:

Pearson's correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y},$$

The above formula defines the *population* correlation coefficient, commonly represented by the Greek letter ρ (rho). Substituting estimates of the covariances and variances based on a sample gives the *sample correlation coefficient*, commonly denoted r :

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}.$$

An equivalent expression gives the correlation coefficient as the mean of the products of the standard scores. Based on a sample of paired data (X_i, Y_i), the sample Pearson correlation coefficient is

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{X_i - \bar{X}}{s_X}\right)\left(\frac{Y_i - \bar{Y}}{s_Y}\right)$$

where

$$\frac{X_i - \bar{X}}{s_X}, \bar{X}, \text{ and } s_X$$

MATLAB Implementation

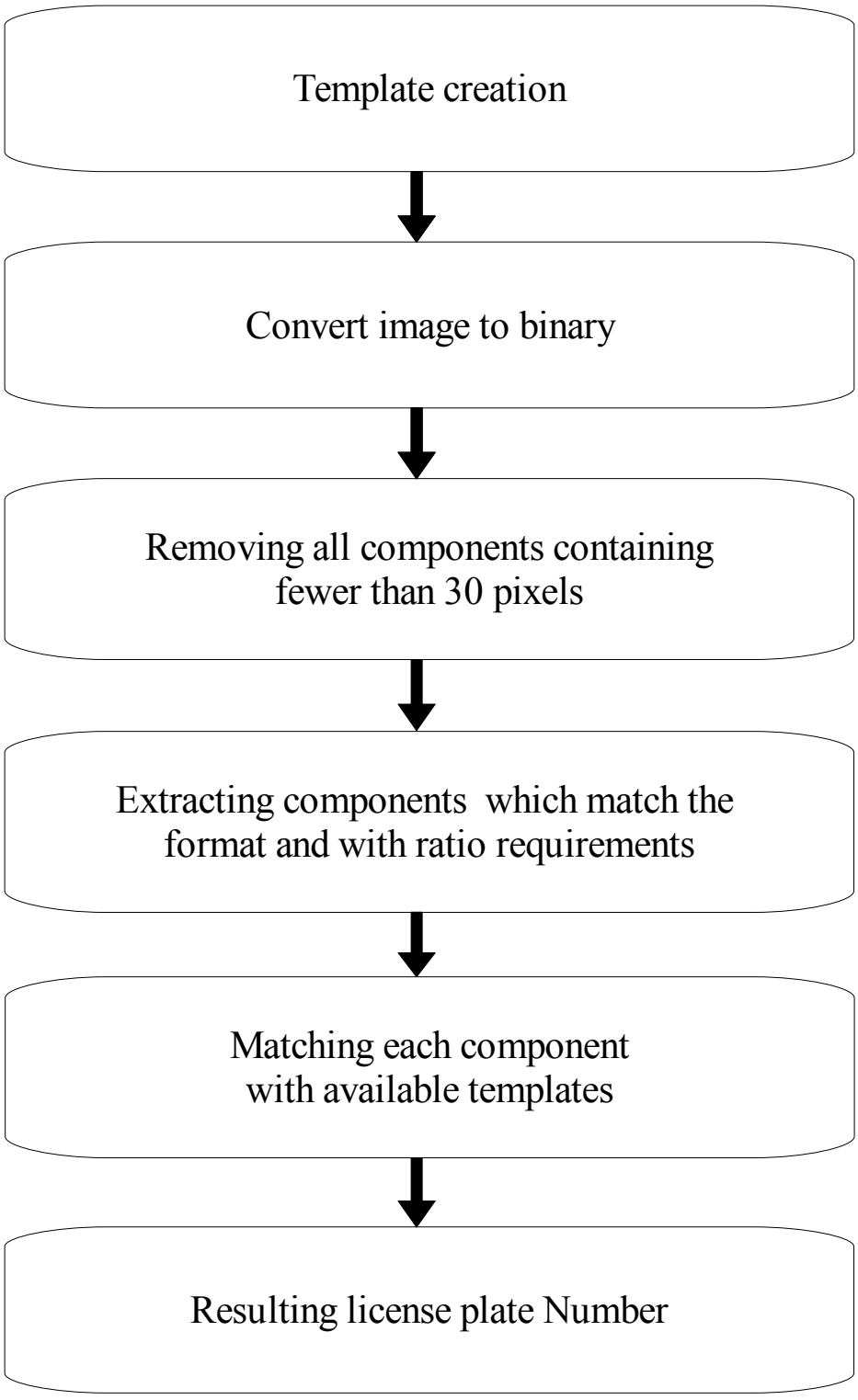
In the practical implementation of the algorithm using MATLAB, we have used the corr2() function to calculate the correlation coefficient. For example, r = corr2(A,B) computes the correlation coefficient between A and B, where A and B are matrices or vectors of the same size, and stores the value in r.

Internally, Corr2 computes the correlation coefficient using

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

where $\overline{A}$ = the mean of the values in A, and $\overline{B}$ = the mean of the values in B.

A flowchart showing the basic implementation of the OCR module is shown in the next page -



Flowchart showing Optical Character Recognition Using MATLAB

Template Creation

The overall success or failure rate of the algorithm mostly depends on the available templates – against which the input image is compared to extract the text from it. The more the number of different fonts present in the templates, higher is the chance to detect text successfully in an image. The following steps have been taken to create the templates used in this project.

First of all, binary images are prepared for all the characters which are to be included in the templates (This includes the letters A-Z and the numbers 0-9). All of them are then resized into same resolution (in this case – 24*42). These images are then read in MATLAB as individual two-dimensional arrays. Finally they are converted into cells using the inbuilt MATLAB function mat2cell(), and are stored in the file templates.mat. This template file is used in the OCR module for comparison purposes.

Examples of character templates

Preprocessing

Preprocessing of the image includes two major steps – 1.Converting the image into binary format, and 2.Removal of small objects from the image.

1. Converting the image into binary format - The input image is first converted to grayscale, and then to binary format. A binary image is required because the later part of the module uses a function to find the connected

components in the image which only accepts binary images as input. For binary conversion, the in-built MATLAB function im2bw() is used.

The im2bw() function requires two input arguments – the input image, and a THRESHOLD value which is used to convert the input image into a binary one. The output binary image has values of 1 (white) for all pixels in the input image with luminance greater than THRESHOLD and 0 (black) for all other pixels. To calculate the threshold value dynamically for each input image, another built-in MATLAB function, graythresh() is used. This function only takes grayscale image for input – therefore the input image is first converted from RGB format to Grayscale format.

2. Removal of small objects from the image – Due to noise level and various other factors in the input picture, a large numbers of small connected objects may be present in the input picture. Comparing all of them with the templates is unnecessary, time-consuming and often it may produce incorrect results. Therefore, as a part of pre-processing, all the connected components containing fewer than 30 pixels (a value derived after experimenting with a number of input images) is removed from the picture. This step both increases the accuracy and the efficiency while significantly reducing the runtime. After this step, the rest of the connected components are labeled for the next step.

Candidate Extraction

In a standard license plate, it is expected that all the characters present in the plate will be of almost same dimensions. Therefore, after a probable candidate is found which is successfully matched with a present template, the size of all the other components are measured before passing them for template matching. If the size of a connected component differs greatly from the first

successfully found character, it is discarded. Once a connected component of suitable dimensions is found, it is first resized into the dimensions of the template files, and then sent to the next step.

Matching with available templates

A small matching module computes the correlation coefficient of the input component against all the template entries, and stores the respective values into an array. Depending on the position of the component in the string, it is either compared with the character set, or the number set. After the comparison, the input candidate is recognized as the character corresponding to the greatest value of the correlation coefficient.

Output text

After all the characters have been successfully recognized, the obtained string of alphanumeric characters is stored into a text file.

MATLAB IMPLEMENTATION

Introduction

This chapter describes 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, imhist, 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, we have implemented our 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. The MATLAB code of the algorithm is shown below:

List of all the functions and headers used in the code :

Inbuilt functions of MATLAB used :-

clc: Clear Command Window

clear -Remove items from workspace, freeing up system memory.

clear all - removes debugging breakpoints in M-files and reinitializes persistent variables.

imread - Read image from graphics file.

rgb2gray - Convert RGB image or colormap to grayscale.

graythresh - Global image threshold using Otsu's method.

im2bw - Convert image to binary image, based on threshold. The output image BW replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black).

size - returns the sizes of each dimension of an image in matrix a and b.

mat2cell - Divide matrix into cell array of matrices.

medfilt2 - Performs median filtering of the matrix A using the default 3-by-3 neighborhood.

bwlabel - returns in num the number of connected objects found in BW.

bwareaopen - Morphologically open binary image (remove small objects).

fopen - Opens files in the required mode (generally read/write).

max - Largest elements in array.

min - Smallest elements in array.

find - Find indices and values of nonzero elements.

zeros - Create array of all zeros.

resize - Resize image.

load - Load workspace variables from disk.

corr2 - 2-D correlation coefficient.

isempty - Determine whether array is empty.

fclose - Close one or more open files.

imshow - Display image.

List of user-defined functions

lpr() - Finds one or more candidate regions.

crop() - Crops an image suitably for the OCR function.

create_templates() - Creates the template file from per-existing images.

OCR() - Extracts characters from the input image.

lines() - Divides the whole image containing text into lines.

read_numbers() - Matches an input component with a number template.

read_alphabet() - Matches an input component with an alphabet template.

MATLAB Code for user-defined functions

1. lpr()

```
function [text] = lpr(name)

%%INITIALIZATION

clc;                        % Clear command window.
%clear all;                  % Delete all variables.
warning off;

I = imread (name);  % Read Image and make a backup
I2=I;

k=1;                        % k will be used to display figures

figure(k); imshow(I); k=k+1;

%% CONVERT THE IMAGE TO GRAYSCALE

[m1 m2 m3]=size(I);
if m3>1                                 %if the image is RGB
    I = rgb2gray(I);                    %then convert it to grayscale
end

[rows cols] = size(I);
figure(k); imshow(I); k=k+1;

%% ENHANCE THE CONTRAST

[m1 m2] = size(I);
for i=2:m1-1
```

```
    for j=2:m2-1
        if I(i,j)<50
            I(i,j)=0;
        end
         if I2(i,j)<50
            I2(i,j)=0;
        end
    end
end

for i=1:m1
    for j=1:m2
        if I(i,j)>200
            I(i,j)=255;
        end
        if I2(i,j)>200
            I2(i,j)=255;
        end
    end
end
 figure(k); imshow(I); k=k+1;

Idilate = I;
for i = 1:rows
    for j = 2:cols-1
        temp = max(I(i,j-1), I(i,j));
        Idilate(i,j) = max(temp, I(i,j+1));
    end
end

I = Idilate;


%% PROCESS EDGES IN HORIZONTAL DIRECTION

difference = 0;
sum = 0;
total_sum = 0;
difference = uint32(difference);


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(k);
% Plot the Histogram for analysis
subplot(3,1,1);
plot (horz1);
title('Horizontal Edge Processing Histogram');
xlabel('Column Number ->');
ylabel('Difference ->');
k=k+1;

%% Smoothen the Horizontal Histogram by applying Low Pass
Filter
disp('Passing Horizontal Histogram through 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('Horizontal 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) < 1.2*average)
        horz(i) = 0;
        for j = 1:rows
            I(j, i) = 0;
        end
    end
end

subplot(3,1,3);
plot (horz);
title('Horizontal Histogram after Filtering');
xlabel('Column Number ->');
```

```
ylabel('Difference ->');

figure(k); imshow(I); k=k+1;
%% 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(k);
subplot(3,1,1);
plot (vert1);
title('Vertical Edge Processing Histogram');
xlabel('Row Number ->');
ylabel('Difference ->');
k=k+1;

%% 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('Vertical 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) < 1.2*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(k); imshow(I); k=k+1;

%% 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

%% Check each probable candidate
flagm=0;
for p = 1:2:row_size
     for q = 1:2:column_size

        II=I2(row(p):row(p+1),column(q):column(q+1));
        II2=I(row(p):row(p+1),column(q):column(q+1));

        [n1 n2] = size(II);
        white=0;
        for xx=1:n1
            for yy=1:n2
                if(II2(xx,yy)>240)
                    white=white+1;
                end
            end
        end

        ratio=white/(n1*n2);

        if (n1>20 && n2>60 && ratio<0.8)
            figure(k); imshow(II); k=k+1;
            imwrite(II,'plate.jpg');
            k=crop(k);
            k=crop(k);
            [k,text,flagm] = OCR(k);
        end
        if flagm
            break;
        end
     end
     if flagm
         break
     end
end
end
```

2. crop()

```
function [k] = crop(k)

im=imread('plate.jpg');
threshold = graythresh(im);
imbw =im2bw(im,threshold);



[L Ne]=bwlabel(imbw);
```

```
[r c]=size(imbw);

m=0;
pos=0;
for i=1:Ne
    counter=0;
    for jj=1:r
        for kk=1:c
            if L(jj,kk)==i
                counter=counter+1;
            end
        end
    end
    if counter>=m
        m=counter;
        pos=i;
    end
end


[rr cc]=find(L==pos);
I1=im(min(rr):max(rr),min(cc):max(cc));

imwrite(I1,'plate.jpg');

end
```

3. create_templates()

```
%CREATE TEMPLATES
%Letter
A=imread('letters_numbers\A.bmp');
B=imread('letters_numbers\B.bmp');
C=imread('letters_numbers\C.bmp');
D=imread('letters_numbers\D.bmp');
E=imread('letters_numbers\E.bmp');
F=imread('letters_numbers\F.bmp');
G=imread('letters_numbers\G.bmp');
H=imread('letters_numbers\H.bmp');
I=imread('letters_numbers\I.bmp');
J=imread('letters_numbers\J.bmp');
K=imread('letters_numbers\K.bmp');
L=imread('letters_numbers\L.bmp');
M=imread('letters_numbers\M.bmp');
N=imread('letters_numbers\N.bmp');
O=imread('letters_numbers\O.bmp');
P=imread('letters_numbers\P.bmp');
Q=imread('letters_numbers\Q.bmp');
R=imread('letters_numbers\R.bmp');
S=imread('letters_numbers\S.bmp');
T=imread('letters_numbers\T.bmp');
U=imread('letters_numbers\U.bmp');
V=imread('letters_numbers\V.bmp');
```

```
W=imread('letters_numbers\W.bmp');
X=imread('letters_numbers\X.bmp');
Y=imread('letters_numbers\Y.bmp');
Z=imread('letters_numbers\Z.bmp');
%Number
one=imread('letters_numbers\1.bmp');
two=imread('letters_numbers\2.bmp');
three=imread('letters_numbers\3.bmp');
four=imread('letters_numbers\4.bmp');
five=imread('letters_numbers\5.bmp');
six=imread('letters_numbers\6.bmp');
seven=imread('letters_numbers\7.bmp');
eight=imread('letters_numbers\8.bmp');
nine=imread('letters_numbers\9.bmp');
zero=imread('letters_numbers\0.bmp');
%*-*-*-*-*-*-*-*-*-*-*-*-*-
letter=[A B C D E F G H I J K L M...
    N O P Q R S T U V W X Y Z];
number=[one two three four five...
    six seven eight nine zero];
character=[letter number];
templates=mat2cell(character,42,[24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24]);
save ('templates','templates')
clear all
```

4. OCR()

```
function [k,text flagm] = OCR(k)

temp1=k;
text=[];

% Read image
imagen=imread('plate.jpg');

% Convert to gray scale
if size(imagen,3)==3 %RGB image
    imagen=rgb2gray(imagen);
end

% Convert to BW
threshold = graythresh(imagen);
imagen =~im2bw(imagen,threshold);

% Remove all object containing fewer than 30 pixels
imagen = bwareaopen(imagen,30);

%Storage matrix word from image
word=[ ];
```

```
re=imagen;

%Opens text.txt as file for write
fid = fopen('text.txt', 'wt');

% Load templates
load templates
global templates

% Compute the number of letters in template file
num_letras=size(templates,2);
flag=0;
count=0;
while 1

     %Fcn 'lines' separate lines in text
    [fl re]=lines(re);
    imgn=fl;

     %Uncomment line below to see lines one by one
    figure(k), imshow(fl);pause(0.1)
    k=k+1;

    % Label and count connected components
    [L Ne] = bwlabel(imgn);

     for n=1:Ne
        [r,c] = find(L==n);

        % Extract letter
        n1=imgn(min(r):max(r),min(c):max(c));
        s=size(n1);
        [rr cc] = size(n1);

        % Resize letter (same size of template)
        img_r=imresize(n1,[42 24]);

        % Call fcn to convert image to text
        if count==0 | count==1 | count==4
            letter=read_alphabet(img_r,num_letras);
        else
            letter=read_numbers(img_r,num_letras);
        end
        if letter=='W'
               flag=1;
               nn=s;
               nnr=rr;
               nnc=cc;
        end

        % Letter concatenation
        if flag==1 & (s<1.5*nn) & count~=9
           if ((letter=='1') | (letter=='Y') | (letter=='B'))
& rr>=(0.6*nnr)
               word=[word letter];
               count=count+1;
               figure(k), imshow(img_r);pause(0.1)
               k=k+1;
```

```
            elseif s>=(0.6*nn) & count~=9
                word=[word letter];
                count=count+1;
                figure(k), imshow(img_r);pause(0.1)
                k=k+1;



            end
        end

    end

    %fprintf(fid,'%s\n',lower(word));%Write 'word' in text
file (lower)
    fprintf(fid,'%s',word);%Write 'word' in text file (upper)
    text=[text word];

     % Clear 'word' variable
    word=[ ];

     %*When the sentences finish, breaks the loop
    if isempty(re)  %See variable 're' in Fcn 'lines'
        break
    end
end
temp2=k;
flag=1;

if ftell(fid)<8
    flag=0;
    text=[];
end
fclose(fid);

%Open 'text.txt' file
if flag==0
    for c=temp1-1:1:temp2-1
        close (c);
    end
    k=temp1-1;
    flagm=0;
else
    flagm=1;
end
end
```

5. read_numbers()

```
function letter=read_numbers(imagn,num_letras)
global templates
comp=[ ];
for n=27:num_letras
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
vd=find(comp==max(comp));
vd=vd+26;
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
if vd==27
    letter='1';
elseif vd==28
    letter='2';
elseif vd==29
    letter='3';
elseif vd==30
    letter='4';
elseif vd==31
    letter='5';
elseif vd==32
    letter='6';
elseif vd==33
    letter='7';
elseif vd==34
    letter='8';
elseif vd==35
    letter='9';
elseif vd==36
    letter='0';
else
    letter='*';
end
```

6. read_alphabet()

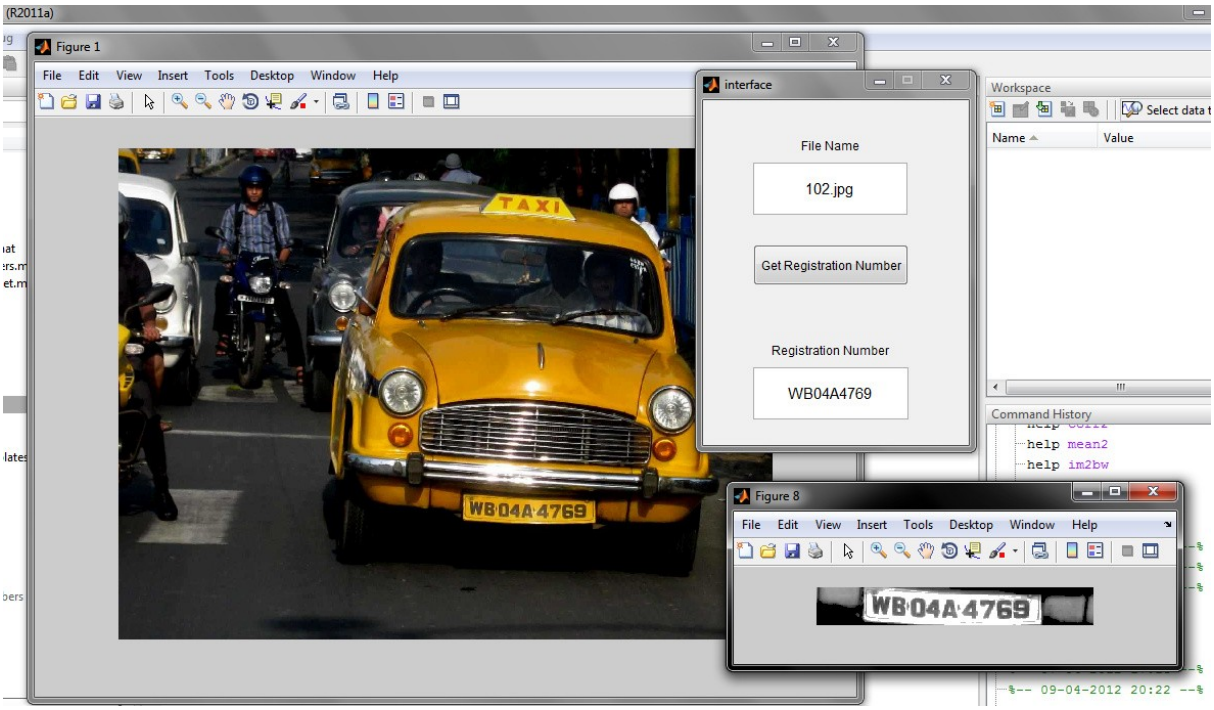
```
function letter=read_alpahbet(imagn,num_letras)

global templates
comp=[ ];

for n=1:26
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
vd=find(comp==max(comp));
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
if max(comp)<0.3
    letter='*';
```

```
elseif vd==1
    letter='A';
elseif vd==2
    letter='B';
elseif vd==3
    letter='C';
elseif vd==4
    letter='D';
elseif vd==5
    letter='E';
elseif vd==6
    letter='F';
elseif vd==7
    letter='G';
elseif vd==8
    letter='H';
elseif vd==9
    letter='I';
elseif vd==10
    letter='J';
elseif vd==11
    letter='K';
elseif vd==12
    letter='L';
elseif vd==13
    letter='M';
elseif vd==14
    letter='N';
elseif vd==15
    letter='O';
elseif vd==16
    letter='P';
elseif vd==17
    letter='Q';
elseif vd==18
    letter='R';
elseif vd==19
    letter='S';
elseif vd==20
    letter='T';
elseif vd==21
    letter='U';
elseif vd==22
    letter='V';
elseif vd==23
    letter='W';
elseif vd==24
    letter='X';
elseif vd==25
    letter='Y';
else
    letter='Z';
end
```

Screenshot of the Working:



CONCLUSION

Though the resulting algorithm is very efficient and robust and has a high success rate, it does have a few limitations as well. Here we have stated a number of limitations that we faced, along with the solutions we came up with:

1. Recognizes only specified font

As previously mentioned, the algorithm extracts characters from an image by comparing them to a pre-existing database. Therefore, if a license plate uses a font that is drastically different from the fonts present in the database, chances are they would be recognized incorrectly, or won't be recognized at all.

Solution - The solution of this problem is simply increasing the size of the database used for comparison. The more the number of different fonts present in the database, greater is the success rate of the algorithm.

2. Character ambiguity - This one is pretty self-explanatory in nature. As some of the letter and number looks alike (B and 8, I and 1, O and 0, Z and 2 etc.), there is always a chance that one of these characters will be mis-interpreted by the algorithm.

Solution – While comparing the candidates with the database, we keep in mind that a license plate has a specific format (in India, for example, it is LLNNLNNNN, where L stands for a letter from the English alphabet, and N stands for a decimal digit). Therefore, while scanning a picture, we also calculate the position of a candidate in a string, and accordingly compare it only to the alphabet set, or to the number set, but not both at the same time.

3. Variable Lighting Conditions – The algorithm mainly depends on the difference of the contrast levels of bright and dark areas of the picture.

Therefore, poor lighting conditions at the time of capturing the image is a real problem as it significantly lowers the chance of the license plate being successfully recognized by the LPR module. Also, inadequate light at the time of capturing the picture adds a lot of noise to the image, thus in turn reducing the efficiency of the algorithm itself.

Solution – This is one serious problem and there is no perfect solution available to overcome the situation. Though the algorithm does incorporate noise-removing as well as contrast-enhancing sub modules, they only improve the success rate by a factor – they can't eliminate the problem completely.

The project was designed keeping in mind the automation of the number plate detection system for security reason that could replace the current system of manual entry.

This project was a success in recording the number plate of a vehicle although it has got it’s own limitation of image processing and other hardware requirements. From this project we learn about image processing and OCR (Optical Character Recognition)

The basic focus of the project was in the image processing using MATLAB. The MATLAB v7.12.0 was used as programming tool which provides many features related to image processing.

Practical implementation of the project range from small use such as keeping record of incoming and outgoing vehicles from the parking area to a vast implementation such as security system. The project was completed within 10 months time period.

REFERENCES

[1] Halina Kwasnicka and Bartosz Wawrzyniak, “License plate localization and recognition in camera pictures”, AI-METH 2002, November 13-15, 2002.

[2] Naikur Bharatkumar Gohil, “Car license plate detection”, B.E.Project Report, Dharmsinh Desai University, India, 2006

[3] Pramod Kapadia, “Car License Plate Recognition Using Template Matching Algorithm”, Master Project Report, California State University, Sacramento, Fall 2010.

[4] F. Martín, M. García, J. L. Alba, (2002, Jun.), New Methods for Automatic Reading of VLP's (Vehicle License Plates), presented at IASTED International Conference Signal Processing, Pattern Recognition, and Applications, SPPRA 2002.http://www.gpi.tsc.uvigo.es/pub/papersppra02.pdf.

[5] Kwang-Baek Kim, Si-Woong Jang and Cheol-Ki Kim, “Recognition of Car License Plate by Using Dynamical Thresholding Method and Enhanced Neural Networks”, Lecture Notes on Computer Science 2756, N. Petkov and M.A. Westenberg , Ed. Springer-Verlag, 2003, pp. 309–319.