ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project supervisor and guide Ms. Shamama Anwar, Assistant Professor, BIT Mesra, who guided us, all through the project and showed us the right path. We are really gratified that he has spent so much time with us during our project work.

Our honourable thanks to Dr. Sandip Dutta, Associate Professor and Head of Department of CSE, BIT Mesra for providing us access to excellent labs and library.

We thank BIT Mesra for providing us with the necessary infrastructure and technical support that was required for this project.

Finally, we express our gratitude and appreciation towards our parents and friends for their valuable suggestions and moral support.

Mohammad Ashhar Jawaid (BE/1069/2011)

Prakashdeep Singh Golan (BE/1099/2011)

CONTENTS

1. INTRODUCTION

Text Extraction is a process by which we convert printed document / scanned page or image in which text are available to ASCII Character that a computer can recognize.

Text detection and recognition in general have quite a lot of relevant application for automatic indexing or information retrieval such document indexing, content-based image retrieval, and license car plate recognition which further opens up the possibility for more improved and advanced systems.

Off-line handwriting recognition involves the automatic conversion of text in an image into letter codes which are usable within computer and text-processing applications. The data obtained by this form is regarded as a static representation of handwriting. Off-line handwriting recognition is comparatively difficult, as different people have different handwriting styles. And, as of today, OCR engines are primarily focused on machine printed text and ICR for hand "printed" (written in capital letters) text. There is no OCR/ICR engine that supports handwriting recognition as of today.

The aim of this project is to detect, extract and recognize text from images, via two processes, namely, Segmentation –which is the separation of individual characters and Recognition -which is recognition of character in the detected text region using a suitable algorithm.

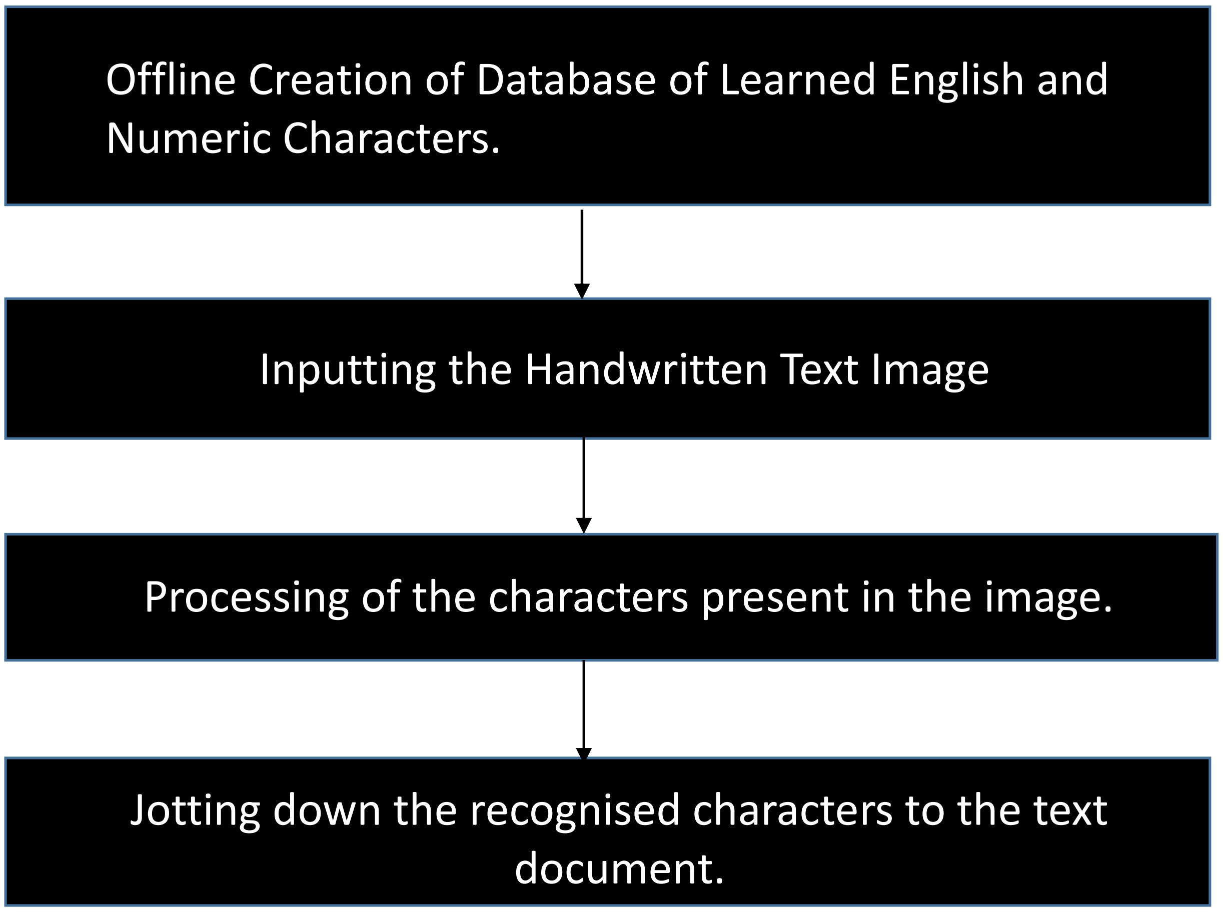
The objective of this paper to segment and recognize characters in an image have been achieved. Even though the segmentation accuracy from the experiment is 100%, the result during real application may be lower due to limited set of picture used in experiment. However, this project shows that segmentation using connected components is best method of segmenting the image.

1.1 System Modules – an overview

Handwritten Text conversion System consists of the following modules:

1) Creation of database of training.
2) Extraction of characters from the given image.
3) Recognition of extracted characters.
4) Conversion to text document.

The System is designed based on the flowchart as shown below:


Offline Creation of Database of Learned English and Numeric Characters.
Inputting the Handwritten Text Image
Processing of the characters present in the image.
Jotting down the recognised characters to the text document.

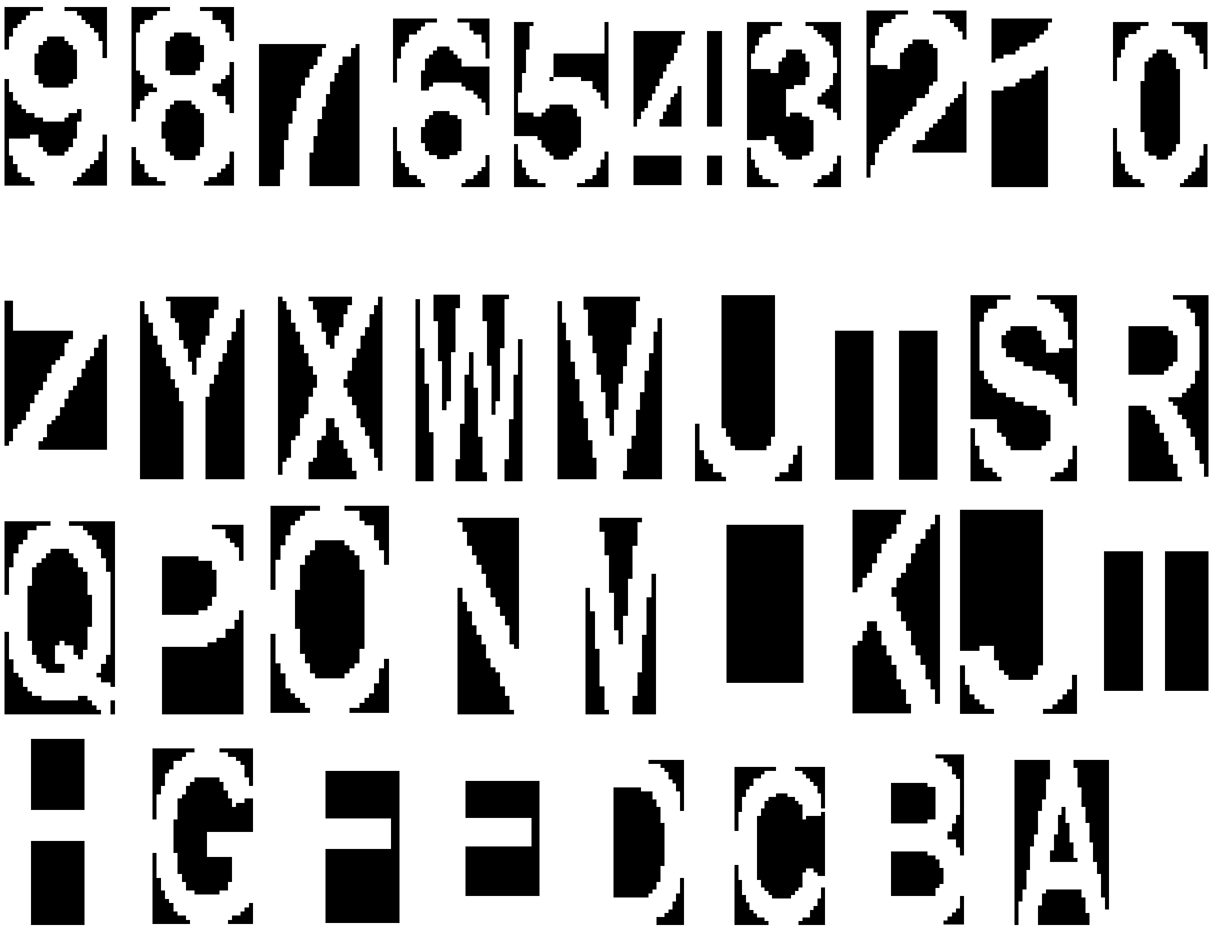
2. Learning Characters

The recognition algorithm relies on a set of learned characters and their properties. It compares the characters in the scanned image file to the characters in this learned set.

Generating the learned set is quite simple. It requires that an image file with the desired characters in the desired font be created, and a text file representing the characters in this image file. If a character such as pi, has a multicharacter translation, angled brackets should be placed around the translation.

Once the learned set has been read in from the image file and its properties recognized, it can be written out to a "learn" file. This file stores the properties of the learned characters in abbreviated form, eliminating the need for retaining the images of the learned characters, and can be read in very quickly.

The Database of Learned Characters, which was used in the MATLAB program, comprises of 36 characters of size 42 X24 pixels. Following were the 36 characters used for learning:



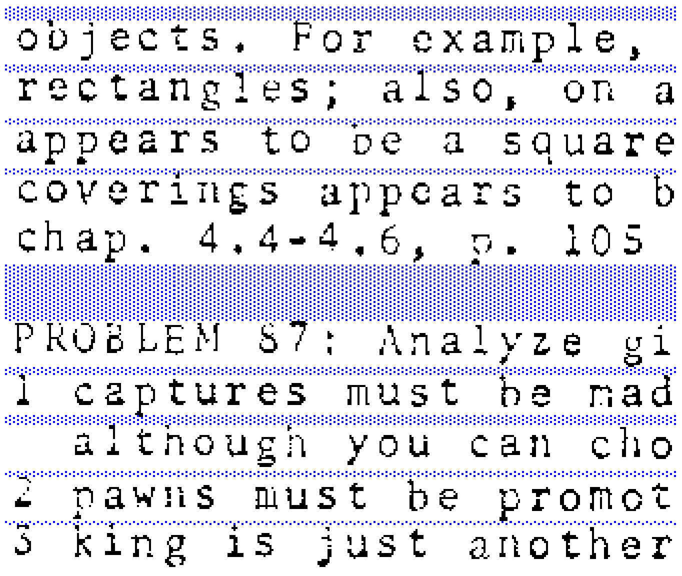
3. Character Extraction

Character extraction has three phases:

- Detection of lines of text.
- Detection of connected components.
- Extracting out each detected component from each line.

3.1 Line Detection

To detect lines of text (which is later useful in determining the order of characters and possibly their layout on the page), we do a horizontal projection of the page. Our original plan was to detect line breaks using some kind of statistical analysis, but for now, adjacent lines in the image having a very small number of pixels constitute a line break. Noise can deceive this simple algorithm, but adjusting the Noise Tolerance global variable can usually allow the user to overcome this shortcoming.



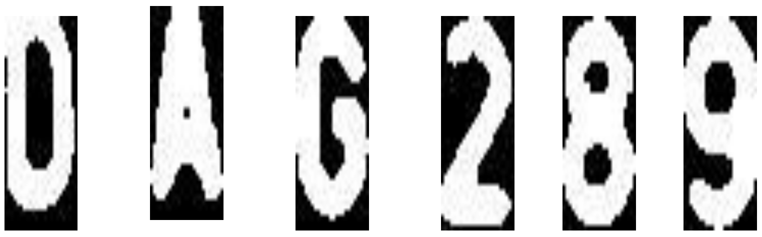
```
objects. For example,
rectangles; also, on a
appears to be a square
coverings appears to b
chap. 4.4-4.6, p. 105

PROBLEM 87: Analyze gi
1 captures must be mad
  although you can cho
2 pawns must be promot
3 king is just another
```

3.2 Segmentation and Extraction:

The process of locating regions of printed or handwritten text is segmentation. Segmentation differs text from figures and graphics. When segmentation is applied to text, it isolates characters or words. The mostly occurred problem in segmentation is: it causes confusion between text and graphics in case of joined and split characters. Usually, splits and joints in the characters causes due to scanning. If document is dark photocopy or if it scanned at low threshold, joints in characters will occur. And splits in characters will occur if document is light photocopy or scanned at high threshold.
The segmentation and further extraction of characters involves the following steps:

- Scan the image from left to right to find 'on' pixel.
- If 'on' pixel been found, all 'on' pixel connected to the detected on pixel will be extracted and segmented as a pixel.
- The process will be repeated until it reach end right of the image.
- Extract the connected components. The extracted components will look as follows:



4. Character Recognition - Concept and Implementation

After extracting the individual characters in a document and determining their properties, we compare them to the learned set. The comparison algorithm is simple: it sums the squares of the differences between each property in the extracted character and each property in the learned character, returning a "Confidence". Initially, we compared each extracted character to the entire linked list of learned characters, but in the interest of speed, we modified this slightly to classify characters in relation to the baseline of their line of text.

Each of the Extracted character from the extraction process is now subjected to an " all in all " matching technique in which the extracted character image is now compared with all the character images we had in our learning database. The "correlation (CORR2)" operator is used in the process, defined as follows:

$$\boldsymbol{r} = \frac{\sum_{\mathrm{m}}\sum_{\mathrm{n}}(\boldsymbol{i}_{mn} - \bar{\boldsymbol{\iota}})(\boldsymbol{j}_{mn} - \bar{\boldsymbol{\jmath}})}{\sqrt{\left(\sum_{\mathrm{m}}\sum_{\mathrm{n}}(\boldsymbol{i}_{\mathrm{mn}} - \bar{\boldsymbol{\iota}})^{2}\right)\left(\sum_{\mathrm{m}}\sum_{\mathrm{n}}(\boldsymbol{j}_{\mathrm{mn}} - \bar{\boldsymbol{\jmath}})^{2}\right)}}$$

Where $\bar{\boldsymbol{\iota}}$ is the mean of the input matrix *i* and $\bar{\boldsymbol{\jmath}}$ is the mean of the input matrix *j*.

- $0 < r < 1$
- 1 mean *i* and *j* is exactly same while 0 mean the *i* and *j* *not* same at all.

For every learned character its matching value is stored and after its comparison with all the characters in the learned database, the one with which it had the most correlation with is assigned its label.

Only after all this is done for each and every image, the recognised characters are then sent down to be written down in a text file which is then to be presented as the OUTPUT of the whole program.

5. EXPECTED RESULTS AND CONCLUSION

When we look back on the entire process of our project, I can combine the conclusions into many points:

- Text recognition and conversion is a complex problem, which is not easily solvable .The argument which is prevalent in the initial stage is that of forming the database. A simple solution of providing a 42X24 character image of standard size. The main problem lies in pre-processing of data. Datasets from the "internet" were quite large, they contained a lot of data (a lot of data is excellent for training, validation and testing) , but too little information on attributes and image extractions (which were used by them -authors of datasets) . From that reason, I was able to form a small recognisable database, but not able to test it on my own example. Final result was only X percent of true classification, which was not what I expected. Next, I decided to go on my own and make my own recognition example, which will include also self-testing algorithm. The bad side of own application is that, I will not have a lot of training data, therefore output from the MATLAB code will be questionable.

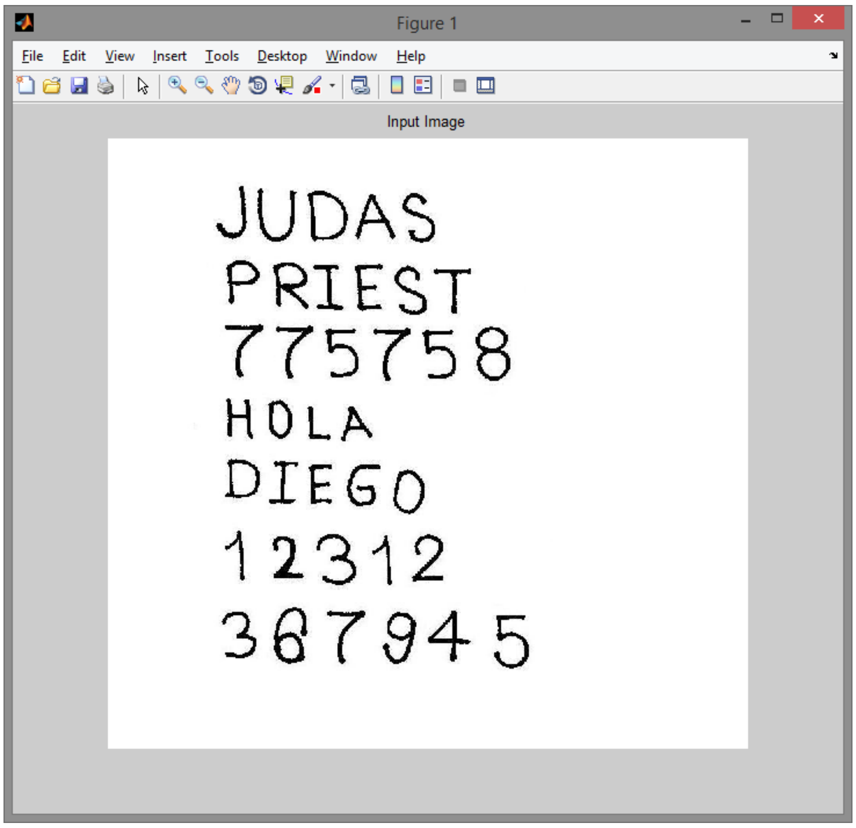
- Described application of character recognition can be divided into three main parts. Formation of the learned database, Image pre-processing to get the training data and at the end testing with final recognition results. Database Description was not a tedious task as we simply saved the character images in a 42X24 format.

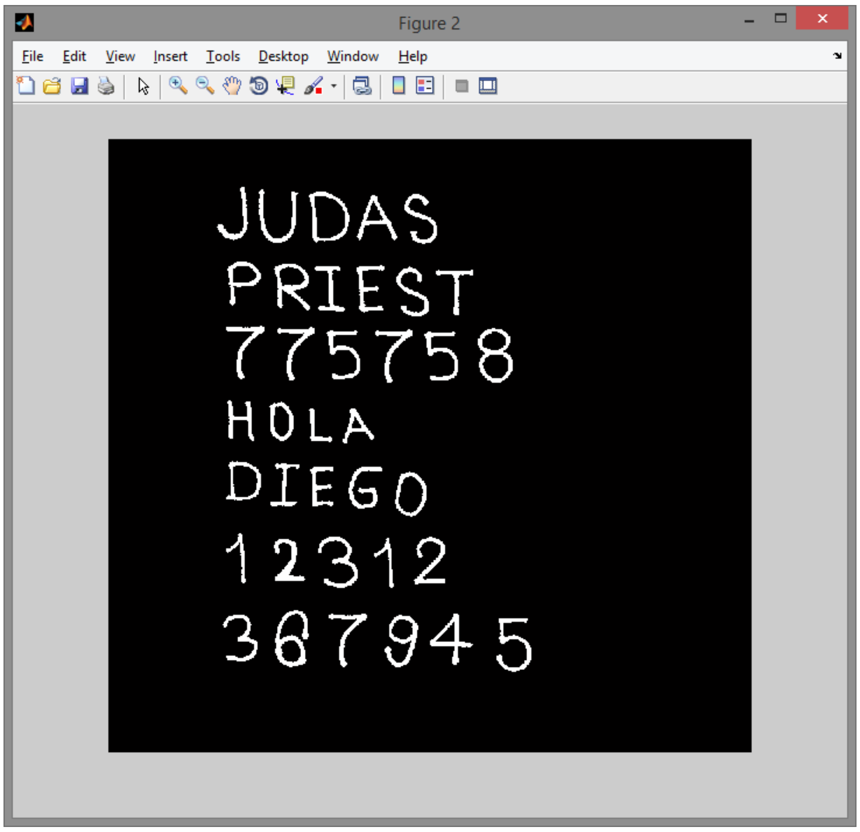
- Image Processing and end recognition was done in MATLAB with the help of some MATLAB commands like im2bw, bwlabel & corr2.

- The End result of this is an output in the text format which comprises of the text present in the given image.

6. Samples of Work Till Now

Following is the serial snapshot of our working model:





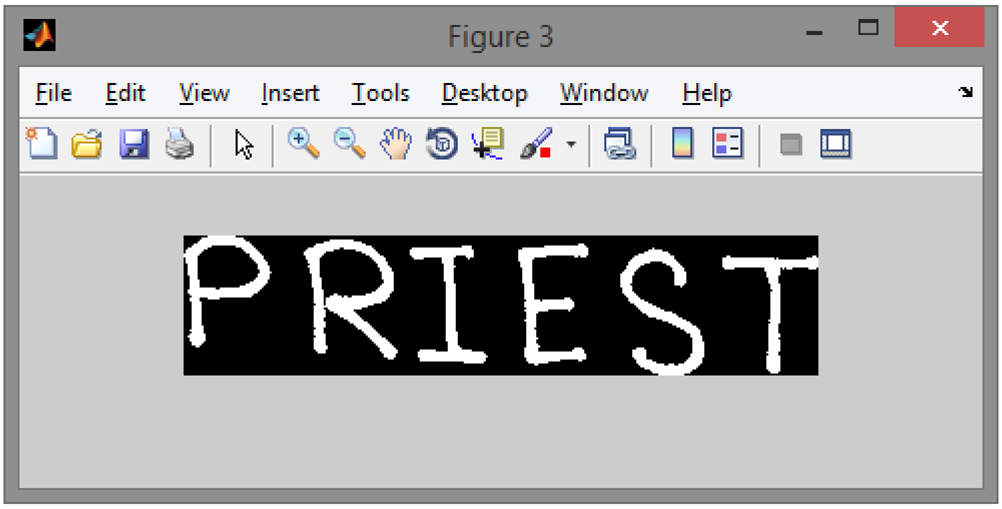


Figure 3
File
Edit
View
Insert
Tools
Desktop
Window
Help
PRIEST

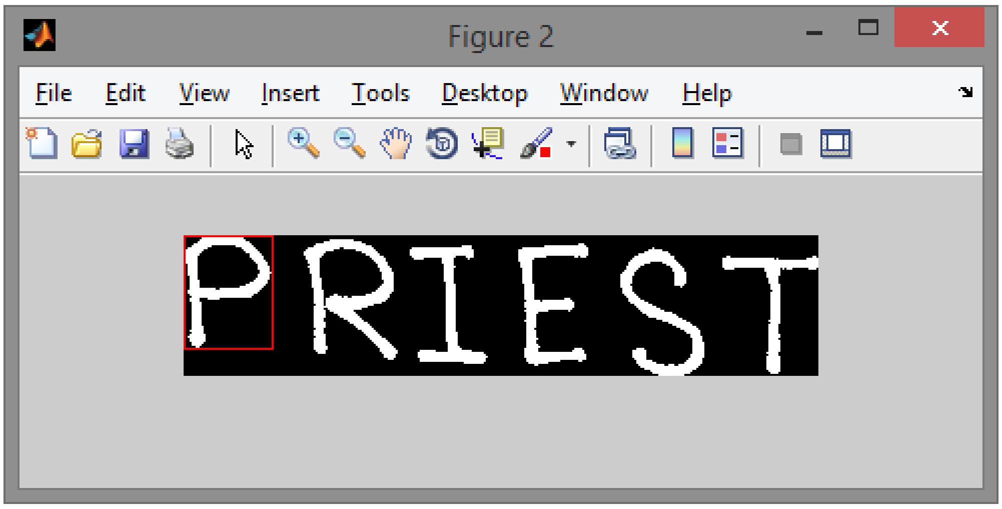


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
PRIEST

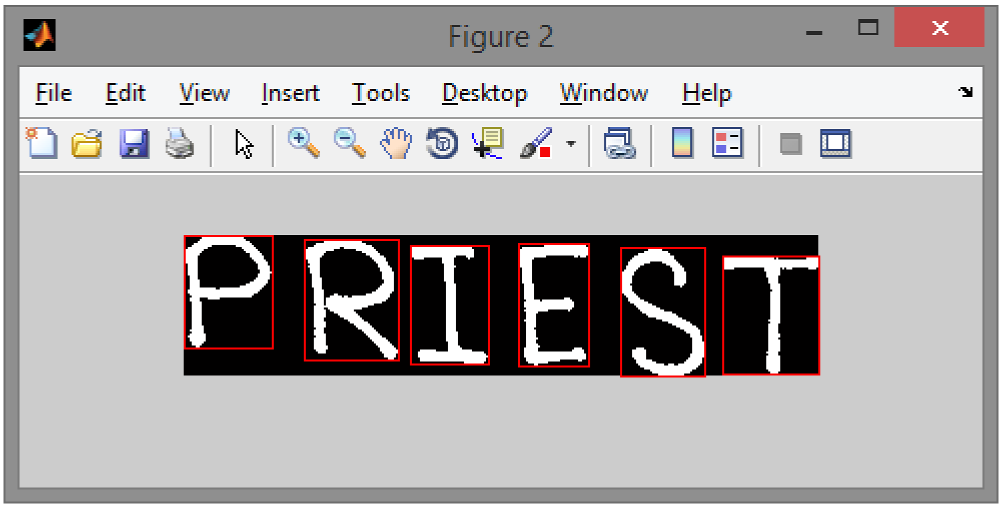
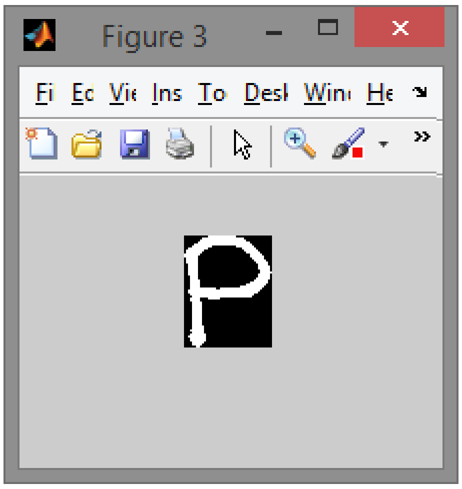
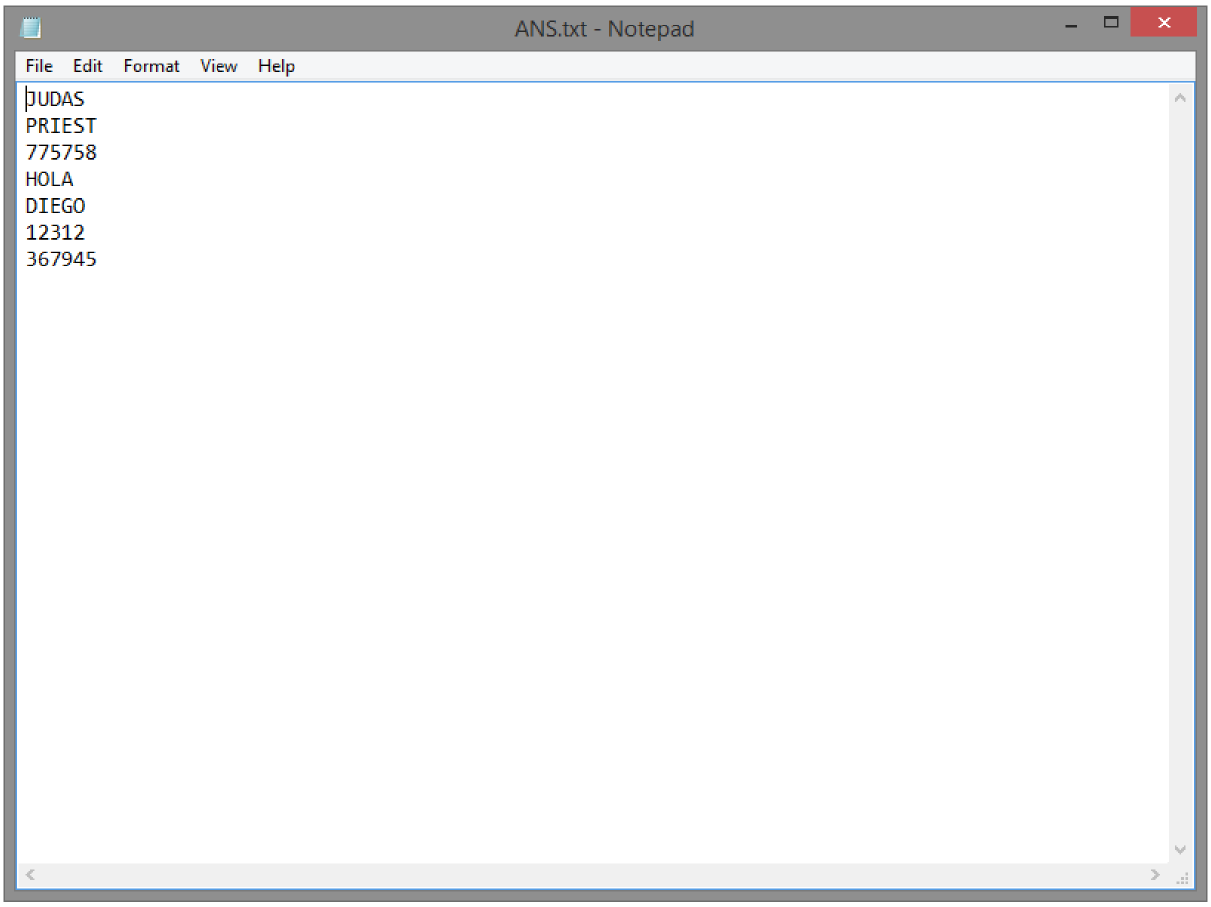


Figure 2
File
Edit
View
Insert
Tools
Desktop
Window
Help
PRIEST





```
JUDAS
PRIEST
775758
HOLA
DIEGO
12312
367945
```

7. Code Snippet

```
%READ INPUT IMAGE
normal = imread('Input\image0003.jpg');
imshow(normal);
title('Input Image');

%CONVERT TO GRAYSCALE
%gray = rgb2gray(normal);
if size(normal,3)==3 %RGB image
    gray=rgb2gray(normal);
end

%CONVERT TO BINARY IMAGE
%binary = im2bw(gray);
threshold = graythresh(gray);
binary = im2bw(gray,threshold);
%figure,imshow(binary);

%INVERT IMAGE
binary = ~binary;
figure,imshow(binary);
pause(2);

%COUNT LETTER IN TEXT
re=binary;
pause(1);
figure;
cnt=1;

fid = fopen('Output/ANS.txt', 'wt');
while 1

    %EXTRACT ONE LINE
    [fl re]=divide_line(re);
    imshow(fl);
    pause(0.5);
    word = [ ];

    %COUNT CONNECTED COMPONENTS i.e TOTAL CHARACTERS IN THE LINE
    [Ilabel, num]=bwlabel(fl);
    Iprops = regionprops(Ilabel, 'BoundingBox')
    Ibox = round(reshape([Iprops.BoundingBox],[4 num]));

    for cnt2=1:num
        %figure,imshow(normal(Ibox(:,cnt)));
        rectangle('position',Ibox(:,cnt2),'edgecolor','r');
        pause(0.5);
    end

    w=strcat('Output/line',int2str(cnt),'.jpg');
    imwrite(fl,w);

    for n=1:num
        [r,c] = find(Ilabel==n);
        %Extract letter
        n1=fl(min(r):max(r),min(c):max(c));
        imshow(n1);
```

```
            %recognise text & write to file
              n1=imresize(n1,[42 24]);
              letter=match(n1);%save extracted letter to text
              w=strcat('Output/',int2str(cnt),'.jpg');
              cnt=cnt+1;
              imwrite(n1,w);
              word=[word letter];
          end
          fprintf(fid,'%s\n',word);
          if isempty(re)
              break;
          end
      end
      winopen('Output/ANS.txt');
      fclose(fid);



      function ans=match(image)

          %load all data
          %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');
              I2=imread('letters_numbers\I2.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');
```

```
%create checking matrix
     train=[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...
            one two three four five...
            six seven eight nine zero I2];

     train=mat2cell(train,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 24]);

      %match
      num_letras = size(train,2)
      x=[ ];
        for n=1:num_letras
            Y=corr2(image,train{1,n});
            x=[x Y];
        end
        ANS=find(x==max(x));
        switch ANS
            case 1
                ans='A';
            case 2
                ans='B';
            case 3
                ans='C';
            case 4
                ans='D';
            case 5
                ans='E';
            case 6
                ans='F';
            case 7
                ans='G';
            case 8
                ans='H';
            case 9
                ans='I';
            case 10
                ans='J';
            case 11
                ans='K';
            case 12
                ans='L';
            case 13
                ans='M';
            case 14
                ans='N';
            case 15
                ans='O';
            case 16
                ans='P';
            case 17
                ans='Q';
            case 18
                ans='R';
            case 19
                ans='S';
            case 20
                ans='T';
            case 21
```

```
                ans='U';

            case 22
                ans='V';
            case 23
                ans='W';
            case 24
                ans='X';
            case 25
                ans='Y';
            case 26
                ans='Z';
            case 27
                ans='1';
            case 28
                ans='2';
            case 29
                ans='3';
            case 30
                ans='4';
            case 31
                ans='5';
            case 32
                ans='6';
            case 33
                ans='7';
            case 34
                ans='8';
            case 35
                ans='9';
            case 36
                ans='0';
            case 37
                ans='I';

        end



function [fl re]=divide_line(image)

    %FUNCTION CLIP CROPS THE IMAGE TO REMOVE SURROUNDING BLACK BORDER
    image=clip(image);
    %imshow(image);

    %TOTAL NO OF PIXELS IN A COLUMN
    n=size(image,1);

    for s=1:n
        % SUM OF ALL PIXEL VAlUES IN A ROW
      % IF SUM == 0 THEN THERE IS NO WHITE PIXEL IN THE ROW AND HENCE
      BEGINS THE NEW LINE
        if sum(image(s,:))==0

            % EXTRACT FIRST LINE
            fl=image(1:s-1, :);
            fl = clip(fl);
            %imshow(fl);
            %pause(0.5);

            % REMAINING LINES
            re=image(s:end, :);
            re=clip(re);
            %imshow(re);
            %pause(0.5);

            break

        %IF THERE IS ONLY ONE LINE OR EVEN NO LINE
        else
            fl=image;
            re=[ ];
        end
    end

function img_out=clip(img_in)
    [f c]=find(img_in);
    img_out=img_in(min(f):max(f),min(c):max(c));
```

8. Applications and Future Enhancements

There are various applications of Text recognition. Some of them are listed as follows:

- Undesirable Text removal from images.
- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt.
- Automatic number plate recognition
- Automatic insurance documents key information extraction
- Extracting business card information into a contact list
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR
- Assistive technology for blind and visually impaired users

These are the possible applications which can use our character recognition implementation to support them.

But we, as our final semester project, have intended to enhance this character recognition in the following uses:

1) Automated Number Plate Recognition System
2) Recognition of small letters and special characters
3) Text to speech conversion

9. Automatic Number Plate Recognition

ABSTRACT

Automatic Identification of vehicles is a very challenging area, which is in contrast to the traditional practice of monitoring the vehicles manually. Automatic license plate (LP) recognition is one of the most promising aspects of applying computer vision techniques towards intelligent transportation system. In Location of the vehicle plate, a method of vehicle license plate character segmentation and extraction based on improved edge detection and Mathematical morphology was presented. In the first place, color images were changed into grey images, secondly through calculates the difference of each pixel and neighbourhood pixels to build up images edge and it can make the license plate stand out; Sobel operator is used to extract the edge of objects in image; then the algorithm applies the dilation and erosion mathematical morphology of binary images to get the image smooth contour. The segmentation result which is sent forward to LP recognition stage will improve further processing's efficiency. Neural Network is used to recognize the license plate character. Because of the accuracy of the plate region extraction, the character can be extracted exactly from the plate region.

Automatic number plate recognition (**ANPR**) is a mass surveillance method that uses optical character recognition on images to read vehicle registration plates. They can use existing closed-circuit television or road-rule enforcement cameras, or ones specifically designed for the task. They are used by various police forces and as a method of electronic toll collection on pay-per-use roads and cataloguing the movements of traffic or individuals.

ANPR can be used to store the images captured by the cameras as well as the text from the license plate, with some configurable to store a photograph of the driver. Systems commonly use infrared lighting to allow the camera to take the picture at any time of the day. ANPR technology tends to be region-specific, owing to plate variation from place to place.

Concerns about these systems have centred on privacy fears of government tracking citizens' movements, misidentification, high error rates, and increased government spending.

The software aspect of the system runs on standard home computer hardware and can be linked to other applications or databases. It first uses a series of image manipulation techniques to detect, normalize and enhance the image of the number plate, and then optical character recognition (OCR) to extract the alphanumeric of the license plate. ANPR systems are generally deployed in one of two basic approaches: one allows for the entire process to be performed at the lane location in real-time, and the other transmits all the images from many lanes to a remote computer location and performs the OCR process there at some later point in time. When done at the lane site, the information captured of the plate alphanumeric, date-time, lane identification, and any other information required is completed in approximately 250 milliseconds. This information can easily be transmitted to a remote computer for further processing if necessary, or stored at the lane for later retrieval. In the other arrangement, there are typically large numbers of PCs used in server to handle high workloads, such as those found in the London congestion charge project. Often in such systems, there is a requirement to forward images to the remote server, and this can require larger bandwidth transmission media.

9.1 INTRODUCTION

The whole system into three following steps:

1. Plate location or finding location of plate in the vehicle image and cropping plate image from it.
2. Plate segmentation or cutting plate image to character's images.
3. Character recognition or convert character's images to final distinguished characters among them.

Due to the diversity of parameters involved in car images, License plate detection is considered the most crucial stage in the whole LPR system. In the past, a number of techniques have been proposed for locating the desired plate through visual image processing. Now there are some algorithms about the locating of license plate, such us the methods based on color feature, edge extracting, histogram analysis, symmetry, morphological operators. In a method of Chinese license plate recognition has been proposed that structural verification is used for plate locating and projection segmentation is exploited for incising plate image into character's images. Sobel operator is the first and best-known choice for most LPD algorithms to convert the grayscale scene image to the gradient image at the initial stage. In general, conventional LPD algorithms focus only on vertical edge detection through vertical Sobel operator, because the horizontal edge density of license plates is not more outstanding than that of the surrounding background objects.

Character recognition of the number-plate is a fairly well developed field in computer vision in which template matching and neural networks are often used and can produce satisfactory results. However, template matching has its drawbacks in some aspects comparing with neural networks. For example, when characters of number-plate are segmented, neural network approach is preferred to template match one.

9.2 PRE-PROCESSING

Pre-processing mostly is necessary to facilitate further high performance recognition, in this proposed methodology, the character is binaries and the noise is eliminated in the pre-processing stage.

Grey Conversion

We have taken colour image of car clearly showing its License plate for experiment. We firstly convert this RGB colour input image to a 256 grayscale image using formula (1).

GREY=J(:,:,1)+J(:,:,2)+J(:,:,3) ....(1)

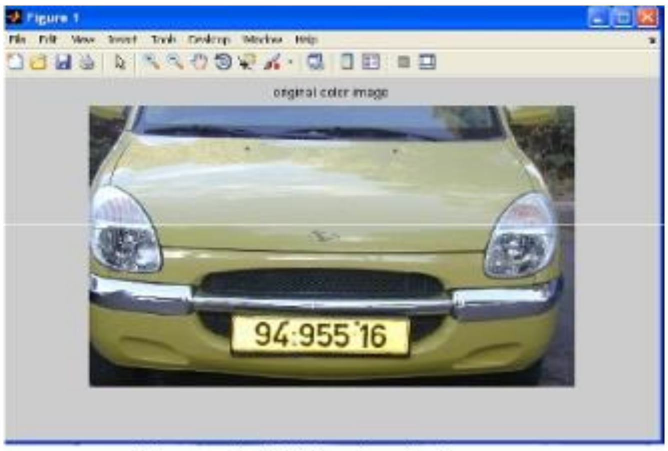


Figure 1. Original color image

The grey-scale could take every pixel of the picture to a number between 0-255 and the purpose of the linearization is to take every pixel into the number 0 or 255. To remove the tonal variation between Red, Green and Blue channels of input images and converting it into grey scale flatness to a single hue.

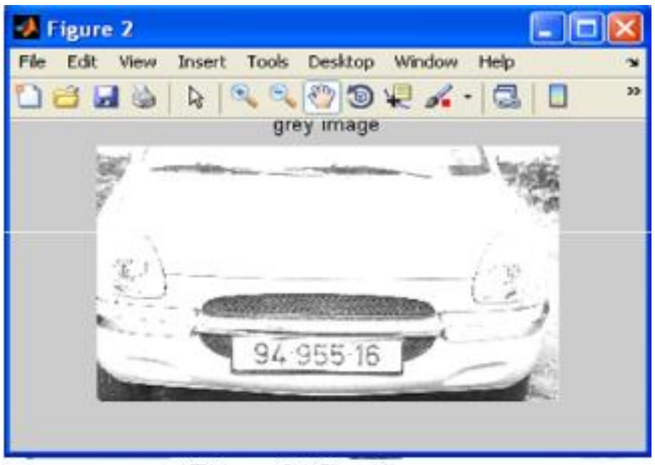


Figure 2. Grey image

Median Filtering

It is inevitable for containing noises of original image. We use median filtering to eliminate the noises. Using median filtering not only can eliminate the noises, but also make the high frequency more concentrated. There by, it is beneficial for us to detect the edges in images. The salt and pepper characteristic of vehicle license plate is presented, So Median filter is best known choice for removing such noise.

9.3 LICENSE PLATE EXTRACTION

To extract the license plate, we mainly utilize the characteristic of the plate in vehicle images like the colour in the region is blue. We project the binary difference image horizontally and vertically, properly smooth these two projection curves, and search their peaks to find the accurate rectangle that includes the plate.

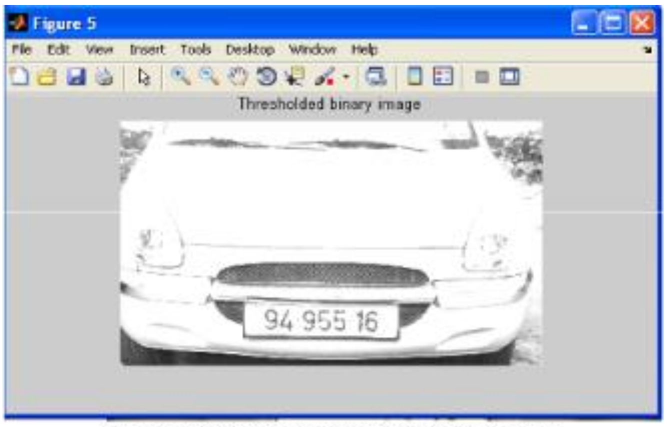


Figure.3. Thresholded Binary image

Opening and Closing

Opening generally smoothest the contour of an object, breaks narrow isthmuses and eliminates thin protrusions. Closing also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour.

The opening of set A by structuring element B, denoted A_ B, is defined as

$$A\_B= (A\_B) \oplus B \qquad (2)$$

Thus, the opening of A by B is the erosion of A by B, followed by a dilation of the result by B. Similarly, closing is defined as

$$A \bullet B= (A \oplus B)\_B \qquad (3)$$

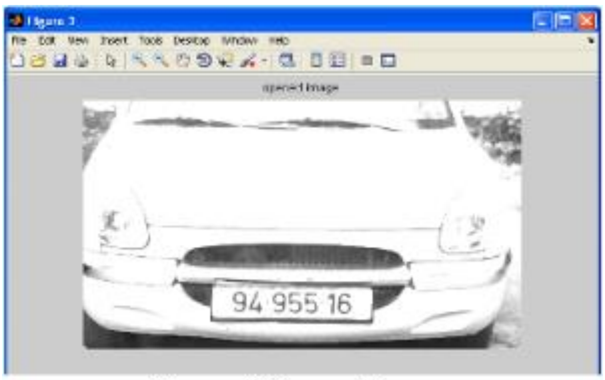


Figure 4. Opened image

This says that the closing of A by B is simply the dilation of A by B followed by the erosion of the result by B.

9.4 CHARACTER SEGMENTATION

Character segmentation is an important stage in many license plate recognition systems. There are many factors that cause the character segmentation task difficult, such as image noise, plate frame, rivet, and rotation and illumination variance. Object segmentation is an essential task in computer vision and object recognitions. Image segmentation is the process of partitioning a digital image into multiple regions or sets of pixels. These partitions represent different objects in the image, usually having the same texture or colour. Segmentation is quite essential to image feature extraction and subsequent classification of the resultant features. This step is very significant due to overlapping characters that form the license plate. There are three main forms of characters that are overlapping vertically, ligature, diacritics, horizontal overlap, and two connected characters. The task will be more difficult for those different forms of which are joined.

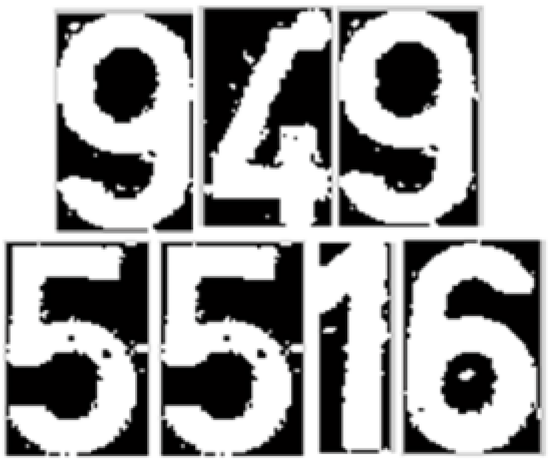
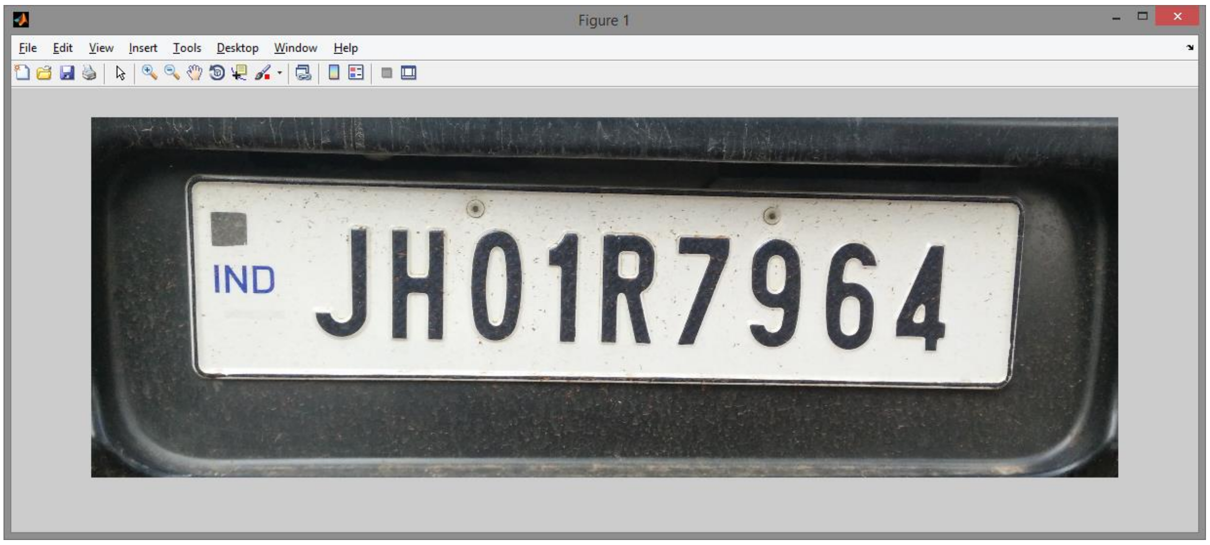


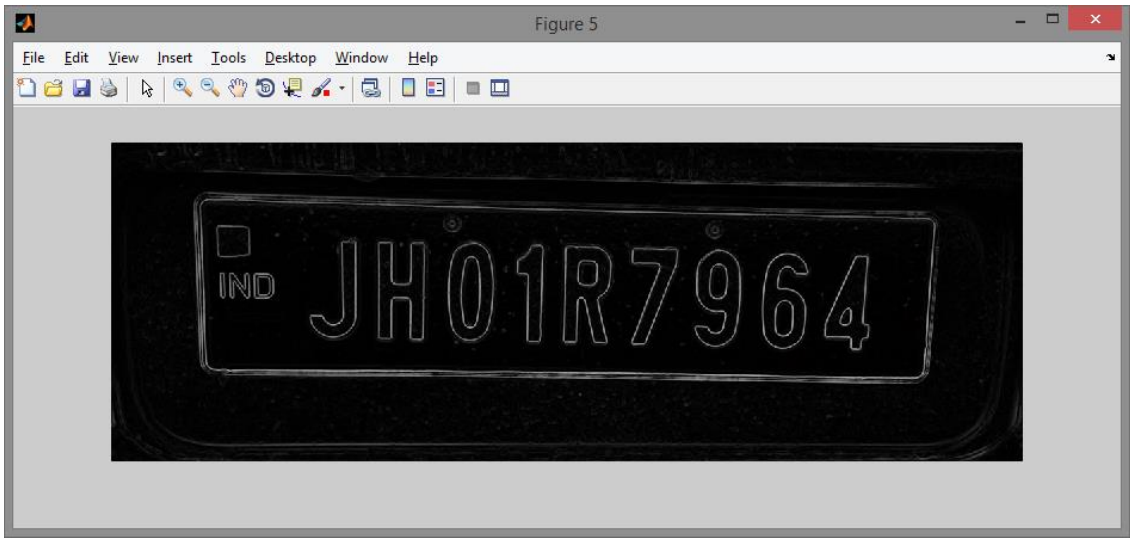
Figure 5. Segmented characters of License plate.

9.5 CHARACTER RECOGNITION

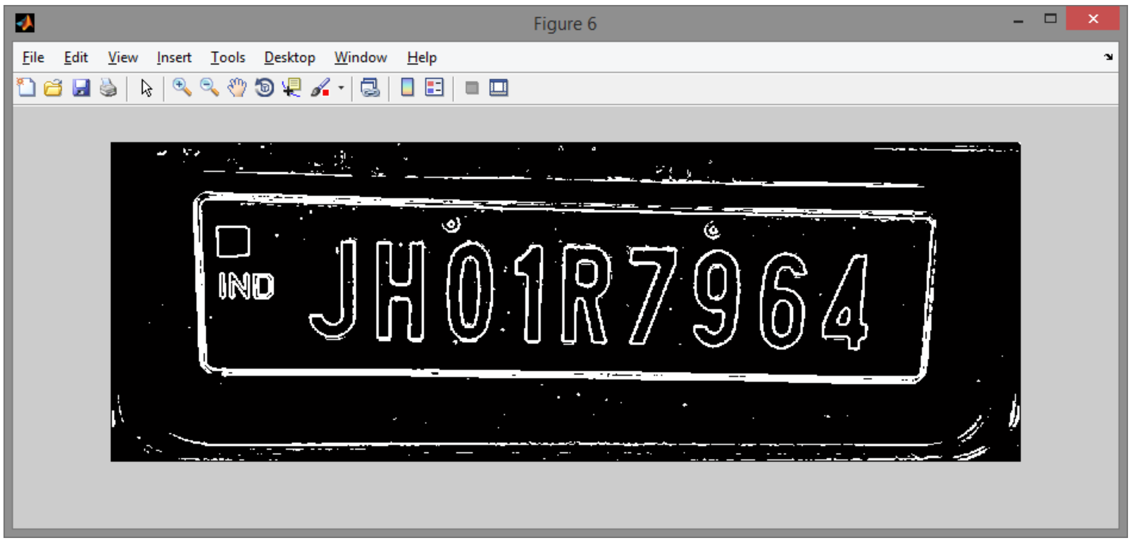
The two methods are template-matching method and neural network method. Here we are evaluating using template-matching method. Character recognition is final step in vehicle license plate detection and recognition is reading of single characters and numbers. This step is very important for example at the entrance to car-park or for the police for stolen cars search. Single elements on license plate must be segmented and analysed. The analysis is called as Optical Character Recognition (OCR).

The steps involved in this stage are the method that was devised for the intital OCR System. Refer to page 8.

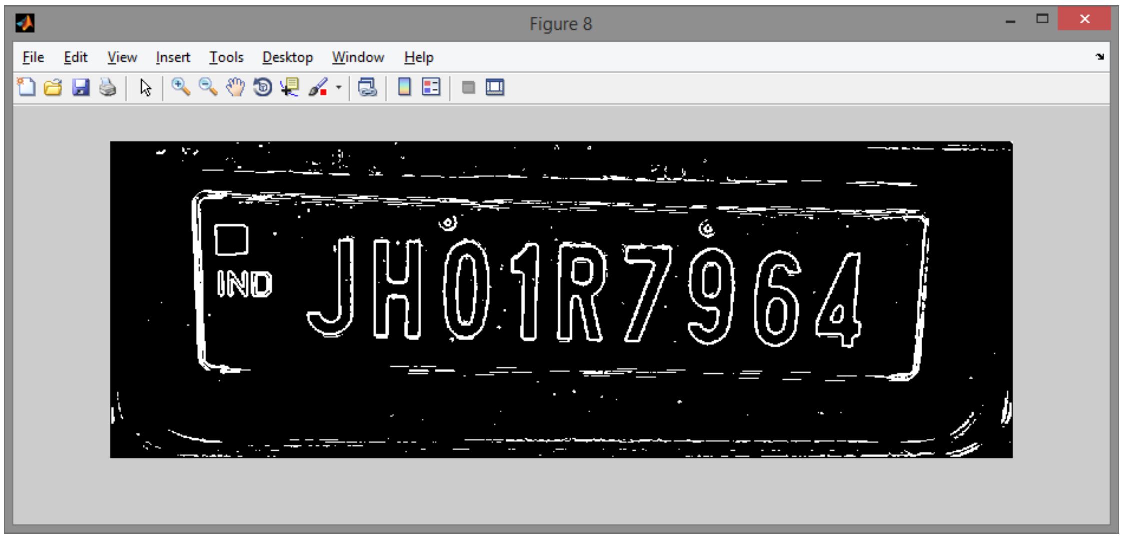




$\downarrow$



$\downarrow$



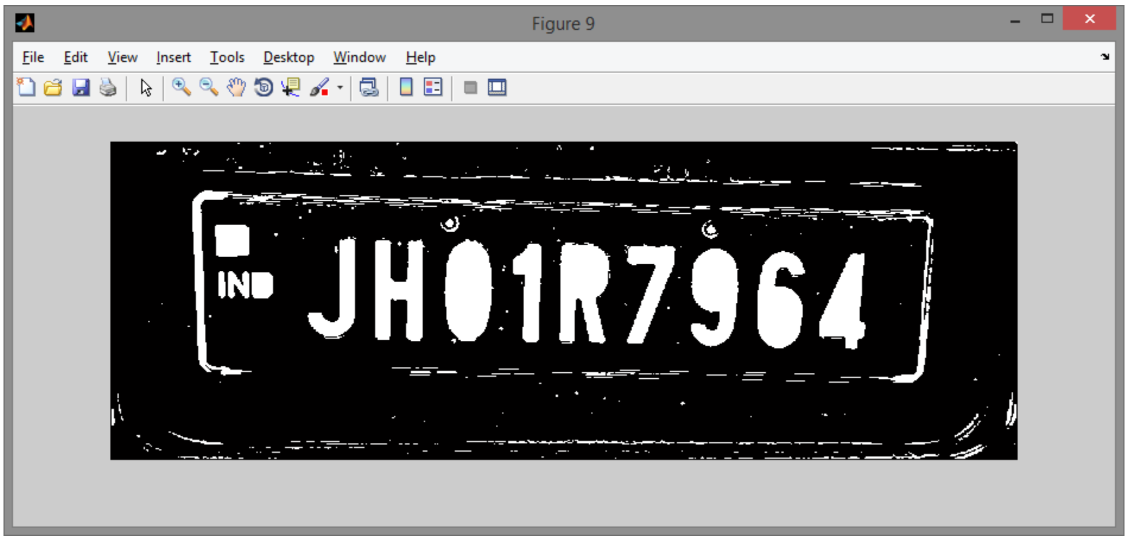


Figure 9
File Edit View Insert Tools Desktop Window Help
IND JH01R7964

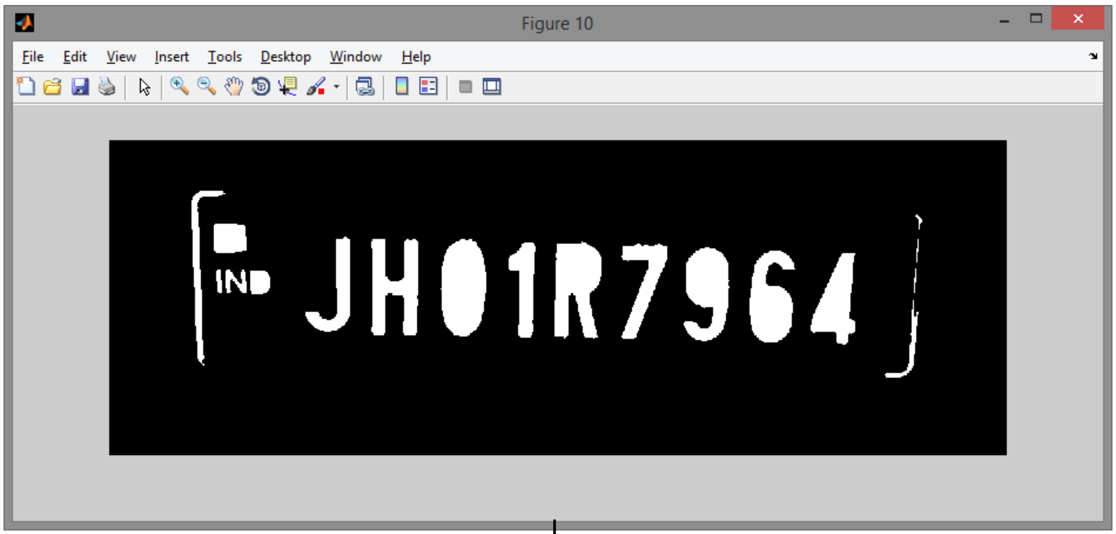
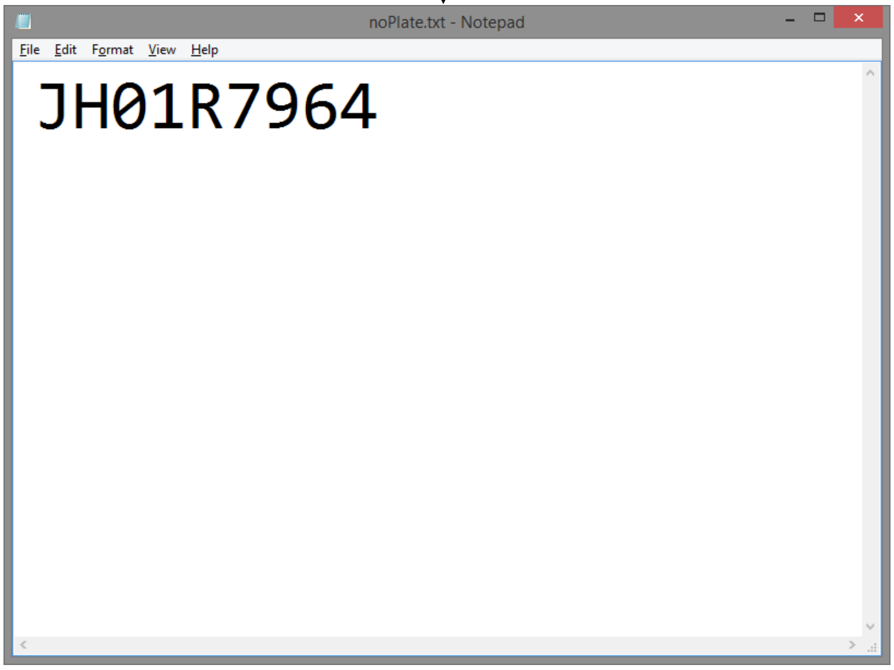


Figure 10
File Edit View Insert Tools Desktop Window Help
IND JH01R7964


noPlate.txt - Notepad
File Edit Format View Help

```
JH01R7964
```

9.6 Code Snippet

```
function numberPlateExtraction

f=imread('Input/car3.jpg');
f=imresize(f,[400 NaN]);
g=rgb2gray(f);
g=medfilt2(g,[3 3]);
se=strel('disk',1);
gi=imdilate(g,se);

ge=imerode(g,se);
gdiff=imsubtract(gi,ge);
gdiff=mat2gray(gdiff);
gdiff=conv2(gdiff,[1 1;1 1]);
gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1);
B=logical(gdiff);

% Eliminating the possible horizontal lines from the output image of
regiongrow that could be edges of license plate.

er=imerode(B,strel('line',50,0));
out1=imsubtract(B,er); figure,imshow(out1);

% Filling all the regions of the image.
F=imfill(out1,'holes'); figure,imshow(F);

% Thinning the image to ensure character isolation.
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));

% Selecting all the regions that are of pixel area more than 100.
final=bwareaopen(H,100);

Iprops=regionprops(final,'BoundingBox','Image');

% Selecting all the bounding boxes in matrix of order numberofboxesX4;
NR=cat(1,Iprops.BoundingBox);

% Calling of controlling function.
r=controlling(NR);
if ~isempty(r)
    I={Iprops.Image};
    noPlate=[];
    for v=1:length(r)
        N=I{1,r(v)};
        letter=readLetter(N);
        %end
        noPlate=[noPlate letter];
    end
    fid = fopen('noPlate.txt', 'wt');
    fprintf(fid,'%s\n',noPlate);
    fclose(fid);
    winopen('noPlate.txt')

else
    fprintf('Unable to extract the characters from the number plate.\n');
    fprintf('The characters on the number plate might not be clear or
touching with each other or boundries.\n');
end
end
```

10. Text to Speech Synthesis

Text to Speech (TTS) refers to the ability of computers to read text aloud. A **TTS Engine** converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. TTS engines with different languages, dialects and specialized vocabularies are available through third-party publishers.

Speech Synthesis is the artificial production of human speech. A computer system used for this purpose is called a **speech computer** or **speech synthesizer**, and can be implemented in software or hardware products. A **text-to-speech (TTS)** system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech.

Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. Systems differ in the size of the stored speech units; a system that stores phones or diaphones provides the largest output range, but may lack clarity. For specific usage domains, the storage of entire words or sentences allows for high-quality output. Alternatively, a synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output

A text-to-speech system (or "engine") is composed of two parts:[3] a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called *text normalization*, *pre-processing*, or *tokenization*. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called *text-to-phoneme* or *grapheme-to-phoneme* conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end—often referred to as the *synthesizer*—then converts the symbolic linguistic representation into sound. In certain systems, this part includes the computation of the *target prosody* (pitch contour, phoneme durations),[4] which is then imposed on the output speech.

Modern Windows desktop systems can use SAPI 4 and SAPI 5 components to support speech synthesis and speech recognition. SAPI 4.0 was available as an optional add-on for Windows and Windows 98. Windows 2000 added Narrator, a text–to–speech utility for people who have visual impairment. Third-party programs such as CoolSpeech, Textaloud and Ultra Hal can perform various text-to-speech tasks such as reading text aloud from a specified website, email account, text document, the Windows clipboard, the user's keyboard typing, etc. Not all programs can use speech synthesis directly. Some programs can use plug-ins, extensions or add-ons to read text aloud. Third-party programs are available that can read text from the system clipboard.

Here In this Project of ours, we implemented TTS using a .NET library of Microsoft and Implemented TTS in the following fashion :

1. First extract the text using the OCR algorithm we developed.
2. Use the following code snippet to covert text-to speech

```
function tts( text )
% This function converts text into speech.
% You can enter any form of text (less than 512 characters per line) into
% this function and it speaks it all.
if nargin<1
    text = 'Please call this function with text';
end
try
    NET.addAssembly('System.Speech');
    Speaker = System.Speech.Synthesis.SpeechSynthesizer;
    if ~isa(text,'cell')
        text = {text};
    end
    for k=1:length(text)
        Speaker.Speak (text{k});
    end
catch
    warning(['If this is not a Windows system or ' ...
        'the  .Net  class  exists  you  will  not  be  able  to  use  this
function.']);
end
```

11. Challenges faced

❖ Construction of Database

Construction of the Learned Database, having the most appropriate character image of a particular size, was a tedious task.

❖ Segmentation and Recognition of Disconnected Characters

Characters such as "I" and "j" have a TITTLE which was being extracted as a different character altogether.

❖ Line Extraction for irregular line breaks

Images which did not have a clear demarcation between two lines failed in the segmentation process.

❖ Number Plate Recognition

Image should not be blurred, no reflection of light from any character, number plate should be dominant, prominent and distinct in the image. Also the recognition system is country specific (no of characters on the number plate are different in different countries).

References

- ❖ Edge Based Text Extraction From Complex Images by Xiaoqing Liu and Jagath Samarbandhu
- ❖ Character Recognition IEEE paper from KATHEY, University of berkely.
- ❖ Automatic Text Detection using Morphological Operations and Inpainting by *Khyati Vaghela*
- ❖ Font and Background Color Independent Text Binarization by *T.Kasar , J.Kumar , A.G. Ramkrishnan*
- ❖ OCR for Devnagari Script by *Mahesh Goyani*
- ❖ Sheroz Khan, Rafiqul Islam Othman khalifa, "Malaysian Vehicle License Plate Recognition," The International Arab Journal of Information Technology , pp. 359-364, 2007.
- ❖ International Journal on Cybernetics & Informatics (IJCI) Vol.2, No.1, February 2013
- ❖ Rong Li, Musa Yassin Fort, and Georgis C. Anagnostopolous , Multi-stage Automatic License Plate Location & Recognition
- ❖ MATLAB Documentation
- ❖ http://cnx.org/content/m12531/latest/#
- ❖ http://lab.fs.uni-lj.si/lasin/wp/IMIT_files/neural/doc/seminar5.pdf