VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

OPTICAL CHARACTER RECOGNISTION USING MATALB

Signals and Systems

SUBMITTED BY:

V J Venkat Akash 19BEC0567

Fatema Malik 19BEC0662

T Rushika 19BEC0725

Apurva Singh 19BEC0753

Abstract- *In todays' world advancement in sophisticated scientific techniques is pushing further the limits of human outreach in various fields of technology. One such field is the field of character recognition commonly known as OCR which comes under signal processing and system environment (Optical Character Recognition).*

In this fast paced world there is an immense urge for the digitalization of printed documents and documentation of information directly in digital form. And there is still some gap in this area even today. OCR techniques and their continuous improvisation from time to time is trying to fill this gap. This project is about devising an algorithm for recognition of hand written characters also known as HCR (Handwritten Character Recognition) leaving aside types of OCR that deals with recognition of computer or typewriter printed characters.

A novel technique is proposed for recognition English language characters using Artificial Neural Network including the schemes of feature extraction of the characters and implemented. The persistency in recognition of characters by the AN network was found to be more than 90% of times.

Introduction –

Character recognition, usually abbreviated to optical character recognition or shortened OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. It is a field of research in pattern recognition, artificial intelligence and machine vision. Though academic research in the field continues, the focus on character recognition has shifted to implementation of proven techniques. Optical character recognition is a scheme which enables a computer to learn, understand, improvise and interpret the written or printed character in their own language, but present correspondingly as specified by the user. Optical Character Recognition uses the image processing technique to identify any character computer/typewriter printed or hand written. A lot of work has been done in this field. But a continuous improvisation of OCR techniques is being done based on the fact that algorithm must have higher accuracy of recognition, higher persistency in number of times of correct prediction and increased execution time.

The idea is to device efficient algorithms which get input in digital image format. After that it processes the image for better comparison. Then after the processed image is compared with already available set of font images. The last step gives a prediction of the character in percentage accuracy.

The objective of this project is to identify handwritten characters with the use of neural networks. We have to construct suitable neural network and train it properly. The program should be able to extract the characters one by one and map the target output for training purpose. After automatic processing of the image, the training dataset has to be used to train “classification engine” for recognition purpose. The program code has to be written in MATLAB and supported with the usage of Graphical User Interface (GUI).

PROBLEM DESCRIPTION-

Before OCR can be used, the source material must be scanned using an optical scanner (and sometimes a specialized circuit board in the PC) to read in the page as a bitmap (a pattern of dots). Software to recognize the images is also required.

The character recognition software then processes these scans to differentiate between images and text and determine what letters are represented in the light and dark areas.

Older OCR systems match these images against stored bitmaps based on specific fonts. The hit -or-miss results of such pattern -recognition systems helped establish OCR's reputation for inaccuracy.

Today's OCR engines add the multiple algorithms of **neural network technology** to analyze the stroke edge, the line of discontinuity between the text char acters, and the background. Allowing for irregularities of printed ink on paper, each algorithm averages the light and dark along the side of a stroke, matches it to known characters and makes a best guess as to which character it is. The OCR software then averages or polls the results from all the algorithms to obtain a single reading .

OCR software can recognize a wide variety of fonts, but handwriting and script fonts that mimic handwriting are still problematic, therefore additional help of neural networ k power is required. Developers are taking different approaches to improve script and handwriting recognition. As mentioned above, one possible approach of handwriting recognition is with the use of neural networks.

Neural networks can be used, if we have a suitable dataset for training and learning purposes. Datasets are one of the most important things when constructing new neural network. Without proper dataset, training will be useless. There is also a saying abou t pre-processing and training of data and neural network: “Rubbish -in, rubbish-out”. So how do we produce (get) a proper dataset? First we have to scan the image. After the image is scanned, we define processing algorithm, which will extract important attributes from the image and map them into a database or better to say dataset. Extracted attributes will have numerical values and will be usually stored in arrays. With these values, neural network can be trained and we can get a good end results.

The most important attributes for handwriting algorithms are:

Negative image of the figure, where the input is defined as 0 or 1. 0 is black, 1 is white, values in between shows the intensity of the relevant pixel.

The horizontal position, counting pixels from the left edge of the image, of the center of the smallest rectangular box that can be drawn with all "on" pixels inside the box.

The vertical position, counting pixels from the bottom, of the above box.

The width, in pixels, of the box.

The height, in pixels, of the box.

The total number of "on" pixels in the character image.

The mean horizontal position of all "on" pixels relative to the center of the box and divided by the width of the box. This feature has a negative value if the image is "leftheavy" as would be the case for the letter L.

The mean vertical position of all "on" pixels relative to the center of the box and divided by the height of the box.

The mean squared value of the horizontal pixel distances as measured in 6 above. This attribute will have a higher value for images whose pixels are more widely separated in the horizontal direction as would be the case for the letters W or M.

The mean squared value of the vertical pixel distances as measured in 7 above.

The mean product of the horizontal and vertical distances for each "on" pixel as measured in

6 and 7 above. This attribute has a positive value for diagonal lines that run from bottom left to top right and negative value for diagonal lines from top left to bottom right.

The mean value of the squared horizontal distance tunes the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.

The mean value of the squared vertical distance times the horizontal distance for each "on" pixel. This measures the correlation of the vertical variance with the horizontal position.

The mean number of edges (an "on" pixel immediately to the right of either an "off pixel or the image boundary) encountered when making systematic scans from left to right at all vertical positions within the box. This measure distinguishes between letters like "W" or "M" and letters like "I" or "L."

The sum of the vertical positions of edges encountered as measured in 13 above. This feature will give a higher value if there are more edges at the top of the box, as in the letter "Y."

The mean number of edges (an "on" pixel immediately above either an "off pixel or the image boundary) encountered when making systematic scans of the image from bottom to top over all horizontal positions within the box.

The sum of horizontal positions of edges encountered as measured in 15 above.

APPROACH

To solve the defined handwritten character recognition problem of classification we used MATLAB computation software with Neural Network Toolbox and Image Processing Toolbox add-on. The computation code is divided into the next categories:

Pre-processing of the image

Feature extraction

Creating an Artificial Neural Network

Training & Testing of the network

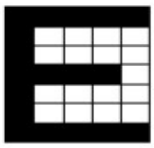
Recognition

CREATING AND TRAINING OF NETWORK

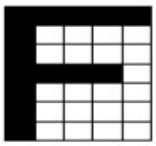
In case of character recognition we have to create a 2D vector of character images which can be fed to the network as ideal set of input variables. In our case there is a total of 26 capital English letters which we are to recognize.

Below is a set of characters written in binary form of 7x5 sized matrix of 26 capital English letters

1	1	1	1	1
1	0	0	0	0
1	0	0	0	0
1	1	1	1	0
1	0	0	0	0
1	0	0	0	0
1	1	1	1	1

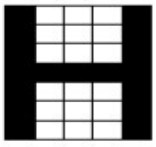


1	1	1	1	1
1	0	0	0	0
1	0	0	0	0
1	1	1	1	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0

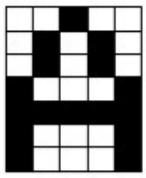


0	1	1	1	0
1	0	0	0	1
1	0	0	0	0
1	0	0	0	0
1	0	1	1	1
1	0	0	0	1
0	1	1	1	0

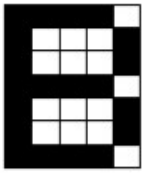
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1



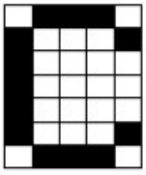
0	0	1	0	0
0	1	0	1	0
0	1	0	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1



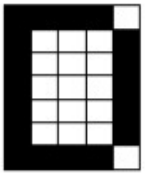
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0



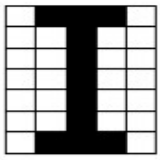
0	1	1	1	0
1	0	0	0	1
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	1
0	1	1	1	0



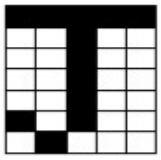
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0



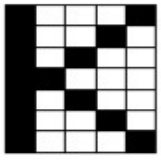
0	1	1	1	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	1	1	1	0



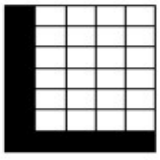
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
1	0	1	0	0
0	1	0	0	0



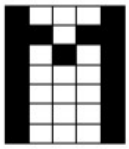
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1



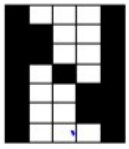
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
1	1	1	1	1



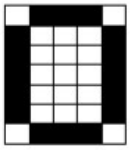
1	0	0	0	1
1	1	0	1	1
1	0	1	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1



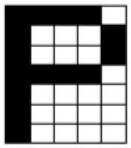
1	0	0	0	1
1	1	0	0	1
1	1	0	0	1
1	0	1	0	1
1	0	0	1	1
1	0	0	1	1
1	0	0	0	1



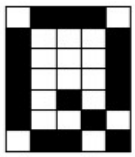
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0



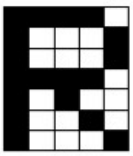
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0



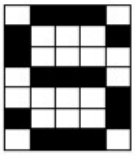
0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	1	0	1
1	0	0	1	0
0	1	1	0	1



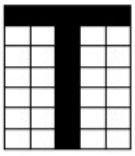
1	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	1	1	1	0
1	0	1	0	0
1	0	0	1	0
1	0	0	0	1



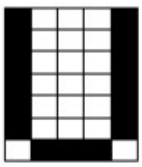
0	1	1	1	0
1	0	0	0	1
1	0	0	0	0
0	1	1	1	0
0	0	0	0	1
1	0	0	0	1
0	1	1	1	0



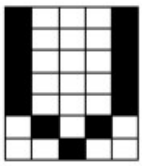
1	1	1	1	1
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0



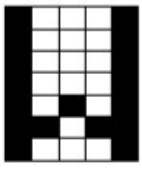
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0



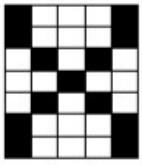
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	0	1	0
0	0	1	0	0



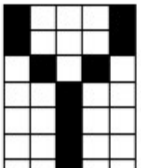
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	1
1	0	0	0	1



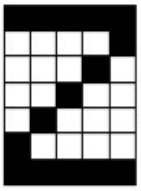
1	0	0	0	1
1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	0	0	0	1



1	0	0	0	1
1	0	0	0	1
0	1	0	1	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0



1	1	1	1	1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0
1	1	1	1	1

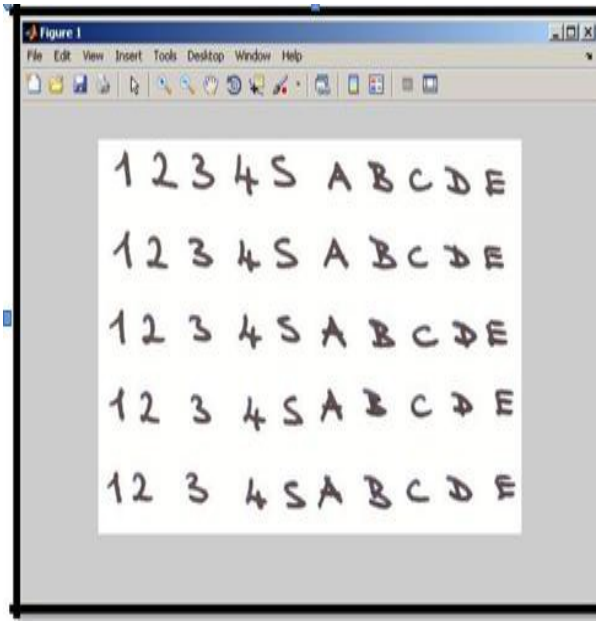


Pre-processing of the image

First of all the image on which the characters are written by hand is required. Below is the example of one case in which an image sample is taken

I = imread('training.bmp');

imshow(I)



Grey-scaling and Binarization of image

This cell of codes convert the RGB to gray.

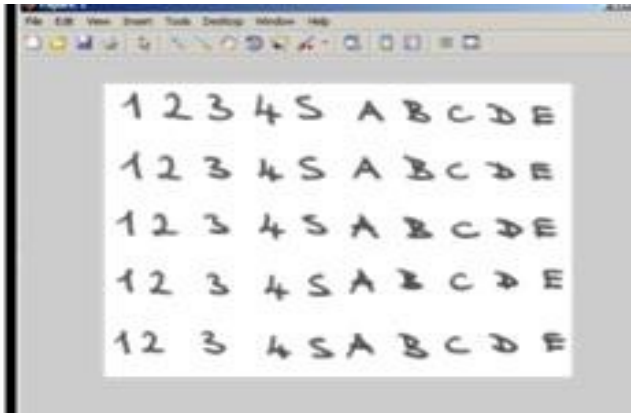
Igray = rgb2gray(I);

imshow(Igray)

This cell of codes convert the gray to binary image.

Ibw = im2bw(Igray,graythresh(Igray));

imshow(Ibw)

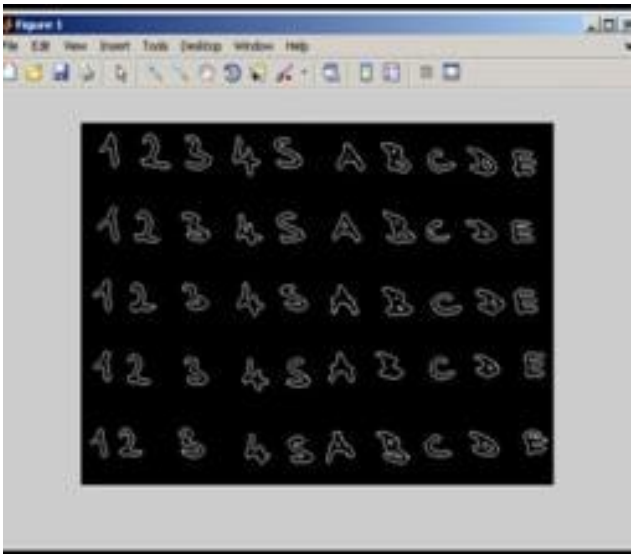


Edge detection

This cell of codes detect the edge of the image.

Iedge = edge(uint8(Ibw));

imshow(Iedge)



Inversion and Dilation of binary image

se = strel('square',2);

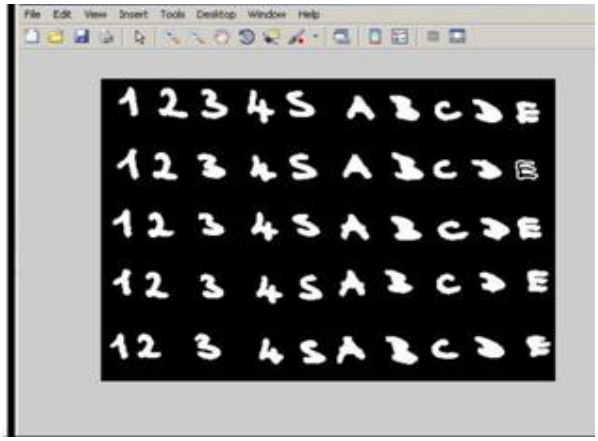
Iedge2 = imdilate(Iedge, se);

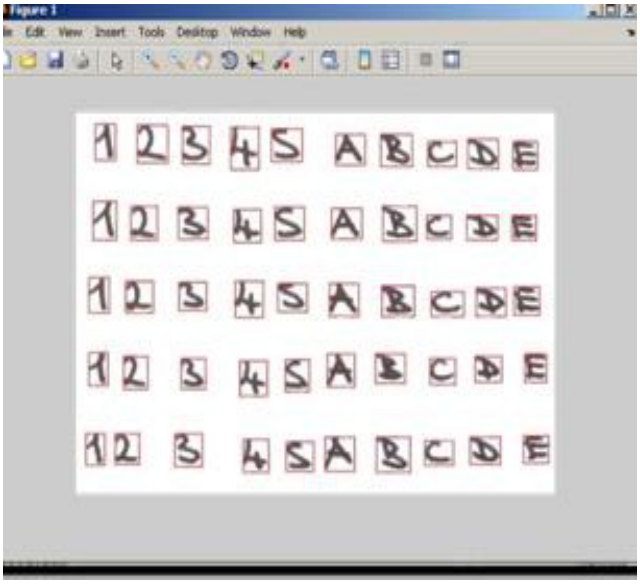
imshow(Iedge2);

Image Filling

Ifill= imfill(Iedge2,'holes');

imshow(Ifill)





Indexing and Boxing of characters

[Ilabel num] = bwlabel(Ifill);

disp(num);

Iprops = regionprops(Ilabel);

Ibox = [Iprops.BoundingBox];

Ibox = reshape(Ibox,[4 50]);

imshow(I)

Plot the Object Location

This cell of codes plot the object locations.

hold on;

for cnt = 1:50
rectangle('position',Ibox(:,cnt),'edgecolor','r');

end

Text Recognition Using the ocr Function

```
I=imread("training.png")
results=ocr(I);
word=results.Words{2};
wordBox=results.WordBoundingBoxes(
5,:) figure;
Iname=insertObjectAnnotation(I,'re
ctangle',wordBox,word);
imshow(Iname);
```

Information Returned By The ocr Function

```
lowConfidenceIdx=
results.CharacterConfidences <
0.5;
lowConfBBoxes
=result.CharacterBoundingBoxes(low
ConfidenceIdx, :);
lowConfVal =
result.CharacterConfidences(lowCon
fidenceIdx);
str=
sprintf('confidence=%f',lowConfVal
);
ILowconf=
insertObjectAnnotation(I,'rectangl
e',lowConfBBoxes,str);
figure;
imshow(ILowconf);
```

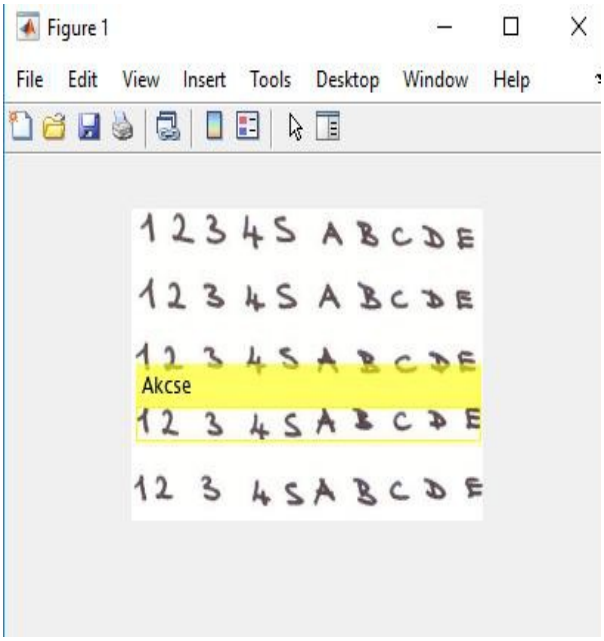
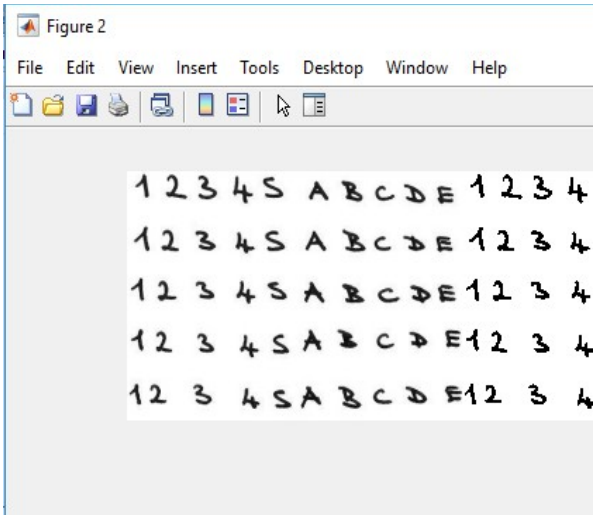
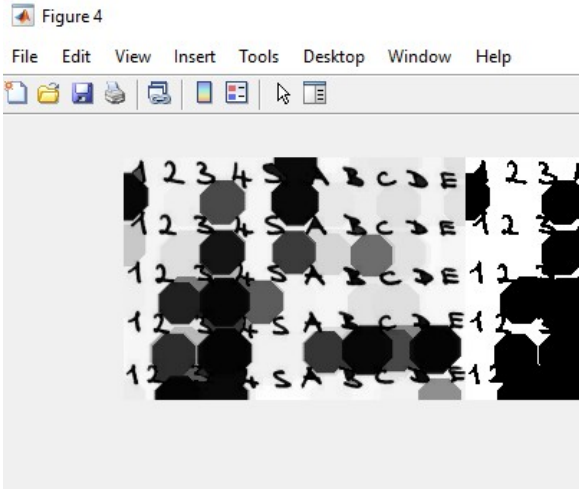


Image Pre-processing Techniques To Improve Results

```
I=imread("training.png")
I=rgb2gray(I);
figure;
imshow(I);
results=ocr(I);
results.Text;
results=ocr(I,'TextLayout','Block'
); results.Text;
BW=imbinarize(I);
figure;
imshowpair(I,BW,'montage');
Icorrected=imtophat(I,strel('disk'
,15));
BW1=imbinarize(Icorrected);
figure;
imshowpair(Icorrected,BW1,'montage
');
marker = imerode(Icorrected,
strel('line',10,0));
Iclean = imreconstruct(marker,
Icorrected);
BW2 = imbinarize(Iclean); figure;
imshowpair(Iclean,BW2,'montage');
results =
ocr(BW2,'TextLayout','Block');
results.Text
regularExpr = '\d';
bboxes =
locateText(results,regularExpr,'Us
eRegexp',true);
```

```
digits =
regexp(results.Text,regularExpr,'m
atch');
Idigits =
insertObjectAnnotation(I,'rectangl
e',bboxes,digits);
figure;
imshow(Idigits);
```

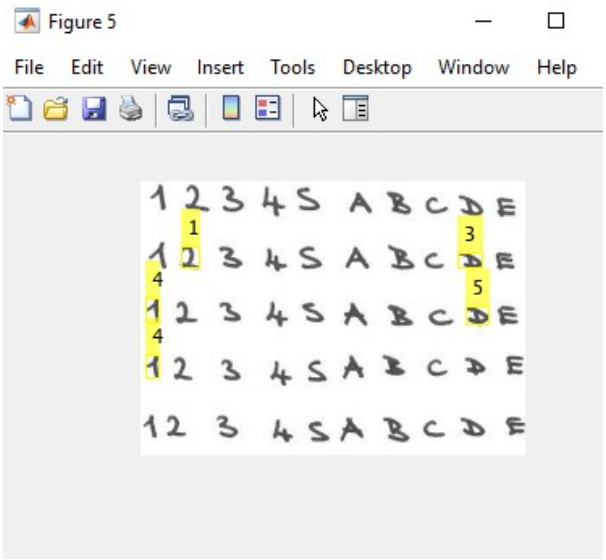


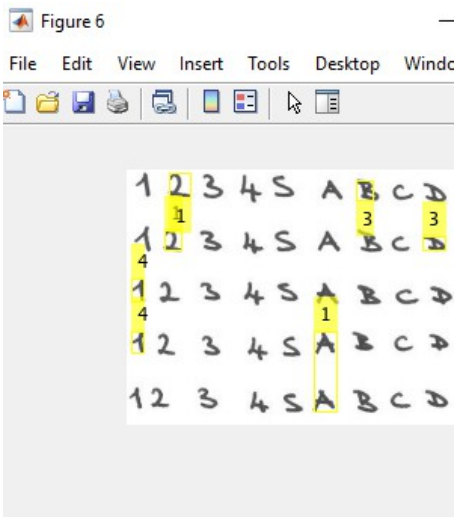


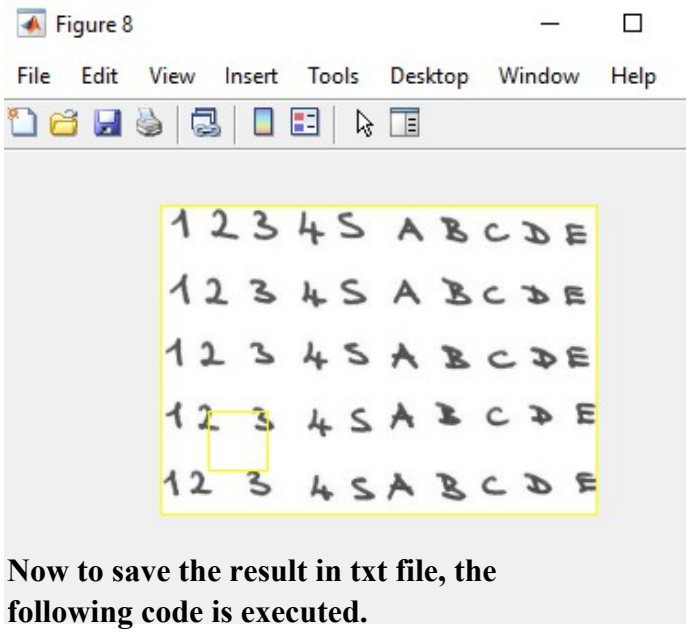
ROI-based Preprocessing To Improve Results

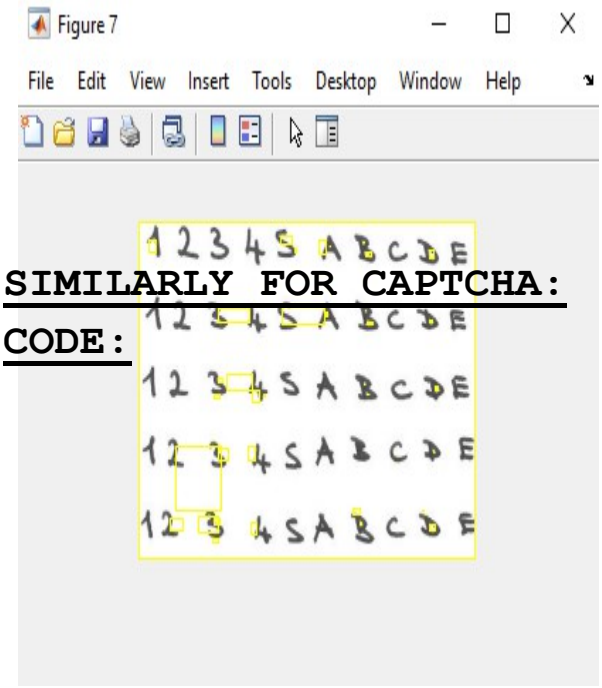
```
results = ocr(BW2,
'CharacterSet','0123456789','TextL
ayout','Block');
results.Text
[sortedConf, sortedIndex] =
sort(results.CharacterConfidences,
'descend');

indexesNaNsRemoved = sortedIndex(
~isnan(sortedConf) );

topTenIndexes =
indexesNaNsRemoved(1:10);

digits =
num2cell(results.Text(topTenIndexe
s));
bboxes =
results.CharacterBoundingBoxes(top
TenIndexes, :);

Idigits =
insertObjectAnnotation(I,'rectangl
e',bboxes,digits);

figure;
imshow(Idigits);
blobAnalyzer =
vision.BlobAnalysis('MaximumCount'
,500);

[area,centroids,roi] =
step(blobAnalyzer,BW1);

img =
insertShape(I,'rectangle',roi);
figure;
imshow(img);
areaConstraint = area > 300;

roi = double(roi(areaConstraint,
:));

img =
insertShape(I,'rectangle',roi);
figure;
imshow(img);
width  = roi(:,3);
height = roi(:,4);
aspectRatio = width ./ height;


roi = roi( aspectRatio > 0.25 &
aspectRatio < 1 ,:);
img =
insertShape(I,'rectangle',roi);
figure;
imshow(img);
```

```
roi(:,1:2) = roi(:,1:2) - 4;
roi(:,3:4) = roi(:,3:4) + 8;
results = ocr(BW1,
roi,'TextLayout','Block');

text = deblank( {results.Text} );
img  =
insertObjectAnnotation(I,'rectangl
e',text);
figure;
imshow(img)
```









Now to save the result in txt file, the following code is executed.

```
c=rand(14,20)
dlmwrite('apurva.txt',c);
```

SIMILARLY FOR CAPTCHA:

CODE:

```
clc
clear all
close all
x=imread('Captchaone.png');
imshow(x);
title('Original Image');
figure;
x=rgb2gray(x);
imshow(x);
title('Grayscale Image');
figure;
x=imbinarize(x);
imshow(x);
title('Binary Image');
x=~x;
g=strel('disk',0);
x=imclose(x,g);
figure;
imshow(x);
title('Image after applying Morphological
Closing');
k=bwmorph(x,'thin',1);
figure;
imshow(k);
title ('Image after Morphological Thinning')
M=ocr(k);
disp(M.Text);
```

COMMAND WINDOW

New to MATLAB? See resources for Getting Started.

```
W6 8HP
```

Conclusion-

Classification of characters and learning of image processing techniques is done in this project. Also the scheme through which project is achieved is Artificial Neural Network scheme. The result which was got was correct up to more than 90% of the cases, but it would be improved at the end. This work was basically focused on envisaging methods that can effici ently extract feature vectors from each indiv idual character.This method gave more or less required and effective result

both for feature extraction as well as recognition.
There are also differentmethods through which
‘handwrittencharacter recognition’ is achieved.

REFERENCES:

.[1] JAMSHED MEMON 1 , MAIRA SAMI 2 , RIZWAN AHMED KHAN 3 , AND MUEEN UDDIN 4, “Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)”, July 28, 2020, date of current version August 14, 2020.

[2] M. Kumar, S. R. Jindal, M. K. Jindal, and G. S. Lehal, ‘‘Improved recognition results of medieval handwritten Gurmukhi manuscripts using boosting and bagging methodologies,’’ Neural Process. Lett., vol. 50, pp. 43–56, Sep. 2018.

[3] H. Zhang and Z. Cheng, ‘‘An advanced pyramid network technology for optical character recognition,’’ J. Phys., Conf. Ser., vol. 1302, Aug. 2019, Art. no. 022042

[4] R. Ptucha, F. P. Such, S. Pillai, F. Brockler, V. Singh, and P. Hutkowski, ‘‘Intelligent character recognition using fully convolutional neural networks,’’ Pattern Recognit., vol. 88, pp. 604–613, Apr. 2019

[5]S. Rajasekaran and G. V. Pai, Neural Networks, Fuzzy Systems and Evolutionary Algorithms: Synthesis and Applications. New Delh, India: PHI Learning, 2017.