Image Processing Using Artificial Intelligence

Apoorva Kulkarni, Swastika Gayen, Vedire Chandu Reddy, Harshith Raj Boddu, Tuttagunta Lakshmi Narasimha Swamy, Mandadapu Sai Mohana Manikanta , Donthi Gnaneshwar Reddy, Mandarapu Lokesh

INTRODUCTION

1.1 IMAGE PROCESSING

Image processing usually refers to digital image processing, but optical and analog image processing also are possible. The acquisition of images (producing the input image in the first place) is referred to as imaging.

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems.

In imaging science, image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

Closely related to image processing are computer graphics and computer vision. In computer graphics, images are manually made from physical models of objects, environments, and lighting, instead of being acquired (via imaging devices such as cameras) from natural scenes, as in most animated movies. Computer vision, on the other hand, is often considered high-level image processing out of which a machine/computer/software intends to decipher the physical contents of an image or a sequence of images (e.g., videos or 3D full-body magnetic resonance scans).

In modern sciences and technologies, images also gain much broader scopes due to the ever growing importance of scientific visualization (of often large-scale complexscientific/experimental data). Examples include microarray data in genetic research, or real-time multi-asset portfolio trading in finance

2.1 OCR (Optical Character Recognition)

Optical character recognition (OCR) is the mechanical or electronic conversion of images of typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems that have a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

SYSTEM DESIGN

3.1 System Architecture

Optical Character Recognition (OCR) refers to the technique of analyzing images to find and recognize the text in them. The input to an OCR system is therefore an image file and the output is the extractor text, in any machine readable form. Optical Character Recognition is extremely relevant in today's world in the backdrop of the digital revolution. While 92-96% of human knowledge is locked up in printed and handwritten form, the amazing computing power that we seem to have at our disposal is entirely in the digital realm. OCR systems play an important part in bridging the gap and bridging these together. This technology assumes great significant in the local context, where OCR systems in do not exist. OCR is a software device which will convert a scanned image of printed/handwritten document into a computer editable text. With the help of such a program we can digitalize old books in so that it can be preserved and we can have new copies very easily. Handwriting recognition will have breath taking impacts in the world of data entry. OCR will make the text machine readable and should therefore be considered if the text is to be reused, edited or reformatted. The text should be available for full text information retrieval. The text is to be coded in HTML

or SGML. The text should be available to adaptive equipment for the visually impaired. File size is of concern. Resources are available to perform OCR and correct the output. The following figure in 3.1 illustrates about the Block diagram of OCR.

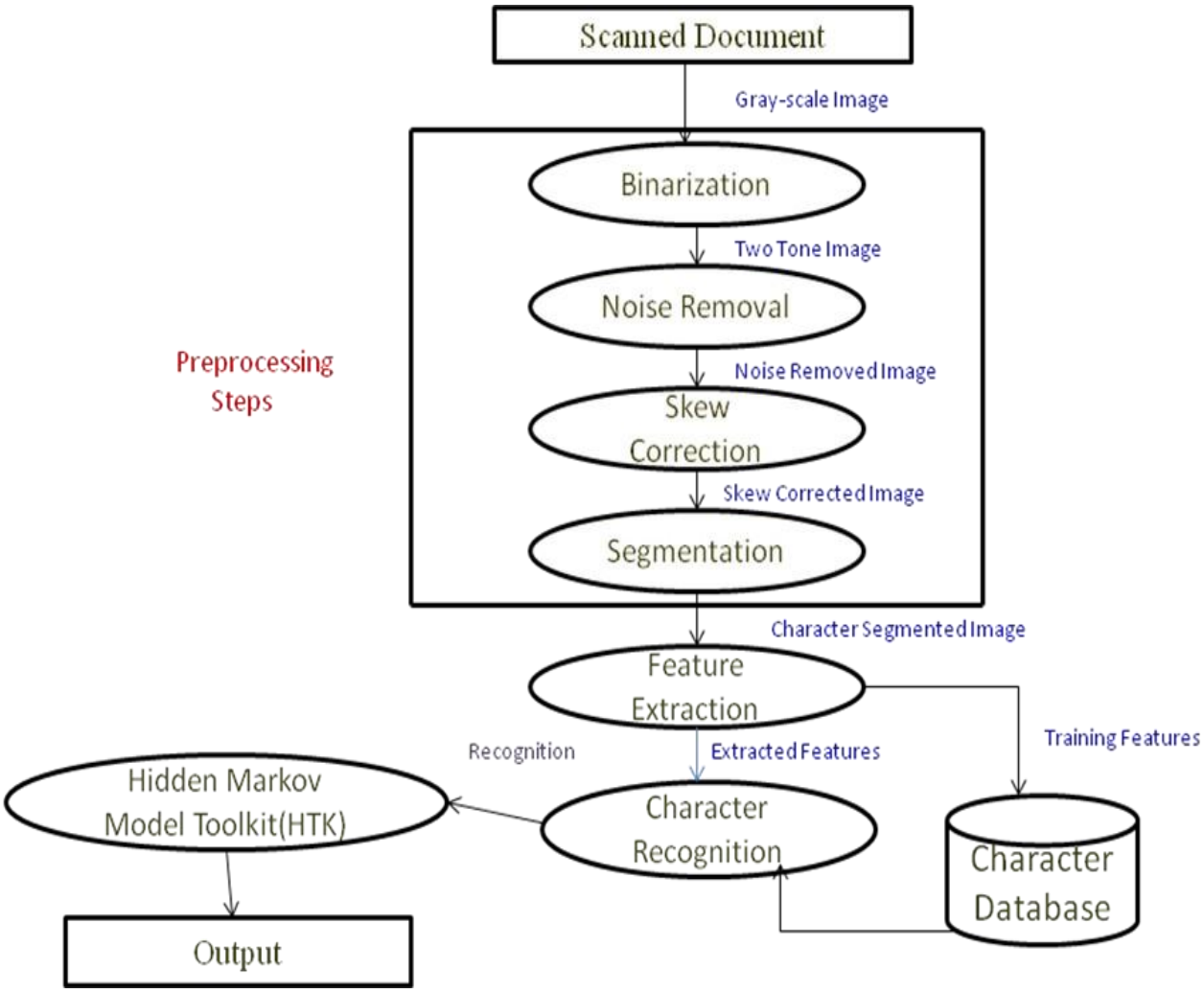


Figure 5.1: Block Diagram of the Optical Character Recognition

In this software, a `**scanned image of the printed/handwritten document**` can be converted into a computer **editable text file**. This will avoid cumbersome task of typing it again. The handwritten document is first scanned into an image file of any format (preferably Bitmap file). Next the user will load the image file into the OCR (Modified Optical Character Recognition) and OCR will recognize the glyphs in the document and save it as a computer editable text file. Correcting

the errors or editing the text content in the recognized text will be made possible in OCR. There will be facility to recognize text from documents having colored text. Facilities to correct errors using a dictionary, to recognize text from documents having embedded images, remove salt and pepper noise from the scanned image and correcting the skew of the scanned image can be incorporated later OCR can be further integrated as a plug-in into a document processing software like Microsoft Word. For the recognition lines and words are isolated based on the distance between the glyph groups. Then the glyphs in each word are isolated by identifying the pixel groups. The seven 'Hu moments' of each glyph is then extracted and these moments are matched with the library of feature. Character having the highest degree of match will be then the recognized character. Feature Library of each font can be created by a training process. A scanned image of alphabet or a font file can be given as the training data. The potential of OCR systems is enormous in situations like digitalization of old books in library which got no electronic copies and can't be preserved. Handwriting recognition will allow very easy and fast input of data than typewriting.

3.2 System Overall Architecture

The scanned document image file is loaded into the preprocessing stages that involves

- Binarization
- Noise Removal
- Skew Correction
- Segmentation

After the preprocessing is over, features are extracted from the characters and then they are sent into the recognition stage, which is done by Hidden Markov Model (HMM) to get the recognized output. The following figure in 3.2 explains the architecture of the proposed model.

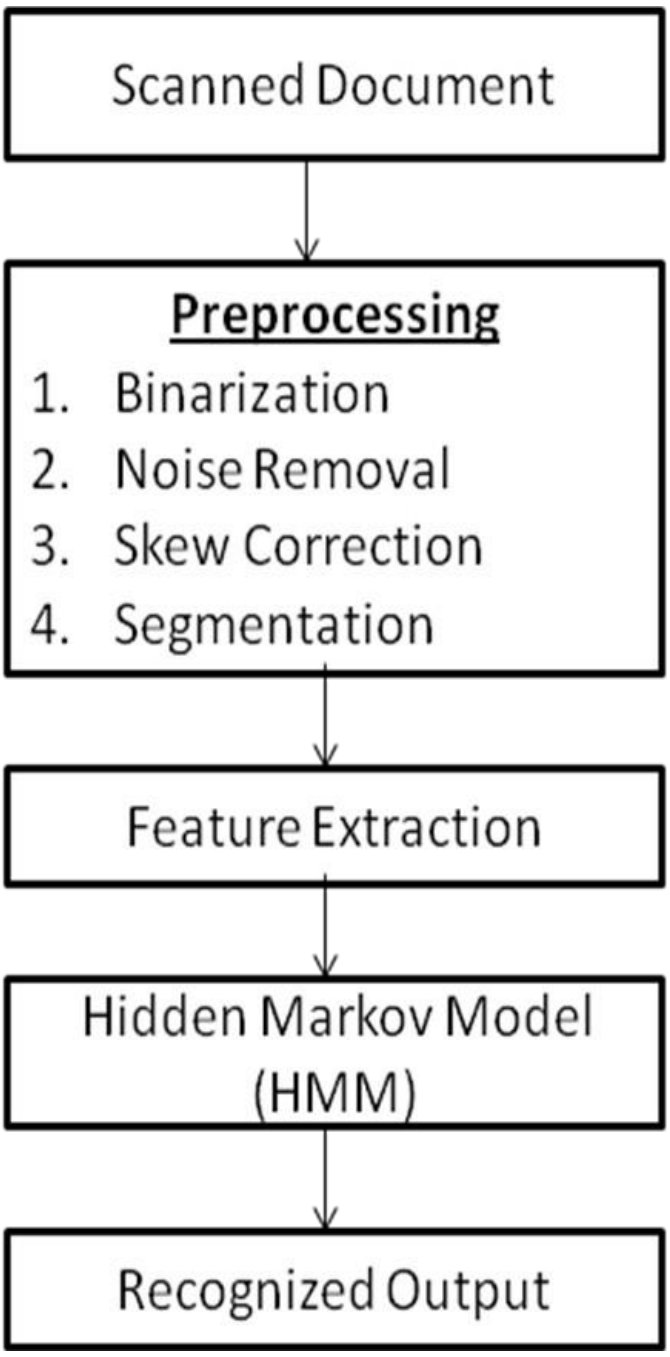


Figure 3.2: Architecture of the Proposed System

3.3 Context Diagram

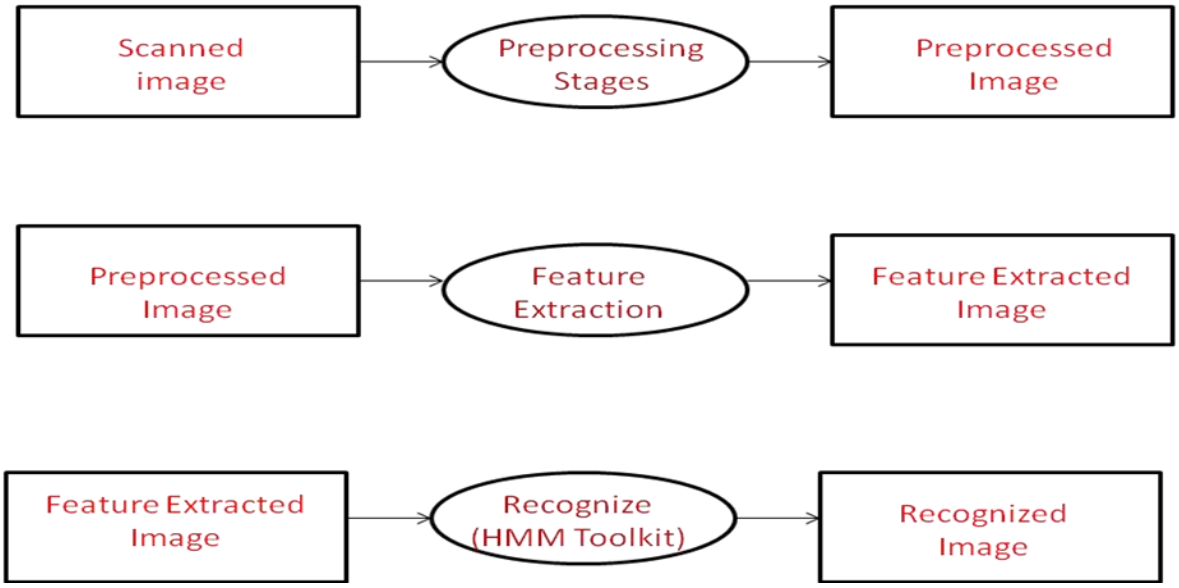


Figure 3.3: Context Diagram

Data Flow Diagram

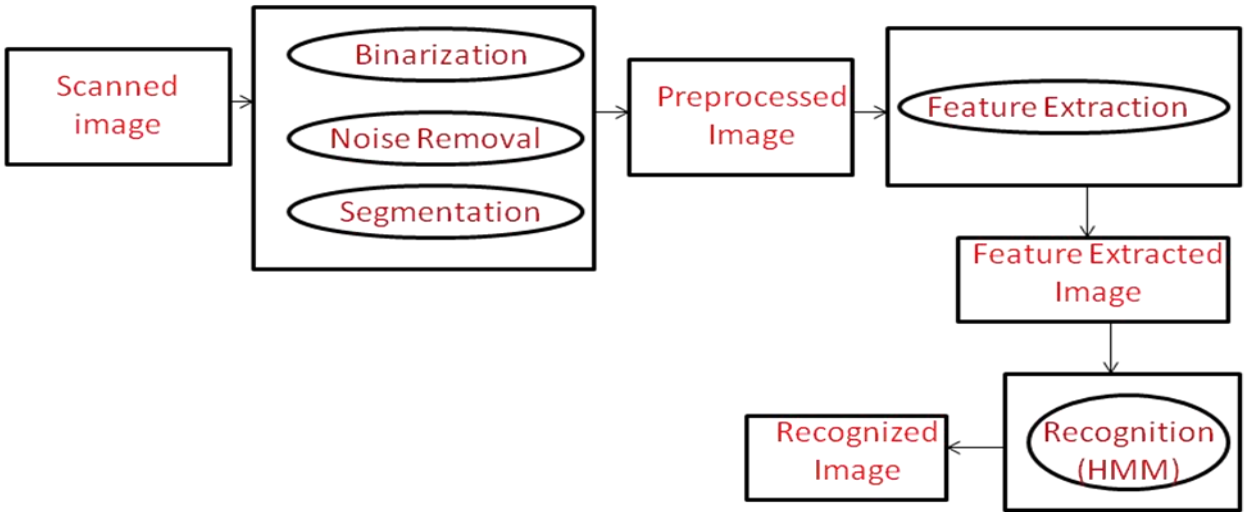


Figure 3.4: Data Flow Diagram

OBJECTIVE OF PROJECT:

The whole world now wants accuracy with speed. *'Image sensing and processing'* is one of those techniques to achieve this goal. The input for this technique is an image and the output will be a processed image or information related to that input image. QR scanning, Image cryptography, security issues are some of the applications.

4.1 Objective:

The input for this application wills be an image of a word. The relative translation of that word will be retrieved from the database and displayed.

4.2 Procedure:

Firstly, this application will capture the image which has some text in it .it scans it and processes it. The text from the image is to be retrieved. From the database, the translations of that text in the image are to be retrieved and displayed as output.

CHAPTER 5
System Specifications

1 Software Requirements

- Front End :Mat Lab 7.10(R2012a)
- Back End :Hidden Markov Model Toolkit (HTK)
- Operating System :Windows XP Professional(Service Pack 2)
- Eclipse software with TTS package

5.2 Hardware Requirements

- Hard Disk :10 GB and above
- RAM :512 MB and above
- Processor :Pentium 4 and above

CHAPTER 6 SYSTEM IMPLEMENTATION

The scanned image is preprocessed, i.e., the image is **checked for skew correction**, then the image is **binarized**, then **unwanted noise is removed** and finally the **characters are segmented.**

6.1 Preprocessing Steps

The preprocessing of the character involves the following steps.

1 Binarization

2 Noise Removal

3 Skew Correction

4 Segmentation

6.1.1 Binarization

Image binarization converts an image (up to 256 gray levels) to a black and white image (0 or 1). The simplest way to use image binarization is to choose a threshold value, and classify all pixels with values above this threshold as white, and all other pixels as black.

6.1.1.1 Modified Otsu Global Algorithm

This algorithm is the **combination of Otsu Global algorithm and Sauvola algorithm**

This method is both simple and effective. The algorithm assumes that the image to be threshold contains two classes of pixels (e.g. foreground and background) and calculates the optimum threshold separating those two classes so that their combined spread (intra-class variation) is minimal. As in Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

- 4.1

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

Weights ω*i* are the probabilities of the two classes separated by a threshold *t* and variances of these classes. Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance

$$\sigma_b^2(t) = \sigma^2 - \sigma_{\omega b}^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$$

- 4.2

$$\sigma_b^2(t) = \sigma^2 - \sigma_{\omega b}^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$$

Which is expressed in terms of class probabilities ω*i* and class means μ*i* which in turn can be updated iteratively.

6.1.1.2 Algorithm 1

Input: Scanned Document Image

Output: Binarized Image

Step 1: Compute histogram and probabilities of each intensity level Step 2: Set up initial and

Step 3: Step through all possible thresholds t=1….maximum intensity

1 Update and

2 Compute

Step 4: Desired threshold corresponds to the maximum (t)

In the case of the **bad quality image global thresholding cannot work well**. For this, we would like to apply a technique. Sauvola binarization technique (window-based), which calculates a local threshold for each image pixel at (x, y) by using the intensity of pixels within a small window W (x, y). Here we have taken the window of size 19x19 pixels with (x, y) as centre except at the edge pixels of the image frame. So, we start

Computation from x =10, y =10. The **Threshold T (x, y)** is computed using the following formula

$$T(x,y) = Int[X.(1 + k/(R-1))]$$

$$T(x,y) = Int[X.(1 + k/(R - 1))]T(x,y) = Int[X.(1 + k/(R - 1))]$$

- 4.3

Where **X** is the mean of gray values in the considered window **W (x, y**), is the standard deviation of the gray levels and **R** is the dynamic range of the variance, **k** is a constant (usually 0.5 but may be in the range 0 to 1).

6.1.2 Noise Removal

6.1.2.1 Morphological Image Cleaning Algorithm(MIC)

MIC is useful for grayscale images corrupted by dense, low-amplitude, random or patterned noise.

6.1.2.2 Algorithm 2

Input: Noise Added Image

Output: Noise Removed Image

Step 1: Consider a noisy grayscale image I.

Step 2: Let **S** be the image with smoothing I which has openings and closings. Step 3: Assume **S** is noise free.

Step 4: Then the difference image, D contains all the noise in I.

D=I-S - 4.4

Step 5: Again the residual image (D) is added to the S smoothed image, so that the resulting image will be as sharp as the original image with smoothened regions between them.

By means of this algorithm, the image will be **free from noise, dense, amplitude**.

CHAPTER 7 DEVELOPMENT PROGRAM:

7.1 SEGMENTATION:

As a first step, the image is cropped to fit the text. After that, line by line is separated. The function performed by the image cropping shown below:

```
function imgn = clip (image)
% Crops to black letter with white background.
% Example:
% = Imread image ('metal.bmp');
% Imgn = clip (image);
% Subplot (2,1,1); imshow (image); title ('INPUT IMAGE')
% Subplot (2,1,2); imshow (~ imgn); title ('OUTPUT IMAGE')
if ~ ISLOGICAL (image)
image = im2bw (image, 0.99); end
a = ~ image; [Fc] = find (a);
lmaxc = max (c); lminc = min (c);
lmaxf = max (f); lminf = min (f);
imgn = a (lminf: lmaxf, lminc: lmaxc); % Crops image
```

Example:

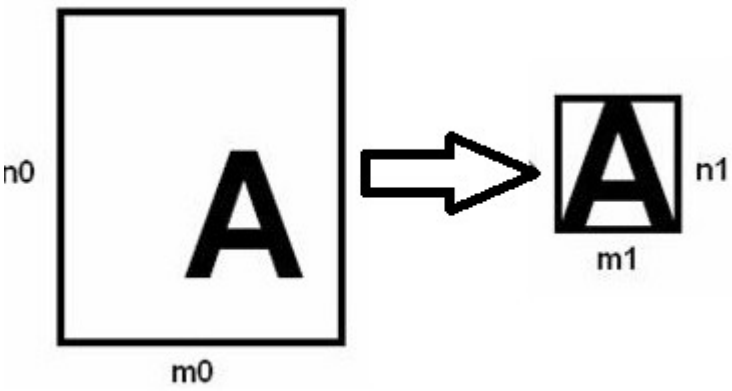


FIG 8.1

As shown in the function, the threshold for binary image transformation is 0.99 (= bn im2bw (image, 0.99)). This threshold was taken to colors with close to 255 (maximum value) RGB values are regarded as 0 in the binary image.

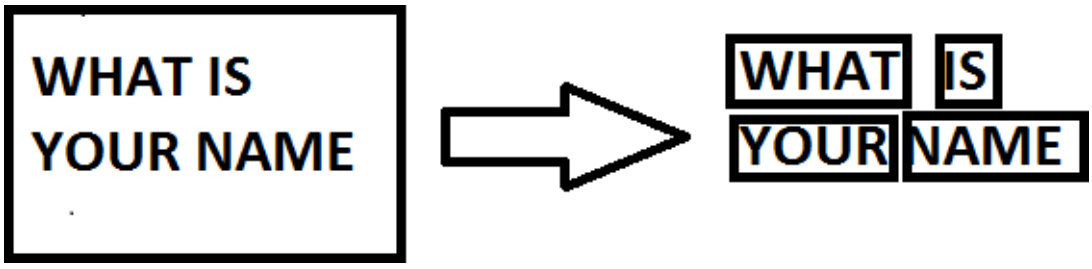


FIG 8.2

Once trimmed image, the next step is to separate each line. For this, the following function is used:

```
function [re fl] = lines (aa)
% Divide text in lines.
% AA-> input image; fl-> first line; re-> REMAIN line
% Example:
% Pa = imread ('heavy_metal.bmp');
% [Re fl] = lines (aa);
% Subplot (3,1,1); imshow (aa); title ('INPUT IMAGE')
% Subplot (3,1,2); imshow (fl); title ('FIRST LINE')
% Subplot (3.1.3); imshow (re); title ('REMAIN LINES')
aa = clip (aa);
```

```
r = size (aa, 1); for s = 1: r
if sum (aa (s,:)) == 0
nm = aa (1: s-1.1: end); % First line matrix
rm = aa (s: end, 1: end); % Remain line matrix
~ fl = clip (~ nm); re = ~ clip (~ rm);
% * - * - * Uncomment lines below to see the result * - * - * - * -
% Subplot (2,1,1); imshow (fl);
% Subplot (2,1,2); imshow (re); break
else
fl = ~ aa; % Only one line. re = [];
```

```
   end
end
```

Once obtained separately each line of the image, it is necessary to remove one letter from the image matrix fl. For this function the bwlabel used, which label connected components of the image. In other words, this function has the solid lines and lists. To separate each letter used the following code:

```
% * - * - * - * - * - Calculating connected components * - * - * - * - * -
% Code from:
% Http: //www.mathworks.com/matlabcentral/fileexchange/ loadFile.do?objectId=8031&objectType=FILE
L = bwlabel (imgn); x = max (max (L));
BW = edge (double (imgn), 'sobel' ); [IMX, imy] = size (BW);
rc = [rc];
[Sx s] = size (rc);
n1 = zeros (IMX, imy); for i = 1: sx
x1 = rc (i, 1);
y1 = rc (i, 2);
n1 (x1, y1) = 255;
for n = 1: x
[R, c] = find (L == n);
end
% * - * - * - * - * - END Calculating connected components * - * - * - * - * -
```

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

Then each letter is normalized to a size of 42 x 24 pixels, which is the size of the template that will perform the correlation. For normalization function is used as follows:

```
function img_r = same_dim (imagen_g)
```

```
% Example:
% Imagen_g = imread ('a_reducir.bmp');
% Img_r = same_dim (imagen_g);
% Subplot (2,1,1); imshow (imagen_g); title ('Image mx n')
% Subplot (2,1,2); imshow (img_r); title ('Image 42 x 24')


img_r = imresize (imagen_g, [42 24]);
```

7.2 CLASSIFICATION

The main operation that was used to classify the two-dimensional correlation. This operation gives a value of the similarity between two matrices (images). The function corr2 develops this operation the following equation [4]:

The following function performs the correlation between each extracted templates and letter:

```
function read_letter letter = (PICTURE)
---------------------------------------------------------------
% Computes the correlation Between template and input image
% And Its output is a string container containing the letter.
% Size of 'PICTURE' must be 42 x 24 pixels
% Example:
% Imagn = imread ('D.bmp');
% = Read_letter letter (PICTURE) comp = [];


load templates
for n = 1: 36
```

```
sem = corr2 (templates {1, n}, PICTURE); comp = [comp sem];
end
vd = find (comp == max (comp));
% * - * - * - * - * - * - * - * - * - * - * - * - * - * - if 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 = 'A' ; 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' ; elseif vd == 26
letter = 'Z' ;
% * - * - * - * - * elseif 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' ;
else
   letter = '0' ;
end
```

7.3 TEMPLATES

Each template is a binary image bmp of 42 x 24 pixels. The script to store templates in a cell structure is as follows:

```
% TEMPLATES CREATE
% Letter
A = imread ( 'A.bmp' ); B = imread ( 'B.bmp' );
C = imread ( 'C.bmp' ); D = imread ( 'D.bmp' );
E = imread ( 'E.bmp' ); F = imread ( 'F.bmp' );
G = imread ( 'G.bmp' ); H = imread ( 'H.bmp' );
I = imread ( 'I.bmp' ); J = imread ( 'J.bmp' );
K = imread ( 'K.bmp' ); L = imread ( 'L.bmp' );
M = imread ( 'M.bmp' ); N = imread ( 'N.bmp' );
O = imread ( 'O.bmp' ); P = imread ( 'P.bmp' );
U = imread ( 'U.bmp' ); V = imread ( 'V.bmp' );
W = imread ( 'W.bmp' ); X = imread ( 'X.bmp' );
Y = imread ( 'Y.bmp' ); Z = imread ( 'Z.bmp' );
% Number
one = imread ( '1.bmp' ); two = imread ( '2.bmp' ); three = imread ( '3.bmp' ); four = imread ( '4.bmp' ); five
```

```
= imread ( '5.bmp' ); six = imread ( '6.bmp' ); seven = imread ( '7.bmp' ); eight = imread ( '8.bmp' ); nine = imread ( '9.bmp' ); zero = imread ( '0.bmp' );
% * - * - * - * - * - * - * - * - * - * - * - * - letter = [ABCDEFGHIJKLM ...
NOPQRSTUVWXYZ];
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' )
```

CONCLUSION :

The font size should not be less than 42 x 24 pixels.

- The input image can be colored letters or not.
- If the image has more noise at the edges which can filter medfilt2 function imcrop be necessary to use the function to crop the image.
- If the font size is small, use the function imdilate to increase its thickness before moving the program.
- To result in a text to lowercase, the function of the variable lower word is used in the main program.
- The processing time of the test text of Figure 6 was: >> tic; OCR; toc Elapsed time is 3.806446 seconds. However, on a PC with 1GB of RAM processing time it was 2.4 seconds.

The processing time is reduced if a staff of 20 x 20 pixels is used

- There is still a long way to HAL, the brilliant talking computer of '2001, a space odyssey'. A number of advances in the area of NLP or DSP, however, have recently boosted up the quality and naturalness of available voices, and this is likely to continue. Important issues need now be further addressed in that purpose.

CHAPTER 11

Test Examples:

Example 1:

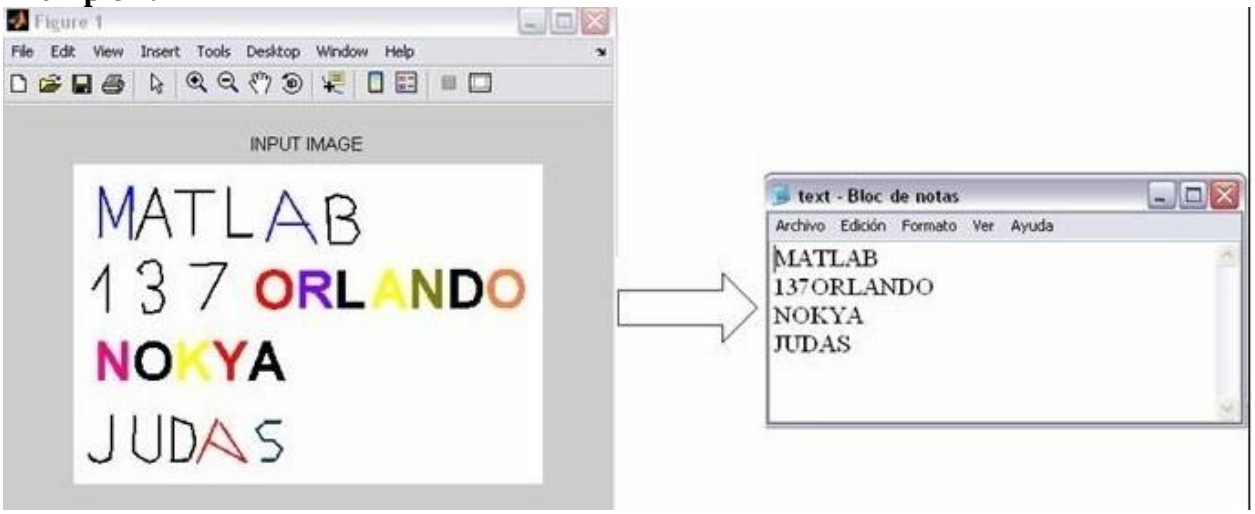
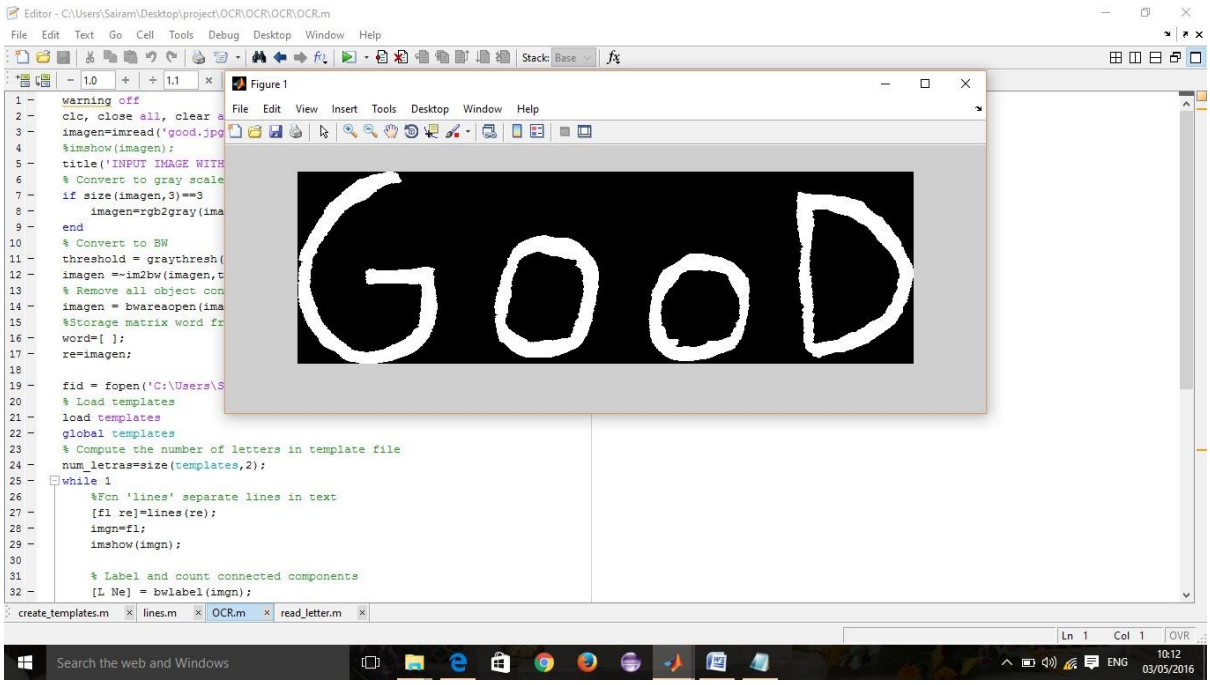


FIG 11.1

Example 2:

FIG 11.2

EXAMPLE 3



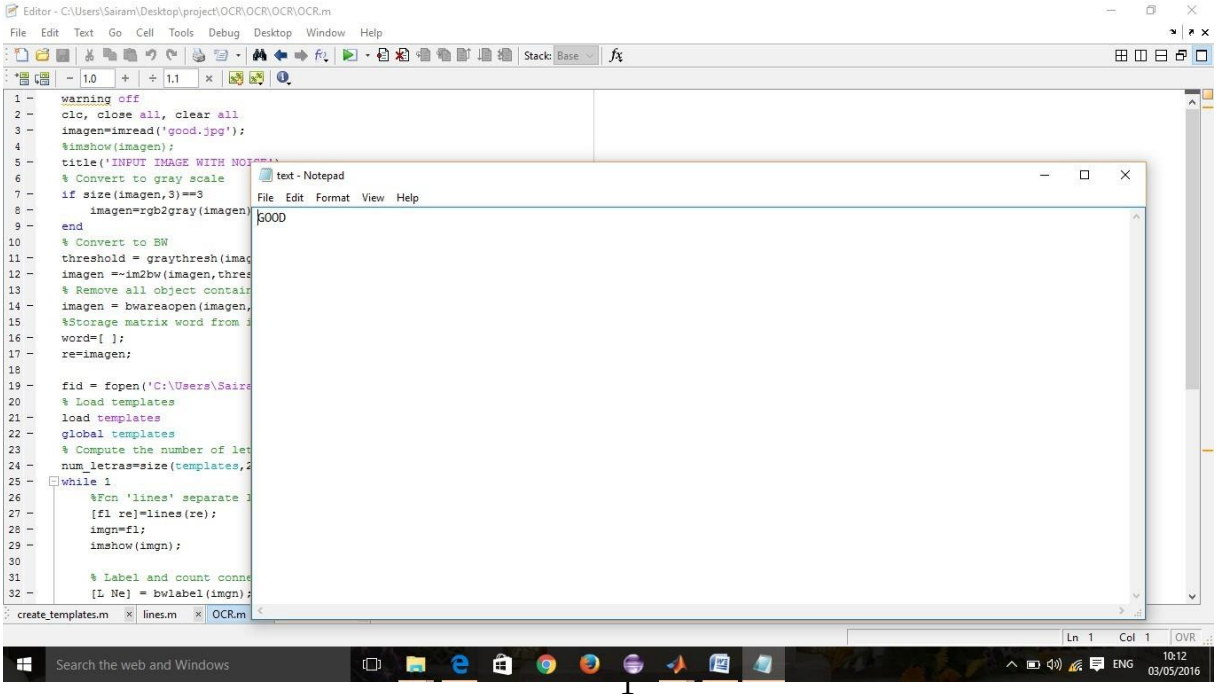
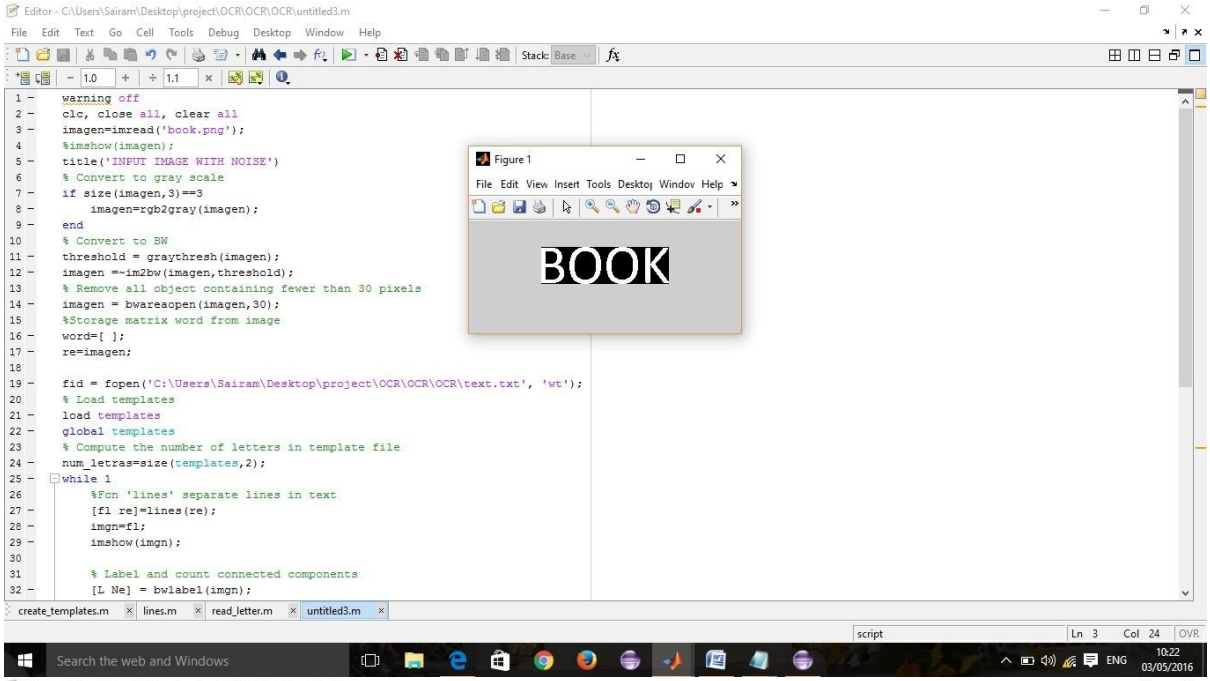
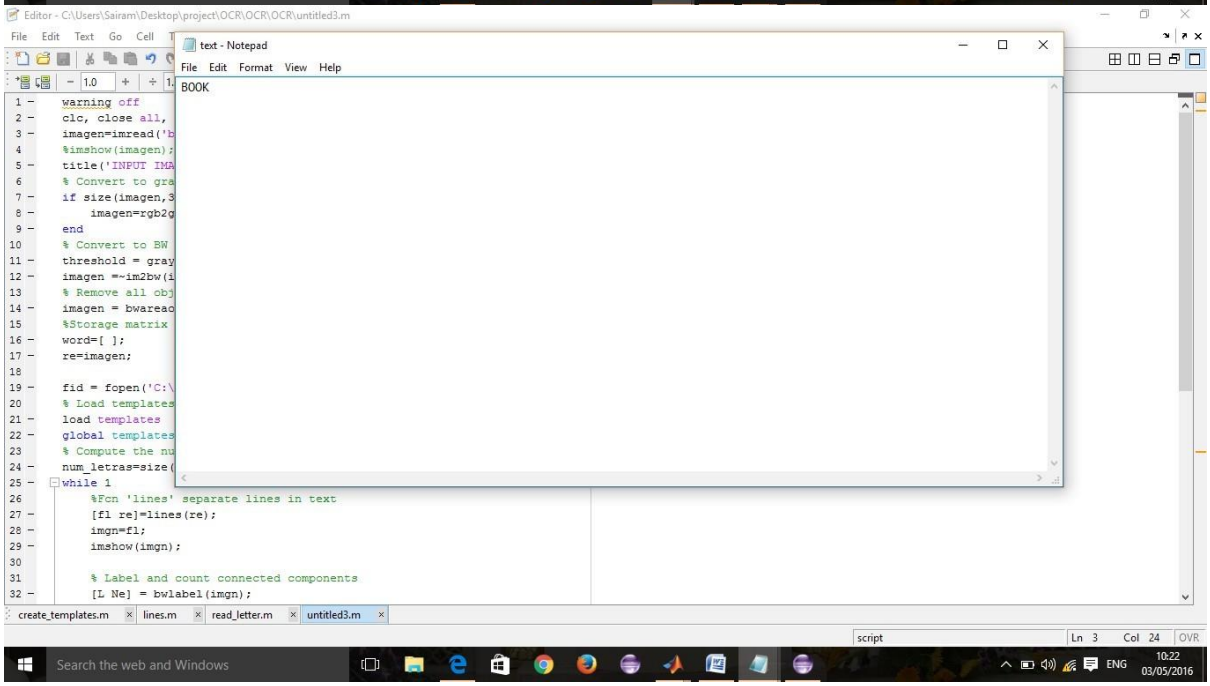


FIG 11.3

EXAMPLE 4





REFERENCES

- Bunke .H, Roth .M, and Schukat-Talamazzini .E.G :- OfflineCursive Handwriting Recognition using Hidden Markov Models. Pattern Recognition, 28(9):1399–1413, **1995.**
- Chaudhury .S, Sheth .R, "Trainable script identification strategies for Indian languages", In Proc. 5th Int. Conf. on Document Analysis and Recognition (IEEE Comput. Soc. Press), pp. 657–660, **1999.**
- Colin Higgens, and Dave Elliman, :- Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach- Somaya Alma'adeed, Colin Higgens, and Dave Elliman Proceedings of the 16th International Conference on Pattern Recognition (ICPR'02) © **2002** IEEE.
- Coulmas .F :- The Blackwell Encyclopedia of Writing Systems. Blackwell, Oxford, **1996.**
- Dan Povey et al :-Discriminative Training for HMM-Based Offline Handwritten Character Recognition - -Proceedings of the 7th
- Dehghani et al .A :- Off-line Recognition of Isolated Persian Handwritten Characters Using Multiple Hidden Markov Models

-Proceedings of the International Conference on Information Technology: Coding and Computing © **2001** IEEE

- Hewavitharana et al .S, :- Off-line Sinhala Handwriting Recognition using Hidden Markov Models - Third Indian Conference On Computer Vision, Graphics, © Image Processing – **2002**
- Ishwarya M.V, Jagadeesh Kannan .R, "An Improved Online Character Recognition Using Neural Networks," ace, pp.284-288,

2010 International Conference on Advances in Computer Engineering, © **2010.**

- Jaeger .S, Manke .S, Reichert .J and Waibel :- A Online Handwriting Recognition: NPen++ Recognizer. International Journal on Document Analysis and Recognition, 169–180, © **2001.**
- A.J. ABRANTES, J.S. MARQUES, I.M. TRANSCOSO,"Hybrid Sinusoïdal Modeling of Speech without Voicing Decision", EUROSPEECH 91, pp. 231-234.
- T. DUTOIT, H. LEICH, "MBR-PSOLA : Text-To-Speech Synthesis based on an MBE Re-Synthesis of the Segments Database"