OPTICAL CHARACTER RECOGINATION(OCR) AND GESTURE RECOGINATION

Submitted by- DEEPANSH ARORA (7102257)

DEEPAK SHARMA (7102198)

VISHNU TEJA.Y (7102289)

Name of supervisor- Mr.Abhishek srivastava

Submitted in partial fulfilment of the Degree of

Bachelor of Technology

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY UNIVERSITY, NOIDA

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work titled "**OPTICAL CHARACTER RECOGINATION AND GESTURE RECOGINATION"** submitted by "**DEEPANSHARORA (7102257), DEEPAK (7102198) and VISHNU TEJA (7102289)**" in partial fulfilment for the award of degree of bachelor of technology of Jaypee Institute of Information Technology University, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor:

Name of Supervisor: **Mr. Abhishek Srivastva**

Designation: **Senior lecturer**

Date:

ACKNOWLEDGEMENT

We wish to express our sincere gratitude to M**r. Abhishek srivastva**, senior lecturer, Department of ECE, Jaypee Institute of Information Technology University, Noida for providing us an opportunity to do this project work on "**OPTICAL CHARACTER RECOGINATION AND GESTURE RECOGINATION**" .This project bears an imprint of many research works and Journals. We sincerely thank him as a mentor who was always there for guidance and encouragement in carrying out this project work.

We also wish to express our deep gratitude to the officials and other staff members of ECE Department who rendered their help by providing a good environment throughout the period of this project work.

.

Yours faithfully:

DEEPANSH ARORA (7102257) Batch-A4

DEEPAK SHARMA (7102198) Batch-A2

VISHNU TEJA .Y (7102289) Batch-A6

CHAPTER 1

SUMMARY

In our Project, Optical character recognition (OCR) technology will offer blind and visually impaired persons the capacity to scan printed text and then speak it back in synthetic speech or save it to a hardware which has features of Recording, Rewinding etc.

There are three essential elements to OCR technology—scanning, recognition, and reading text. Initially, a printed document is scanned by a camera. OCR software then converts the images into recognized characters and words. The synthesizer in the OCR system then speaks the recognized text. Finally, the information is stored in an electronic form, either in a personal computer (PC) or the memory of the OCR system (which have IC APR9600) itself.

All OCR systems create temporary files containing the texts' characters and page layout. In our OCR's these temporary files will be converted into formats retrievable by commonly used computer software such as word processors and spreadsheet and database software. The blind or visually impaired user can access the scanned text by using adaptive technology beased on image enhancement and provide speech output.

Speech synthesizers and text-to-speech programs will convert any words stored in a computer to speech. However, much of the information people need is in books. To fully use the capability of computers to convert text to speech, we developed efficient ways of transferring text from books to computers.

Special cameras and pattern recognition programs have been used for recognizing specially designed letters and numbers, such as the account numbers on checks. The camera converts the pattern of each letter into a binary code. A computer is programmed to process the binary code and determine which letter it represents.

In the last few years, devices and programs have been developed which make it possible for computers to recognize most typewritten characters and to adjust automatically for different type styles and sizes. In the next few years, this technology is likely to be perfected and become more widely available. (Only very limited success can be expected with handwritten letters, due to the large variations found in even one person's handwriting.)

1.1 Converting Print to Speech

This machine combines sophisticated pattern recognition, speech synthesis, and text-to-speech conversion capabilities. It lets blind users control how the material is read. They can set the speed of reading and adjust the tonality of the voice. They can stop the reading at any time, have the last few words or lines repeated, request the machine to spell out words or announce punctuation and capitalization, and mark certain words or phrases for later reference.

One of the big problems for blind people always has been the very small number of books, magazines and other printed materials accessible to them. Few books are translated into Braille, the system of raised dots read with the fingers, or issued as audio recordings.

1.2 Gesture Recognition

Parameterized gestures open-up an additional input channel in human-computer interaction. FINGER can act as a tool for the specification of gestures. An interpreter recognizes parallel finger-movements according to the gesture definition .Gestures are a natural and intuitive form of human communication, which blind can use. With Gestures we want to present a language that formally describes the movements of fingers, and specifies the resultant operation, thereby formalizing the interaction. Fingers movement followed by mouse pointer movement is implemented in MATLAB.

1.3 Assumptions

Due to challenges faced referred in chapter 13 we have taken following assumptions while implementing OCR algorithm

First, the **font size** of the text in the captured image should be sufficiently large, otherwise it may be ignored.

Secondly, the **background** should be uniform or at least near uniform, and it should constitute a major part of the whole image. Portion outside of the text boundary should not be larger than the actual background, otherwise it may be treated as the background instead, and the entire entity within and including the boundary will be wrongly regarded as an object.

CHAPTER 2

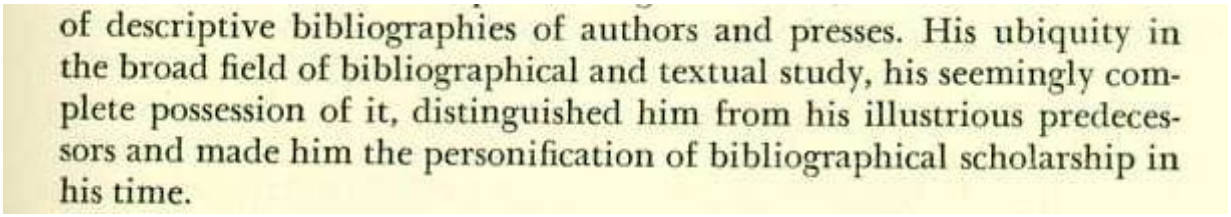
INTRODUCTION TO OCR AND GESTURE RECOGNITION

Our project is divided into 3 phases-

1. Optical character recognition
2. Gesture recognition
3. Integration of above 2 to develop a sound application

First we define optical character recognition (OCR) whose goal is to classify optical patterns (often contained in a digital image) corresponding to alphanumeric or other characters. The process of OCR involves several steps including segmentation, feature extraction, and classification. Each of these steps is a field unto itself, and is implemented in MATLAB. OCR is mechanical or electronic translation of images of handwritten, typewritten or printed text (captured by a scanner or webcam) into machine-editable text. We will use text extraction to convert paper books and documents into electronic files, for instance, to computerize an old record-keeping system in an office or to help blind man to dictate particular page he want to read.

One example of OCR is shown below. A portion of a scanned image of text, borrowed from the web, is shown along with the corresponding (human recognized) characters from that text


of descriptive bibliographies of authors and presses. His ubiquity in the broad field of bibliographical and textual study, his seemingly complete possession of it, distinguished him from his illustrious predecessors and made him the personification of bibliographical scholarship in his time.

of descriptive bibliographies of authors and presses. His ubiquity in the broad field of bibliographical and textual study, his seemingly complete possession of it, distinguished him from his illustrious predecessors and made him the personification of bibliographical scholarship in his time.

Figure 1: Text extracted from scanned image

A few examples of OCR applications are listed here. The most common for use OCR is the first item; people often wish to convert text documents to some sort of digital representation.

1. People wish to scan in a document and have the text of that document available in a word processor.
2. Recognizing license plate numbers
3. Post Office needs to recognize zip-codes

Second, we define Gesture recognition whose goal is to classify optical patterns of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Current focuses in the field include emotion recognition from the face and hand gesture recognition. Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse.

Gesture recognition enables humans to interface with the machine (HMI) and interact naturally without any mechanical devices. Using the concept of gesture recognition, it is possible to point a finger at the computer screen so that the cursor will move accordingly. This could potentially make conventional input devices such as mouse, keyboards and even touch-screens redundant.

We have implemented this mouse movement using hand motion which integrating with OCR will result in sound application described below.

A few examples of gesture recognition are listed here

1. Facial feature recognition (airport security) – Is this person a bad-guy?
2. Speech recognition – Translate acoustic waveforms into text.
3. A Submarine wishes to classify underwater sounds – A whale? A Russian sub? , A friendly ship?

OCR converts a graphical image of a document to word converts graphical image of a document to a word processor file without typing the characters.In other words,you can change printed text into an editable, searchable document. Prior to OCR technology human was forced to type all the information available on the paper in printed format for applications such as mailing, correction etc.

2.1 The Classification Process

(Classification in general for any type of classifier) There are two steps in building a classifier-training and testing.

These steps can be broken down further into sub-steps.

1. **Training**

    a. Pre-processing – Processes the data so it is in a suitable form.
    b. Feature extraction – Reduce the amount of data by extracting *relevant* information—usually results in a vector of scalar values.
    (We also need to NORMALIZE the features for distance measurements)
    c. Model Estimation – from the finite set of feature vectors, need to estimate a model (usually statistical) for each class of the training data

2. **Testing**

    a. Pre-processing
    b. Feature extraction – (both same as above)
    c. Classification – Compare feature vectors to the various models and find the closest match. One can use a distance measure.

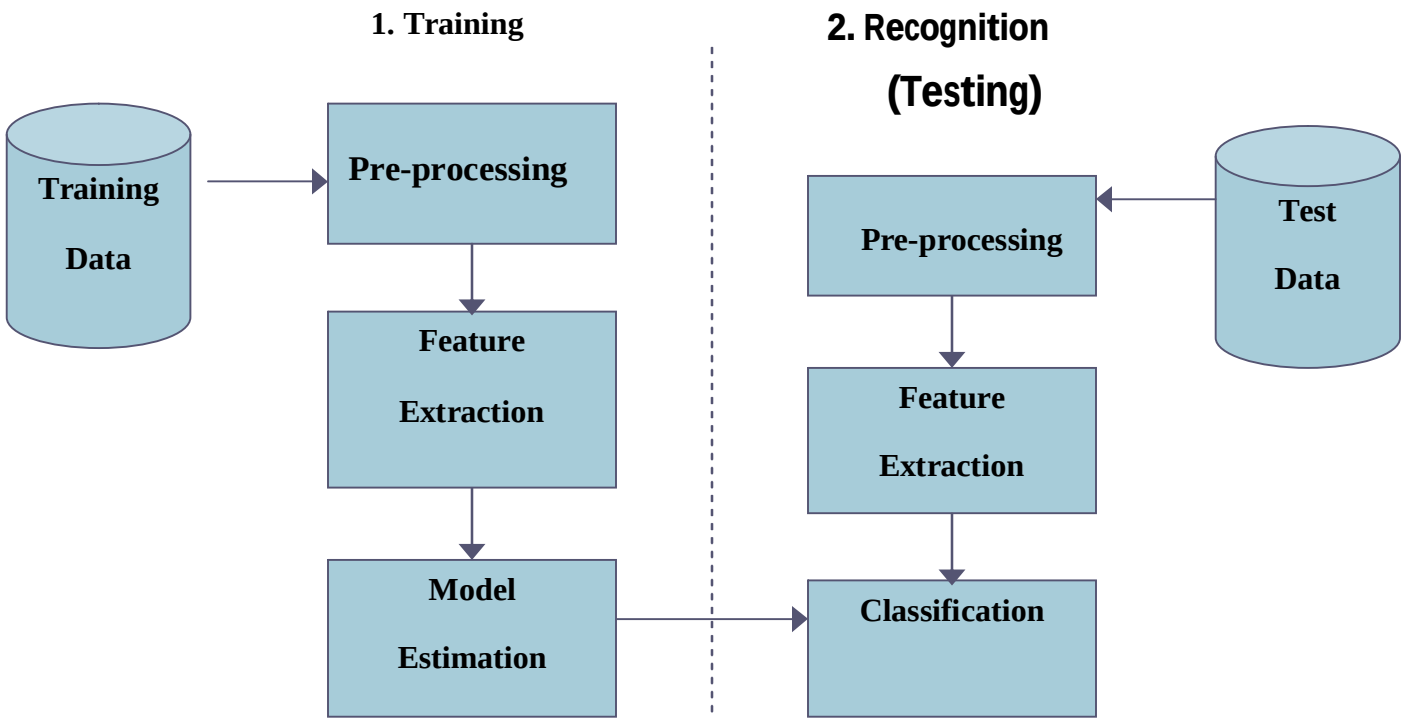


Figure 2: The pattern classification process

CHAPTER 3

FLOW OF THE OCR SYSTEM

The flow of our OCR system is shown in figure (3), each step in which is introduced in the following subsections.

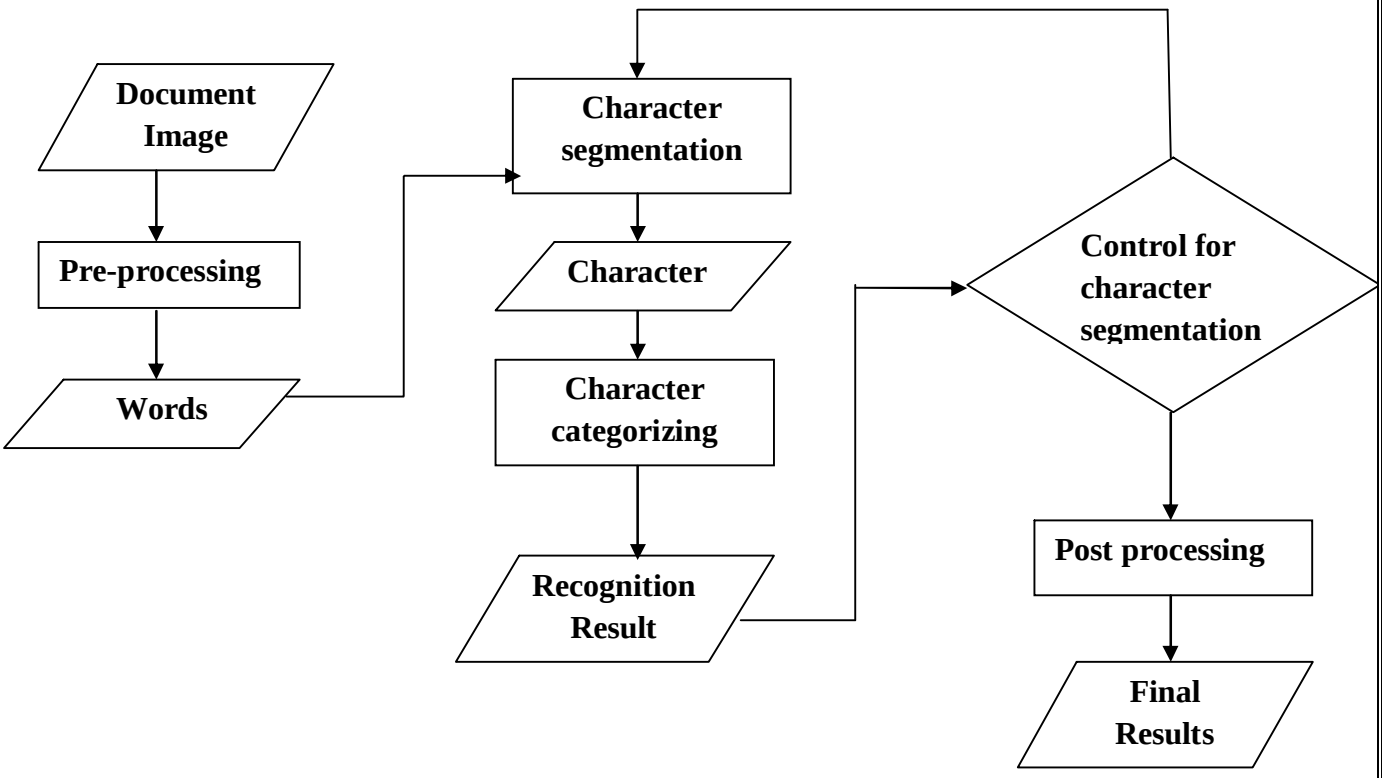


Fig.3-Design flow of OCR system

2.1 Pre-processing

The pre-processing consists of the following functions basically:

1. Image reading.

2. Image processing for better OCR results, including binarization, deskew, auto-orientation,

language recognition and so on.

3. Layout analysis: The image is divided into text regions, picture regions and other regions (e.g. table regions, mathematical regions, etc.).

4. Line segmentation: Each text region is divided into several text lines.

5. Word segmentation: Each text line is divided into several words.

2.2 Character Extraction

The character segmentation-recognition consists of following steps:

1. Character segmentation: Each word image is segmented into several single character images.

2. Character recognition: The different pattern matching algorithm either correlation or artificial neural network is used to for the recognition of characters.

3. The control for character segmentation: The quality of the segmentation result is evaluated according to the spelling check and the confidence of character recognition. The character segmentation recognition will repeat until the quality is good enough in sense that no 2 or more letter combined to form single character or the maximum number of the iterations is reached.

2.3. Post-processing

The functions of the post-processing are mainly that:

1. Restoring the layout according to the processing results.

2. Exporting the processing results with the user-specified format.

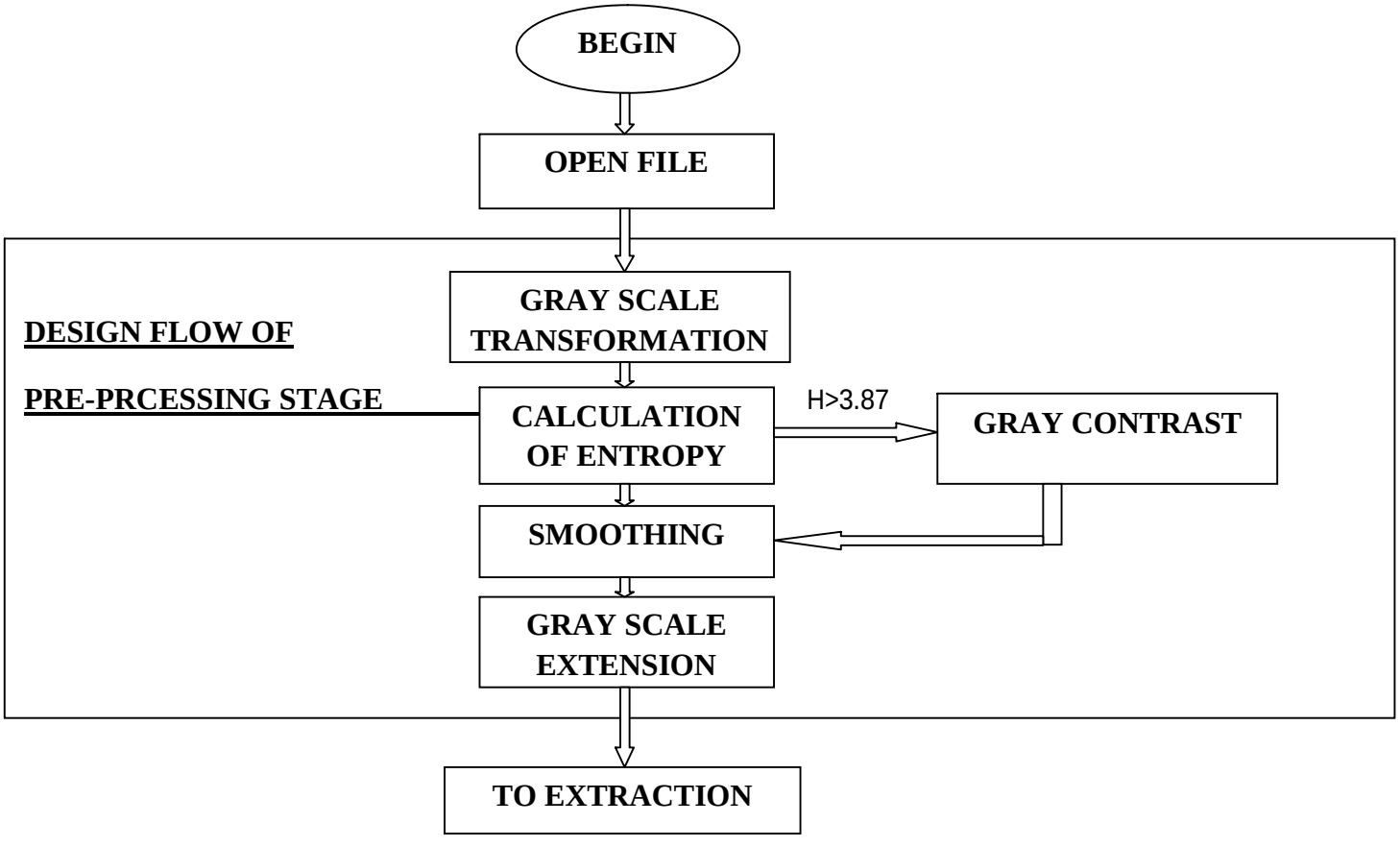
CHAPTER 4

PRE-PROCESSING FOR THE OCR SYSTEM

In the processing stage, an intelligent and adaptive algorithm will yield the essential information from the raw image. The process for enhancing images of scanned documents identifies a variety of items in the scanned document which could make optical character recognition and other document image processing difficult or impossible. These items include identifying skew, registration, speck, lines intersecting printed matter, reverse printing, shaded printed matter.

In the processing stage, the original colour image will first be transformed to an adequate grey image, which is sufficient for the text extraction purpose. Next, the sharp transition portions of the image (the edges) are detected using a revised Prewitt edge detection algorithm (Prewitt, 1970), which is specifically developed to meet the specific requirements of this application. The algorithm assumes the edges will be black. After the edges are detected and closed-shapes formed, the image will be segmented into several white regions. The zone with the largest area will be regarded as the background, and the difference of the colour value of the regions in the original image from the value associated with the background will be used to determine whether it is an object, or it is part of the background. Following these steps, signs and noise may still exist in the processed image. Decision-making and elimination algorithm will be employed to remove these remaining unwanted parts. This completes the processing stage and at this time, only the desired characters will be left in the image, and they are ready to be sent to an OCR engine for interpretation.

We will describe the pre-processing stage, which contains four steps in order to transform the original colour image to a grey level image, and increase its contrast when necessary. The key part of the processing stage includes edge detection and region segmentation so as to segregate the region of interest (RoI) from the rest of the image.



Pre-Processing stage is the first stage of our OCR project. Image enhancement techniques are implemented here for row camera image so that it can be handed over to the second stage of text extraction.

The camera images are coupled with an uncontrolled environment, uneven illumination and obvious reflection in the image captured, and a possibly odd image capturing angle, the target text captured can be blurred, all of these posing difficulties to text extraction. For this purpose we had implemented below steps for proper enhancement of image

Figure 4: Input Image

3.1 Colour to greyscale transformation

The transformation function to transform a colour image to the grey level image is given in equation below. Let f(x, y) be the colour value of a pixel original image at the (x, y) position. (We will refer to f as the original colour image). Then f(x, y).R, f(x, y).G and f(x, y).B, denotes the corresponding value of its red (R), green (G) and blue (B) components respectively.

T(x, y) represents the greyscale value of that pixel of the transformed image.

$$T(x,y) = 0.114 \times f(x,y).R + 0.587 \times f(x,y).G + 0.299 \times f(x,y).B$$

fig (a) sample image for processing stage

fig (b) Image after grayscale transformation

fig (c) Effects of smoothing

fig (d) Effects of grayscale extension

Figure 5:Grey Scale Extension

3.2 Entropy computation for contrast enhancement

Entropy, in the context of thermodynamics, is a measure of the disorder or randomness in a closed system. When applied to image processing, entropy represents the amount of information or the level of monotone of an image. Entropy can be computed as follows:

$$H = -\sum \leq p_i \times \log p_i = \sum \leq p_i \times \log\left(\frac{1}{p_i}\right)$$

H denotes the entropy of an image, pi represents the proportion of greyscale values in the range i∈ [0.255] over the entire image. Due to the limited computational ability of cameras and scanner, when an image is captured under poor lighting condition, the image will appear dark and monotonous compared to another which is taken under good lighting condition.

Therefore, entropy can be used as an indication if an increase in the contrast of the image will be necessary. If the entropy calculated of the image is too low, the contrast can be increased, otherwise the detected edges of characters may not form closed shapes. From extensive experiments, an empirical value of Hthres = 38 is recommended to identify images, which needs to be enhanced in terms of the contrast. A computationally efficient way to enhance contrast can be achieved through an exponential transformation:

$$C(x,y) = \frac{255}{1+\exp\left(\frac{aver\_T - T(x,y)}{\upsilon}\right)}$$

C(x, y) denotes the transformed results. aver_T is the average grey values in the image represented by T. The parameter υ can be fixed at υ = 15. Essentially, through this transformation, the range of grey values will be expanded to a wider range. Otherwise, if the entropy is less than Hthres, C(x, y) = T(x, y).

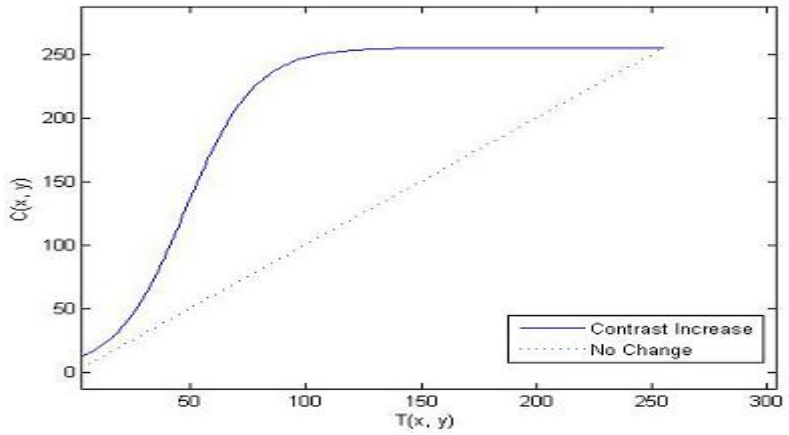


Figure 6: Matlab output to show contrast enhancement

Figure 7: Contrast Enhancement

3.3 Smoothing

Smoothing is also sometimes termed as blurring. The aim of a smoothing function is to smooth the image prior to edge detection. The edge detection process aims to highlight the sharp transition parts in an image. Therefore, a simple smoothing algorithm may help to minimise the effects of these disturbances on the edge detection process. Smoothing is mainly achieved based on a mask which dictates the weightage of the surrounding pixels in influencing the pixel of concern. A possible mask is shown below:

1	1	1
1	2	1
1	1	1

Fig5: Mask for smoothing

Let C(x, y) and S(x, y) denote the greyscale value at position (x, y) of the image before and after the transformation.

$$S(x,y) = \frac{1}{10}\left(C(x,y) \otimes M\right)$$

Figure 8: Smoothing og Image

3.4 Grayscale extension

Histogram analysis is an important tool in image processing. After the colour image has been transformed to the greyscale equivalent and conditioned to image S according to above eqn . the maximum max_S and minimum min_S greyscale values can be computed. If max_S and min_S are close to each other, e.g., max_S – min_S < 80, the greyscale image will be monotonous with a low contrast. Greyscale extension will increase contrast via exploiting the full range max_S – min_S = 255. If the original max_S and min_S are already 255 and 0 respectively, then the image will remain unchanged following the transformation.

$$min\_S = \min\left\{S(x,y): 1 \le \times M \wedge 1 \le y \le N\right\} \text{ and}$$
$$max\_S = \max\left\{S(x,y): 1 \le \times M \wedge 1 \le y \le N\right\}$$

Let

$$\alpha = -min\_S \text{ and } \beta = \frac{255}{max\_S - min\_S}.$$

Then the transformation equation is given by:

$$G(x,y) = \left(S(x,y) + \alpha\right) \times \beta$$

Figure 9: GrayScale Extension

3.5 Region segmentation

The main objective behind region segmentation is to segregate the grey image into several regions, so as to be able to separate the background from the objects. It involves four key steps which will be explained in the ensuing subsections.

3.5.1 Edge detection

The key issue is to ensure that the detected edges form a closed shape. Otherwise, two components will be labelled as one through the following step. This problem will compound when it comes to the subsequent filling process. In this section, we will present a new detection algorithm based on the revision of an existing one, to guarantee a closed shape for such images.

The eight kernels to be used in this algorithm are listed in Figure 6. In the literature, kernels can also be referred to as operators, detectors or masks.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

First, through the convolution operations, D(x, y) can be obtained by the following convolution computation:

$$D(x,y) = \max\left(F_i \otimes G(x,y)\right)$$

D(x, y) denotes the results following edge detection at (x, y), and Fi is the ith kernel listed above.D(x, y) is determined by the maximum value of (Fi $\otimes$ G(x, y). In the proposed algorithm, we will use a hard-limiter to differentiate an edge from the background and connected parts. The following decision making process is proposed for this purpose:

1. After computing D(x, y) over the whole image, calculate the average edge value:

$$aver\_E = \frac{\sum_{x=1}^{M}\sum_{y=1}^{N} D(x,y)}{M \times N}.$$

3. Compare D(x, y) to $\gamma \times$ aver_E, if D(x, y) > $\gamma \times$ aver_E, then the pixel at (x, y) will be turned to black, which means we regard this point as part of the edge. Otherwise, it will be kept as white. Through experiments, an empirical value $\gamma = 2.4$ is recommended.

Figure 10: Edge Detection

3.5.2 Fake edge reduction

Through edge detection, sharp transition parts are turned to black and the rest will remain as white. However, many fake edges may exist in the image due to the non-uniform texture in the original image A reduction function is needed to remove these fake edges. Let aver_D(x, y) denotes the average value of the eight-neighbours of the pixel of the image D located at (x, y), and D(x, y) denotes the value of the pixel. Then, the following operations can be done to realise the reduction function, D denotes the output.

$$\text{if } \left|aver\_D(x,y) - D(x,y)\right| > 127.5 \; \tilde{D}(x,y) = 255 - D(x,y)$$

$$\text{Otherwise} \qquad \tilde{D}(x,y) = D(x,y).$$

Figure 11: Fake Edge Detection

3.5.3 Labelling

Following edge detection, labelling is a usual method to identify the connected components. When labelling is carried out adequately, each of the identified characters is associated with a sequential number. Thus, through this method, we can label the connected white zones in the image. The white background may be marked as 0 for convenience. Meanwhile, we should note that some characters such as ‘P’, ‘A’, ‘R’, ‘D’ have two connected white areas since they have a ‘hole’ as part of the characters.

3.5.4 Sub-zone filling

Following the previous steps, the RoI are now available. The next step would be to fill the character so as to derive black characters on a white background. This is the crucial step in the whole algorithm, and it can be accomplished by a fusion of colour and edge information.

The filling step essentially involves a binary decision-making process, based on a comparison between the average colour values of each sub-zone in the original colour image to the average colour value of the background. The largest sub-zone is regarded as the background. Thus, it is kept as white. Meanwhile, we trace the corresponding location of the background in the original

image [i.e., the green portion of Figure 13(a)], and compute the average colour value of the background. Let averRbg, averGbg and averBbg denotes the average value of red, green and blue.

Figure 12: Sub Zone Filling

Pseudo code for sub-zone filling algo can be summed up as:

BEGIN

1. compute averRbg, averGgb and averBbg
2. i = 1
3. compute averRi, averGi and averBi
4. compute

$$dis[i] = \frac{\left|averR_i - avrR_{bg}\right|}{averR_{bg}} + \frac{\left|averG_i - avrG_{bg}\right|}{averG_{bg}} + \frac{\left|averB_i - averB_{bg}\right|}{averB_{bg}}$$

5. i + +
6. compute max_dis and min_dis
7. compute thres = 0.45 * (max_dix – min_dis)
8. i = 1
9. if dis[i] > thres draw the ith sub-zone black
10. i + +

END

3.5.6 Dilation and erosion for noise removal and to make letters connected

We have first change to change the gray scale image into a binary image, in which each pixel is restricted to a value of either 0 or 1.In erosion, every object pixel that is touching a background pixel is changed into a background pixel. In dilation, every background pixel that is touching an object pixel is changed into an object pixel. Erosion makes the objects smaller, and can break a single object into multiple objects. Dilation makes the objects larger, and can merge multiple objects into one. The algorithm for determining if the neighbors are connected or unconnected is based on counting the black-to-white transitions between adjacent neighboring pixels, in a *clockwise* direction.

Figure 13: Dilation and Erosion

CHAPTER 5

CHARACTER EXTRACTION

In this part, the characters will be identified as letters and the image will sent for recognition and further processing. After the image is cleaned up becomes a binary image which contains only the text, the binary image is then saved and the memory is cleaned up. This step is very important to increase the speed of system. After that the following steps should be done.

5.1 Removal the borders

The borders have been removed; this will reduce the image size. Only the rectangular part of the image which contains the text will remain. As shown in figure, the gray region will be removed which will make the image size smaller and consequently make the program faster.

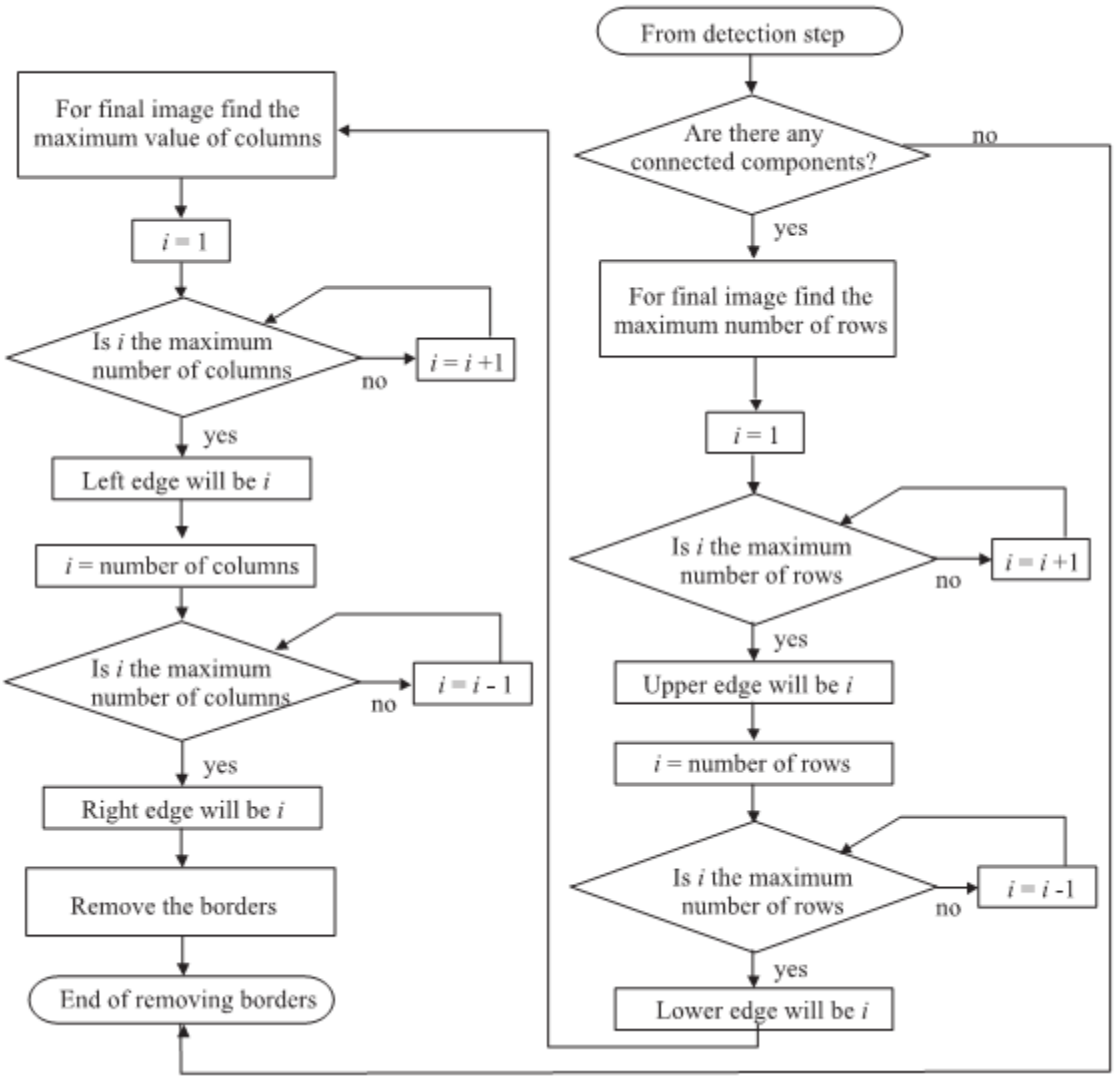


Figure 14: Flowchart for Removal of Border

5.2 Divide the text into rows

After moving the border, the area has been divided into rows. Each row is saved in the next step figure.

Both above steps were done in a single nested loop for image. We have scanned the image vertically and horizontally in a single nested loop in this way we increases the processing speed of application and all the starting ending coordinates of row were stored in multidimensional array system we will describe it letter.

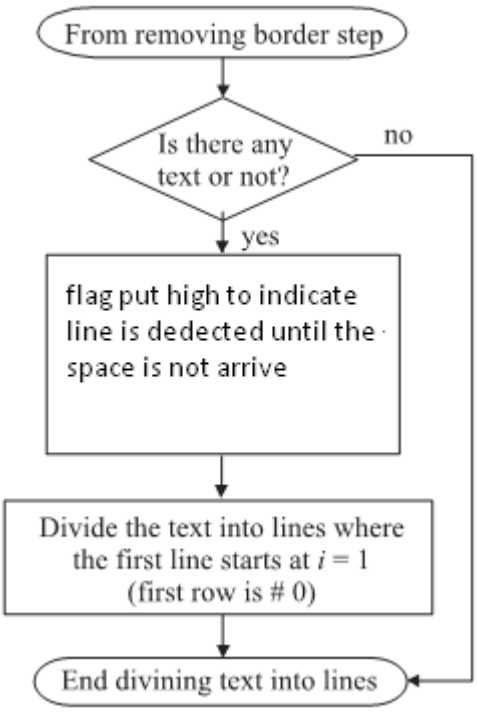


Figure 15: Flowchart for dividing Text in to Lines

5.3 Divide the rows (lines) into words

The single line is then divided into words. Before that**,** the empty area before and after the text are removed. Word and letters are distinguished behave of the spaces between the two letters. Again all the information of coordinates of word is stored in array again.

Flowchart for dividing rows into words is drawn below

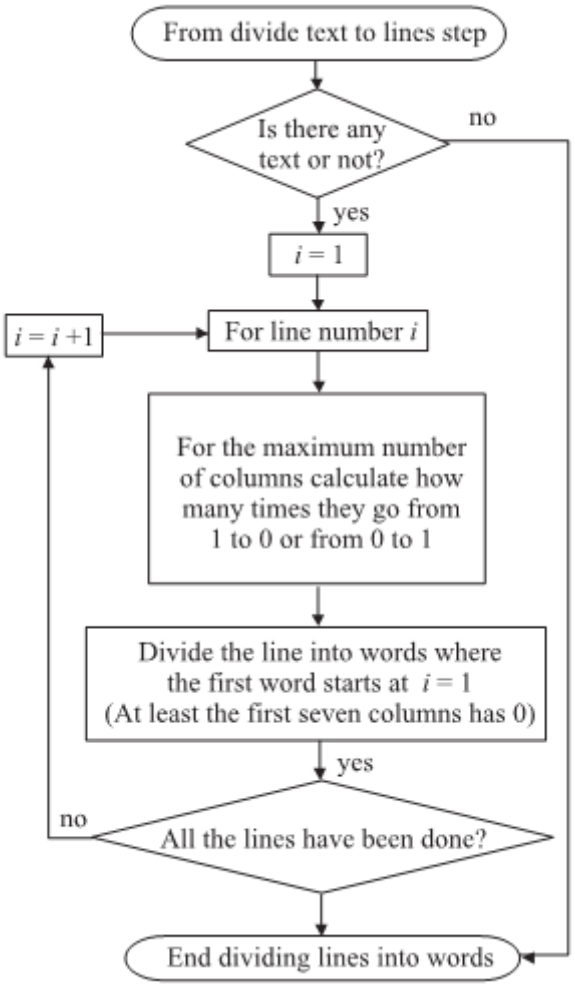


Figure 16: Flowchart for dividing line into words

5.4 Divide the word into letters

Each word is then divided into characters and saved in an array. This array is represents an image of latter. This image is again resized as we know that a text material can have different font size letters. So before we move this letter in reorganization system we resize each letter.

Flowchart for dividing words into characters is drawn below

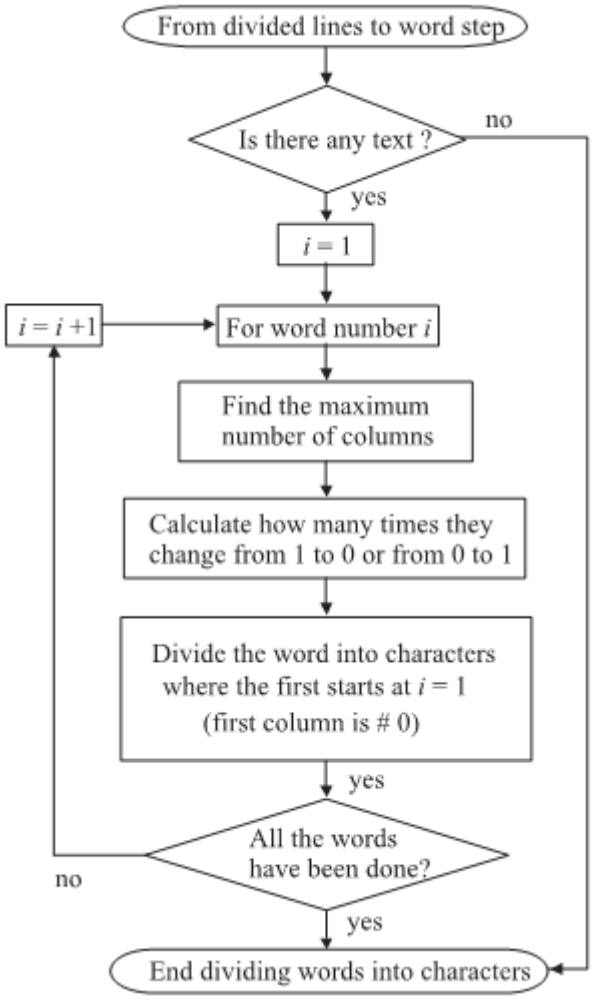


Figure 17: Flowchart for dividing word into Characters

As per coding aspect above both step again implemented in a single nested loop, here we excess each row(lines) from previous records, then each row is vertically scanned we distinguish the letters and words in each row and the process is repeated for all the rows. After finding the horizontal boundary of letters we recheck again for the vertical boundary as we know letters can be uppercase or lowercase in same line so the vertical height of different letters can vary in same line so we introduce concept of chopping in extracted letters.

CHAPTER 6

PATTERN RECOGNITION IN OCR SYSTEM

It is often useful to have a machine perform pattern recognition which is important in sense to apply to extracted images in extraction stages. In particular, machines that can read symbols are very cost effective. A machine that reads banking checks can process many more checks than a human being in the same time. This kind of application saves time and money, and eliminates the requirement that a human perform such a repetitive task

6.1 Pattern recognition using correlation method

We have applied correlation technique for Text recognition in our project. It finds the correlation of extracted character with that of every possible letter in database. Thus, finds the letter in database to which extracted letter correlation is maximum and prints that letter to text file which is further converted to speech using Window Api application in MATAB.

6.1.1Description

Correlation quantifies the strength of a linear relationship between two variables. When there is no correlation between the two quantities, then there is no tendency for the values of one quantity to increase or decrease with the values of the second quantity.

The main result of a correlation is called the **correlation coefficient** (or "r"). It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related. If r is close to 0, it means there is no relationship between the variables. If r is positive, it means that as one variable gets larger the other gets larger. If r is negative it means that as one gets larger, the other gets smaller (often called an "inverse" correlation).

In MATLAB r = corr2 (A, B) computes the correlation coefficient between A and B, where A and B are matrices or vectors of the same size.

Corr2 computes the correlation coefficient using

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

Where = mean2 (A), and = mean2 (B).

6.2 Character Recognition using neural network

A network is to be designed and trained to recognize the 26 letters of the alphabet. An imaging system that digitizes each letter centered in the system's field of vision is available. The result is that each letter is represented as a 5 by 7 grid of Boolean values. However, the imaging system is not perfect, and the letters can suffer from noise.

A network can be designed and trained to recognize the 26 letters of the alphabet. An imaging system that digitizes each letter centred in the system's field of vision is available. The result is that each letter is represented as a 5 by 7 grid of Boolean values. However, the imaging system is not perfect, and the letters can suffer from noise.

Perfect classification of ideal input vectors is required and reasonably accurate classification of noisy vectors. The twenty-six 35-element input vectors are defined as a matrix of input vectors called alphabet. The target vectors are also defined in this file with variable called targets. Each target vector is a 26-element vector with a 1 in the position of the letter it represents, and 0's everywhere else. For example, the letter A is to be represented by a 1 in the first element (as A is the first letter of the alphabet), and 0's in elements two through twenty-six.

6.2.1 Neural Network

The network receives the 35 Boolean values as a 35-element input vector. It is then required to identify the letter by responding with a 26-element output vector. The 26 elements of the output vector each represent a letter. To operate correctly, the network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0.

In addition, the network should be able to handle noise. In practice, the network does not receive a perfect Boolean vector as input. Specifically, the network should make as few mistakes as possible when classifying vectors with noise of mean 0 and standard deviation of 0.2 or less.

6.2.2 Architecture

The neural network needs 35 inputs and 26 neurons in its output layer to identify the letters. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values.

The hidden (first) layer has 25 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer. If the network solves the problem well, but a smaller more efficient network is desired, fewer neurons could be tried.

The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input vectors can result in the network's not creating perfect 1's and 0's. After the network is trained the output is passed through the competitive transfer function compet. This makes sure that the output corresponding to the letter

most like the noisy input vector takes on a value of 1, and all others have a value of 0. The result of this post processing is the output that is actually used.

6.2.3 Training

To create a network that can handle noisy input vectors, it is best to train the network on both ideal and noisy vectors. To do this, the network is first trained on ideal vectors until it has a low sum squared error. Training is very crucial since in this weights are assigned for different inputs.

Then the network is trained on 10 sets of ideal and noisy vectors. The network is trained on two copies of the noise-free alphabet at the same time as it is trained on noisy vectors. The two copies of the noise-free alphabet are used to maintain the network's ability to classify ideal input vectors. All training is done using back propagation with both adaptive learning rate and momentum.

CHAPTER 7

GESTURE RECOGNITION

A primary goal of gesture recognition research is to create a system which can identify specific human gestures and use them to convey information or for device control. We also explore how humans use gestures to communicate with and command other people. Speech and handwriting recognition research provide methods for designing recognition systems and useful measures for classifying such systems. Gesture recognition systems which are used to control memory and display, devices in a local environment, and devices in a remote environment are examined for the same reason.

In our Project we have recognize human gesture on basis of

1. Edge detection of human body

2. Following the centroid of an object

The prototype of hand gesture recognition using Edge detection will provide information about edges in such a way that can recognize one's gesture. These recognized gestures can be used for recognizing defined signals like thumb up, down etc. from blind man and perform desired task accordingly as assigned. A wave of the hand, for instance, might terminate the program recognizing gestures as input allows computers to be more accessible for the physically-impaired and makes interaction more natural in a 3-D virtual world environment.

In addition to the technical challenges of implementing gesture recognition, there are also social challenges. Gestures must be simple, intuitive and universally acceptable.

For calculation of centroid image is traversed by a line parallel to x and y axis. For a given color intensity collected in a group, maximum and minimum coordinates are stored. Taking average of maximum and minimum gives the mid-point of both(parallel to x and y axis) set of parallel lines. Further average is taken of these mid points to get the mid point of the object.

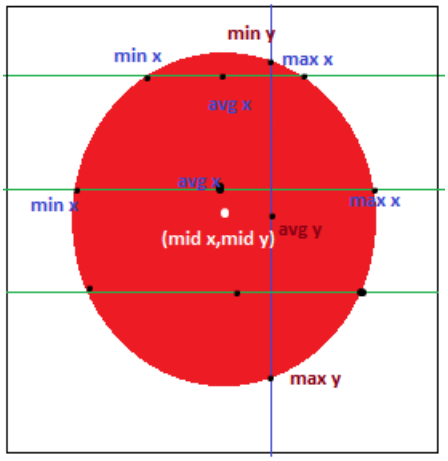


Figure 18:Centroid Calculation

Avg x=(min x +max x)/2

Mid x=(avg x_1 +avg x_2 +…+avg x_n)/n

Avg y=(min y +max y)/2

Mid y=(avg y_1 +avg y_2 +…+avg y_n)/n

After aquiring the centre cordinates of the object,a java inbuilt class has to be included in matlab to interface with the mouse pointer.

CHAPTER 8

TEXT TO SPEECH CONVERSION

Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech synthesizer, and can be implemented in software or hardware. 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 diphones 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.

The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood. An intelligible text-to-speech program allows people with visual impairments or reading disabilities to listen to written works on a home computer. Many computer operating systems have included speech synthesizers since the early 1980s.

Text-to-Speech (TTS) capabilities for a computer refers to the ability to play back text in a spoken voice. **TTS** is the ability of the operating system to play back printed text as spoken words.An internal (installed with the operating system) driver (called a TTS engine): recognizes the text and using a synthesized voice (chosen from several pre-generated voices) speaks the written text. Additional engines (often use a certain jargon or vocabulary) are also available through third-party manufacturers.

In course of our project we found two alternative ways to convert text to speech. The basic work has been started in .NET.An interface has been made beween Matlab and .Net, in this the text is extracted from the images in matlab which is further preprocessed to get better information. All the text extracted is stored in notepad which could be further processed to get voice as output. The response time was relatively high, So inorder to obtain better results all the work has been shifted in to matlab.An interface was created between windows Text-to-Voice engine and our program to make the task easier . It is the ability of the operating system to play back printed text as spoken words. An internal driver, called a Text to Voice engine, recognizes the text and using a synthesized voice chosen from several pre-generated voices, speaks the written text. It is installed with the operating system. Additional engines are also available through third-party manufacturers. These engines often use a certain jargon or vocabulary; for example, a vocabulary specializing in medical or legal terminology. They can also use different voices allowing for regional accents.

Modern Windows systems use SAPI4- and SAPI5-based speech systems that include a speech recognition engine (SRE). SAPI 4.0 was available on Microsoft-based operating systems as a third-party add-on for systems like Windows 95 and Windows 98. Windows 2000 added a speech synthesis program called Narrator, directly available to users. All Windows-compatible programs could make use of speech synthesis features, available through menus once installed on

the system. Microsoft Speech Server is a complete package for voice synthesis and recognition, for commercial applications such as call centers.

8.1 Overview of text processing

A text-to-speech system (or "engine") is composed of two parts: 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.

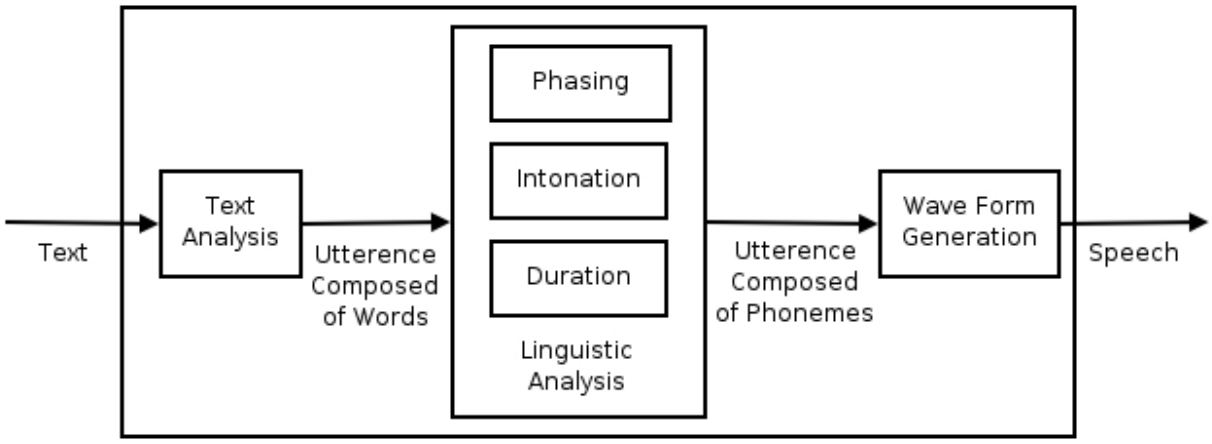


Figure 19: High level design of Text to Speech Conversion

CHAPTER 9

HARDWARE DESIGN

Hardware module has been included in the project to make the speech task simpler and easier to understand.Apr9600 module has been selected as it has playback and record time of 60 sec.This module can be interfaced with microcontroller which inturn is connected in computer through serial communication.The microcontroller controls the operations such as record and playback according the requirement. The data that is recorded can also be processed further,this could also provide user interface and convenience.To control the data flow in Apr9600 the ADC port of Atmega16 is configured as Output.To make the message ports active '0' should be sent through ADC ports ,if the data overflow can be identified with a busy indication .

We can work in two modes

1.Random Access Mode

2.Tape Mode

9.1 Random Access Mode

In this mode recording or playback can be made randomly in any of the selected messages. The length of each message segment is the total recording length available (as defined by the selected sampling rate) divided by the total number of segments enabled.

9.1.1 Functional Description of Recording in Random Access Mode

On power up, the device is ready to record or play back, in any of the enabled message segments. To record, /CE must be set low to enable the device and /RE must be set low to enable recording. Recording can be initiated by applying a low level on the message trigger pin that represents the message segment we intend to use.

9.1.2 Functional Description of Playback in Random Access Mode

To playback, /CE must be set low to enable the device and /RE must be set high to disable recording & enable playback.Playback can be initiated by applying a high to low edge on the message trigger pin that representing the message segment you intend to playback. Playback will continue until the end of the message is reached. If a high to low edge occurs on the same message trigger pin during playback, playback of the current message stops immediately.

9.2 Tape Mode

Tape mode manages messages sequentially much like traditional cassette tape recorders. Within tape mode two options exist, auto rewind and normal. Auto rewind mode configures the device to automatically rewind to the beginning of the message immediately following recording or playback of the message. In tape mode, using either option, messages must be recorded or played back sequentially, much like a traditional cassette tape recorder.

9.2.1 Function Description Recording in Tape Mode

On power up, the device is ready to record or play back, starting at the first address in the memory array. To record, /CE must be set low to enable the device and /RE must be set low to enable recording. A falling edge of the /M1_Message pin initiates voice recording (indicated by one beep). A subsequent rising edge of the /M1_Message pin during recording stops the recording (also indicated by one beep). If the /M1_Message pin is held low beyond the end of the available memory.

9.2.2 Function Description of Playback in Tape Mode

On power-up, the device is ready to record or play back, starting at the first address in the memory array. Before we begin playback, the /CE input must be set to low to enable the device and /RE must be set to high to disable recording and enable playback. The first high to low going pulse of the /M1_Message pin initiates playback from the beginning of the current message; on power up the first message is the current message. When the /M1_Message pin pulses low the second time, playback of the current message stops immediately.

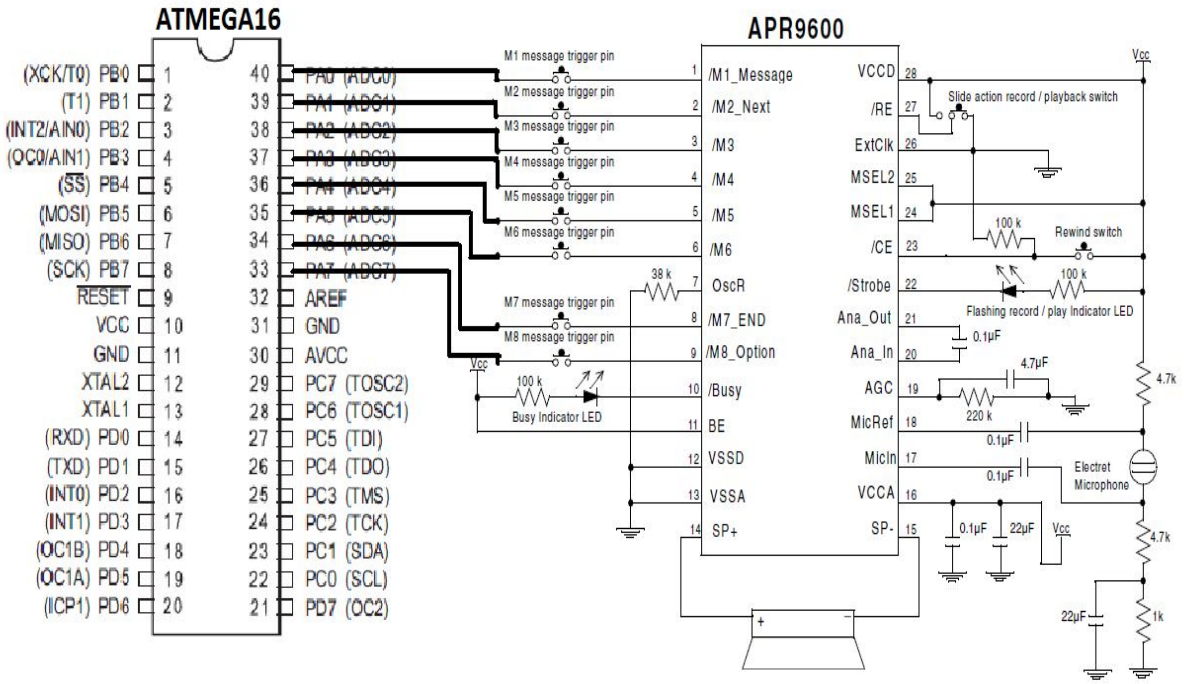


Figure 20: Interfacing of APR9600 with ATMEGA16

PIN NO	PIN NAME	**Normal Option**	**Auto Rewind Option**
1,2,3,4 5,6	Message1,2, 3,4,5,6	Message: A low edge on this pin plays or records the next message.	Message: A low edge on this pin plays or records the current message.
8	M7_END	During playback a low level on this pin indicates that all recorded messages have been played. During recording a low level on this pin indicates that the end of the memory array was reached.	During playback a low level on this pin indicates that all recorded messages have been played. During recording a low level on this pin indicates that the end of the memory array was reached.
9	M8_Option	Option: This pin in conjunction with MSEL1 and MSEL2 sets record and playback operating mode. Consult table 1 for decoding information.	MSEL1 and MSEL2 sets record and playback operating mode. .

CHAPTER 10

INTEGRATION

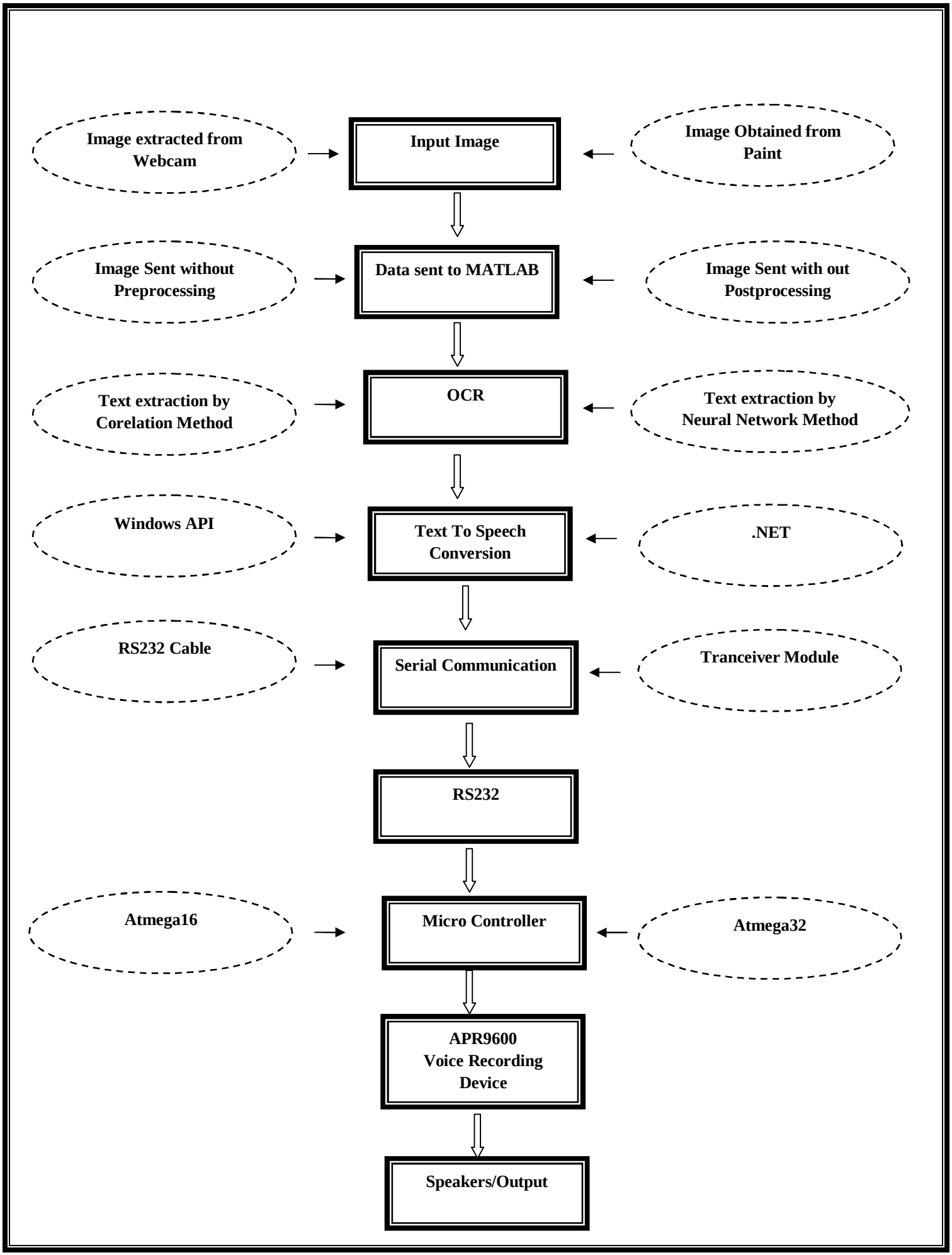


Image extracted from Webcam
Input Image
Image Obtained from Paint
Image Sent without Preprocessing
Data sent to MATLAB
Image Sent with out Postprocessing
Text extraction by Corelation Method
OCR
Text extraction by Neural Network Method
Windows API
Text To Speech Conversion
.NET
RS232 Cable
Serial Communication
Tranceiver Module
RS232
Atmega16
Micro Controller
Atmega32
APR9600 Voice Recording Device
Speakers/Output

CHAPTER 11

CHALLENGES FACED AND SOLUTIONS SUGGESTED

1. Bubbles may exist inside the extracted characters due to scratches, reflection or nails on the picture taken by camera. In these situations, a bubble filling algorithm can be used to fill the bubbles. Therefore, the filling algorithm can be used based on both labelling and area calculation. However, the labelling algorithm will pose additional computational burden with no considerable effect on recognition. Thus, this step has not been adopted, in our project, in order to reduce the processing time.

2. There can be large variation in both the font size and font type of the text expected in the diverse forms of images captured. Therefore the threshold box for segmentation cannot be fixed at a specific size.

3. Resolution of such images will be typically modest. Coupled with an uncontrolled environment, uneven illumination and reflection and a possibly odd image capturing angle, the target text captured can be blurred, all of these posing difficulties to text extraction.

4. The text extraction and recognition function will inevitably be limited by the nature of the small-screen mobile devices, which will restrict the span of the image which can be captured. Finally, images taken under a poor lighting condition may result in low entropy. Low entropy may also cause problems in processing. Far more intensive computation will be necessary when the text is embedded in a complex background

5. Calculating the histogram, and then smooth it to find the threshold points. These methods are not working well when the text and background colours are very similar, and also if it may create an area of negative text. Therefore histogram technique is not implemented rather mask defined in pre-processing stage is implemented.

6. **No support for handwritten text** OCR has severe limitations when come to handwritten text. Characters must be handwritten with separate characters in different box.

7.We found better solution to existing solutions when connected components are considered .If length and width ratio of extracted character is greater to that of critical ratio then image can be divided in to partial images, the way of dividing connected characters and treating them as partial images has increased the accuracy and time consumption has been decreased.

8.Implementation of Erosion and Skeletonisation technique together for the identification of characters has solved the problem of fake connections of two different characters to some extent.

CHAPTER 12

FUTURE SCOPE

OCR is one of the most emerging technology and its reliability is continually improving. Soon OCR will become a powerful tool for data entry applications which will lead to automated data entry by OCR,thus reducing labor.Incorporating OCR will be an attractive feature of any Data Entry System.However in past due to limited availability of a capital and short environment was restricting the growth of this technology,but today more and more enterprises are working on this technology and that will definitely lead to 100% accuracy in this technology thus making the dream of paperless world true.

Our work has been defined to only fewer fonts at present,In future we would extract the text from hand written images and also from all the other fonts.Templates were being designed to make the task easier.TTS accuracy can also be increased to such extent that it could become user interactive.

Following OCR algorithm can be used for development of a human-machine interactive software application, specifically useful for text extraction from images, which are captured using mobile and digital devices with cameras which can be used to translate the text in such a captured image to another language.

1. The approach is also viable as an alternative way to send SMS from images captured and to extract URL text from a complex background and directly link the user to the website via his mobile device.

2. Implementation of **spell checking** technique can greatly enhance the working of OCR system by increasing its recognition accuracy.

3. Increasing the efficiency or enhancing the box called creating zone will also increase the efficiency of software.

4. Colour detection will also be useful to help user to reduce job of formatting in any sort of text.

Applications of OCR

- Avoids Re-Typing of a Page
- Format a paper that was Faxed.
- Enter an article into bibliographic database.
- Can make the softcopies for all the documents easily.
- Preservation of old manuscript.
- Computer meter reading.

REFERENCES

1) Text Detection and Character Recognition using Fuzzy image processing by Mohanad Alata — Mohammad Al-Shabi, Journal of ELECTRICAL ENGINEERING, VOL. 57, NO. 5, 2006, 258–267.

2) YUAN, Q.—TAN, C. : Text Extraction from Gray Scale Document Images Using Edge Information, Proc. Sixth Int. Conf. on Document Analysis and Recognition, 302–306.

3) WOLF,C.—JOLION,J.—CHASSAING,F. : Text Localization, Enhancement and Binarization in Multimedia Documents,In Proceedings of the International Conference on Pattern Recognition (ICPR). IEEE Computer Society, Canada: Quebec City. 2002, 4, 1037–1040.

4) Canny, J. (1995) 'A computational approach to edge detection', *IEEE Trans. on Pattern Recognition*, Vol. 17, No. 12.

5) Gonzalez, R.C. and Woods, R.E. (2001) *Digital Image Processing*, 2nd ed., Chapter 6, Chapter 10,Prentice-Hall, Englewood Cliffs, NJ.

6) Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,23(8):800{810, 2001.

7) Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *Proc. of First International Workshop on Multimedia Intelligent Storage and Retrieval Management*, 1999.

8) Arth, C., Limberger, F. and Bischof, H., 2007. Real-time license plate recognition on an embedded DSP-platform. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR '07) pp. 1–8..

9) Wolf, C., michel Jolion, J. and Chassaing, F., 2002. Text localization,enhancement and binarization in multimedia documents.In Proceedings of the International Conference on Pattern Recognition (ICPR) 2002, pp. 1037–1040.

10) Anderson, J. A., 1995, *Introduction to Neural Networks* (Cambridge, MA:MIT Press).

11) Using Neural Networks to Create an Adaptive Character Recognition System Alexander J. Faaborg Cornell University, Ithaca NY.

APPENDIX

A. CODES

1.pre processing (m file)

```
tic
clear all;
close all;
im=imread('C:\Users\deepak\Desktop\hehehe.jpg');
mask=[1 1 1
    1 3 1
    1 1 1];
mask1=[1 1 1
    1 -2 1
   -1 -1 -1];
mask2=[1 1 1
    1 -2 -1
    1 -1 -1];
mask3=[1 1 -1
    1 -2 -1
    1 -1 -1];
mask4=[1 -1 -1
    1 -2 1
    1 1 1];
mask5=[-1 -1 -1
    1 -2 1
    1 1 1];
mask6=[-1 -1 1
   -1 -2 1
    1 1 1];
mask7=[-1 1 1
   -1 -2 1
   -1 1 1];
mask8=[1 1 1
   -1 -2 1
   -1 -1 1];
 tempp=0;                          %datatypeconvert
 img=rgb2gray(im);
aakar=size(img);
xxx=aakar(2);
yyy=aakar(1);
figure, imshow(img);
img2=img;
avg1=0;
e=0;
```

```
x=1;
h=0;
avg1=double(mean(mean(img)));
e=entropy(img);
if e < 38
   for j=1:yyy+9

      for i=1:xxx
      if j<=yyy
      tempp=double(img(j,i));
      c(j,i)=255/(1+exp((avg1-tempp)/15));
    end

if j>3 && j<=yyy+3

  if j-3-1==0 && i-1==0
    c2(j-3,i)=(c(j-3,i)*mask(2,2)+c(j-3,i+1)*mask(2,3)+c(j-3+1,i)*mask(3,2)+c(j-
3+1,i+1)*mask(3,3))/5;
  elseif j-3+1== (yyy+1) && i+1== (xxx+1)
    c2(j-3,i)=(c(j-3-1,i-1)*mask(1,1)+c(j-3-1,i)*mask(1,2)+c(j-3,i-1)*mask(2,1)+c(j-
3,i)*mask(2,2))/5;
  elseif i+1==(xxx+1) && j-3-1==0
    c2(j-3,i)=(c(j-3,i-1)*mask(2,1)+c(j-3,i)*mask(2,2)+c(j-3+1,i-1)*mask(3,1)+c(j-
3+1,i)*mask(3,2))/5;
  elseif  i-1==0 && j-3+1==(yyy+1)
    c2(j-3,i)=(c(j-3-1,i)*mask(1,2)+c(j-3-1,i+1)*mask(1,3)+c(j-3,i)*mask(2,2)+c(j-
3,i+1)*mask(2,3))/5;
  elseif j-3-1==0 %&& i-1~=0 && i+1<=xxx
    c2(j-3,i)=(c(j-3,i-1)*mask(2,1)+c(j-3,i)*mask(2,2)+c(j-3,i+1)*mask(2,3)+c(j-3+1,i-
1)*mask(3,1)+c(j-3+1,i)*mask(3,2)+c(j-3+1,i+1)*mask(3,3))/7;
  elseif j-3+1==(yyy+1) %&& i+1<=xxx && i-1>0
    c2(j-3,i)=(c(j-3-1,i)*mask(1,2)+c(j-3-1,i+1)*mask(1,3)+c(j-3,i-1)*mask(2,1)+c(j-
3,i)*mask(2,2)+c(j-3,i+1)*mask(2,3)+c(j-3-1,i-1)*mask(1,1))/7;
  elseif i+1==(xxx+1)% && j-3-1>0 && j-3+1<=yyy
    c2(j-3,i)=(c(j-3-1,i-1)*mask(1,1)+c(j-3-1,i)*mask(1,2)+c(j-3,i-1)*mask(2,1)+c(j-
3,i)*mask(2,2)+c(j-3+1,i-1)*mask(3,1)+c(j-3+1,i)*mask(3,2))/7;
  elseif i-1==0 %&& j-3+1<=yyy && j-3-1>0
    c2(j-3,i)=(c(j-3-1,i)*mask(1,2)+c(j-3-1,i+1)*mask(1,3)+c(j-3,i)*mask(2,2)+c(j-
3,i+1)*mask(2,3)+c(j-3+1,i)*mask(3,2)+c(j-3+1,i+1)*mask(3,3))/7;
  else
    c2(j-3,i)=(c(j-3-1,i-1)*mask(1,1)+c(j-3-1,i)*mask(1,2)+c(j-3-1,i+1)*mask(1,3)+c(j-3,i-
1)*mask(2,1)+c(j-3,i)*mask(2,2)+c(j-3,i+1)*mask(2,3)+c(j-3+1,i-1)*mask(3,1)+c(j-
3+1,i)*mask(3,2)+c(j-3+1,i+1)*mask(3,3))/10;
  end
end
if j>6 && j<=yyy+6
        if j-6-1==0 && i-1==0
```

```
        imgn(j-6,i)=(c2(j-6,i)*mask1(2,2)+c2(j-6,i+1)*mask1(2,3)+c2(j-6+1,i)*mask1(3,2)+c2(j-
6+1,i+1)*mask1(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask2(2,2)+c2(j-6,i+1)*mask2(2,3)+c2(j-
6+1,i)*mask2(3,2)+c2(j-6+1,i+1)*mask2(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask3(2,2)+c2(j-6,i+1)*mask3(2,3)+c2(j-
6+1,i)*mask3(3,2)+c2(j-6+1,i+1)*mask3(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask4(2,2)+c2(j-6,i+1)*mask4(2,3)+c2(j-
6+1,i)*mask4(3,2)+c2(j-6+1,i+1)*mask4(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask5(2,2)+c2(j-6,i+1)*mask5(2,3)+c2(j-
6+1,i)*mask5(3,2)+c2(j-6+1,i+1)*mask5(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask6(2,2)+c2(j-6,i+1)*mask6(2,3)+c2(j-
6+1,i)*mask6(3,2)+c2(j-6+1,i+1)*mask6(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask7(2,2)+c2(j-6,i+1)*mask7(2,3)+c2(j-
6+1,i)*mask7(3,2)+c2(j-6+1,i+1)*mask7(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i)*mask8(2,2)+c2(j-6,i+1)*mask8(2,3)+c2(j-
6+1,i)*mask8(3,2)+c2(j-6+1,i+1)*mask8(3,3));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));
     elseif j-6+1==yyy+1 && i+1==xxx+1
        imgn(j-6,i)=(c2(j-6-1,i-1)*mask1(1,1)+c2(j-6-1,i)*mask1(1,2)+c2(j-6,i-
1)*mask1(2,1)+c2(j-6,i)*mask1(2,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask2(1,1)+c2(j-6-1,i)*mask2(1,2)+c2(j-6,i-
1)*mask2(2,1)+c2(j-6,i)*mask2(2,2));
            imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask3(1,1)+c2(j-6-1,i)*mask3(1,2)+c2(j-6,i-
1)*mask3(2,1)+c2(j-6,i)*mask3(2,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask4(1,1)+c2(j-6-1,i)*mask4(1,2)+c2(j-6,i-
1)*mask4(2,1)+c2(j-6,i)*mask4(2,2));
            imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask5(1,1)+c2(j-6-1,i)*mask5(1,2)+c2(j-6,i-
1)*mask5(2,1)+c2(j-6,i)*mask5(2,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask6(1,1)+c2(j-6-1,i)*mask6(1,2)+c2(j-6,i-
1)*mask6(2,1)+c2(j-6,i)*mask6(2,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask7(1,1)+c2(j-6-1,i)*mask7(1,2)+c2(j-6,i-
1)*mask7(2,1)+c2(j-6,i)*mask7(2,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask8(1,1)+c2(j-6-1,i)*mask8(1,2)+c2(j-6,i-
1)*mask8(2,1)+c2(j-6,i)*mask8(2,2));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));

     elseif i+1==xxx+1 && j-6-1==0
        imgn(j-6,i)=(c2(j-6,i-1)*mask1(2,1)+c2(j-6,i)*mask1(2,2)+c2(j-6+1,i-1)*mask1(3,1)+c2(j-
6+1,i)*mask1(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask2(2,1)+c2(j-6,i)*mask2(2,2)+c2(j-6+1,i-
1)*mask2(3,1)+c2(j-6+1,i)*mask2(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask3(2,1)+c2(j-6,i)*mask3(2,2)+c2(j-6+1,i-
1)*mask3(3,1)+c2(j-6+1,i)*mask3(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask4(2,1)+c2(j-6,i)*mask4(2,2)+c2(j-6+1,i-
1)*mask4(3,1)+c2(j-6+1,i)*mask4(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask5(2,1)+c2(j-6,i)*mask5(2,2)+c2(j-6+1,i-
1)*mask5(3,1)+c2(j-6+1,i)*mask5(3,2));
```

```
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask6(2,1)+c2(j-6,i)*mask6(2,2)+c2(j-6+1,i-
1)*mask6(3,1)+c2(j-6+1,i)*mask6(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask7(2,1)+c2(j-6,i)*mask7(2,2)+c2(j-6+1,i-
1)*mask7(3,1)+c2(j-6+1,i)*mask7(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask8(2,1)+c2(j-6,i)*mask8(2,2)+c2(j-6+1,i-
1)*mask8(3,1)+c2(j-6+1,i)*mask8(3,2));
            imgn(j-6,i)=uint8(abs(imgn(j-6,i)));

      elseif  i-1==0 && j-6+1==yyy+1
        imgn(j-6,i)=(c2(j-6-1,i)*mask1(1,2)+c2(j-6-1,i+1)*mask1(1,3)+c2(j-6,i)*mask1(2,2)+c2(j-
6,i+1)*mask1(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask2(1,2)+c2(j-6-1,i+1)*mask2(1,3)+c2(j-
6,i)*mask2(2,2)+c2(j-6,i+1)*mask2(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask3(1,2)+c2(j-6-1,i+1)*mask3(1,3)+c2(j-
6,i)*mask3(2,2)+c2(j-6,i+1)*mask3(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask4(1,2)+c2(j-6-1,i+1)*mask4(1,3)+c2(j-
6,i)*mask4(2,2)+c2(j-6,i+1)*mask4(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask5(1,2)+c2(j-6-1,i+1)*mask5(1,3)+c2(j-
6,i)*mask5(2,2)+c2(j-6,i+1)*mask5(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask6(1,2)+c2(j-6-1,i+1)*mask6(1,3)+c2(j-
6,i)*mask6(2,2)+c2(j-6,i+1)*mask6(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask7(1,2)+c2(j-6-1,i+1)*mask7(1,3)+c2(j-
6,i)*mask7(2,2)+c2(j-6,i+1)*mask7(2,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask8(1,2)+c2(j-6-1,i+1)*mask8(1,3)+c2(j-
6,i)*mask8(2,2)+c2(j-6,i+1)*mask8(2,3));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));
      elseif j-6-1==0 %&& i-1~=0 && i+1<=xxx
            imgn(j-6,i)=(c2(j-6,i-1)*mask1(2,1)+c2(j-6,i)*mask1(2,2)+c2(j-
6,i+1)*mask1(2,3)+c2(j-6+1,i-1)*mask1(3,1)+c2(j-6+1,i)*mask1(3,2)+c2(j-
6+1,i+1)*mask1(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask2(2,1)+c2(j-6,i)*mask2(2,2)+c2(j-
6,i+1)*mask2(2,3)+c2(j-6+1,i-1)*mask2(3,1)+c2(j-6+1,i)*mask2(3,2)+c2(j-
6+1,i+1)*mask2(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask3(2,1)+c2(j-6,i)*mask3(2,2)+c2(j-
6,i+1)*mask3(2,3)+c2(j-6+1,i-1)*mask3(3,1)+c2(j-6+1,i)*mask3(3,2)+c2(j-
6+1,i+1)*mask3(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask4(2,1)+c2(j-6,i)*mask4(2,2)+c2(j-
6,i+1)*mask4(2,3)+c2(j-6+1,i-1)*mask4(3,1)+c2(j-6+1,i)*mask4(3,2)+c2(j-
6+1,i+1)*mask4(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask5(2,1)+c2(j-6,i)*mask5(2,2)+c2(j-
6,i+1)*mask5(2,3)+c2(j-6+1,i-1)*mask5(3,1)+c2(j-6+1,i)*mask5(3,2)+c2(j-
6+1,i+1)*mask5(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask6(2,1)+c2(j-6,i)*mask6(2,2)+c2(j-
6,i+1)*mask6(2,3)+c2(j-6+1,i-1)*mask6(3,1)+c2(j-6+1,i)*mask6(3,2)+c2(j-
6+1,i+1)*mask6(3,3));
```

```
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask7(2,1)+c2(j-6,i)*mask7(2,2)+c2(j-
6,i+1)*mask7(2,3)+c2(j-6+1,i-1)*mask7(3,1)+c2(j-6+1,i)*mask7(3,2)+c2(j-
6+1,i+1)*mask7(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6,i-1)*mask8(2,1)+c2(j-6,i)*mask8(2,2)+c2(j-
6,i+1)*mask8(2,3)+c2(j-6+1,i-1)*mask8(3,1)+c2(j-6+1,i)*mask8(3,2)+c2(j-
6+1,i+1)*mask8(3,3));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));

      elseif j-6+1==yyy+1 %&& i+1<=xxx && i-1>0
        imgn(j-6,i)=(c2(j-6-1,i)*mask1(1,2)+c2(j-6-1,i+1)*mask1(1,3)+c2(j-6,i-
1)*mask1(2,1)+c2(j-6,i)*mask1(2,2)+c2(j-6,i+1)*mask1(2,3)+c2(j-6-1,i-1)*mask1(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask2(1,2)+c2(j-6-1,i+1)*mask2(1,3)+c2(j-6,i-
1)*mask2(2,1)+c2(j-6,i)*mask2(2,2)+c2(j-6,i+1)*mask2(2,3)+c2(j-6-1,i-1)*mask2(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask3(1,2)+c2(j-6-1,i+1)*mask3(1,3)+c2(j-6,i-
1)*mask3(2,1)+c2(j-6,i)*mask3(2,2)+c2(j-6,i+1)*mask3(2,3)+c2(j-6-1,i-1)*mask3(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask4(1,2)+c2(j-6-1,i+1)*mask4(1,3)+c2(j-6,i-
1)*mask4(2,1)+c2(j-6,i)*mask4(2,2)+c2(j-6,i+1)*mask4(2,3)+c2(j-6-1,i-1)*mask4(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask5(1,2)+c2(j-6-1,i+1)*mask5(1,3)+c2(j-6,i-
1)*mask5(2,1)+c2(j-6,i)*mask5(2,2)+c2(j-6,i+1)*mask5(2,3)+c2(j-6-1,i-1)*mask5(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask6(1,2)+c2(j-6-1,i+1)*mask6(1,3)+c2(j-6,i-
1)*mask6(2,1)+c2(j-6,i)*mask6(2,2)+c2(j-6,i+1)*mask6(2,3)+c2(j-6-1,i-1)*mask6(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask7(1,2)+c2(j-6-1,i+1)*mask7(1,3)+c2(j-6,i-
1)*mask7(2,1)+c2(j-6,i)*mask7(2,2)+c2(j-6,i+1)*mask7(2,3)+c2(j-6-1,i-1)*mask7(1,1));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask8(1,2)+c2(j-6-1,i+1)*mask8(1,3)+c2(j-6,i-
1)*mask8(2,1)+c2(j-6,i)*mask8(2,2)+c2(j-6,i+1)*mask8(2,3)+c2(j-6-1,i-1)*mask8(1,1));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));

      elseif i+1==xxx+1 %&& j-6-1>0 && j-6+1<=250
        imgn(j-6,i)=(c2(j-6-1,i-1)*mask1(1,1)+c2(j-6-1,i)*mask1(1,2)+c2(j-6,i-
1)*mask1(2,1)+c2(j-6,i)*mask1(2,2)+c2(j-6+1,i-1)*mask1(3,1)+c2(j-6+1,i)*mask1(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask2(1,1)+c2(j-6-1,i)*mask2(1,2)+c2(j-6,i-
1)*mask2(2,1)+c2(j-6,i)*mask2(2,2)+c2(j-6+1,i-1)*mask2(3,1)+c2(j-6+1,i)*mask2(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask3(1,1)+c2(j-6-1,i)*mask3(1,2)+c2(j-6,i-
1)*mask3(2,1)+c2(j-6,i)*mask3(2,2)+c2(j-6+1,i-1)*mask3(3,1)+c2(j-6+1,i)*mask3(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask4(1,1)+c2(j-6-1,i)*mask4(1,2)+c2(j-6,i-
1)*mask4(2,1)+c2(j-6,i)*mask4(2,2)+c2(j-6+1,i-1)*mask4(3,1)+c2(j-6+1,i)*mask4(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask5(1,1)+c2(j-6-1,i)*mask5(1,2)+c2(j-6,i-
1)*mask5(2,1)+c2(j-6,i)*mask5(2,2)+c2(j-6+1,i-1)*mask5(3,1)+c2(j-6+1,i)*mask5(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask6(1,1)+c2(j-6-1,i)*mask6(1,2)+c2(j-6,i-
1)*mask6(2,1)+c2(j-6,i)*mask6(2,2)+c2(j-6+1,i-1)*mask6(3,1)+c2(j-6+1,i)*mask6(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask7(1,1)+c2(j-6-1,i)*mask7(1,2)+c2(j-6,i-
1)*mask7(2,1)+c2(j-6,i)*mask7(2,2)+c2(j-6+1,i-1)*mask7(3,1)+c2(j-6+1,i)*mask7(3,2));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask8(1,1)+c2(j-6-1,i)*mask8(1,2)+c2(j-6,i-
1)*mask8(2,1)+c2(j-6,i)*mask8(2,2)+c2(j-6+1,i-1)*mask8(3,1)+c2(j-6+1,i)*mask8(3,2));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));
      elseif i-1==0 %&& j-6+1<=yyy && j-6-1>0
```

```
        imgn(j-6,i)=(c2(j-6-1,i)*mask1(1,2)+c2(j-6-1,i+1)*mask1(1,3)+c2(j-6,i)*mask1(2,2)+c2(j-
6,i+1)*mask1(2,3)+c2(j-6+1,i)*mask1(3,2)+c2(j-6+1,i+1)*mask1(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask2(1,2)+c2(j-6-1,i+1)*mask2(1,3)+c2(j-
6,i)*mask2(2,2)+c2(j-6,i+1)*mask2(2,3)+c2(j-6+1,i)*mask2(3,2)+c2(j-6+1,i+1)*mask2(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask3(1,2)+c2(j-6-1,i+1)*mask3(1,3)+c2(j-
6,i)*mask3(2,2)+c2(j-6,i+1)*mask3(2,3)+c2(j-6+1,i)*mask3(3,2)+c2(j-6+1,i+1)*mask3(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask4(1,2)+c2(j-6-1,i+1)*mask4(1,3)+c2(j-
6,i)*mask4(2,2)+c2(j-6,i+1)*mask4(2,3)+c2(j-6+1,i)*mask4(3,2)+c2(j-6+1,i+1)*mask4(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask5(1,2)+c2(j-6-1,i+1)*mask5(1,3)+c2(j-
6,i)*mask5(2,2)+c2(j-6,i+1)*mask5(2,3)+c2(j-6+1,i)*mask5(3,2)+c2(j-6+1,i+1)*mask5(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask6(1,2)+c2(j-6-1,i+1)*mask6(1,3)+c2(j-
6,i)*mask6(2,2)+c2(j-6,i+1)*mask6(2,3)+c2(j-6+1,i)*mask6(3,2)+c2(j-6+1,i+1)*mask6(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask7(1,2)+c2(j-6-1,i+1)*mask7(1,3)+c2(j-
6,i)*mask7(2,2)+c2(j-6,i+1)*mask7(2,3)+c2(j-6+1,i)*mask7(3,2)+c2(j-6+1,i+1)*mask7(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i)*mask8(1,2)+c2(j-6-1,i+1)*mask8(1,3)+c2(j-
6,i)*mask8(2,2)+c2(j-6,i+1)*mask8(2,3)+c2(j-6+1,i)*mask8(3,2)+c2(j-6+1,i+1)*mask8(3,3));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));
    else
        imgn(j-6,i)=(c2(j-6-1,i-1)*mask1(1,1)+c2(j-6-1,i)*mask1(1,2)+c2(j-6-
1,i+1)*mask1(1,3)+c2(j-6,i-1)*mask1(2,1)+c2(j-6,i)*mask1(2,2)+c2(j-6,i+1)*mask1(2,3)+c2(j-
6+1,i-1)*mask1(3,1)+c2(j-6+1,i)*mask1(3,2)+c2(j-6+1,i+1)*mask1(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask2(1,1)+c2(j-6-1,i)*mask2(1,2)+c2(j-6-
1,i+1)*mask2(1,3)+c2(j-6,i-1)*mask2(2,1)+c2(j-6,i)*mask2(2,2)+c2(j-6,i+1)*mask2(2,3)+c2(j-
6+1,i-1)*mask2(3,1)+c2(j-6+1,i)*mask2(3,2)+c2(j-6+1,i+1)*mask2(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask3(1,1)+c2(j-6-1,i)*mask3(1,2)+c2(j-6-
1,i+1)*mask3(1,3)+c2(j-6,i-1)*mask3(2,1)+c2(j-6,i)*mask3(2,2)+c2(j-6,i+1)*mask3(2,3)+c2(j-
6+1,i-1)*mask3(3,1)+c2(j-6+1,i)*mask3(3,2)+c2(j-6+1,i+1)*mask3(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask4(1,1)+c2(j-6-1,i)*mask4(1,2)+c2(j-6-
1,i+1)*mask4(1,3)+c2(j-6,i-1)*mask4(2,1)+c2(j-6,i)*mask4(2,2)+c2(j-6,i+1)*mask4(2,3)+c2(j-
6+1,i-1)*mask4(3,1)+c2(j-6+1,i)*mask4(3,2)+c2(j-6+1,i+1)*mask4(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask5(1,1)+c2(j-6-1,i)*mask5(1,2)+c2(j-6-
1,i+1)*mask5(1,3)+c2(j-6,i-1)*mask5(2,1)+c2(j-6,i)*mask5(2,2)+c2(j-6,i+1)*mask5(2,3)+c2(j-
6+1,i-1)*mask5(3,1)+c2(j-6+1,i)*mask5(3,2)+c2(j-6+1,i+1)*mask5(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask6(1,1)+c2(j-6-1,i)*mask6(1,2)+c2(j-6-
1,i+1)*mask6(1,3)+c2(j-6,i-1)*mask6(2,1)+c2(j-6,i)*mask6(2,2)+c2(j-6,i+1)*mask6(2,3)+c2(j-
6+1,i-1)*mask6(3,1)+c2(j-6+1,i)*mask6(3,2)+c2(j-6+1,i+1)*mask6(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask7(1,1)+c2(j-6-1,i)*mask7(1,2)+c2(j-6-
1,i+1)*mask7(1,3)+c2(j-6,i-1)*mask7(2,1)+c2(j-6,i)*mask7(2,2)+c2(j-6,i+1)*mask7(2,3)+c2(j-
6+1,i-1)*mask7(3,1)+c2(j-6+1,i)*mask7(3,2)+c2(j-6+1,i+1)*mask7(3,3));
        imgn(j-6,i)=imgn(j-6,i)+(c2(j-6-1,i-1)*mask8(1,1)+c2(j-6-1,i)*mask8(1,2)+c2(j-6-
1,i+1)*mask8(1,3)+c2(j-6,i-1)*mask8(2,1)+c2(j-6,i)*mask8(2,2)+c2(j-6,i+1)*mask8(2,3)+c2(j-
6+1,i-1)*mask8(3,1)+c2(j-6+1,i)*mask8(3,2)+c2(j-6+1,i+1)*mask8(3,3));
        imgn(j-6,i)=uint8(abs(imgn(j-6,i)));
    end

end
 if j>9
```

```
  if j-9-1==0 && i-1==0
     davg(j-9,i)=(double(imgn(j-9,i+1))+double(imgn(j-9+1,i))+double(imgn(j-9+1,i+1)))/3;
       if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
         dbar(j-9,i)=(255-double(imgn(j-9,i)));
       else
         dbar(j-9,i)=double(imgn(j-9,i));
       end
   elseif j-9+1== (yyy+1) && i+1== (xxx+1)
         davg(j-9,i)=(double(imgn(j-9-1,i-1))+double(imgn(j-9-1,i))+double(imgn(j-9,i-1)))/3;

if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
   dbar(j-9,i)=(255-double(imgn(j-9,i)));
else
   dbar(j-9,i)=double(imgn(j-9,i));
end
   elseif i+1==(xxx+1) && j-9-1==0
         davg(j-9,i)=(double(imgn(j-9,i-1))+double(imgn(j-9+1,i-1))+double(imgn(j-9+1,i)))/3;
              if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                  dbar(j-9,i)=(255-double(imgn(j-9,i)));
              else
                  dbar(j-9,i)=double(imgn(j-9,i));
              end
    elseif  i-1==0 && j-9+1==(yyy+1)
                 davg(j-9,i)=(double(imgn(j-9-1,i))+double(imgn(j-9-1,i+1))+double(imgn(j-
9,i+1)))/3;
        if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                        dbar(j-9,i)=(255-double(imgn(j-9,i)));
        else
                        dbar(j-9,i)=double(imgn(j-9,i));
           end

    elseif j-9-1==0
                        davg(j-9,i)=(double(imgn(j-9,i-1))+double(imgn(j-9,i+1))+double(imgn(j-
9+1,i-1))+double(imgn(j-9+1,i))+double(imgn(j-9+1,i+1)))/5;

         if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                                dbar(j-9,i)=(255-double(imgn(j-9,i)));
                else
                                dbar(j-9,i)=double(imgn(j-9,i));
                end

    elseif j-9+1==(yyy+1) %&& i+1<=xxx && i-1>0
                 davg(j-9,i)=(double(imgn(j-9-1,i))+double(imgn(j-9-1,i+1))+double(imgn(j-9,i-
1))+double(imgn(j-9,i+1))+double(imgn(j-9-1,i-1)))/5;
                 if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                                dbar(j-9,i)=(255-double(imgn(j-9,i)));
                        else
```

```
                        dbar(j-9,i)=double(imgn(j-9,i));
                    end

    elseif i+1==(xxx+1)% && j-9-1>0 && j-9+1<=yyy
                davg(j-9,i)=(double(imgn(j-9-1,i-1))+double(imgn(j-9-1,i))+double(imgn(j-9,i-
1))+double(imgn(j-9+1,i-1))+double(imgn(j-9+1,i)))/5;
                    if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                        dbar(j-9,i)=(255-double(imgn(j-9,i)));
                    else
                        dbar(j-9,i)=double(imgn(j-9,i));
                    end
    elseif i-1==0 %&& j-9+1<=yyy && j-9-1>0
                davg(j-9,i)=(double(imgn(j-9-1,i))+double(imgn(j-9-1,i+1))+double(imgn(j-
9,i+1))+double(imgn(j-9+1,i))+double(imgn(j-9+1,i+1)))/5;
                    if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                        dbar(j-9,i)=(255-double(imgn(j-9,i)));
                    else
                        dbar(j-9,i)=double(imgn(j-9,i));
                    end
    else
                davg(j-9,i)=(double(imgn(j-9-1,i-1))+double(imgn(j-9-1,i))+double(imgn(j-9-
1,i+1))+double(imgn(j-9,i-1))+double(imgn(j-9,i+1))+double(imgn(j-9+1,i-1))+double(imgn(j-
9+1,i))+double(imgn(j-9+1,i+1)))/8;
                    if (davg(j-9,i)-double(imgn(j-9,i)))>127.5
                        dbar(j-9,i)=(255-double(imgn(j-9,i)));
                    else
                        dbar(j-9,i)=double(imgn(j-9,i));
                    end
    end
 end
end
end
new=mean(mean(dbar))*2.3;
for j=1:yyy
   for i=1:xxx
       if dbar(j,i) < uint8(new)
          dbarb(j,i)=1;
       else
          dbarb(j,i)=0;
       end


    end
end
figure, imshow(dbarb);
end
toc
```

appendix 2

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

appendix 3

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

% Computes the correlation between template and input image
```

```
% and its output is a string containing the letter.
% Size of 'imagn' must be 42 x 24 pixels
% Example:
% imagn=imread('D.bmp');
% letter=read_letter(imagn)
global templates
comp=[ ];
for n=1:num_letras
   sem=corr2(templates{1,n},imagn);
   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='R';
elseif vd==19
    letter='S';
elseif vd==20
```

```
    letter='T';
elseif vd==21
    letter='U';
elseif vd==22
    letter='V';
elseif vd==23
    letter='W';
elseif vd==24
    letter='X';
elseif vd==25
    letter='Y';
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
```