CHARACTER RECOGNITION USING TEMPLATE MATCHING

PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF THE BACHELOR OF INFORMATION TECHNOLOGY (B.I.T) DEGREE

DEPARTMENT OF COMPUTER SCIENCE

JAMIA MILLIA ISLAMIA

NEW DELHI-25

SUBMITTED BY :

Mr.Danish Nadeem

&

Miss.Saleha Rizvi

1. INTRODUCTION

1.1 PROBLEM DEFINITION

In the proposed system, we shall be dealing with the problem of machine reading typewritten/handwritten characters. This corresponds to the ability of human beings to recognize such characters, which they are able to do little or no difficulty.

The aim is to produce a system that classifies a given input as belonging to a certain class rather then to identify them uniquely, as every input pattern. The system performs character recognition by quantification of the character into a mathematical vector entity using the geometrical properties of the character image. The scope of the proposed system is limited to the recognition of a single character.

In the ensuing section we introduce some background concepts that are necessary to understand the proposed system. Then we proceed to section 1.3 to explain our methodology.

1.2 BACKGROUND

1.2.1 GRAPHIC FILES

A Graphic file is a file containing a picture that may be a line or scanned photograph. Any program that displays or manipulates stored images needs to be able to store image for a later use.

Data in graphic files can be encoded in two different ways

- √ ***ASCII TEXT***

This is a readable text which is easy for humans to read and to some extent to edit and easy for programs to read and write. But it is bulky and slow to read and write from programs.

- √ ***COMPRESSED FORMAT(Binary Formats)***

They are very compact but incomprehensible to human and require complex reading and writing routines. They vary a lot in terms of the flexibility they offer for the image size, shape, colors and their attributes. At one end is the TIFF (Tagged Input File Format) with so many different options and features that not TIFF implementation can read them all and at other end is Mac Paint which allows storing the image in exactly one size, two colors and one way.

The graphic files are further classified as of two types in terms of the manner in which they store the image.

- √ ***BITMAPPED FORMAT***

Here the picture is represented as rectangular array of dots. It stores complete digitally encoded images. They are also called as raster or dot-matrix description. It is used when the images are, in large part, created by hand or scanned from an original document or photograph using some type of scanner.

A few types of bitmapped graphic files formats are:

- TIFF (Tagged Input File Format)
- GIF(Graphics Interchange Format)
- BMP(Bit map Format)
- Mac Paint
- IMG
- TGA(Targa)
- JPEG (Joint Photographic Expert Group)

√ ***VECTOR FORMATS***

They represent a picture as a series of lines and arcs i.e. it stores the individual graphics that make up the image. These images are also called as line images. As most of the lines that are needed could be represented by relatively simple mathematical equations hence, images could be stored economically. For e.g. to specify a straight line all that is needed is a knowledge of the positions of the two end points of the line and for display purposes the line can then be reconstructed knowing the geometrical properties. Similarly to draw a circle all that is needed is knowledge of its center and its radius.

The advantages of vector formats are:

- They require less size.
- Their quality is not affected when the images are magnified as contrasting to the to the pixel images.

1.2.2 PIXEL

A pixel (picture-element) is a dot or the most fundamental unit that makes up the image. All pixels have a value associated with them called as the **pixel value** -representing the color for that point/pixel. For the simplest pictures, each point is black or white so the pixel value is either 0 or 1, a single bit. However commonly, the picture is in grayscale or color, in which case there has to be a large range of pixel values. For a grayscale image, each pixel might be 8 bits, so the value could range from 0 for black to 255 for white.

1.2.3 TRUE COLOR

24 bit color represents the limit of the human eye's ability to differentiate colors, Thus to human eye, there is no perceptible difference between a 24 bit color image of an object and the object viewed directly. Hence it is referred to as the true color.

1.2.4 PALETTE / COLOR MAP

Full color images can be very large. A 600* 800 image may contain 4, 80,000 pixels. If each of the pixel we stored as 24-bit value than the image would consume 1.4 MB. To decrease the amount of space needed to store the image, the concept of **color map** or **palette** is used. Rather than storing the actual color of each pixel in the file, the color maps contains a list of all colors used in the image and the individual pixel values are stored as entry numbers in the color map/palette. A typical color map has 16 or 256 entries, so each pixel value is only 4 or 8 bits, an enormous savings from 24 bits per pixel. Programs can create various screen effects by changing the color map. The advantage of using the color map is that

- The amount of RAM and memory needed to store the image is considerably reduced.
- The image definition is virtualized. The value of the latter can be demonstrated by considering the task of changing one color in the image instead of changing all pixels of the color in image, we need to change only the palette entry for that color.

1.2.5 COLOR MODEL

A *color model* is a formal way for representing and defining colors. A synonymous term is photometric interpretation. There are different types of color models. For details refer to appendix 9.1.1

1.2.6 RESOLUTION

Graphic images on the screen are made up of tiny dots called pixels or picture elements. The display resolution is defined by the no. of rows (called scan line) from top to bottom and no. of pixels from left to right on each scan line. Each mode uses a particular resolution, higher the resolution more pleasing is the picture. Higher resolution means a sharper, clearer picture with less pronounced staircase effect on drawing lines diagonally and better looking text characters. High resolution requires more memory requirement to display the pictures.

1.2.7 COMPRESSION

- It is the special case of a general technique known as encoding.
- Encoding is the technique that takes a string of symbols and outputs a code for each input symbol. The output of this process is called as encoded representation of its input.

- If the encoded output of an encoding scheme is smaller than the uuencoded input, we have a compression algorithm.
- For every encoding algorithm there is an inverse process called a decoding algorithm.
- The decoding algorithm when applied to the output of the encoded algorithm reproduces the original output that was processed by encoding algorithm.
- If the output of the decoding algorithm reproduces each input that was given to encoding algorithm, the encoding algorithm is said to be lossless. Algorithms that are not lossless are said to be lossy.
- A compression algorithm and its corresponding decompression algorithm are collectively referred to as CODEC.
- The compression depends upon the redundancy that is present in the information. If a piece of information possesses sufficiently low redundancy it is not compressible by standard means.

In case of graphic imagery, redundancy is manifest as repeating pattern colors. Most images represent a higher degree of both. For some details on type of compression schemes refer appendix 9.1.3

1.2.8 WINDOWS BITMAP FORMAT (BMP)

The windows BMP format is a general purpose format for storing Device Independent Bitmaps (DIB's). By DIB we mean that the physical interpretation of the image and its palette are fixed without regard to the requirements of any potential display device. It is most often used to store screen and scanner generated imagery.

The BMP file only supports single line bitmaps of 1, 4, 8 or 24 bits per pixel. One annoying aspect of BMP is that image is stored by scan line proceeding from the bottom row to the top. All other formats use the reverse order or at least support top-to-bottom order as an option. Top to bottom is a defacto standard.

BMP breaks the file into four separate components

- File Header
- An image header.
- An array of palette entries.
- Actual bitmap

When dealing with BMP it is recommended to use a palette unless we are dealing with a 24-bit image. BMP supports image compression by RLE (run length encoding) only images with 4 bit and 8 bit per pixel sizes can be encoded. The interpretation of encoded image data slightly depends on which pixel size is present. Scanned file in the BMP format are padded with unused bits in the end so that their length is an integral number of double words i.e. the number of bytes is evenly divisible by 4.

Despite the fact that the format supports compression, it's rare to find an application that actually bothers to encode image data in this format thus, only a few BMP files are compressed.

1.2.9 PBM APPROACH

Though there are dozens of varieties of bitmap files, all the bitmap formats have far more similarities than differences. All the formats store a rectangular array of pixels with some number of bits per pixel. Since the formats are all so similar, we can use a common framework to read and write files. The PBM utilities, which were written mostly by Jef Poskanzer (that's what P stands for) are a group of interrelated programs that read write a large variety of bitmap file formats and permit a variety of transformations on the files, such as scaling or reducing the number of colors.

It defines three simple image formats PBM, PGM and PPM into which all other formats can be translated. PBM defines size similar file formats. All describe simple rectangular array of pixels. Each starts with an ASCII text header consisting of fields separated by white spaces (space and new lines). The first field, the type field contains the letter P followed by a digit that identifies the particular format. The second and third field identifies the width and height of image in pixels, as ASCII digits. After the header comes the image data, row-by-row, from the top of the image to the bottom.

- √ **PBM FORMAT**
- √ **PGM FORMAT**
- √ **PPM FORMAT**

For further details on each of them refer appendix 9.1.4

1.2.10 CHARACTER RECOGNITION (General Idea)

Automatic Character recognition include a number of problems which make mandatory the development of an automatic process of classifying input information according to the specific requirements imposed on such a classification. The problem of character recognition results in the automatic making of the decision on the basis of data which does not directly indicate the best of all possible decisions. In general form the problem may be formulated as

"There exists a set M of some objects which are divided into n nonintersecting subsets called object classes or characters. To each character there corresponds a specific character description x which, without restriction may be considered as multidimensional vector. The description of objects are not necessarily unique i.e. identical description may sometimes correspond to different object classes." The problem is to design an algorithm optimal in some sense which on the basis of given description of the character would indicate the class to which it belongs. In the most general case, a reader which recognize characters must solve a problem equivalent to the calculation of the several logic function(depending on the character being distinguished), such that each function is equal to unity when and only when there is a character corresponding to this function in the field of vision of the character reader. These functions must be invariant with respect to all shifts and changes in the outlines of the characters which are considered permissible for this character reader.

It is possible that in the calculation of functions corresponding to various characters several actions or sequences of action will be repeated. These sequences of action yield values of elementary functions that are common to all the characters. In order to recognize a character it is necessary to determine which value the elementary function has for this character. It is natural to call these elementary function indicators. The indicators must be invariant with respect to the permissible changes in the characters.

The direction of the stroke, which makes up the character, can be taken as indicators, which are invariant with respect to the forward movement and changes in the dimensions and in some cases with respect to the change in the proportion in the shape of the character. If the direction of the strokes is determined approximately, similar directions being taken as identical, then it is possible in addition to obtain variance with respect to slight rotations of a character or part of it. The directions of the stroke alone do not provide exhaustive information about the character e.g. "T" "L" may be characterized by identical stroke directions. Additional information can be obtained about the character if the direction of is determined , not of the stroke themselves but also of the their boundaries i.e. the boundaries between the black and white fields, and if in this process account is taken as to which side of the boundary the black field is on. It is most convenient to analyze characters by means of such indicators moving along the boundary of a stroke and recording the direction of movement in the sequence in which it occurs in the character(Fig 1)

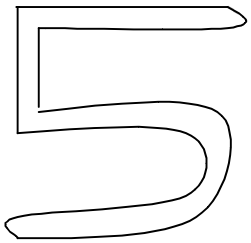


FIG 1:

An analysis of the character is made simple if not all the stroke directions are recorded, but only their most characteristic combinations. Hence during the process, information is lost about the mutual location of the separate groups of sequential direction of strokes which occur in analysis of one character. This difficulty can be overcome if additional characteristic of each character is introduced, namely its position with respect to other strokes or for greater certainty with respect to a rectangle drawn around the character. The algorithm proposed for the recognition of letters and numerals as indicators, each of which is characterized by a definite sequence in direction of movement when passing around the contour of a fixed part of the rectangle drawn around the character.

1.2.11 TYPES OF CHARACTER RECOGNITION SYSTEMS

The constant development of computer tools leads to a requirement of easier interfaces between the man and the computer. CR is one way of achieving this. A CR deal with the problem of reading handwritten/typewritten character offline i.e. at some point in time (in mins, sec, hrs) after it has been written. However recognition of unconstrained handwritten text can be very difficult because characters cannot be reliably isolated especially when the text is cursive handwriting. They are classified as the following two types

- √ **Offline CR.**
- √ **Online CR.**

We shall be discussing them in detail and some of the methods employed to deal with them

Online Character Recognition

In case of online character recognition there is real time recognition of characters. Online systems have better information for doing recognition since they have timing information and since they avoid the initial search step of locating the character as in the case of their offline counterpart. Online systems obtain the position of the pen as a function of time directly from the interface.

Offline recognition of characters is known as a challenging problem because of the complex character shapes and great variation of character symbols written in different modes. In the past decades, a great deal of effort has been made towards solving this problem. ([1], [2], [3] [4], [5]). For a case study of online character recognition systems refer to the appendix section 9.2

Offline Character Recognition

In case of offline character recognition the typewritten/handwritten character is typically scanned in form of a paper document and made available in the form of a binary or gray scale image to the recognition algorithm. Offline character recognition is a more challenging and difficult task as we do not have control over the medium and instrument used. The artifacts of the complex interaction between the instrument medium and subsequent operations such as scanning and binarization present additional challenges to the algorithm for the offline CR. Therefore offline character recognition is considered as a more challenging task then its online counterpart.

The steps involved in character recognition after an image scanner optically captures text images to be recognized is given to the recognition algorithm.

- √ Document Analysis / Preprocessing
- √ Character Recognition / Classification

Document Analysis

The process of extraction of text from the document is called as document analysis. Recognition depends to a great extent on the original document quality and registered image quality.

Character Recognition

The character recognition algorithm has two essential components *feature extractor* and the *classifier*. Feature analysis determines the descriptors, or the feature set used to describe all characters. Given a character image, the *feature extractor* derives the features that the character possesses. The derived features are then used as input to the character classifier.

Template matching or matrix matching, is one of the most common classification methods. Here individual image pixels are used as features. Classification is performed by comparing an input

character with a set of templates (or prototypes) from each character class. Each comparison results in a similarity measure between the input characters with a set of templates. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character's identity is assigned the identity of the most similar template. *Template matching is a trainable process as template characters can be changed*

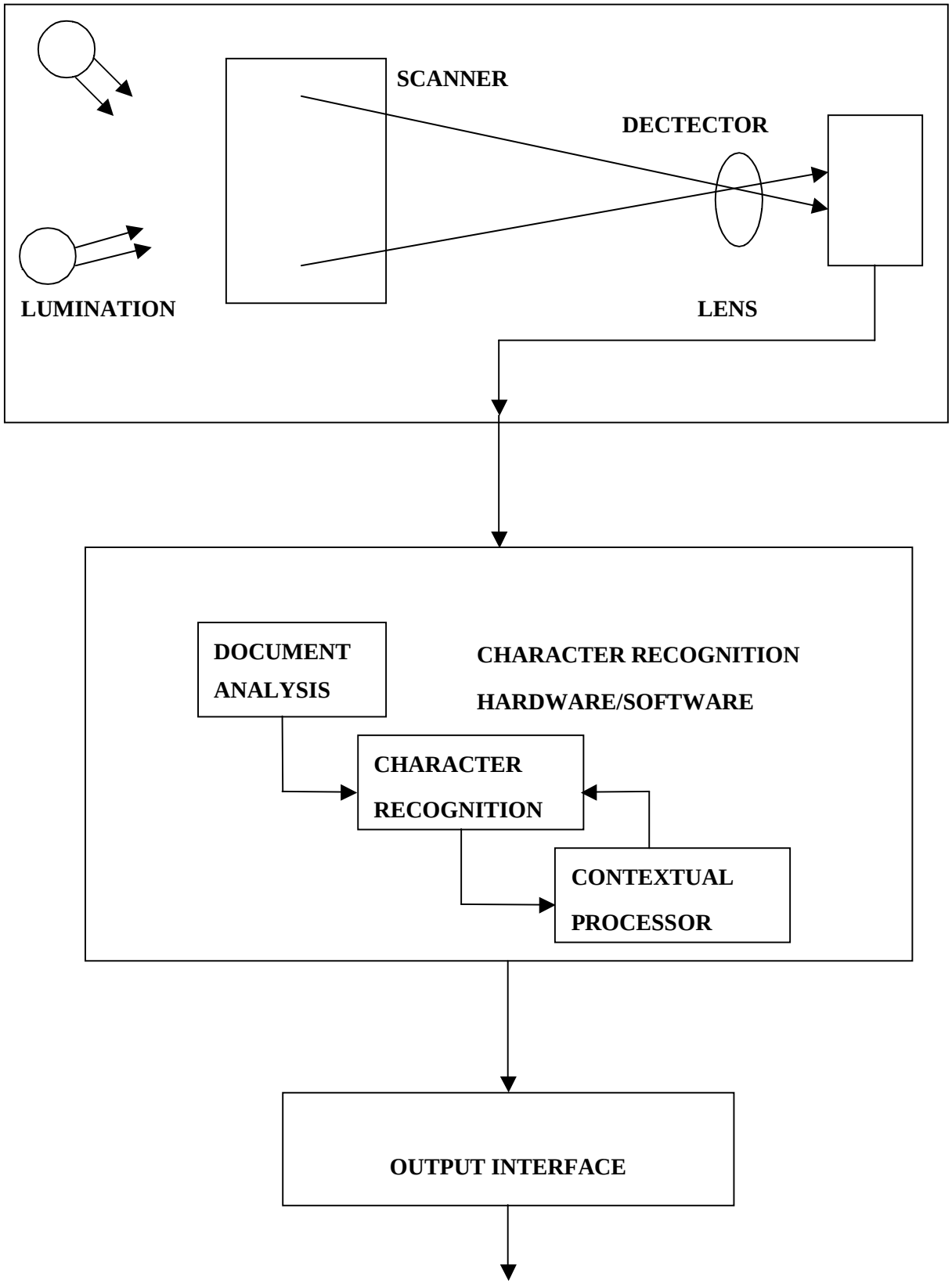

SCANNER
DECTECTOR
LUMINATION
LENS
DOCUMENT ANALYSIS
CHARACTER RECOGNITION HARDWARE/SOFTWARE
CHARACTER RECOGNITION
CONTEXTUAL PROCESSOR
OUTPUT INTERFACE
TO APPLICATION

FIG 2:

Character misclassification stem from two main sources: poor quality character images and poor discriminatory ability. Poor document quality, image scanning and preprocessing all degrade performance by yielding poor quality characters. The character recognition method may not have been trained for a proper response on the character causing the error. This type of error source is difficult to overcome because the recognition method may have its own limitations and all possible character images cannot possibly be considered in training the classifier.

Recognition rates for machine-printed characters can reach over 99% but handwritten character recognition rates are typically lower because every person writes differently. This random nature often manifests itself by resulting in misclassification.

1.2.12 CASE STUDY OF AN OFFLINE CHARACTER RECOGNITION SYSTEM

Fuzzy Logic For Handwritten Numeral Character Recognition [10]

This method considers the handwritten character to be a directed abstract graph of node sets consisting of tips, corners and functions and the branch set consists of line segments connecting pair of adjacent nodes. The segments (branches) are fuzzily classified to branch types (features) such as straight lines, circles or portions of circles. Since the features under consideration are fuzzy in nature the fuzzy set is utilized and the features are treated as fuzzy variables. As handwritten characters are considered to be ill-defined objects, with the art of fuzzy functions such objects can be objectively defined and studied. A handwritten character is represented by fuzzy functions which relates its fuzzy variables and by the node pair involved in each fuzzy variable. The recognition involves two steps :

- √ First the unknown character is preprocessed to produce its representation.
- √ The classification of the unknown character is reduced to finding a character (previously learned of which the representation is isomorphic to the representation of the unknown character).

As there is lack of precision in the definition of the elements of the feature set, the fuzzy concept has been used. The characterization is position, size and distortion variant.

Handwritten Character Recognition System

A. HANDWRITTEN CHARACTER REPRESENTATION

In this system a handwritten character representation consists of

1) Straight Line(vertical, horizontal & slant)
2) Circle

3) A portion of circle of various orientations. Here the class of handwritten lines, circles or portions of circles is considered as fuzzy sets in this correspondence.

This approach is size and position invariant, hence compatible with the visual perception of a man. It follows generalization and needs of only one example to recognize any equivalent variants.

B. PATTERN CLASSIFIER

The decision criteria for pattern classifier is executed in two steps

1) The branch features of the pattern to be classified and the branch features of the prototype are compared. Those prototypes that completely match the branch feature of the patterns are restrained.
2) The node pattern of the same branch type of the pattern to be classified and each retrieved prototype are compared. The fact that the numbering of the nodes in the pattern may not be the same as that of an isomorphic mapping. When the mapping is isomorphic the prototype is accepted otherwise it is rejected.

Recognition System

a) *Input pattern*

The input pattern of the system is digitized into a rectangular picture- frame array

$$P= \{ p=(i,j)|1<=i<=n, 1<=j<=m; m,n \epsilon N\}$$

The pattern is a binary picture i.e. the points on the pattern assume the value of one while the other points of the frame take the value of zero.

b) *Thinning*

Here the algorithm is restricted for binary pattern. The algorithm tests the boundary points on the pattern. Those points on the boundary, the removal of which would not alter the connectivity and those that do not lie on a tip of a line are detected.

c) *Labeling*

i. *Node Detection and Labeling*

A node is the skeleton of a pattern ,defined as the collection of tips (points that have one neighbor), corner (points that have two neighbors) and were an abrupt change of directions occurs and functions (points that have three or more neighbors).

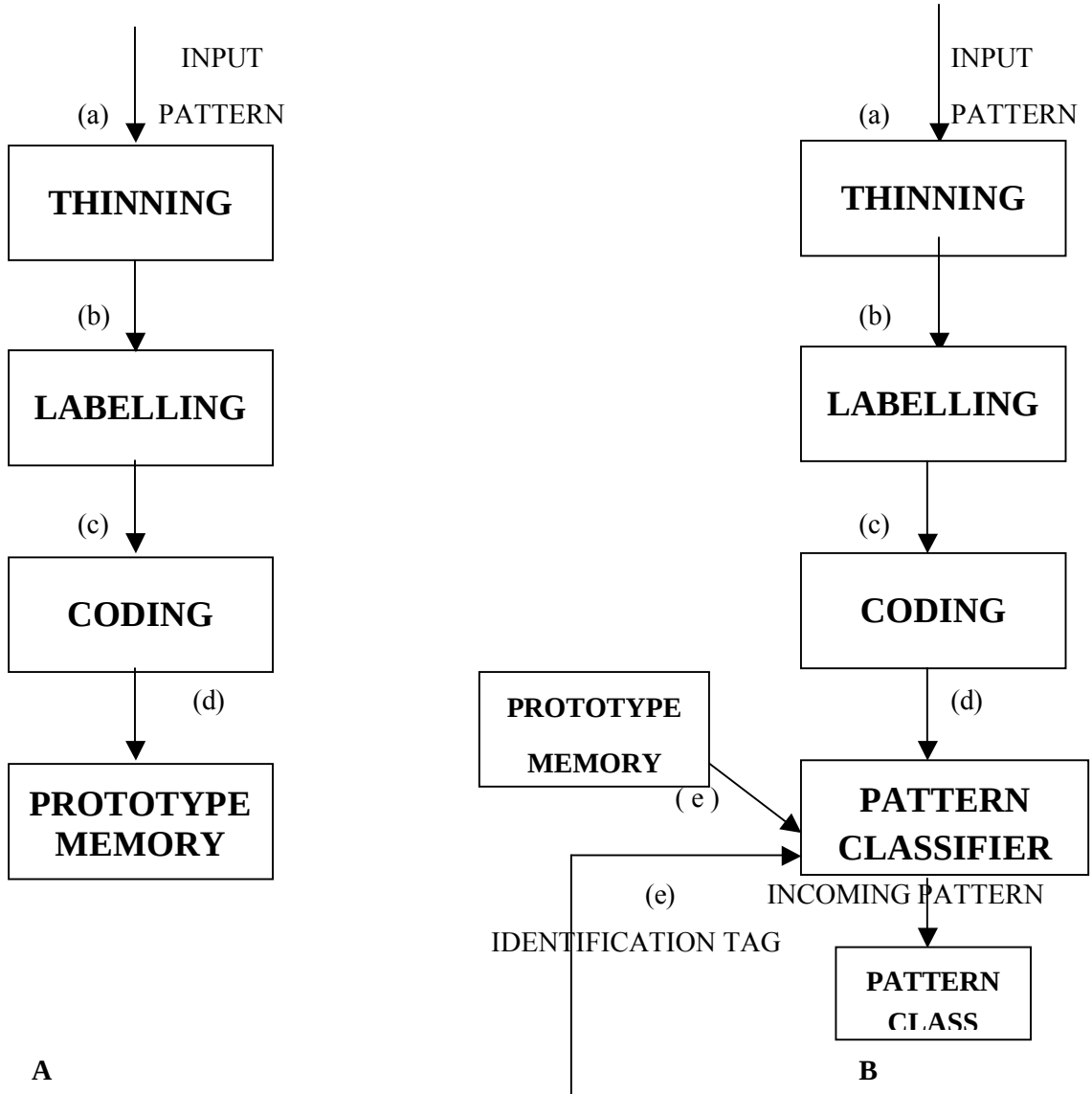


Fig. 3 BLOCK DIAGRAM OF RECOGNITION SYSTEM

A. LEARNING PHASE

B. RECOGNITION PHASE

ii. *Branch Detection and Labeling*

A branch is a line segment connecting a pair of adjacent nodes. For the degenerate case of a line pattern without nodes, the pattern is considered as a circular branch. A branch of which the length is less than threshold S_B is considered extraneous and consequently removed by declaring its nodes equivalent.

In classification of branches two sources of fuzziness are attributed to the measure

1) Straightness
2) Orientation

Both the sources are present in non-circular branches and only second one is present in circular branches with nodes.

The *measure of straightness* is determined by fitting a straight line with the minimum least square errors, and the branch is represented by the line with the least square error.

Measure of Orientation If we know the classification of a branch to the class of the straight line, a portion of a circle or a circle with a node is known, then the measure of orientation can further characterize the branch.

System Operation And Experimentation Results

The operation of this system consists of two phases

a) Learning Phase
b) Recognition Phase

In the *Learning Phase* the prototypes of each pattern class are learned and stored in the prototype memory. The code of the first pattern is stored in the prototype memory and starts a running code count of one, which means there is one patterns classified to this code. If the succeeding code of a pattern is of the same form as the previous one, the code count is incremented by one, while if the code is different, it is stored in the prototype memory and starts a new code count.

This process is repeated for the code of all the incoming patterns and they are accordingly allocated. Upon the end of the learning process, the memory contains all the prototype codes and their corresponding code counts. The code count represents the frequency of the occurrence of each prototype and therefore its relative importance compared to other prototype code. This can be utilized to speed up the recognition time by arranging the prototype code according to their code counts.

In the *Recognition Phase* the learned prototype are used to classify the unknown incoming patterns to the class of the matching prototype. The patterns code in the recognition phase in an actual system does not include any identification tag.

However we need it for generating misclassification statistics. Each input pattern is thinned, labeled and coded. Then the pattern classifier searches for the prototype that matches the pattern code. Three results can be obtained classification, misclassification and unclassification.

Classification occurs if only one of the prototypes is accepted and its identification tag matches that of the incoming pattern.

Misclassification occurs when only one prototype is selected and its identification is different from that of the incoming pattern.

Unclassification occurs when no prototype is selected.

Results

The % of correct classification was 98.4 % and processing time was 5 min 14.32 sec on IBM 370 computer.

1.3 OUR METHODOLOGY

We aim to develop a system that is able to recognize a typewritten/handwritten single character or a digit. The purpose is to produce a system, which classifies a given input as belonging to a certain class rather than to identify them as a fixed input pattern. Later on we can extend this approach for words to entire pages when combined with the segmentation techniques. We quantify the character to be recognized into a mathematical vector entity using the geometrical properties of the character. Then the vectors of the unknown character are compared with the ones which we have considered to be an ideal character. So the unknown character is recognized as one of these characters. ***Our scope is limited to the recognition of a single character or digit.***

1.3.1 PREPROCESSING/IMAGE EXTRACTION

I. During this process we scan the handwritten or printed character from a sheet of paper with the help of a scanning device into graphic/image file format (BMP). As mentioned earlier, the computer images are made up of pixels, which have a specific color associated with each of them. This is identified by the **pixel's value.** These pixels are arranged in the form of a matrix of horizontal rows and columns. Once the image file is scanned, the information about the number of rows and columns in the image, pixel offset, size of image are contained therein collectively called as the **image header**. When we are dealing with a 24-bit bitmap the header is stored in the first 54 bytes as explained in the header file (bmp.h) below.

```
                                        /*BMP.H*/

    typedef unsigned char bmpbyte;
    typedef unsigned short bmpshort;
    typedef unsigned long bmplong;

    struct BmpHeader { /* start of bitmapfileheader */
```

```
    bmpbyte magic1,magic2;
    bmplong filesize;      /*size of the BMP file*/
    bmpshort res1,res2;
    bmplong pixeloffset;   /* Indicates the portion in the file containing the palette
                              and the  header. This is a reliable method of finding
                              from where does the  actual bitmap starts*/
    bmplong bmisize;  /*start of bitmapinfoheader*/
    bmplong cols;      /*Width of the bitmap in pixels*/
    bmplong rows;     /*Height of the bitmap in pixels*
    bmpshort planes;  /*No. of planes in the pixel must be 1*/
    bmpshort bitsperpixel;  /*Indicates the number of bits per pixel */
    bmplong compression;   /*Flag to indicate the type of compression*/
    bmplong cmpsize;  /* size of compressed image */
    bmplong xscale;   /* pixels per meter  horizontally*/
    bmplong yscale;   /* pixels per meter  vertically*/
    bmplong colors;   /* number of colors used */
    bmplong impcolors;  /* number of important/prominent colors */
    /* color map starts here */
};
typedef struct BmpHeader BMPHEADER;
#define  TRUE    1
#define  FALSE   0
```

Each image format stores the pixels and their positions in a specific manner. The 24-bit bitmap uses 3 bytes to store the color (pixel value) associated with each pixel. In BMP files the preprocessing starts from bottom to the top, hence, the color of each pixel is stored starting from the bottom and moving right hand upwards. Once we extract the pixel values associated with each pixel of the image, their decimal equivalent values are stored in a matrix at the corresponding row-column position. The size of this matrix is equal to the (number of rows * no. of columns(of the image)).

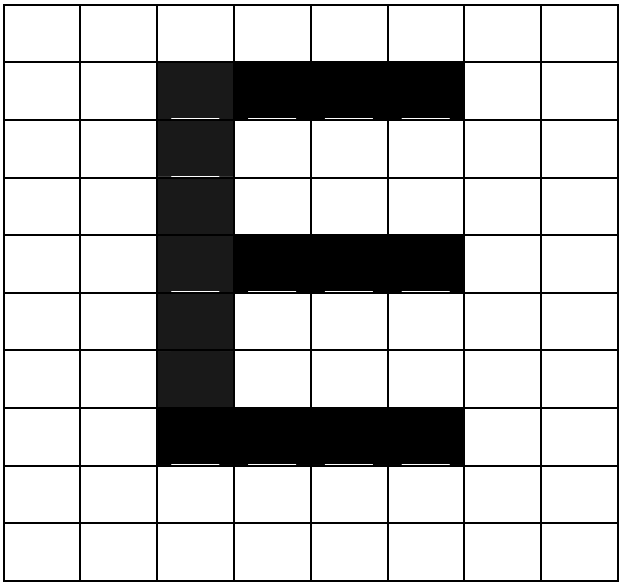


FIG 4: EXTRACTION OF A CHARACTER IN A MATRIX

The extraction of the image "E" into a corresponding matrix is shown above. here w= Numerical value corresponding to white and b= Numerical value corresponding to black. From this matrix we find the element which occurs the most. This will correspond to the most prominent color,

W	W	W	W	W	W	W	W
W	W	B	B	B	B	W	W
W	W	B	W	W	W	W	W
W	W	B	W	W	W	W	W
W	W	B	B	B	B	W	W
W	W	B	W	W	W	W	W
W	W	B	W	W	W	W	W
W	W	B	B	B	B	W	W
W	W	W	W	W	W	W	W
W	W	W	W	W	W	W	W

FIG 5: IDENTIFICATION OF BACKGROUND AND CHARACTER.

and hence the background in our case . This is very obvious as the character will be less prominent than the background of the paper on which it has been written or printed.

Therefore the number of pixels constituting the background will be less than the number of pixels that make up the character. Hence our assumption is that the numerical values other than the background constitute the character.

Our next step is to store the pixels belonging to the character in a table as shown on the next page. This table is called as the initial X-Y table of the unknown character. The number of rows in the table is equal to the number of pixels of the image belonging to the character. This table will differ even for two exactly identical characters at two different positions in the image. In order to make our approach independent of the initial position of the character (on the paper), we consider a suitable logical shift of the scanned character by applying translation on the data in the initial X-Y table, which is discussed in the ensuing section.

To summarize the progress, till now we have scanned the unknown character and formed the corresponding initial x-y table.

COLUMN	ROW
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
.	.
.	.
.	.
.	.
.	.
x_n	y_n

FIG 6 : INITIAL X-Y TABLE

1.3.2 FINDING THE CENTER OF THE CHARACTER

Next we find the center of the character (Xc,Yc) using its initial X-Y Table by the formula

$$Xc = \sum X_i / N$$

$$Yc = \sum Y_i / N;$$

Where N is the number of rows in the initial X-Y table.

Here Xc, is the column position and Yc is the row position of the center of the image. Let us assume that the center of our screen is (Xs, Ys) . We call this the **center of reference.** Now we shift the scanned character logically to this center of reference using the following translation on the initial X-Y table .

$$X \leftarrow X+h; \text{ where } h=Xs- Xc ;$$

$$Y \leftarrow Y+k; \text{ where } k=Ys- Yc ;$$

Hence we get our final **X-Y** Table which is independent of the initial positioning of the character.

COLUMN	ROW
x_1+h	y_1+k
x_2+h	y_2+k
x_3+h	y_3+k
x_4+h	y_4+k
.	.
.	.
.	.
.	.
.	.
x_n+h	y_n+k

FIG 7 : X-Y TABLE

1.3.3 EXTRACTION OF DATA

According to our approach every character has a unique set of geometrical properties associated with it, which differentiates it from the rest of the alphabets. One particular characteristic may not be sufficient to enable recognition but the set as a whole would make every alphabet unique. Larger is the cardinality/ size of this set, higher is the recognition . Now we define certain geometric properties and associate each character with a set containing these characters. Thus, we quantify a character in the form of set. We call this a vector rather as the order of the elements is significant. This set is called as the **characteristic vector** of the character. In addition to this we

create a database where we store the characteristic vectors corresponding to figures, which we consider to be ideal alphabets.

Next we design a method to compare and find the closest match of the characteristic vector from the database. The alphabet that is found to be the closest match is considered to be the recognized character for the unknown character.

Quantification of Character

The character is quantified in the form of a characteristic vector. The **final** X-Y table is used to find the elements of the characteristic vector. The geometric characteristic/properties of the character are defined to be the distances of the pixels present on the character at various direction from the center of the screen which also happens to be the center of the character as we have shifted the image to the center. These distances are measured from the center along the directions which are defined as angles

$$\theta = 0, 2\alpha, 3\alpha \ldots k\alpha, (n-1)\alpha \text{ ; where n is a divisor of 360}$$

Thus, we get 'n' number of distances $X_1, X_2, X_3, X_4 \ldots\ldots, X_n$ called the **characteristic values**. Hence the characteristic vector of the character can be written as $\{ X_1, X_2, X_3, X_4 \ldots\ldots, X_n \}$

Creation Of Square Frame

A hypothetical square frame is created around the image such that the center of the square coincides with the center of reference and the entire character fits into it. Hence we calculate

$$X_O = \max_i | X_s - X_i |$$

$$Y_O = \max_i | Y_s - Y_i |$$

where (X_s, Y_s) is the center of reference and (X_i, Y_i) are the entries in the **final X-Y** table of the character. Suppose that $d = \max \{ X_O, Y_O \}$. Thus we can conclude that the side of the square must be >= 2d

Calculation Of Characteristic Values And Formation Of The Characteristic Vector

To find the characteristic vector along a certain direction, a polar line $\theta = \alpha$ is drawn to find the point at which it intersects the frame. From this point the traversing is done backwards along the line towards the center of the reference character till we get to the first pixel of the character. This

is accomplished by identifying the pixel of the character which is at the farthest distance from the center along this line θ= α. Next from this point we find the distance of the pixel from the center , which will signify the characteristic value corresponding to the polar direction θ= α. In case no pixel is traced and we reach the centre, this characteristic value is set to 0. In the similar way we find characteristic values in other directions for different θ and hence the characteristic vector.

S= { X_1,X_2, X_3,X_4..... ,X_n }. See the figure below

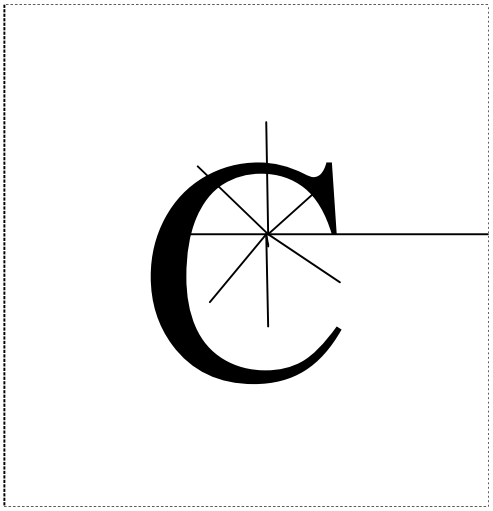


FIG 8 : CHARACTERISTIC VECTORS MEASURED FROM THE CENTER 'O'

Creation Of An Ideal Database

Our system takes into consideration English alphabets of some standard fonts (i.e. Arial, Times New Roman etc.) of a suitable fixed size .A database of their characteristic vectors is formed called as the **Ideal Database**. We are storing fonts of only one size as our system is independent of the size of the alphabet to be recognized. Thereby, making the size of database considerably smaller.

Character Recognition

At this point we have the ideal database with us. Now we are implementing the comparison routine for comparing the characteristic vector of the unknown character with each of the vectors in an ideal database. The alphabet showing the closest match to the unknown character in terms of characteristic vector is declared to be the recognized alphabet for the unknown character. Now we discuss the concept of magnification

Magnification

Whenever an image is magnified , we have a magnification ratio associated with it called as the scaling factor.

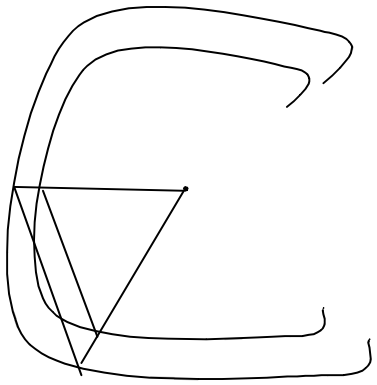


FIG 9: MAGNIFICATION OF CHARACTER 'C'

Here "O" is the centroid of both the figures. In the figure , we have $\Delta OAB \equiv \Delta ODE$,

Therefore , $OA/OE = OB/OD$

Therefore, in terms of the elements of the characteristic vectors of a character image and its magnified image, we have,

$$\text{Magnifying ratio(m)} = X_1/X_1' = X_2/X_2' = X_3/X_3' \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots X_n/X_n'$$

Where X_1 and X_1' are a particular characteristic value of the smaller and the larger character images of the same character.

Comparison Of The Characteristic Vector And Identification Of The Right Match

We define a null index set. Null index set of a character , is the subset of J of N= {1,2,3,……..N} such that $\forall j \in J$,

$X_i = 0$ in the characteristic vector of the character. If β is the character , we denote the null set by N_β. Suppose that the characteristic vector of the unknown character is S and that of English alphabet is D_i in database D are given by

$$S = \{ X_1, X_2, X_3, X_4 \ldots\ldots, X_n \}$$

and

$D_i = \{ d_1, d_2, d_3, d_4 \ldots\ldots, d_n \}$

we construct a set for this pair of characters given by

$$\{ y_1, y_2, y_3, y_4 \ldots\ldots y_n \}$$

where

$$y_i = d_i / x_i \text{ ,if } x_i \neq 0$$

$$y_i = \varnothing, \text{ if } x_i = 0$$

where we consider $\varnothing$ being an **ignorable symbol**.

This set is called as the Initial Variance set , S_{Di} w.r.t D_i.

Final Variance set or simply the variance set: The collection of all non-ignorable elements of initial variance set is called the variance set of the unknown character w.r.t D_i

In our comparison here, we compare the unknown character β with those ideal characters only for each of which the null set index is the superset of N_β

The collection this collection of characteristic vectors of the ideal database as the eligible database denoted by D_β. The collection of the ideal characters is called as eligible database denoted by E_β , Now for each member of $E_i \in E_\beta$, we find out the corresponding variance set SE_i

It is very obvious that the unknown character will be recognized as a character belonging to E_β only. Hence our next work is to find the right match of the unknown character from E_β.

We calculate the variance sets SE_1, SE_2, $SE_3 \ldots$ SE_n where n equal the no. of alphabets in the eligible database. The cardinality of each of these sets is same(8 elements).

When we compare the unknown character with an a character in the ideal database by finding the variance set of the two we assume that at least one of the character is the exact magnification of the unknown character. In that case we must have all the elements in the variance set equal.

i.e.

$SE_i = \{ m,m,m\ldots\ldots,m\}$

In reality it is unlikely that for a handwritten or even a printed character we get an exact magnification relationship with an ideal character. Hence the values of m will show some variance/deviation.

If the values of m show a lot of variance it means that ideal character has very less similarity with the unknown character. So we calculate variance for the elements $SE_i = 1,2,3\ldots k$

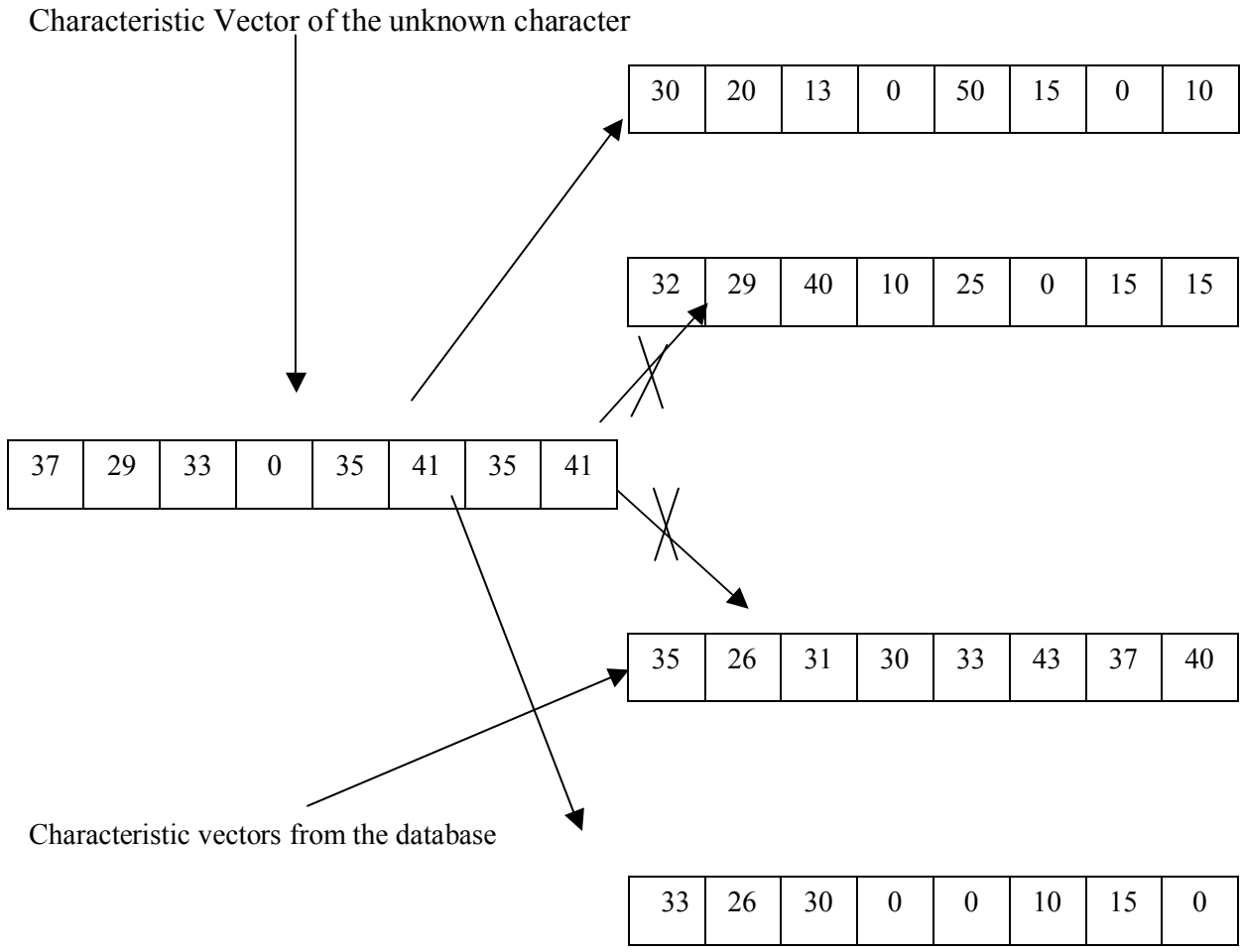


FIG 10: THE 2ND AND 3RD CHARACTERISTIC VECTORS ARE NOT CONSIDERED FOR COMPARISON

Suppose that variance of the elements of the set SEi = v_i, for all i

If v_i = min { v_i } , then we declare the unknown character to be recognized as E_i

2. SOFTWARE REQUIREMENT ANALYSIS

2.1 INTRODUCTION (PROBLEM DISCUSSION AND ANALYSIS)

2.1.1 PURPOSE

The purpose of this document is to produce a report detailing the requirements for the **"Character Recognition System Using Template Matching"**. It describes the function and the constraints that will govern its development.

It lays down the software requirement of **"Character Recognition System Using Template Matching"**. It also helps the user to gain an insight into the system. The SRS is an outcome of the discussion with the user and will serve the following purposes

- ✓ It completely states the functionality of the system in unambiguous terms.
- ✓ It will act as specification of the proposed system in terms of output, input and processes.
- ✓ It will serve as a basis for subsequent design, development between the client and the developer.

2.1.2 GOALS AND OBJECTIVES

The goals and objectives of the proposed software is

"Developing a system that helps in recognizing an unknown character presented to it. The technique is synonymous with the recognition algorithm used to recognize character embedded in various word processors, scanners and other commercial products."

2.2 GENERAL DESCRIPTION

2.2.1 SCOPE AND CONSTRAINTS OF THE SYSTEM

- ✓ The system is meant only to recognize a single character of capital font or legible handwritten ones.
- ✓ For the sake of simplicity only BMP file is used to store the character to be recognized
- ✓ It is limited to the DOS environment i.e. it cannot be executed in VC++
- ✓ Only compressed files can be considered for recognition.
- ✓ Noisy and distorted character cannot be considered for recognition.

2.2.2 GENERAL CONSTRAINTS

- ✓ HARDWARE

- MINIMUN 32 MB RAM

✓ SOFTWARE

- TC 2.0 ONWARDS

2.3 REQUIREMENTS

1) Routine to read a BMP file.
2) Function to find the templates(vectors) of each character presented to the system.
3) Function to calculate the variance with respect to all the vectors stored for the character to be recognized.
4) Master Database file containing all the entries of characteristic vectors of some standard fonts.
5) Routine to match the variance and separate the closest match from the database.

3. SYSTEM DESIGN

System design is the first step in software development, which needs careful and intricate planning .It helps us to prepare detailed technical design of the application based system .It is based on Requirement Analysis. *It provides the specification and design for system design showing flow of work, program and user functions*

3.0 DESIGN OBJECTIVES

The goals that were kept in mind while designing the system are :

- ✓ To make the system user friendly as much as possible.
- ✓ To make the flow of program comprehensible to the user easily.
- ✓ To have transparency in work i.e. to show what all is being done by the system stepwise.

3.1 DESIGN DECISION

The system has been developed using C and C++ as a programming tool.

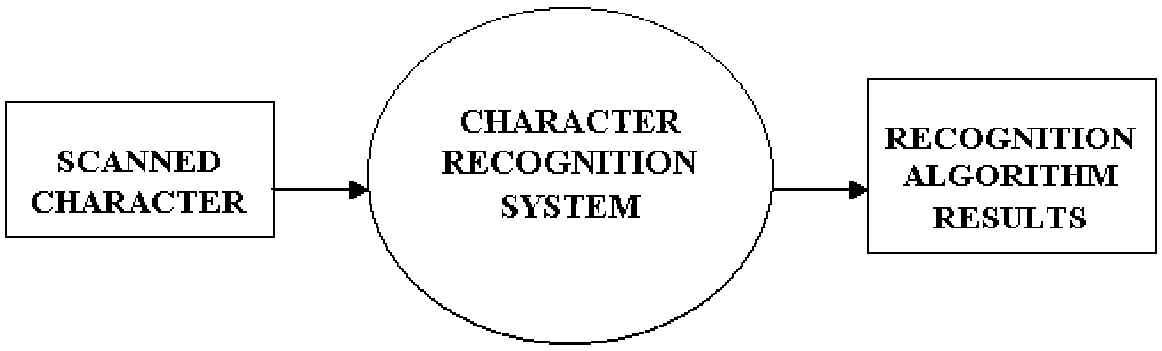
3.2 DATA FLOW DIAGRAM

The DFD serves two purposes

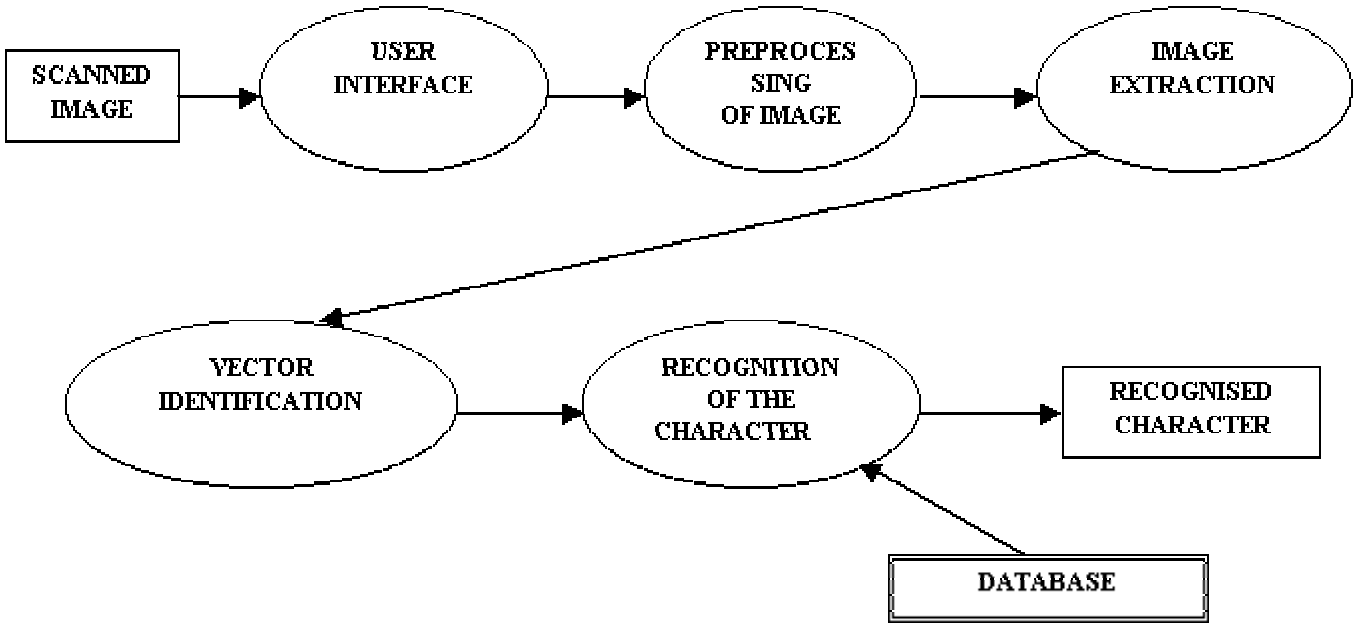
- ✓ To provide an indication of how data are transformed as they move through the system.
- ✓ To depict the function and sub-functions that transforms the data.

They serve as basis for the functional as well as information flow modeling.

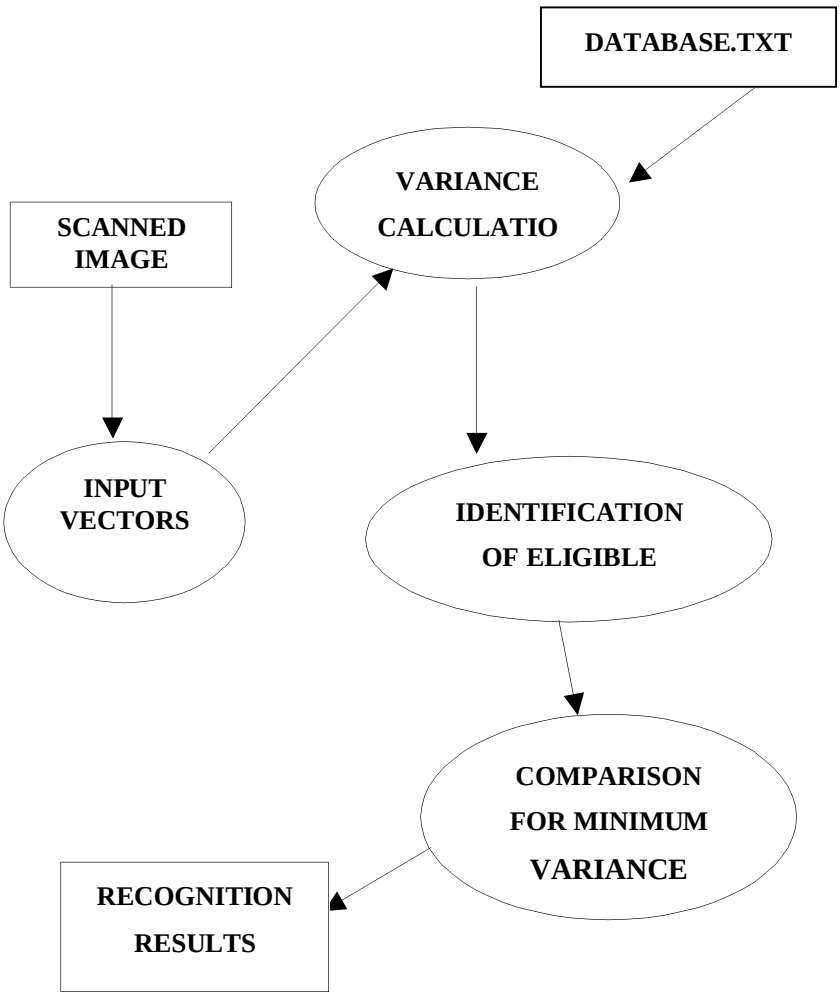
3.2.1 CONTEXT LEVEL DFD FOR THE SYSTEM



3.2.2 1ST LEVEL DFD FOR THE SYSTEM



3.2.3 2ND LEVEL DFD FOR THE SYSTEM



3.2.4 PSPEC for 1ST level DFD

PSPEC:**User Interface**

User interacts with the system to choose among the input BMP files to be recognized

PSPEC: Preprocessing Of Image

The BMP file has four different components namely image header, file header, color palette and the actual bitmap which stores the image. In preprocessing all these four components are read using BMP routines, a validation is done to check the image type as BMP and also check for compressed and uncompressed file format. If the condition comes to be a compressed BMP file the procedure for image extraction is called otherwise the program generates an error message.

PSPEC: Image Extraction

Image extraction is done as the program reads each of the pixels belonging to the image. A pixel has got a color value and a unique x-y coordinate position on the screen. A 2-D array is used to initially store the color value for each pixel and its number of occurrences in the image . It is also assumed that the color having the maximum number of pixels constitutes the background of the character. Thus the remaining pixel value and their position are taken into account as the pixel belonging to the character only. This is stored in a matrix of dimension equal to the number of pixels in a character. This matrix or an array is used to extract the image .The positions of the pixels belonging to the character are stored in it.

PSPEC: Vectors

By means of geometric properties the center of the character is found and from this direction we find the distance in terms of number pixels in specific direction starting from 0˚, 45˚ , 90˚ , 135˚,180˚,225˚,270˚ , 315˚ and 360˚

PSPEC: Recognition

The recognition technique adopted here is minimum variance (degree of deviation from the actual set) with respect to every character set stored in a master database file. The variance of an unknown character is calculated using its vectors set and are compared with the variances of the character set in the master database. The character corresponding to minimum variance is considered to be the closest match of the unknown character.

3.3 ARCHITECTURAL DESIGN

Architectural design represents the data structure and program components that are required to build the computer based system. It considers the structure and properties of the components that constitute the system and relationship that occur between among all Architectural components of the system.

It begins with the data design and then proceeds to the derivation of one or more representations of the architectural structure of the system. Alternative architectural styles or pattern are analyzed to derive the structure that is best suited to the requirements and quality attributes. Once an alternative has been selected, the architecture is elaborated using architectural design method.

3.3.1 Software Components

It can be referred to as a simple program module or can even be extended to databases and middleware that enable the configuration of clients and servers.

3.3.2 Properties of Components

They are the characteristics that are necessary for understanding the components that interact with other components. At the architectural level, internal details of algorithms are not specified. The relationship between components can be as simple as procedure call from one module to another or as complex as a database access protocol.

3.3.3 Summary

Architectural design focuses on the representations of the software components , their properties and interactions.

Architectural Design For Our System

Refer Figure 11 on ensuing page.

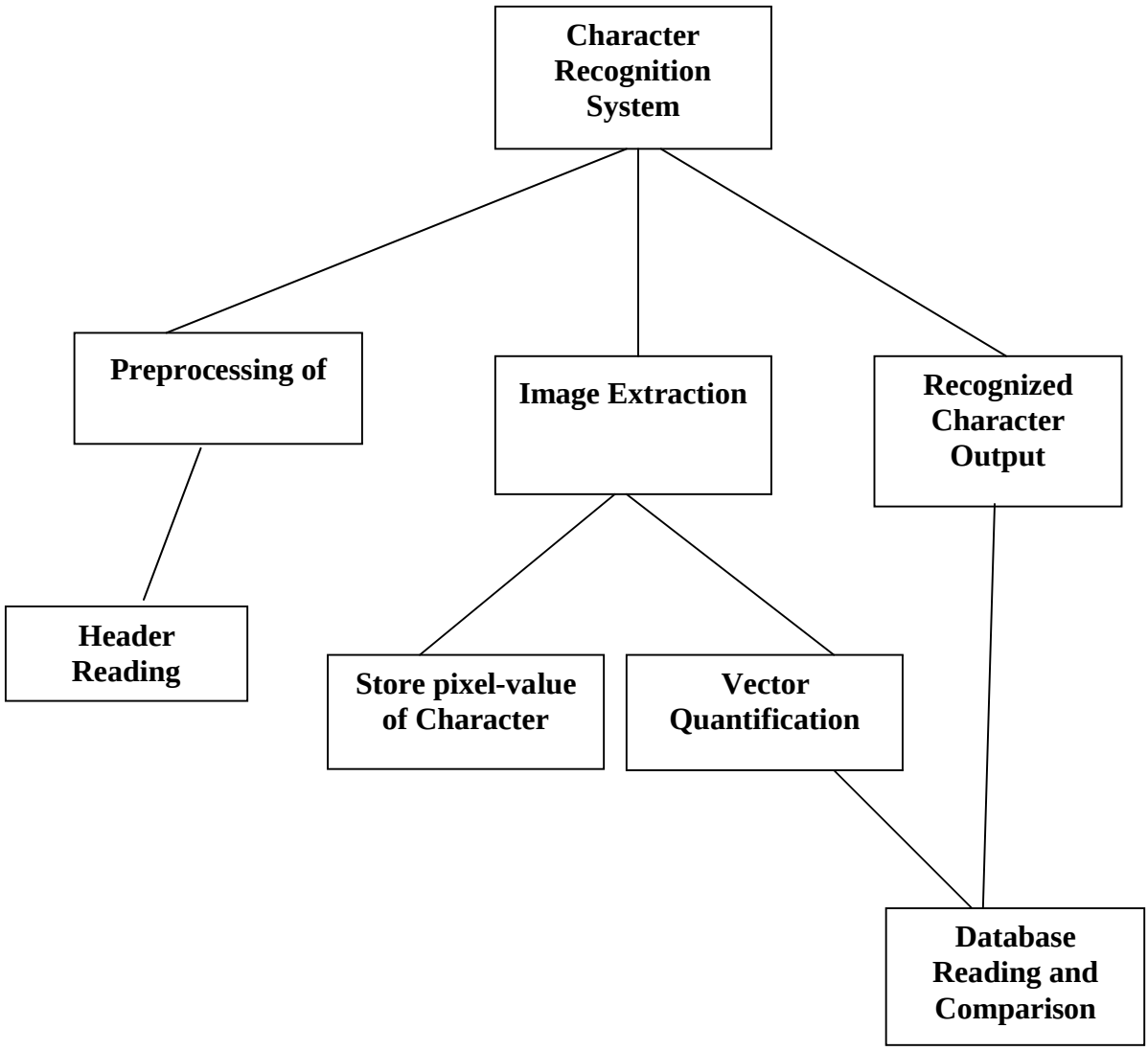


FIG 11 ARCHITECTURAL DESIGN

3.4 IMPLEMENTATION

We discuss ,the details of implementation of our methodology. We exemplify it with the help of digit '4' explaining the implementation algorithm executed in 'C /C++' language.

We consider the implementation step by step.

3.4.1 CREATION OF DATABASE

Initially a database is created for twenty-six English alphabets of 4 standard fonts. Similarly we create a database for 10 digits of these standard fonts. These are alphabets/digits that we consider to be ideal ones. The alphabets/digits are stored in order starting from first font saving vectors for all the twenty-six alphabets and then starting it with the new font. There is no limit to the fonts that can be stored but there exists a trade off between the size of database and accuracy. Greater is

the number of fonts higher the efficiency of the system. The steps involved are explained one by one

A. Saving the image

A BMP image containing the desired character to be recognized is saved.

B. Image Calling

Then this image is called by the C program and opened in the program window as shown in figure no 13

C. BMP Header Reading

Next we read the bitmap image byte by byte. The first 54 bytes constitute the header as explained in section no 1.3.1 . This helps us to calculate the number of rows and columns in the image.

D. Image Reading

The 55^{th} byte onwards constitutes the information pertaining to the bitmap. The pixel reading is done in chunks of 3 bytes each .Each 3 byte value is converted into a corresponding numerical value which signifies the pixel value for that position. As we know BMP reading proceeds from bottom up ,hence, the first three bytes tell the pixel value corresponding to right most pixel at the bottom. This pixel value is stored in a 2D matrix whose size equals the product of number of rows and columns in the image.

E. Identification of dominant color

Next we identify the most frequently occurring color in the image which constitutes the background as explained in section no. 1.3.1 with the help of figures 4 & 5

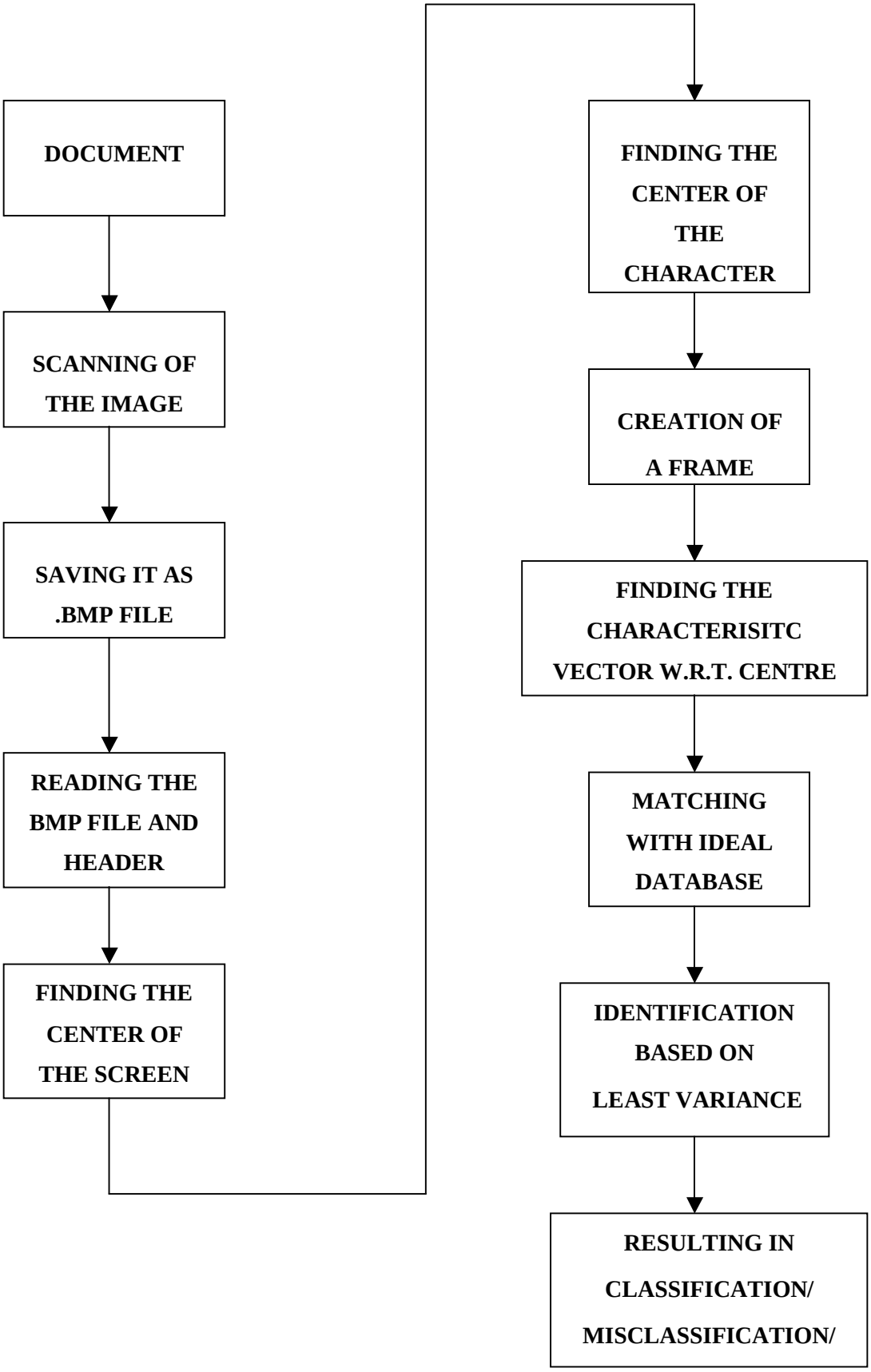


FIG 12 PROGRAM FLOW

F. Formation of Initial and Final X-Y table

With the information gained above we form an initial X-Y table (Fig 6) which contains the pixels corresponding to the character. Next we move the character to the center of the screen as explained in 1.3.1 and compute the final X-Y table(Fig 7).

G. Quantification of Character

We quantify the character by finding the pixel positions in 8 directions starting with θ =0˚, 45˚ **,** 90˚ , 135˚,180˚,225˚,270˚ , 315˚ and 360˚

There is not limit to the no. of vectors that we may use. For the sake of simplicity we are using 8 vectors. In this way we proceed and calculate the vectors for all the alphabets and digits for the fonts whose vectors we wish to store in the ideal database.

H. Creation of a database file

Next we store the pixel values/vectors in a text file called **database.txt** for alphabets and **dataNUM.txt** for digits. The vectors are stored in an ordered and systematic manner such that it keeps exact correspondence with the digits/alphabets viz. the 1st entry in datanum.txt corresponds to 1,11,21,31 belong to 0 and the 10,20,30th … one to 9 for the first, second, third….nth font respectively.

3.4.2 RECOGNITION OF A CHARACTER

During the recognition process we aim to find a match for the unknown character from the database created above. And lastly declare the closest match for the unknown character in the database for the as character recognized. We now discuss the steps involved one by one

A. The steps **A-H** discussed above are repeated for the unknown character. We store the vectors corresponding to one alphabet/digit only.

B. With this vector we eliminate the database for characters having ignorable symbols. There by reducing the efforts on searching considerably. This has already been explained in section 1.3.3.3.b and figure 10.The next step is to form the variance set with every entry of the relevant database.

C. Here we find the statistical variance of elements of these variance sets and determine the one with least variance. The expected recognized character is the corresponding alphabet/digit.

3.4.3 EXAMPLE

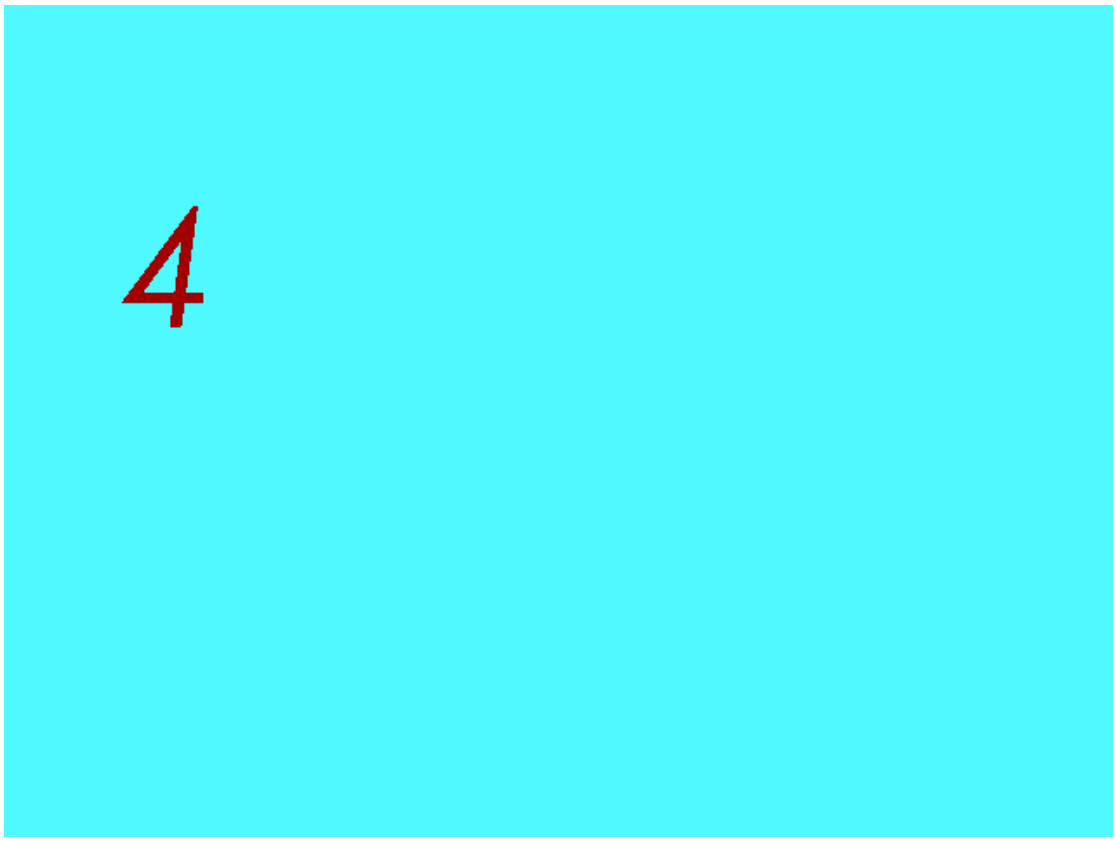


FIG 13: THE DIGIT 4 STORED IN THE MEMORY IS ACCESSED AND OPENED BY THE C ROUTINE IN A PROGRAM WINDOW.

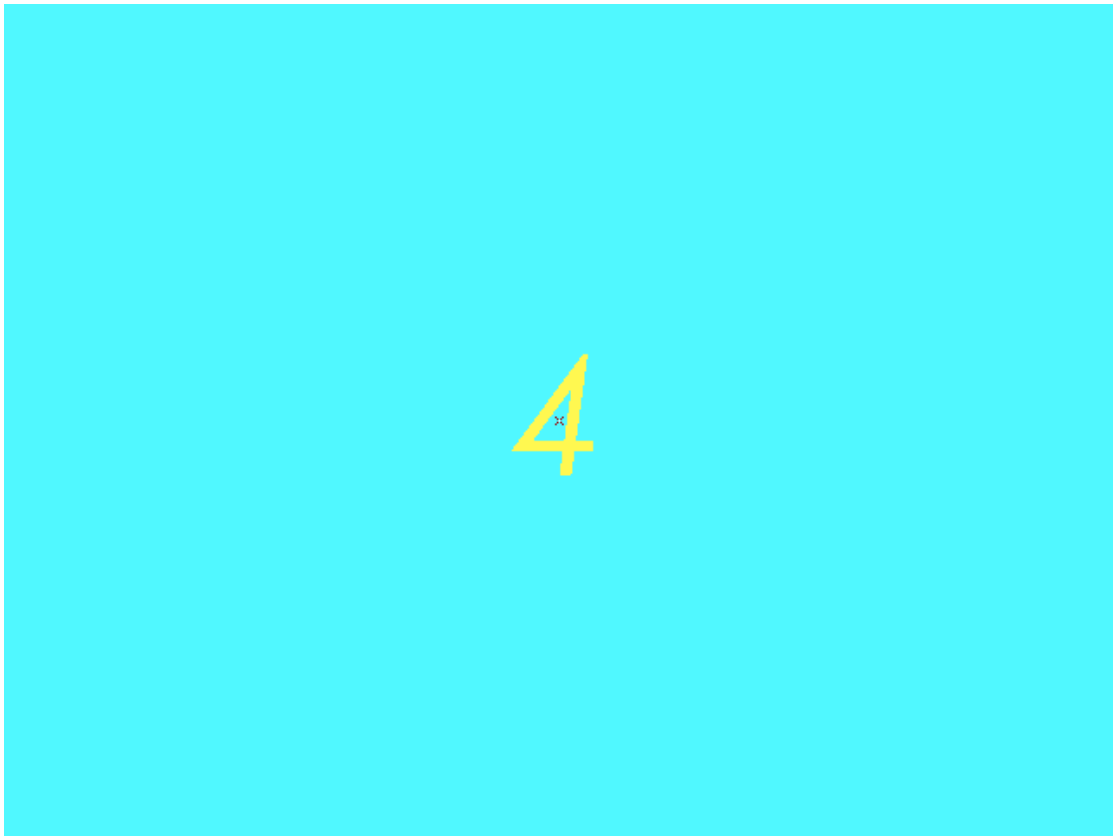


FIG 14: THE DIGIT IS LOGICALLY SHIFTED TO THE CENTER OF THE SCREEN.

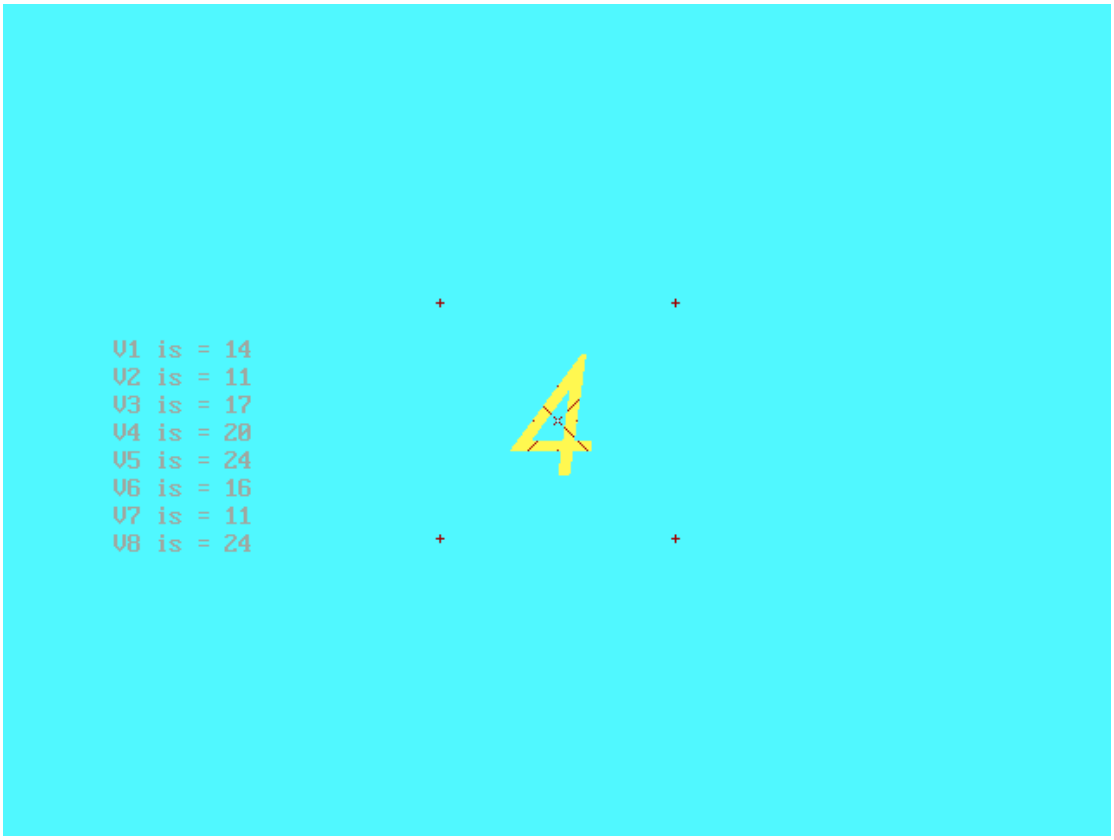


FIG 15: CREATION OF SQUARE FRAME AND QUANTIFICATION OF DIGIT IN THE FORM OF 8 VECTORS.

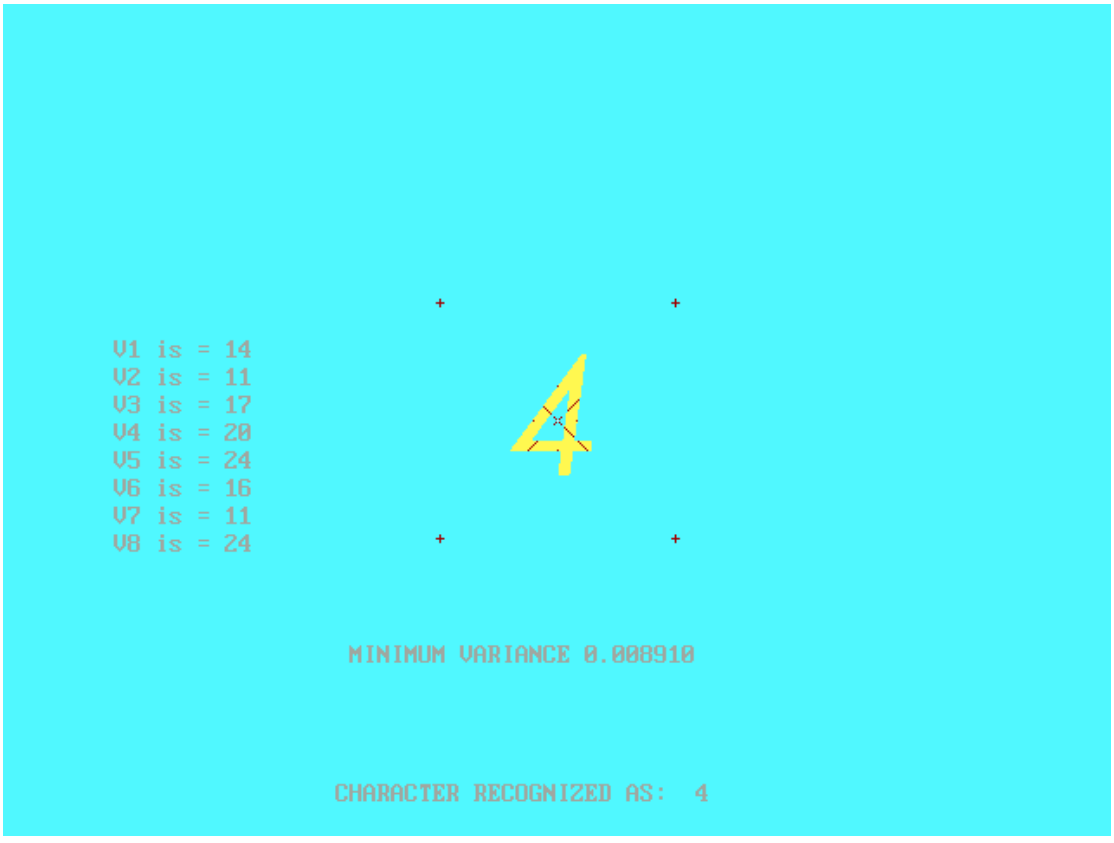


FIG 16 :CALCULATING VARIANCE AND DISPLAYING RECOGNITION RESULTS.

3.4.4 SOME SAMPLE RECOGNITION RESULTS

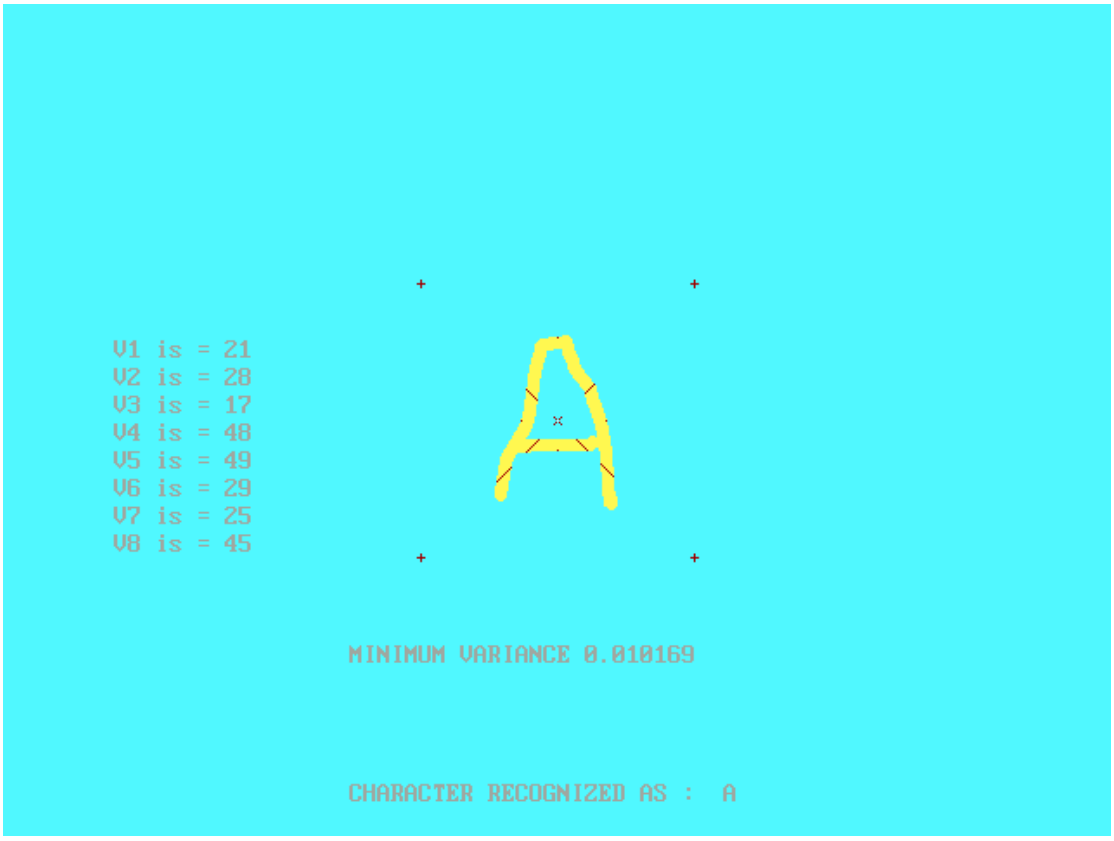


FIG 17 : SAMPLE OF HANDWRITTEN CHARACTER 'A'

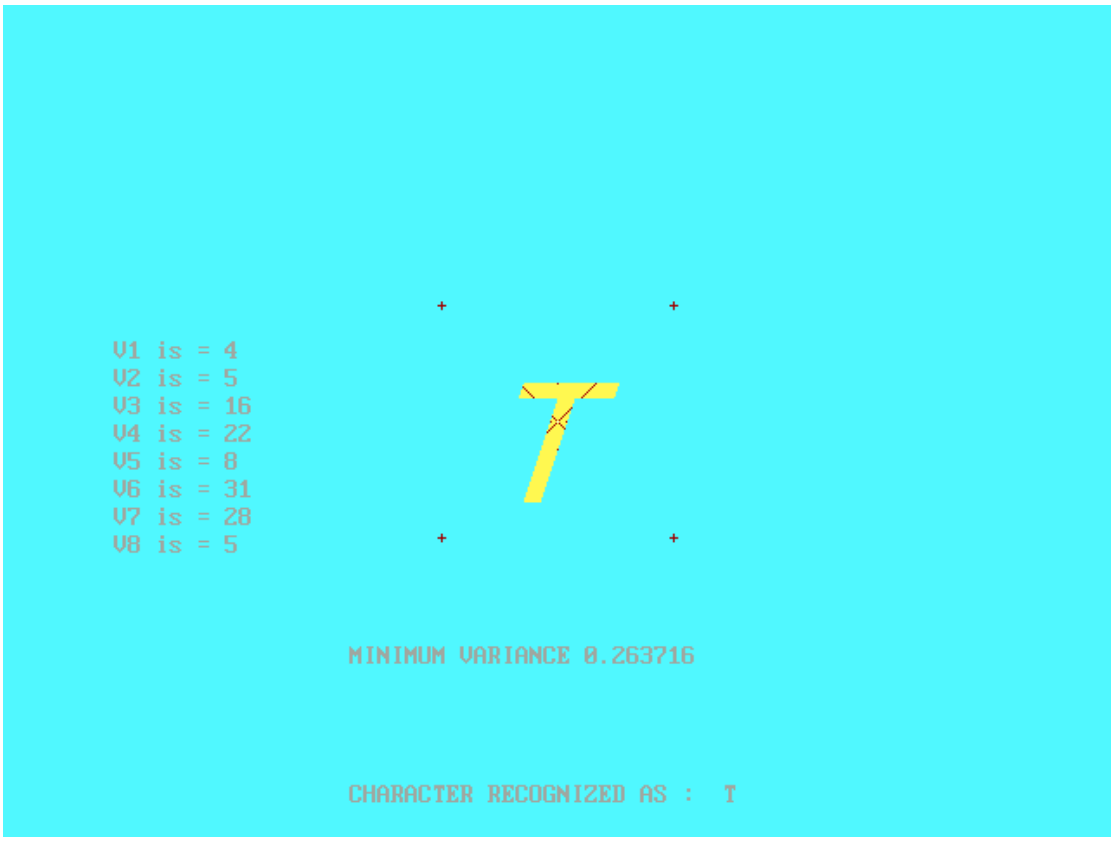


FIG 18 : SAMPLE OF AN ALPHABET IN ITALICS

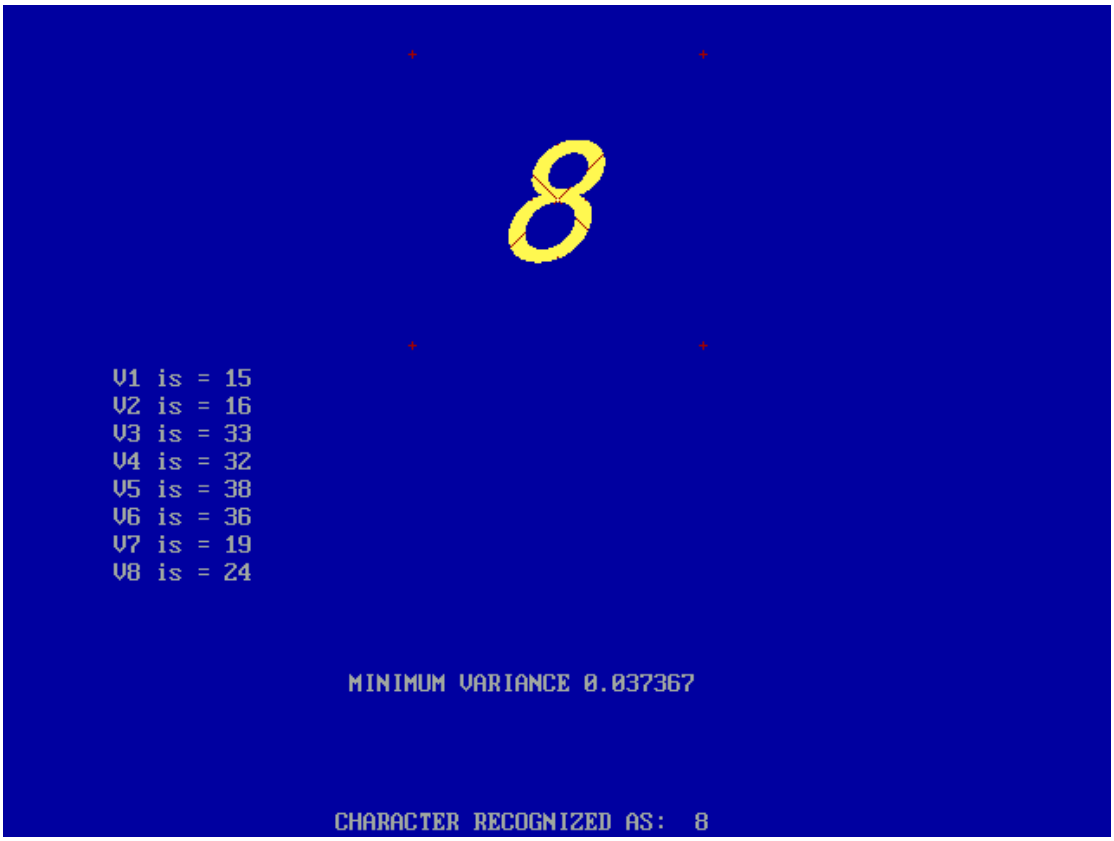


FIG 19 SAMPLE OF A DIGIT IN ITALICS

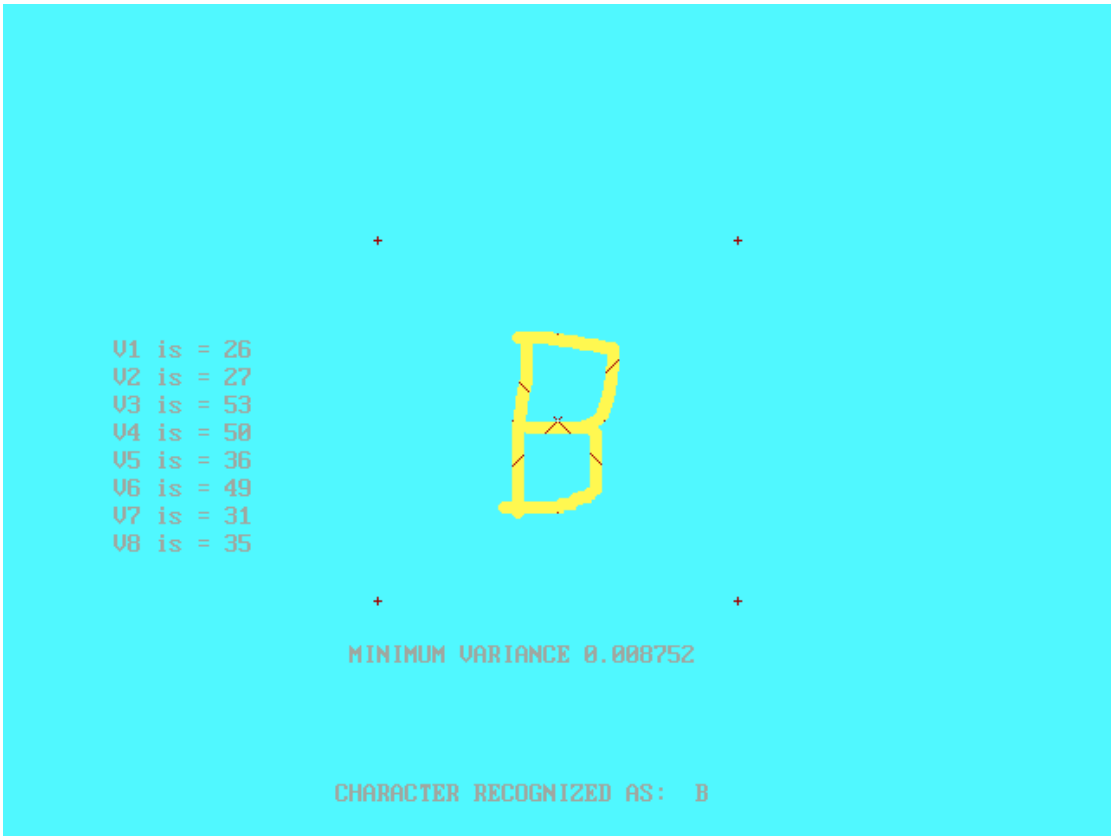


FIG 20 SAMPLE OF HANDWRITTEN ALPHABET 'B'

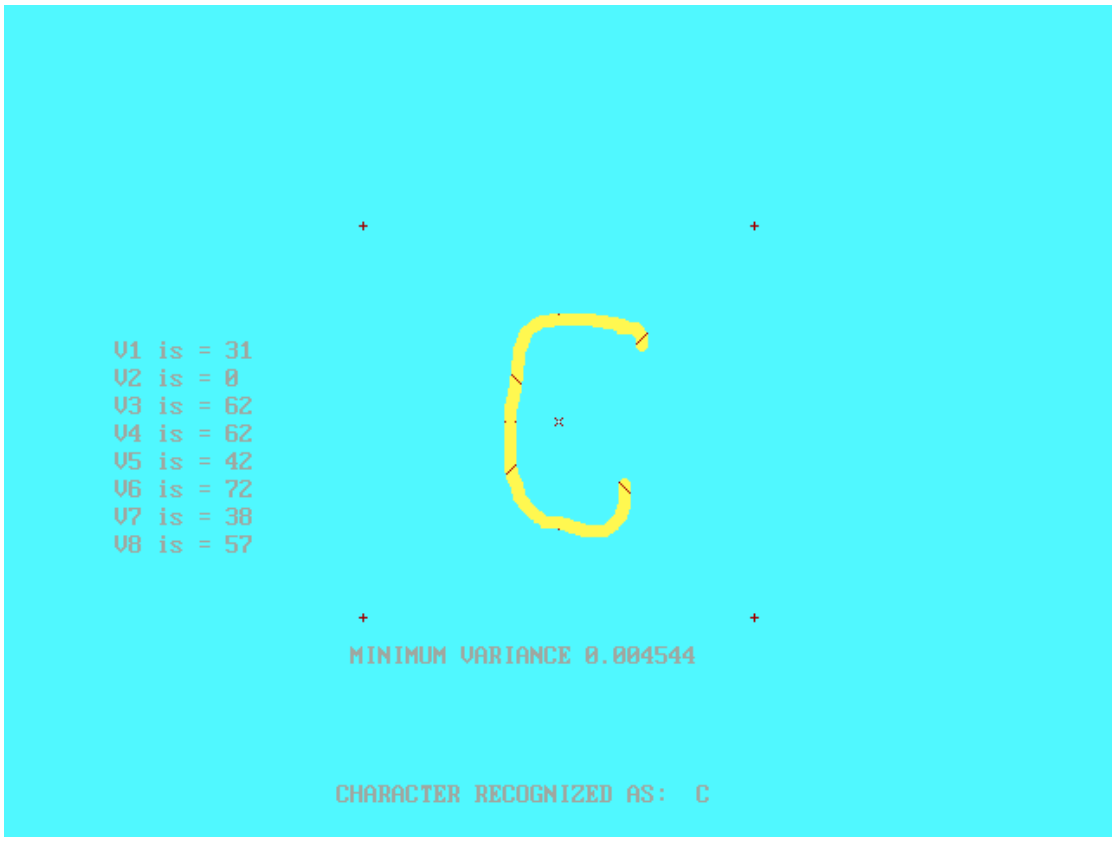


FIG 21 : SAMPLE OF HANDWRITTEN ALPHABET 'C'

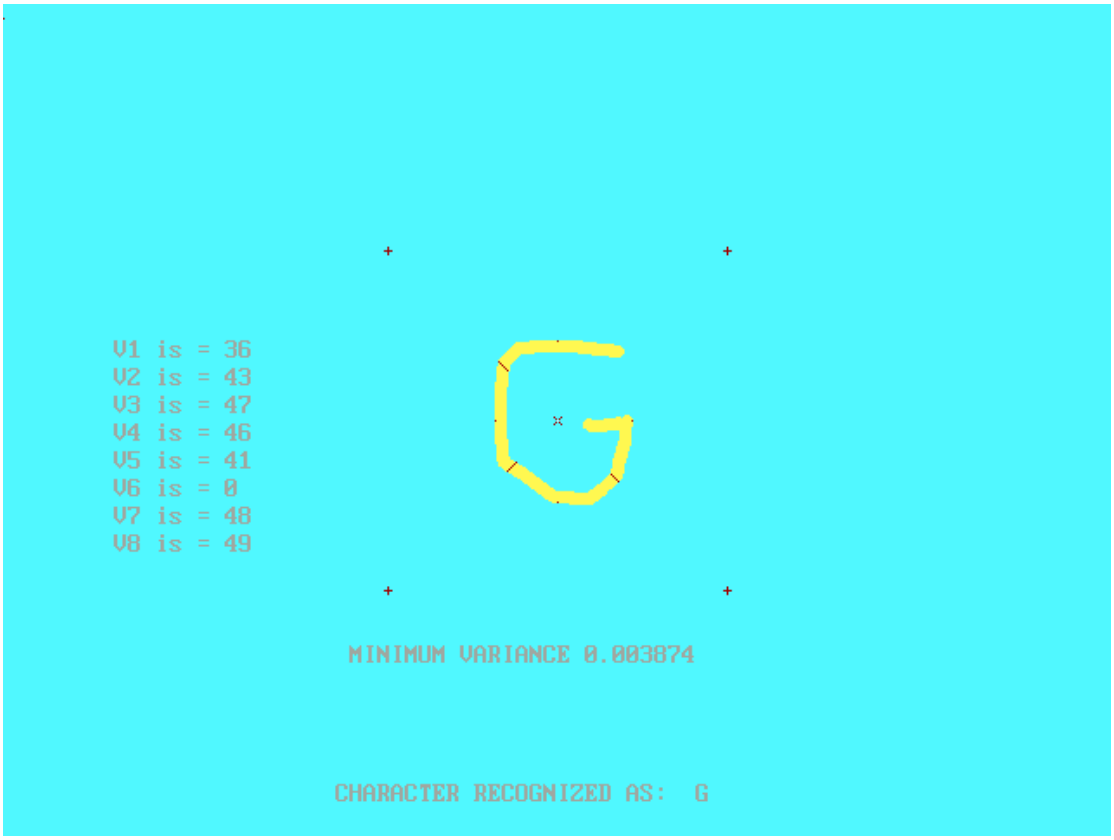


FIG 22 : SAMPLE OF HANDWRITTEN ALPHABET 'G'

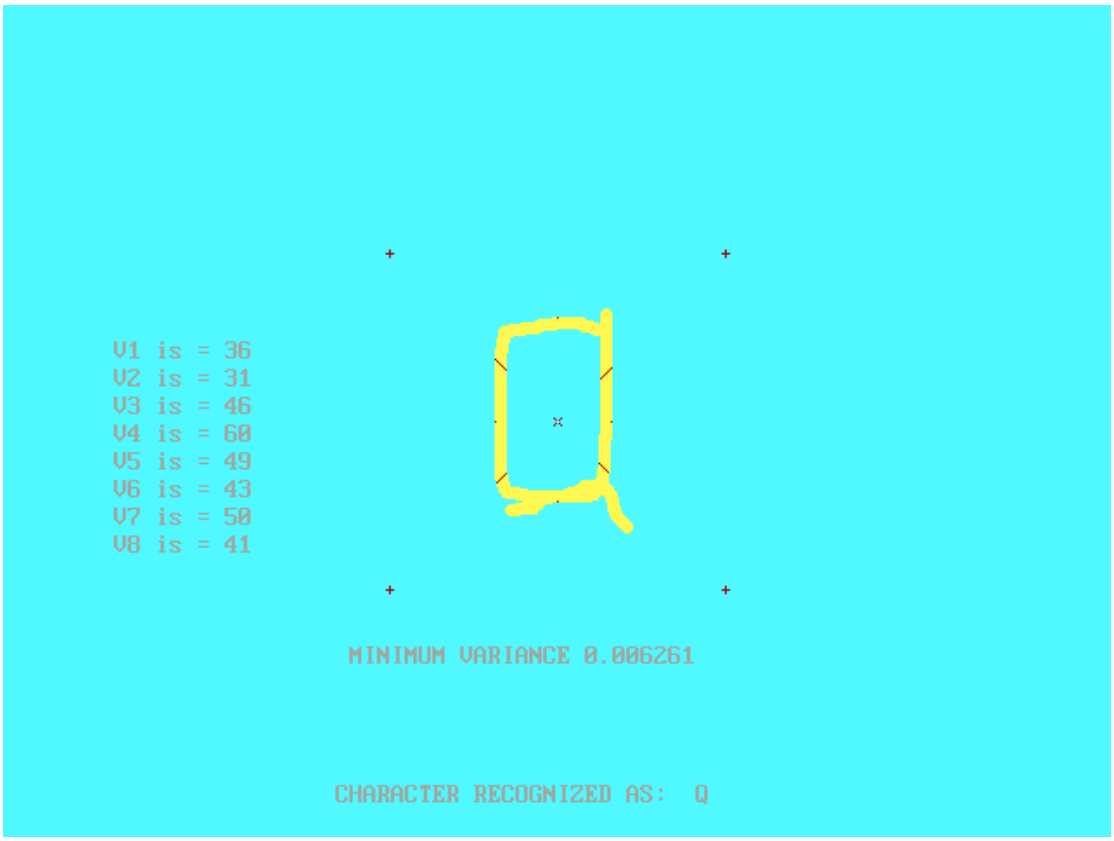


FIG 23 : SAMPLE OF HANDWRITTEN ALPHABET 'Q'

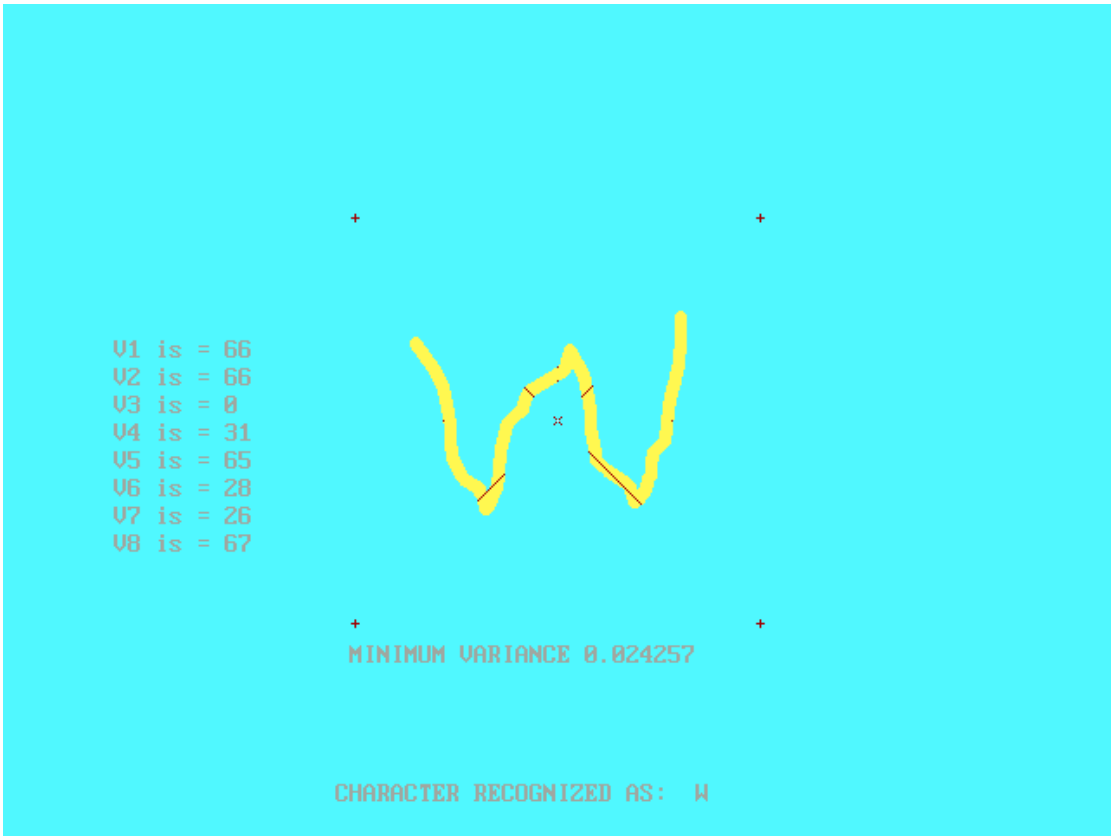


FIG 24 SAMPLE OF HANDWRITTEN ALPHABET 'W'

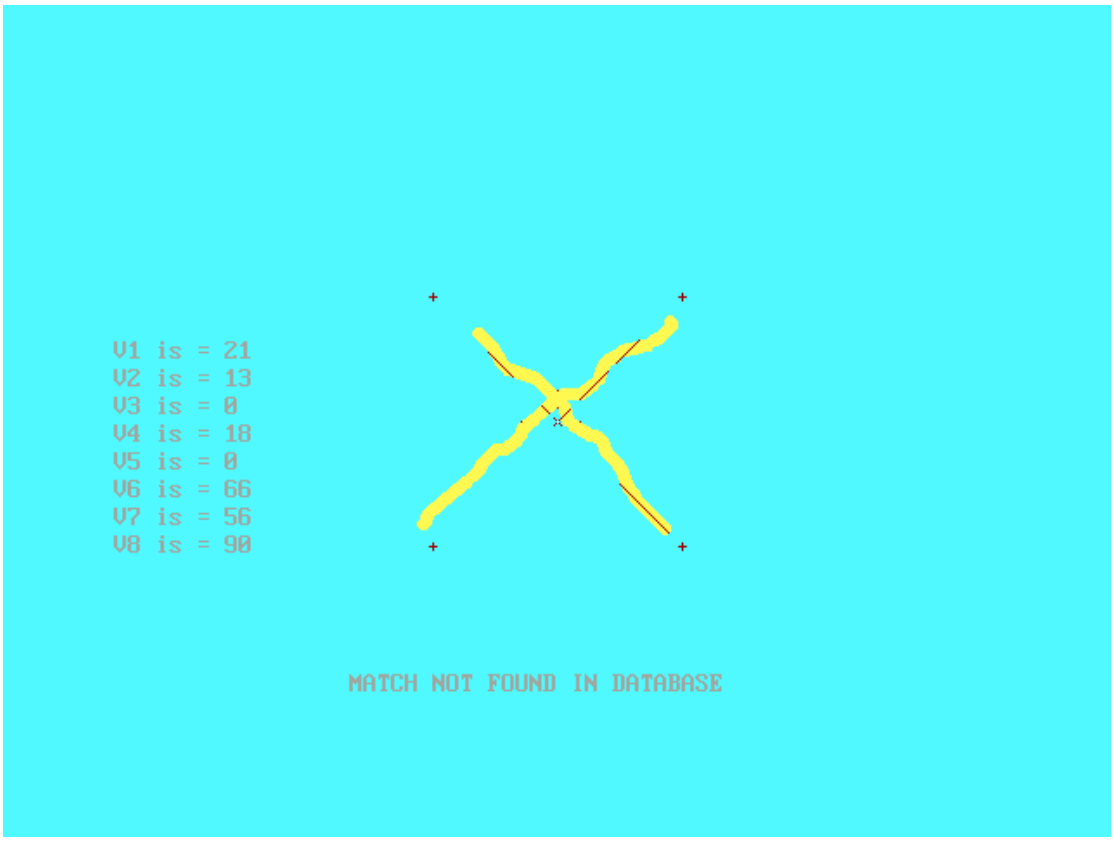


FIG 25: SAMPLE OF HANDWRITTEN ALPHABET 'X'.

4. CODING

4.1 /*ROUTINE TO INITIALIZE THE GRAPHICS MODE*/

```
void  presentation()
{
  int a = getmaxx() / 2;
  int b = getmaxy() / 2;
  settextjustify(CENTER_TEXT,0);
  settextstyle(GOTHIC_FONT,HORIZ_DIR ,3);
  setcolor(4);
  outtextxy(a,b, "Copyright @ 2002 by Danish.N and
  Saleha.R");
  settextstyle(SMALL_FONT,HORIZ_DIR,6);
  setcolor(8);
  outtextxy(a,b+25," Major Project BIT Final Year");
  setcolor(4);
  outtextxy(a,b+50," Department of Computer Science ");
  outtextxy(a,b+75," Jamia Millia Islamia");
  getch();
  clearviewport();
  setcolor(6);
  outtextxy(a, b,"Thank You");
  } //END OF MAIN*/
```

4.2 /* ROUTINE TO READ THE BMP FILE HEADER */

```
    void readBMP(FILE* fd)
    {
      long temp1,temp;
      int i,ncol,j,present;
      struct BmpHeader bh;
      int bytesread;
      int row,col;
      /* read the header word by word to avoid byte order
problems*/
     for(i=0;i<=100;i++)
     {
        for(j=0;j<=2;j++)
        {
        x[i][j]=0;
        }
        }
        bh.magic1= pbm_getrawbyte(fd);
        bh.magic2= pbm_getrawbyte(fd);
        if(bh.magic1 != 'B' || bh.magic2 != 'M')
                  printf("Not a BMP file");
        if( pm_readlittlelong(fd,(long*)&bh.filesize)
             || pm_readlittleshort(fd,(short*)&bh.res1)
             || pm_readlittleshort(fd,(short*)&bh.res2)
             || pm_readlittlelong(fd,(long*)&bh.pixeloffset)
             || pm_readlittlelong(fd,(long*)&bh.bmisize)
             || pm_readlittlelong(fd,(long*)&bh.cols)
             || pm_readlittlelong(fd,(long*)&bh.rows)
             || pm_readlittleshort(fd,(short*)&bh.planes)
             || pm_readlittleshort(fd,(short*)&bh.bitsperpixel)
             || pm_readlittlelong(fd,(long*)&bh.compression)
             || pm_readlittlelong(fd,(long*)&bh.cmpsize)
             || pm_readlittlelong(fd,(long*)&bh.xscale)
             || pm_readlittlelong(fd,(long*)&bh.yscale)
```

```
     || pm_readlittlelong(fd,(long*)&bh.colors)
     || pm_readlittlelong(fd,(long*)&bh.impcolors))
          printf("EOF in header");
if (bh.compression)
printf("Can't handle compressed images, sorry \n");
   if (bh.colors == 0)
bh.colors = 1 << bh.bitsperpixel;
bytesread=54;  /* file position so far */
   switch(bh.bitsperpixel)
{
     int i;
case 24:
     break; /* no color map for true color image*/
case 1: case 4: case 8:
printf(" need to read a color map");
break;
default:
printf("Cannot handle %d bit image",bh.bitsperpixel);
}  /* switch on sizes */
if(bytesread > bh.pixeloffset)
     printf("Corrupt BMP file");
while(bytesread<bh.pixeloffset)
{
     (void) pbm_getrawbyte(fd);
     bytesread++;
}
ncol=0;
for(row=bh.rows;row>=0;row=row-1)
{
     if(bh.bitsperpixel==24)
     {
          /*raw pixels */
          for(col=0;col<bh.cols;col++)
          {       present = 0;
               pixel val;
```

```
                    int r,g,b;
                    r=pbm_getrawbyte(fd);
                    g=pbm_getrawbyte(fd);
                    b=pbm_getrawbyte(fd);
                    bytesread+=3;
          val= ( ((pixel) (r) << 20) | ((pixel) (g) << 10) |
        (pixel) (b) );
                    for(i=0;i<=ncol;i++)
                    {
                         if(x[i][0]==val)
                         {    x[i][1]=x[i][1]+1;
                              i=ncol+1;
                              present = 1;
                         }
                     }
                         if( present == 0)
                     {
                              x[ncol][0] = val;
                              x[ncol][1] = 0;
                              ncol=ncol+1;
                    }
                    for(i=0;i<=ncol;i++)
                    {
                         if(x[i][1]<=x[i+1][1])
                         {
                              temp=x[i][1];
                              temp1=x[i][0];
                              x[i][1]=x[i+1][1];
                              x[i][0]=x[i+1][0];
                              x[i+1][1]=temp;
                              x[i+1][0]=temp1;
                         }
                    }
               }      /*cols*/
          }
```

```
            while((bytesread-bh.pixeloffset)&3)
            {
                 (void) pbm_getrawbyte(fd);
                 bytesread++;
            }
        }    /*rows*/
    }
```

4.3 /*FUNCTION TO FIND THE VECTORS CORRESPONDING TO SCANNED CHARACTER*/

```
void findV1()
{
   arr[1]=0;
   int j,k,count,r1;
   int i,l;
   FILE *f1;
   V1=0;
   r1=(getmaxy()/2)+1;
   count=0;
   for(j=0;j<=size;j++)
   {
   if(dtb[j][1]==r1)
        {
         i=dtb[j][2];
         l=i;
         count=0;
         for(k=j+1;k<=j+8;k++)
          {
            if(dtb[k][1]==r1)
             {
              i = i+1;
             if(dtb[k][2]==i)
             count=count+1;
             }
             else
             {
              k=j+10;
             }
          }
        if(count>=3)
        {
           V1=l;
```

```
                j=size+3;
            }
            else
            {
             V1=0;
            }
            }
    }
if(V1!=0)
{
    putpixel(V1,r1,100);
    V1=(getmaxx()/2+1)-V1;
    arr[1]=V1;
    if(V1<=0)
    {
    arr[1]=-V1;
    V1=0;
    }
    }
    printf("\n\n\n\n\n\n\tV1 is = %d",V1);
    }
```

4.4 /*FUNCTION FOR READING THE DATABASE FILE, CALCULATING AND COMPARING VARIANCES,DISPLAYING THE RESULT AS CLASSIFIED UN-CLASSIFIED AND MIS-CLASSIFIED*/

```
void pickdata()
{
   char ch;
   int num[5],i,j,k,val,coun;
   coun=0;
   int row=0,col=1;
   int bigarr[150][9];
   int n=0;
   for(int l=0;l<150;l++)
   {
   for(n=0;n<=9;n++)
    bigarr[l][n]=0;
   }
   ifstream infile("FILE.txt");
   infile.get(ch);
   col=1;
   while(infile)
   {
        k=0;
        coun=0;
     for(i=0;coun!=-1;i++)
     {
         if(ch!='\t' && ch!='\n')
            {
            num[k]=int(ch)-48;
            k=k+1;
            infile.get(ch);
            }
            else
            {
            coun=-1;
```

```
                }
            }
            val=0;
            if(k>=1)
            {
                for(j=k-1;j>=0;j--)
                val=val+num[j]*pow(10,k-1-j);
                bigarr[row][col]=val;
              if(ch=='\n')
              {
               row=row+1;
               col=1;
              }
              if(ch=='\t')
                  col=col+1;
            }
        infile.get(ch);
        }
           n=0;
           int I[8],large,d,couna;
           couna=0;
           float comp[150][9];
           int tem[8][2];
           for(i=1;i<=8;i++)
           {
                tem[i][1]=0;
           tem[i][2]=0;}
           for(i=0;i<=row-1;i++)
           {
           for(j=1;j<=9;j++)
           {
           comp[i][j]=0;}
           }
           I[0]=0;
           I[1]=V1;
```

```
I[2]=V2;
I[3]=V3;
I[4]=V4;
I[5]=diag1;
I[6]=diag2;
I[7]=diag3;
I[8]=diag4;
int t,e;
t=0;
int a;
int row0=0;
for(i=0;i<=row-1;i++)
{
row0=0;
for(j=1;j<=8;j++)
       {
if(I[j]==0 && bigarr[i][j]!=0)
            {
for(e=1;e<=10;e++)
{
Comp[t][e]=0;
j=9;
row0=-1;
          }
           }
       else
            {
             comp[t][j]=bigarr[i][j];
        }
      }
      if(row0!=-1)
     {
     comp[t][0]=i;
     t=t+1;
     }
```

```
}
float comp1[150][9];
int I1[8];
for(i=0;i<=8;i++)
I1[i]=0;
for(l=0;l<=t-1;l++)
{
comp1[l][0]=comp[l][0];
a=1;
for(n=1;n<=8;n++)
    {   if(I[n]!=0)
       {
        comp1[l][a]=comp[l][n];
        I1[a]=I[n];
         a=a+1;
   }
     }
}
for(l=0;l<=t-1;l++)
{
for(n=1;n<=a-1;n++)
    { comp1[l][n]=comp1[l][n]/(float)I1[n];
    }
}
float sum,sqsum,test[150][2];
for(i=0;i<=t-1;i++)
{
   test[i][2] = comp1[i][0];
   sum=0;
   sqsum=0;
   for(j=1;j<=a-1;j++)
    {
        sum= sum + comp1[i][j];
        sqsum = sqsum + pow(comp1[i][j],2);
    }
```

```
   test[i][1] = ((sqsum)/(a-1)) - pow((sum/(a-1)),2);
}
couna=0;
float smallest=0;
smallest=test[0][1];
couna=test[0][2];
for(i=0;i<=t-1;i++)
{
if(smallest>test[i][1])
{
smallest=test[i][1];
couna=test[i][2];
j=couna;
}
}
char result;
if(couna >25)
couna = couna%26;
result = char(couna + 65);
if(t!=0)
  { printf("\n\n\n\n\t\t\t MINIMUM VARIANCE %f",smallest);
    getch();
    printf("\n\n\n\t\t\tCHARACTER RECOGNISED AS:  %c"
,result);
  }
else
 {
setcolor(BLUE);
outtextxy(150,450,"MATCH NOT FOUND IN DATABASE");
 }
 }// END PICKDATA()
```

5. PHYSICAL DATABASE DESIGN

5.1 DATABASE STRUCTURE

The database consists of two master tables

- ✓ Database.txt(contains the entries for alphabets.)
- ✓ dataNum.txt(contains the entries for digits.)

The structure of Database file is a 2-D array which is considered to be an ideal database and contains the vector entries of some standard English font of similar size. The following figure depicts the actual database used by the system. It consists of 8 entries in each row each corresponding to the vectors for alphabets in different direction. The different vectors are separated by a tab character.

Vectors are stored in an ordered and systematic manner such that it keeps exact correspondence with the digits/alphabets viz. the 1st entry in datanum.txt corresponds to 1,11,21,31 belong to 0 and the 10,20,30th … one to 9 for the first, second, third….nth font respectively.

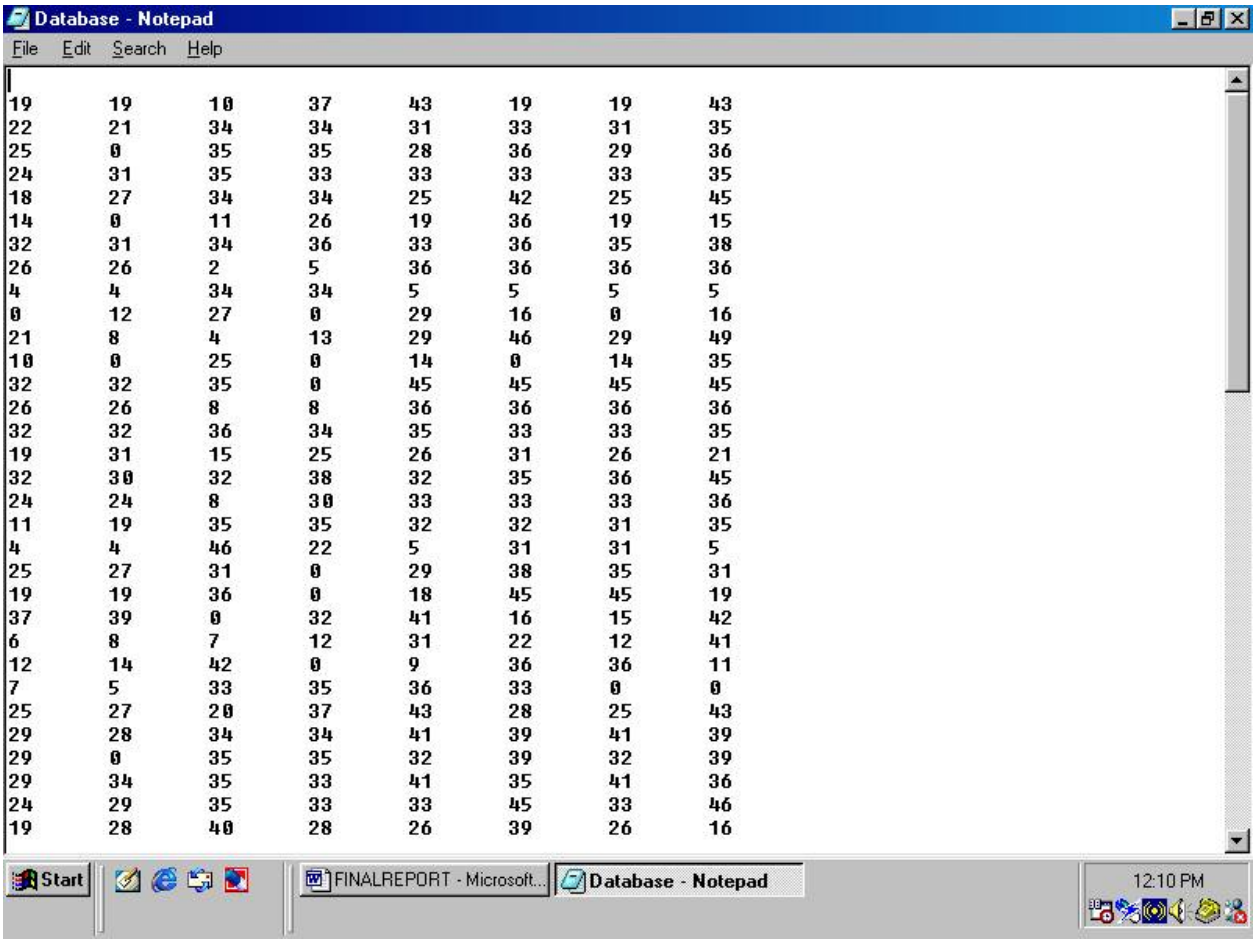


FIG 27 : THE DATABASE FILE

The following figure depicts order of direction in which the vectors corresponding to the character are measured.

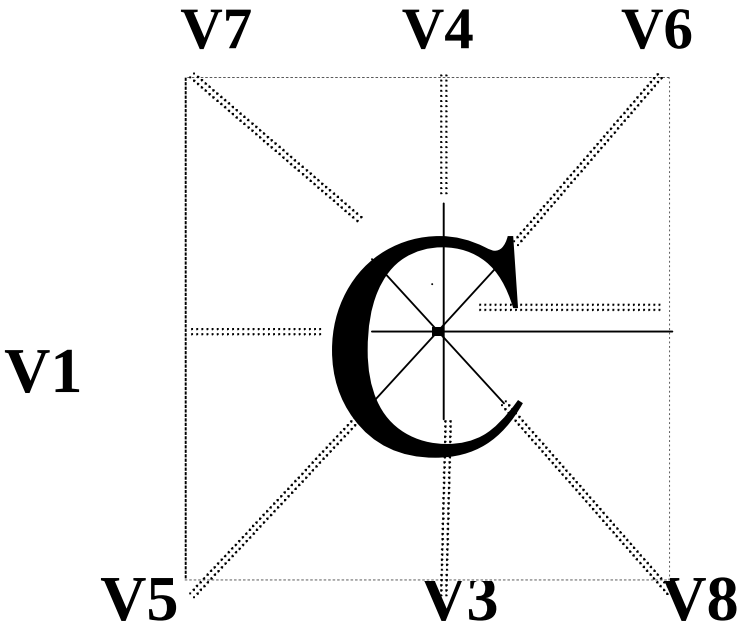


FIG 28 : THE ORDER OF VECTORS AS STORED IN THE DATABASE

The vectors are stored in the database as depicted below

V1	V2	V3	V4	V5	V6	V7	V8
19	19	10	37	43	19	19	43

6. TABULATION OF RESULTS

RESULT / SUMMARY OF IMPLEMENTATION

6.1 RESULTS OF SOME STANDARD ENGLISH ALPHABET FONTS (TYPE-WRITTEN)

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
A	15	15	0	0
B	15	14	1	0
C	15	14	1	0
D	15	14	1	0
E	15	13	2	0
F	15	14	1	0
G	15	14	1	0
H	15	15	0	0
I	15	15	0	0
J	15	14	1	0
K	15	13	2	0
L	15	14	1	0
M	15	14	1	0
N	15	13	2	0
O	15	15	0	0
P	15	15	0	0
Q	15	13	2	0
R	15	13	2	0
S	15	13	2	0
T	15	14	1	0
U	15	13	1	0
V	15	14	1	0
W	15	14	1	0
X	15	13	2	0
Y	15	14	1	0
Z	15	14	1	0

The recognition for 15 samples of some standard English alphabet fonts (type-written) gives the recognition result of: **94.30%**

6.2 RESULTS OF SOME UN-KNOWN ENGLISH ALPHABET FONTS (TYPE-WRITTEN)

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
A	15	15	0	0
B	15	14	1	0
C	15	14	1	0
D	15	14	1	0
E	15	13	2	0
F	15	14	1	0
G	15	14	1	0
H	15	15	0	0
I	15	15	0	0
J	15	14	1	0
K	15	13	2	0
L	15	14	1	0
M	15	14	1	0
N	15	13	2	0
O	15	15	0	0
P	15	15	0	0
Q	15	13	2	0
R	15	13	2	0
S	15	13	2	0
T	15	14	1	0
U	15	13	1	0
V	15	14	1	0
W	15	14	1	0
X	15	13	2	0
Y	15	14	1	0
Z	15	14	1	0

The recognition for 15 samples of some unknown English alphabet fonts (type-written) gives the recognition result of: **88.02%**

6.3 RESULTS OF SOME HAND-WRITTEN ALPHABET

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
A	5	4	1	0
B	5	3	1	1
C	5	4	1	0
D	5	4	1	1
E	5	3	2	0
F	5	3	1	1
G	5	3	1	1
H	5	4	1	0
I	5	5	0	0
J	5	4	1	0
K	5	3	2	0
L	5	4	1	0
M	5	4	1	0
N	5	3	2	0
O	5	5	0	0
P	5	4	1	0
Q	5	3	2	0
R	5	3	2	0
S	5	3	2	0
T	5	4	1	0
U	5	4	1	0
V	5	5	0	0
W	5	5	0	0
X	5	5	0	0
Y	5	4	1	0
Z	5	3	2	0

The recognition for 5 samples of Hand-written English alphabets gives the recognition result of: **75.42%**

6.4 RESULTS OF SOME TYPE-WRITTEN NUMERALS OF STANDARD FONT

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
0	5	5	0	0
1	5	5	0	0
2	5	5	0	0
3	5	5	0	0
4	5	5	0	0
5	5	4	1	0
6	5	5	0	0
7	5	5	0	0
8	5	5	0	0
9	5	5	0	0

The recognition for 5 samples of some standard English numeral fonts (type-written) gives the recognition result of: **98.00%**

6.5 RESULTS OF SOME TYPE-WRITEEN NUMERALS OF SOME UNKNOWN ENGLISH FONTS

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
0	5	5	0	0
1	5	5	0	0
2	5	3	2	0
3	5	4	1	0
4	5	4	1	0
5	5	3	2	0
6	5	4	1	0
7	5	4	1	0
8	5	4	1	0
9	5	5	0	0

The recognition for 5 samples of some unknown English numeral fonts (type-written) gives the recognition result of: **82.00%**

6.6 SUMMARY OF RESULT

6.6.1 ALPHABETS

% AGE CORRECT FOR STANDARD TYPE-WRITTEN FONTS: **94.30 %**

% AGE CORRECT FOR UNKNOWN TYPE-WRITTEN FONTS: **88.02 %**

% AGE CORRECT FOR HAND-WRITTEN FONTS (5 SAMPLES): **75.42 %**

6.6.2 NUMERALS

% AGE CORRECT FOR STANDARD TYPE-WRITTEN FONTS: **98.00 %**

% AGE CORRECT FOR UN-KNOWN TYPE-WRITTEN FONTS: **82.00 %**

7. SCOPE FOR FURTHER IMPROVEMENTS

1. We can incorporate the recognition of lower-case alphabets also.
2. This approach can be extended for the recognition of words, sentences and documents by implementation of segmentation techniques. This will improve the accuracy of the system considerably as in that case semantic checking can also be incorporated.
3. Neural network can also be implemented in the near future to make the system learn and adapt; this would eliminate the need of database.
4. The database structure shall be improved by associating with every characteristic vector,, the corresponding alphabet and font, this would enhance the understandability.
5. Better user interface can be developed.

8. TYPES OF CHARACTER RECOGNITION SYSTEMS AND THEIR POTENTIAL APPLICATIONS

The types of character recognition systems can be classified as follows

- Task Specific Readers
- General Purpose Page Readers

8.0 TASK SPECIFIC READERS

The task specific readers handle only specific document types. Some of the most common task specific reader read bank check, letter mail, or credit card slips. These readers usually utilize custom-made image lift hardware that captures only a few predefined document regions. For e.g. the check readers may read only the courtesy amount field and a postal reader may just scan the address block on a mail piece. Such systems emphasize high throughput rates and low error rates. Most of the task specific readers are able to read both the handwritten and machine-written text. They are used primarily for high volume applications which require high system throughput. Since high throughput rates are desired, handling only the fields of interest helps reduce time constraints. As similar documents possess similar size and layout structure, it is straightforward for the image scanner to focus on those fields where the desired information lies. This considerably reduces the image processing and text recognition time. Some application areas to which task specific readers have been applied are

- Form Readers
- Check Readers
- Bill Processing Systems
- Airline Ticket Readers
- Passport Readers
- Address Readers

8.0.1 FORM READERS

A form reading system needs to discriminate between pre-printed form instructions and filled-in data. The system is first trained with a blank form. The system registers those areas on the form

where the data should be printed. During the form recognition phase. , the system uses spatial information obtained from training to scan the regions that should be filled with data

8.0.2 CHECK READERS

A check reader captures check images and recognizes courtesy amounts and account information on the checks. Some readers also recognize the legal amount on check and use the information in both fields to cross-check the recognition results. An operator can correct misclassified characters by cross-validating the recognition results with the check image that appears on the system console.

8.0.3 BILL PROCESSING SYSTEMS

In general a bill processing system is used to read payment slips, utility bills and inventory documents. The system focuses on certain region on a document where the expected information is located.

8.0.4 AIRLINE TICKET READERS

In order to claim revenue from airline passenger ticket, an airline needs to have three records matched: reservation record, the travel agent record and the passenger ticket. However it is impossible to match all three records for every ticket sold. Current methods which use manual random sampling of tickets are far from accurate in claiming the maximal amount of revenue. Several airlines are using a passenger revenue accounting system to account accurately for passenger revenues. The system reads the ticket number on a passenger ticket and matches it with the one in the airline reservation database. It scans tickets up to 260,000 tickets per day and achieves sorting rate of 17 tickets per second.

8.0.5 PASSPORT READERS

An automated passport reader is used to speed up the returning American passengers through custom inspections. The Reader reads a traveler's name, date of birth and passport number on the passport and checks these against the database records that contain information on fugitive felons and smugglers.

8.0.6 ADDRESS READERS

The address in a postal mail sorter locates the destination address block on a mail piece and read the ZIP code in the address book. If additional fields in the address block are read with high confidence the system may generate a 9 digit ZIP code for the piece. The resulting ZIP code is used to generate a bar code which is sprayed on the envelope as depicted in the figure below.

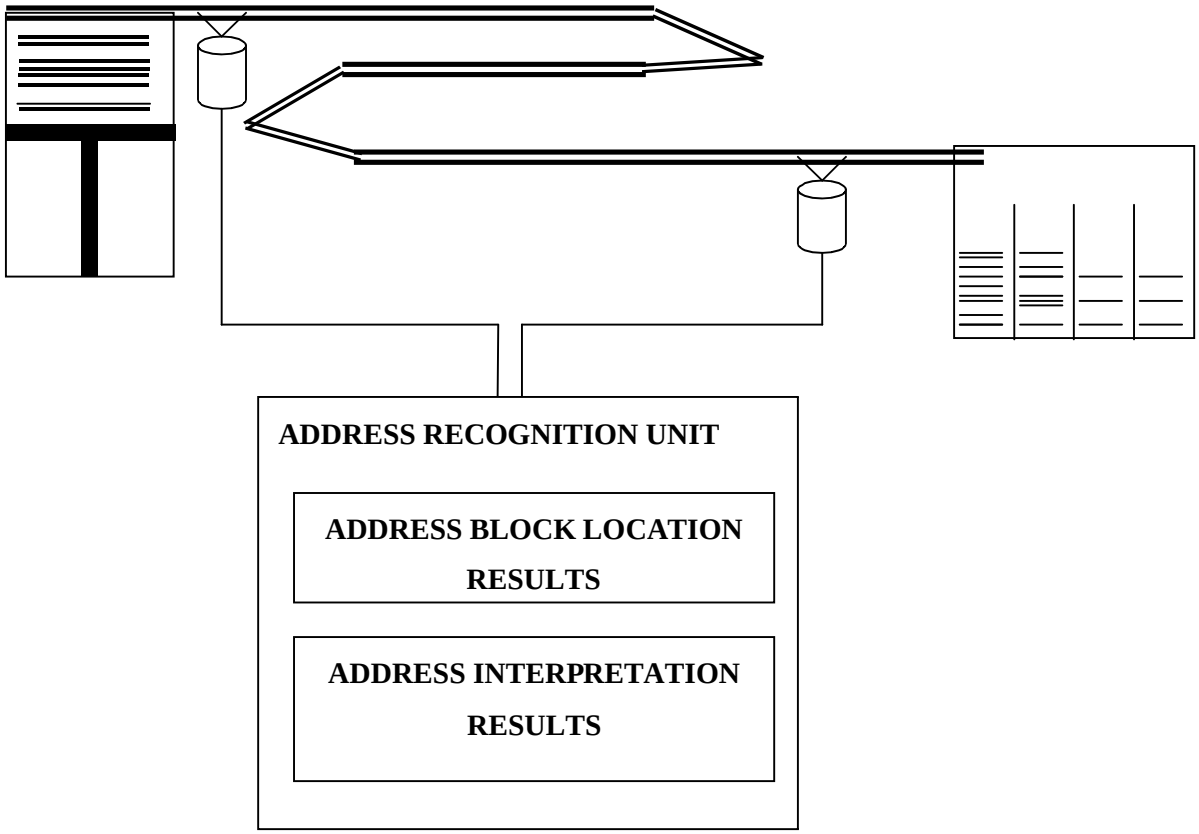


FIG 29 ARCHITECTURE OF POSTAL ADDRESS READING AND SORTING SYSTEM.

8.1 GENERAL PURPOSE PAGE READERS

They are designed to handle a broader range of documents such as business letters, technical writing and newspapers. These systems capture an image of a document page and separate the page into text regions and non text regions. Non-text regions such as graphics and line drawings are often saved separately from the text and associated recognition results. Text regions are segmented into lines, words and characters and the characters are passed to the recognizer. Recognition results are output in a format that can be post-processed by the application software. Most of these page readers can read machine written text only a few can read hand-printed alpha-numeric.

9. APPENDIX

9.0 BACKGROUND (FURTHER DETAILS)

9.0.1 COLOR MODELS

Additive (RGB)

This is the model used by nearly every video screen controller. Each pixel consists of three different components red, green and blue in the pixel. The particular color is created by starting with black and adding in the given amounts of the three component colors. For each component 0 means none of the color and maximum value is the full amount of the color. If there are 8 bits per component then (0, 0, 0) means black, (255, 0, 0) means bright red, (255,255,255) means white. Each pixel would be stored as 24 bits.

Subtractive (CMY)

In the subtractive system the colors are generated by removing or absorbing various components of white light. CMYK stands for cyan, magenta and yellow .This is the color model employed by the color printers. The pigments in the inks deposited on the printing medium absorb specific colors, removing them from the reflected light received by the eye. When no ink is deposited close to 100% of the light is reflected and we perceive white. When all three primaries are deposited, close to zero percent of the visible light spectrum is reflected and we perceive black.

9.0.2 LUMINANCE

It is the measure of color's intrinsic brightness. Luminance of two different colors can be same even when the colors are not.

9.0.3 COMPRESSION TYPES

- **RLE(Run Length Encoding)**
- **Huffman Coding**
- **Dictionary Systems**

Run-Length Encoding (RLE)

This is the most common of all the compression schemes. These schemes tend to take advantage of the fact that images, particularly computer generated images, tend to have areas of solid colors,

meaning that there are often many pixels in a row with the same pixel value. A line of pixels having the same value is referred to as a run. The simplest practical compression scheme replaces all the runs in the images with a two values code representing the pixel's value and the number of times to repeat it.

Huffman Codes

They are also known as variable length codes. Assume that we start with an image consisting of 8 bit pixels. Unless the image is totally random, some pixel values occur far more than others. We assign a set of different lengths to the values, assigning short codes, 4 bits or so, to the most common values, and longer codes, up to 10 or 11 bits, to the less common values. When using this for storing a table that maps code values to the original pixel values, this sort of recoding can end up making the file considerably smaller. These variable length codes are usually called *Huffman Codes*, after David Huffman, who figured out in 1952 how to assign codes given the relative frequency of bytes in a file, so that the total coded length was minimized. JPEG uses this format

Dictionary Systems

They are opposite to the Huffman codes. Huffman schemes start with fixed size inputs and assign variable length codes. Dictionary systems do the opposite- they start with variable length input and assign fixed length codes. The dictionary system most commonly used in graphic files is called as LZW for Lempel, Ziv and Welch , the three people who invented it. An LZW compressor like unlike Huffman, does not need to compute a code table before compressing the input. It is adaptive, meaning that it constructs the dictionary on the fly as it reads the input data. The decoding algorithm can deduce from the sequence of codes in the compressed file what dictionary entries the encoding program used, so the dictionary need not be stored explicitly at all. All GIF files use the LZW compression scheme. Some TIFF files also use the LZW compression scheme.

LZW does work well on computer generated images and particularly well on large solid color areas. However it is not a very effective compression scheme to use, because digitized images tend not to have identical repeating sequences of pixels.

9.0.4 PBM APPROACH (Detailed)

PBM Format (Single Bit)

The simplest format is PBM(Bit Map) for images with 1 bit per pixel. If the type field is P1, the image data is stored as ASCII digits, with 1 being black and 0 white. If the field type is P4, the data is stored as binary, 8 pixel per byte, with the high bit in each type being the leftmost pixel. Even if the width of the image is not a multiple of 8 pixels, each row starts at a byte boundary.

PGM Format (Grayscale)

This is the next simplest format. PGM format for grayscale images has a single component more than 1 bits per pixel. The header contains a third number after the image height and width, which is the maximum pixel value. Pixel values range from 0, meaning black to the maximum value for white. If the type field is P2, the pixel values are stored as ASCII digit strings, separated by white space. If the field type is P5, the pixel values are bytes, one value per byte. In the P5 format , the largest possible pixel value is 255, since that's all there's room in a byte.

PPM Format (Color)

In the case of a PPM(Pixel Map) format, each pixel is stored as a triple of red, green and blue values, in that order. The header contains a third number after the image height and width, which is the maximum component value. If the type field is P3, the pixel values are each three ASCII digit strings, separated by white space. If the field type is P6, the pixel values are triples of bytes, with each byte being a color component. As in the P5 format , the largest possible component value is p6 format is 255.

9.1 CASE STUDIES OF DIFFERENT ONLINE CR SYSTEMS

9.1.1 VISUAL INPUT FOR PEN BASED COMPUTERS [7]

The input surface consists of a camera, a normal piece of paper and a normal pen. The camera focuses on the paper and images the pen tip. The computer analysis of the resulting images enables the trajectory of the pen to be tracked and contact of the pen with the paper to be detected. This approach obtains data from a fixed video camera and it allows the user with maximum flexibility in choosing any pen and writing surface. The pen tip in real time is tracked in order to construct its trajectory accurately and independently of change in the lightning. The pen tip is tracked continuously when the user is writing and when the pen is traveling on top of the paper. In this system only three condition are imposed on the user.

I. The camera should be located so that it has a clear sight of the pen tip.
II. The writing implement should have sufficient contrast with the piece of paper.
III. The paper has to be lighter than the human skin.

The steps involved as per the block diagram on the following page are

Initialization And Preprocessing

After the image has been fed using the camera to the preprocessing stage. This block initializes the algorithm i.e. it finds the initial position of the pen and selects the template(rectangular sub-region

of the image corresponding to the pen tip). The function of this stage is only to extract a region of interest centered around the predicted position of pen tip. The region of interest is used by the following block of the system to detect the actual position of the centroid of the pen tip in the current image.

Pen Tip Detector

The pen tip detector has the task of finding the position of the pen tip in the current frame of the sequence.

Filter / Predictor

The filter predicts the most likely position of the pen tip on the following frame based on the current predicted position, velocity and acceleration of the pen tip on the location of the pen tip given by the pen tip detector. This prediction allows the interface to reduce the research area, saving computations while still keeping a good pen tip detection accuracy. The filter also estimates the most likely position of the pen tip of the missing frames.

Ballpoint Detector

The ball point detector finds the position of the very end of the pen tip i.e. the place where the pen is in contact with the paper when the user is writing It finds the most likely position of the centroid of the pen tip. It is based on the orientation of the axis and of the boundaries of the pen tip in the previous frame.

Pen Up/Down Classifier

The last block of the system checks for the presence of ink on the paper at the positions where the ballpoint of the pen was detected. It is based on local measurements of the brightness.

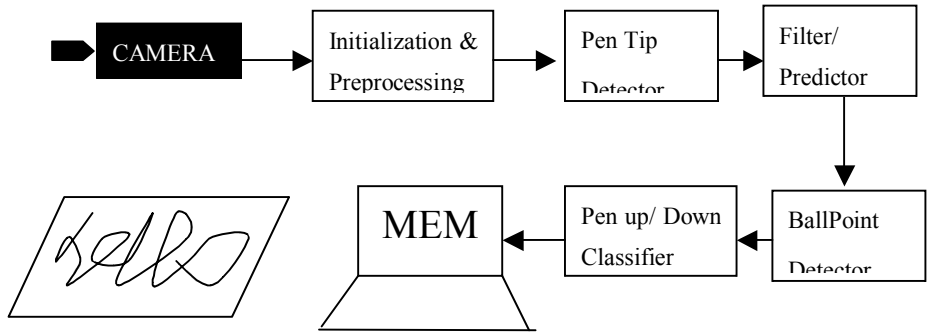


FIG 30 DIAGRAMMATIC REPRESENTATION OF THE METHOD

Implementation

The real time interface was used as a front end for signature verification system .Here 25-30 true signatures and 10 forgeries from 105 subjects were taken adding to an approximate total of 4,000 signatures. The data was collected over months in which the subjects would provide signatures at different times during the day. The interface was placed next to a window , so natural light was used for capturing signatures at day time, while electric lightning was used for acquiring signatures during the night. The subjects were asked to provide data in three different sessions in order to sample their signature variability. Given the number of subjects involved ,the position and orientation of the camera was different from session to session and from subject to subject. The verification error rate achieved was less than 1.5% for skilled forgeries and a verification for less than 0.25% for random forgeries. These rates correspond to the condition of equal false acceptance rate and false rejection rate.

9.1.2 "ONLINE RECOGNITION OF HANDWRITTEN SYMBOLS"[8]

This system represents an online system for recognition of handwritten symbols from a user specified alphabet is described. As a symbol is written on a digitizing tablet, the position of the pen tip and the force exerted by the pen tip is time stamped and recorded at times determined by the sampling rate of the tablet. This system enables recognition of a symbol from a user defined alphabet irrespective of their size as well as the orientation and position of the symbol on the tablet. Here we concentrate on a user dependent approach as the system is expected to recognize symbols from a user-defined alphabet. The methodology employed for recognition is curve matching where each symbol is thought of as a curve(or collection of curve) in the plane and a

curve is represents ed by a sampling of points along the curve. For each symbol in the alphabet there is a representative model curve . The curve matching algorithms try to model the curve that best fits the curve of unknown input symbol. Normalization routine is employed to minimize differences due to one or more of translation, rotation, dilation ,slant or other transformations.

The normalization procedure and definition of distance between two curves used in the recognition scheme result in a method that is capable of recognizing alphabets written in any orientation or scale anywhere on the tablet.

The system can work either in a training mode or recognition mode. To be in the training mode ,the user indicates that he/she wishes to define a new symbol that the system is to be capable of recognizing. The user then writes the symbol on the tablet thus creating a model for that symbol. In the recognition mode , an unknown written symbol is compared to the models to find the model that is most similar to unknown symbols .

Implementation

The system was implemented. We summarize the results from two experiments. In the first experiment, 8 users were asked to write the capital letters A-Z and ten to repeat. One set of the letters was used as the set of models and the other was run through the recognizer. The results were that there were 4 errors out of 8*26=208 recognition tasks giving a 98% recognition rate. Of the eight users , five had 100% recognition rates, two others had one error and one had two errors.

In the second experiment 32 cursively handwritten words were taken as samples. Each of the 19 users wrote the word 15 times. For each of the user one sample each of the 32 words was used as a model. The remaining 14 copies of each of the 32 words were processed by the recognizer. The recognizer correctly recognized 95.5% of the words.

9.2 SOME MORE SAMPLE SCREENS

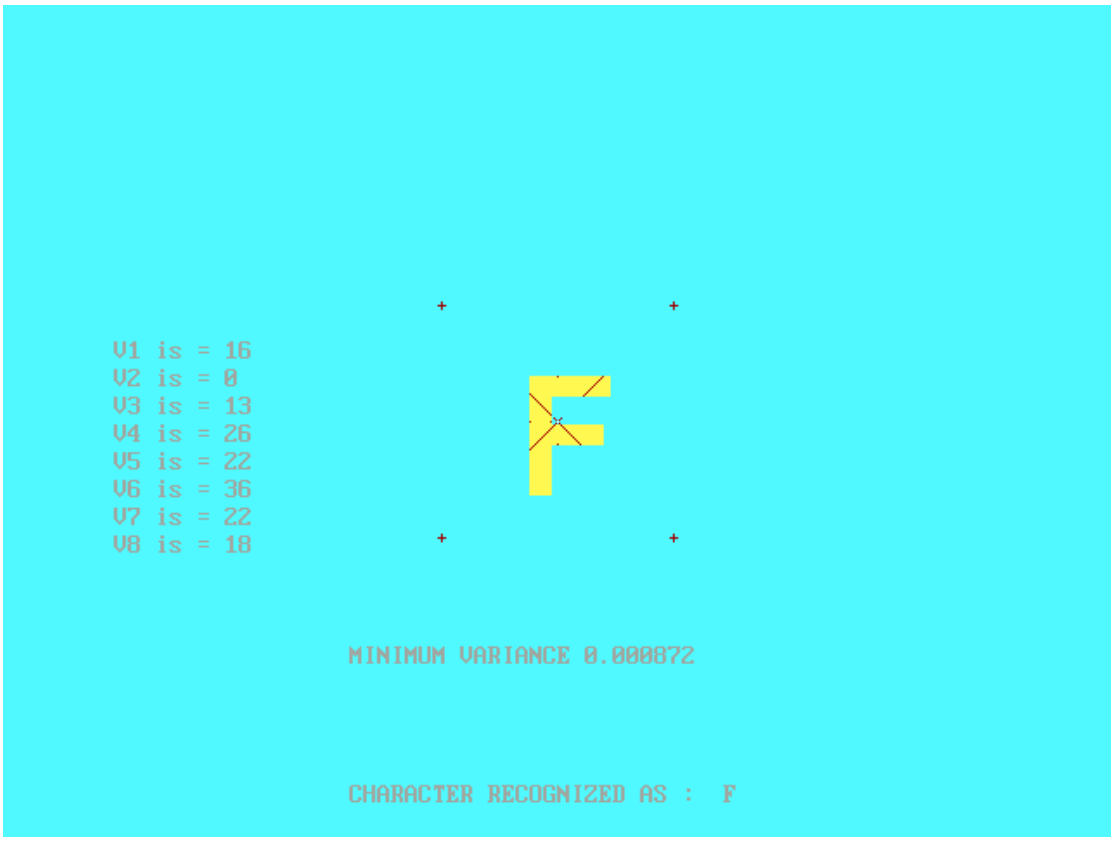


FIG 30 SAMPLE OF A TYPEWRITTEN CHARACTER 'F'

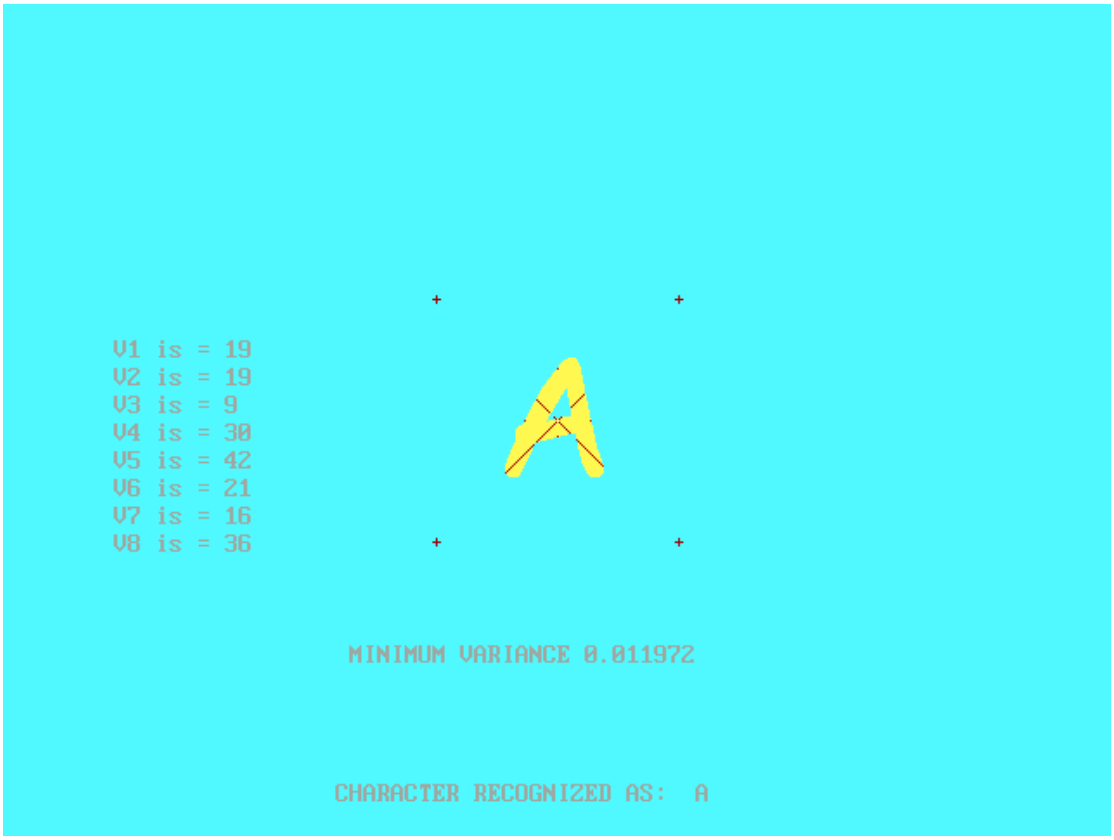


FIG 31 SAMPLE OF ALPHABET 'A'

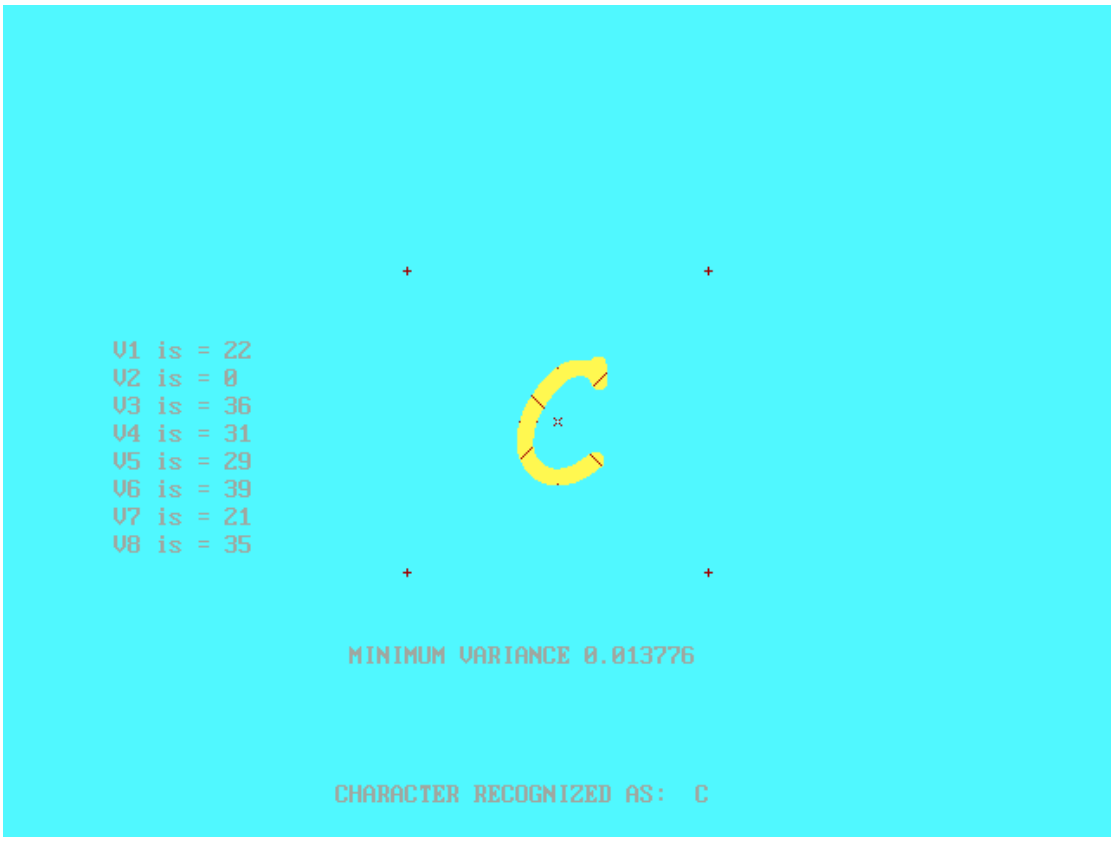


FIG 32 SAMPLE OF ALPHABET 'C'

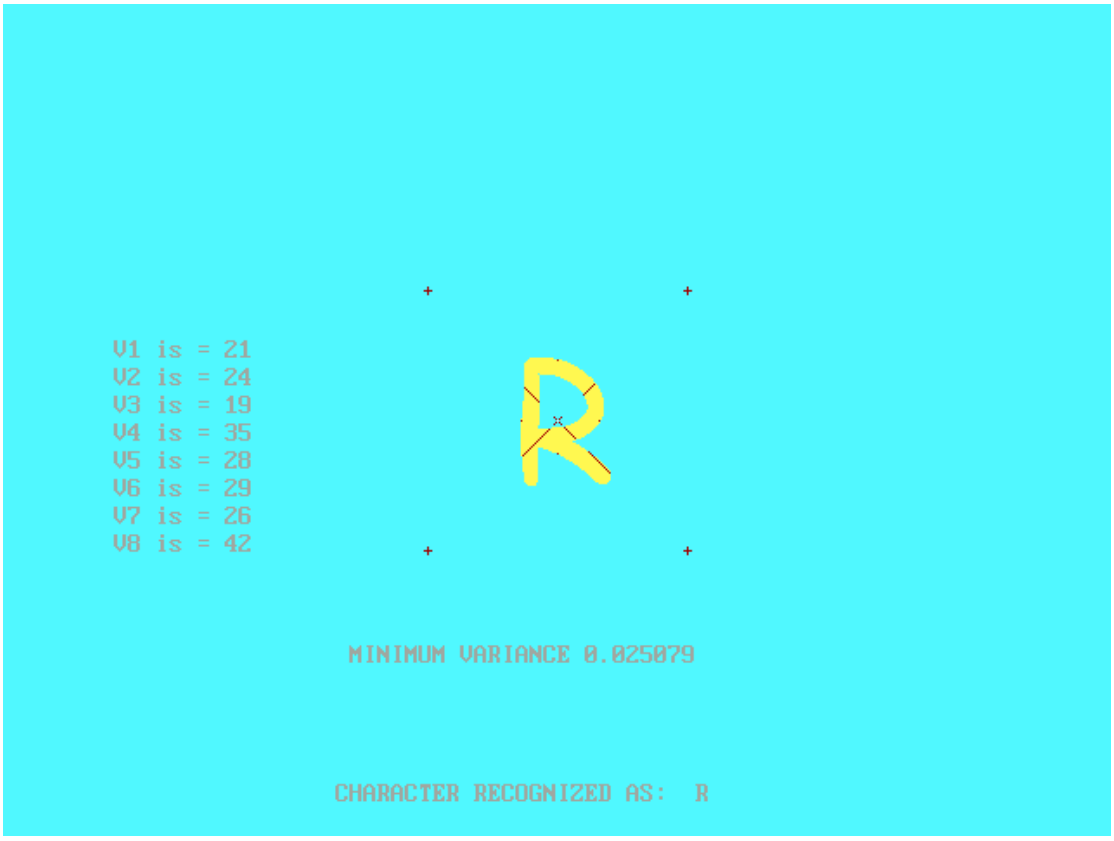


FIG 33 SAMPLE OF ALPHABET 'R'

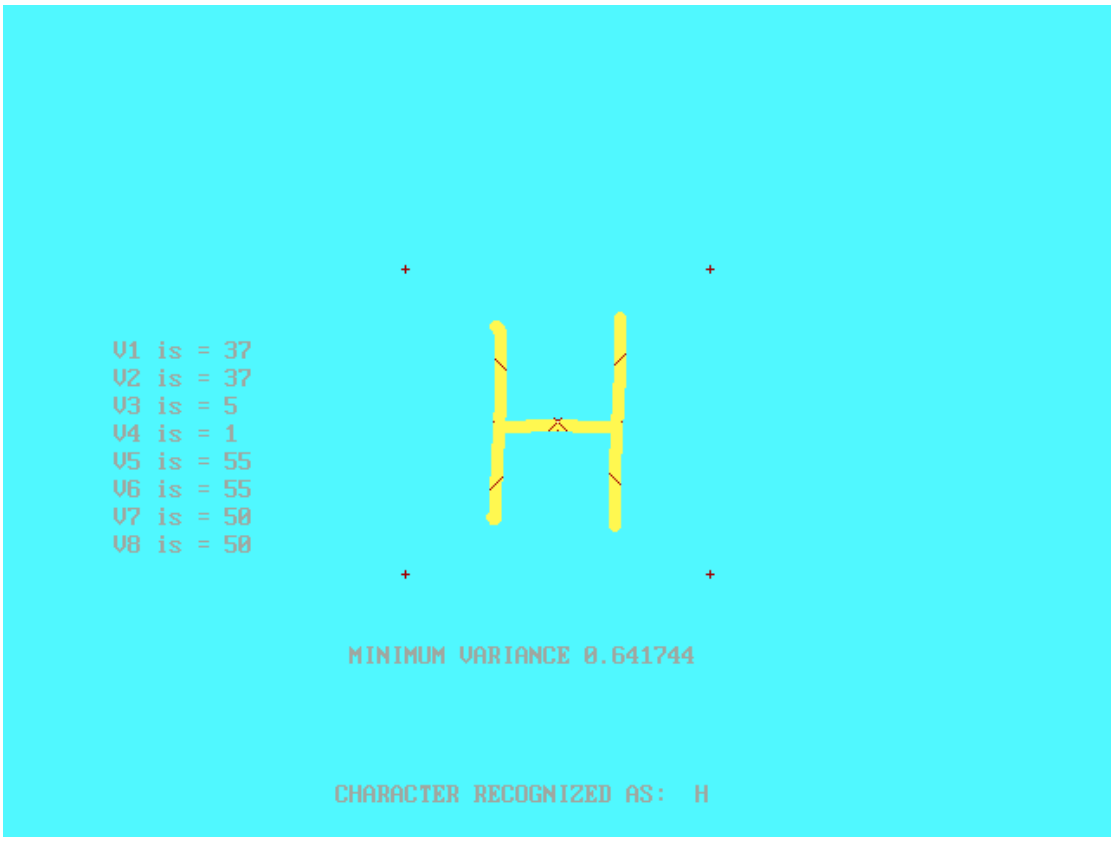


FIG 34 SAMPLE OF HANDWRITTEN ALPHABET 'H'

10. REFERENCES

The following papers and publications have been used, for reference, in the study.

[1] Yan Xiong, Qiang Huo, Chorkin Chan, " A Discrete stochastic model for the offline recognition of Chinese characters.". IEEE Transactions on Pattern Analysis and machine Intelligence. Vol 23, No. 7, July 2001.

[2] T.H Hildebrandt and W.T.Liu, "Optical recognition of Handwritten Chinese characters: Advances since 1980, "Pattern Recognition, vol. 26, pp. 205-225, 1993.

[3] Special Issue on Document analysis and Recognition, IEICE Trans. Information and Systems, vol. E77-D, no. 7, July 1994.

[4] Special Issue on Character Recognition and Document Understanding, IEICE Trans. Information and Systems, vol. E79-D, no. 5, July 1996.

[5] Special Issue on Oriental Character Recognition, Pattern Recognition, vol. 30, no. 8, 1997.

[6] Special Issue on advances in Oriental Document Analysis and Recognition Techniques, Part I, Part II, Int'l J. Pattern Recognition and Artificial Intelligence, vol. 12, nos. 1-2, 1998.

[7] Mario E Munich and Pietro Perona, "Visual Input for Pen-Based Computers", IEEE Transactions on Pattern Analysis and machine Intelligence. Vol. 24, No. 3, March 2002.

[8] Gordon Wilfong, Franl Sinden & Laurence Ruedisueli "Online Recognition of Handwritten Symbols" IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 18, No. 9, Sept 1996. (5)

[9] V.A.Kovalevsky, "Character Readers and Pattern Recognition", Spartan Books ,New York.

[10] Pepe Siy & C.S.Chen, " Fuzzy Logic for Handwritten Numeral Character Recognition " IEEE Trans. System, Man, Cybernetics.,pp. 570-575, November 1974.

[11] Byron Gottfried, " Programming with C", Tata McGraw-Hill Edition.

[12] Y. Kanetkar , "Let us C", BPB Publications