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 :

1. INTRODUCTION

1.1 PROBLEM DEFINITION

1.2 BACKGROUND

1.2.1 GRAPHIC FILES

√ ***ASCII TEXT***

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

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

√ ***BITMAPPED FORMAT***

√ ***VECTOR FORMATS***

1.2.2 PIXEL

pixel value

1.2.3 TRUE COLOR

1.2.4 PALETTE / COLOR MAP

color map **palette**

-
-

1.2.5 COLOR MODEL

color model

1.2.6 RESOLUTION

1.2.7 COMPRESSION

-
-

♦

♦

♦

♦

♦

♦

1.2.8 WINDOWS BITMAP FORMAT (BMP)

-
-
-
-

1.2.9 PBM APPROACH

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

1.2.10 CHARACTER RECOGNITION (General Idea)

"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.

bout the character e.g. "Γ"

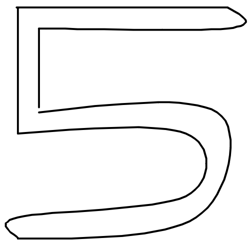


FIG 1:

1.2.11 TYPES OF CHARACTER RECOGNITION SYSTEMS

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

Online Character Recognition

Offline Character Recognition

√

√

Document Analysis

Character Recognition

feature extractor

classifier

feature extractor

Template matching or matrix matching

Template matching is a trainable process as template characters can be changed

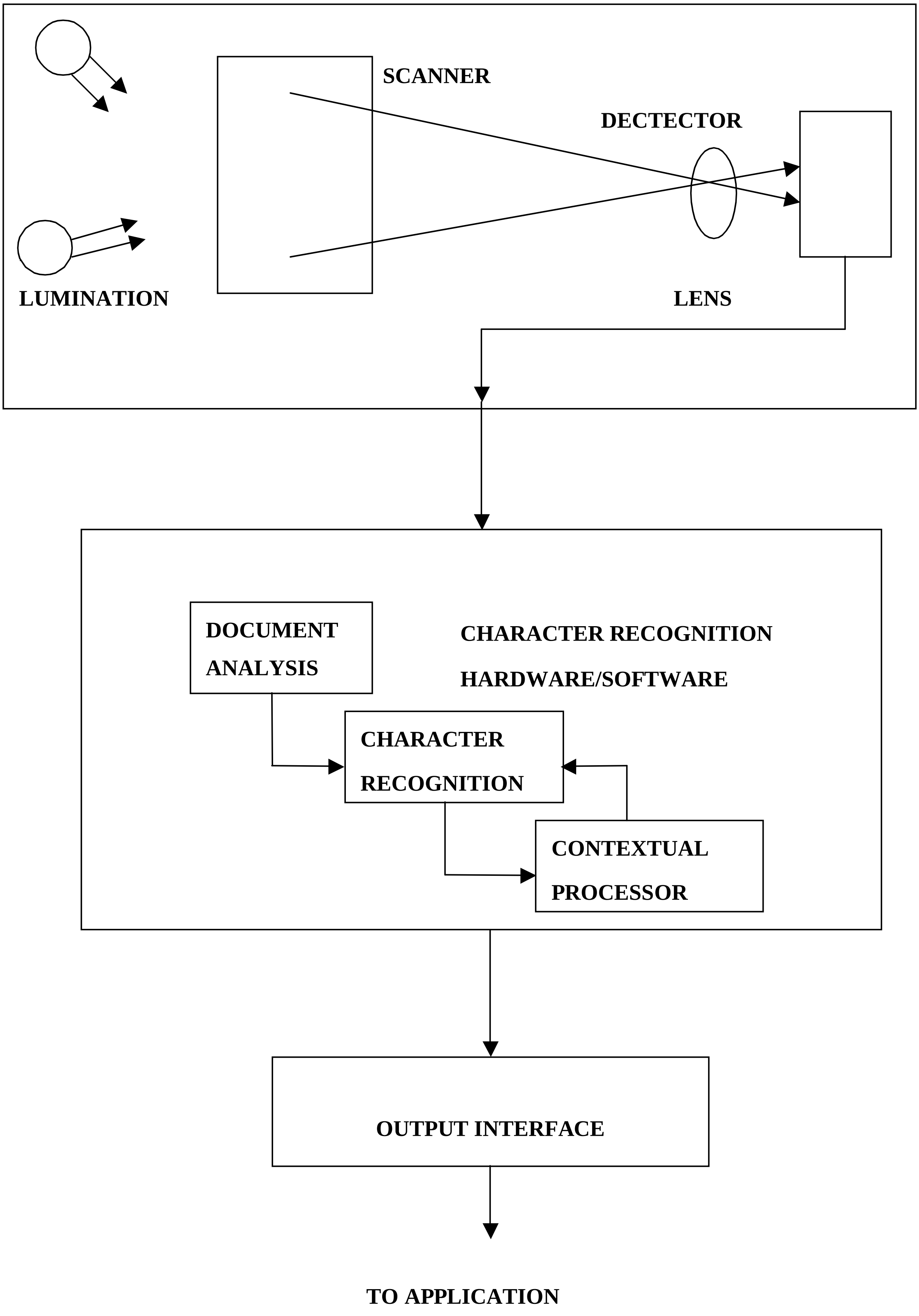

SCANNER
DECTECTOR
LUMINATION
LENS
DOCUMENT
ANALYSIS
CHARACTER RECOGNITION
HARDWARE/SOFTWARE
CHARACTER
RECOGNITION
CONTEXTUAL
PROCESSOR
OUTPUT INTERFACE
TO APPLICATION

FIG 2:

1.2.12 CASE STUDY OF AN OFFLINE CHARACTER RECOGNITION SYSTEM

Fuzzy Logic For Handwritten Numeral Character Recognition [10]

√

√

Handwritten Character Recognition System

A. HANDWRITTEN CHARACTER REPRESENTATION

B. PATTERN CLASSIFIER

Recognition System

Input pattern

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

Thinning

Labeling

Node Detection and Labeling

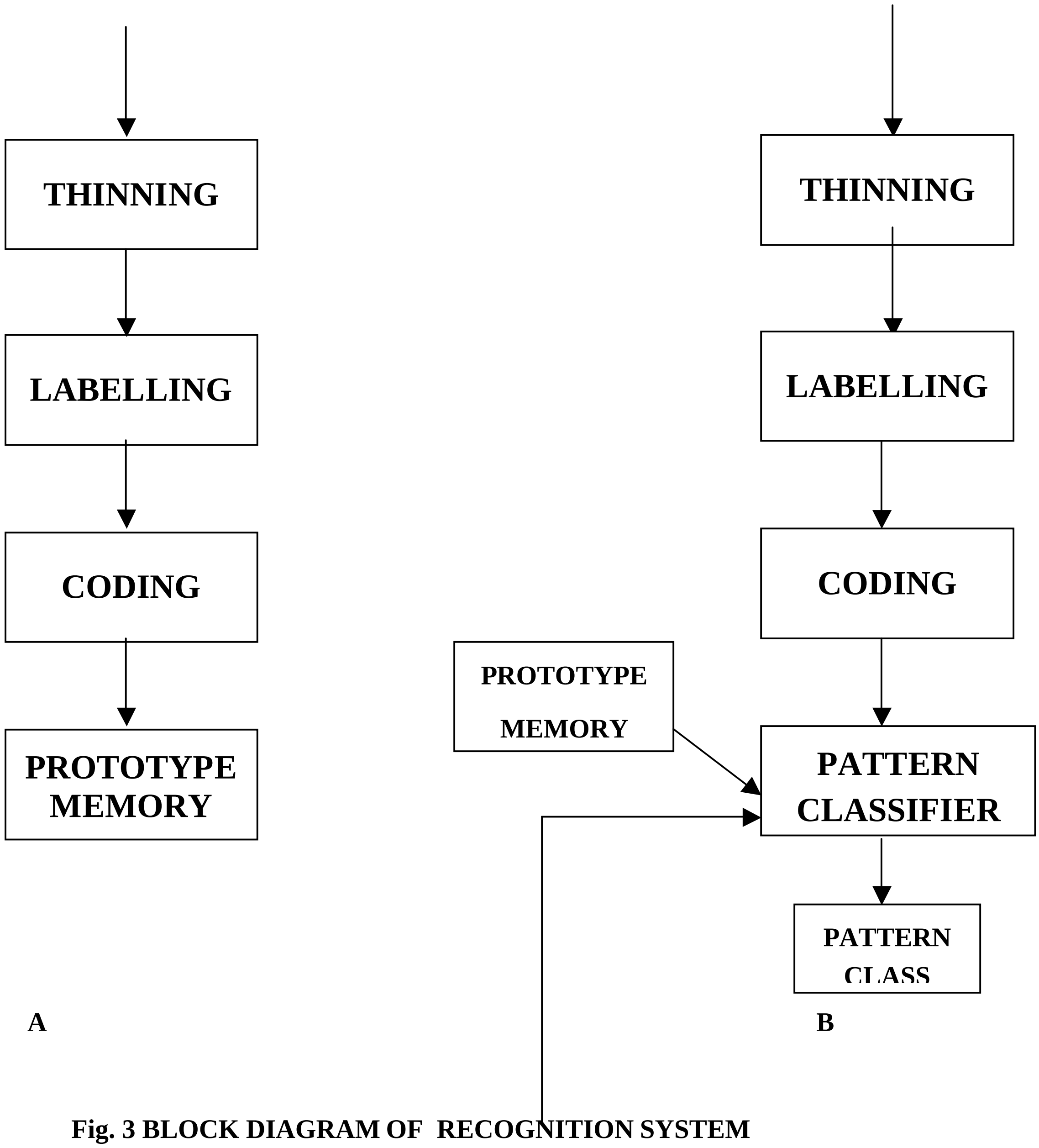


Fig. 3 BLOCK DIAGRAM OF RECOGNITION SYSTEM

A. LEARNING PHASE

B. RECOGNITION PHASE

Branch Detection and Labeling

measure of straightness

Measure of Orientation

System Operation And Experimentation Results

Learning Phase

Recognition Phase

Classification

Misclassification

Unclassification

Results

1.3 OUR METHODOLOGY

Our scope is limited to the recognition of a single character or digit.

1.3.1 PREPROCESSING/IMAGE EXTRACTION

pixel's value.

image header

BMP.H*/

/ start of bitmapfileheader */*

*/*size of the BMP file*/*

/ 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*/*

*/*start of bitmapinfoheader*/*

*/*Width of the bitmap in pixels*/*

*/*Height of the bitmap in pixels**

*/*No. of planes in the pixel must be 1*/*

*/*Indicates the number of bits per pixel */*

*/*Flag to indicate the type of compression*/*

/ size of compressed image */*

/ pixels per meter horizontally*/*

/ pixels per meter vertically*/*

/ number of colors used */*

/ number of important/prominent colors */*

/ color map starts here */*

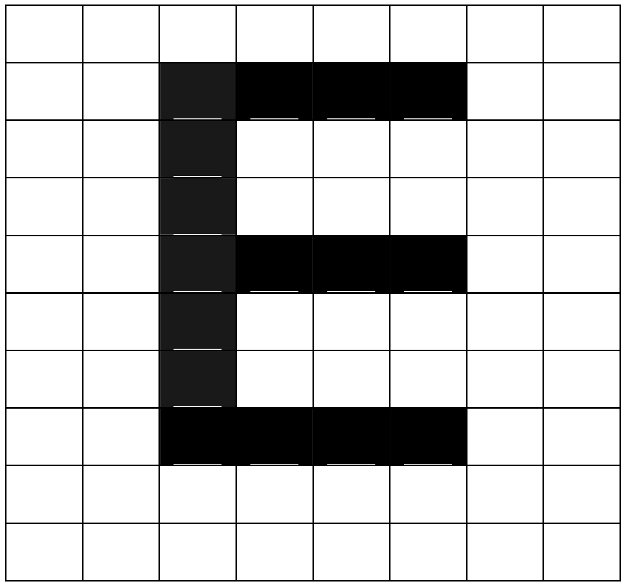


FIG 4: EXTRACTION OF A CHARACTER IN A MATRIX

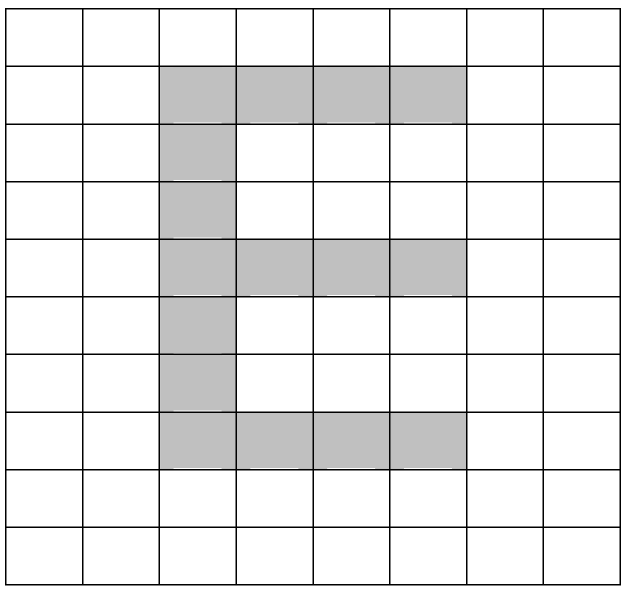


FIG 5: IDENTIFICATION OF BACKGROUND AND CHARACTER.

COLUMN	ROW

FIG 6 : INITIAL X-Y TABLE

1.3.2 FINDING THE CENTER OF THE CHARACTER

$\sum X$

$\sum Y$

center of reference.

$\leftarrow X+h$; where h=X

$\leftarrow Y+k$; where k=Y

X-Y

COLUMN	ROW

FIG 7 : X-Y TABLE

1.3.3 EXTRACTION OF DATA

characteristic vector

Quantification of Character

final

θ= 0,2α, 3α…k α, (n 1)α ; where n is a divisor of 360

characteristic values

Creation Of Square Frame

final X-Y

Calculation Of Characteristic Values And Formation Of The Characteristic Vector

To find the characteristic vector along a certain direction, a polar line θ= α is drawn to find 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

find characteristic values in other directions for different θ and hence the characteristic vector.

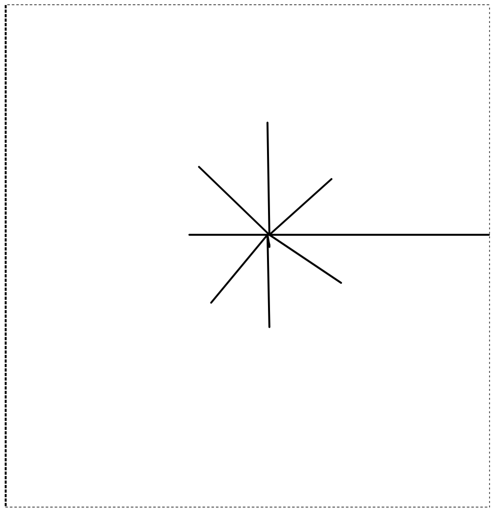


FIG 8 : CHARACTERISTIC VECTORS MEASURED FROM THE CENTER ‘O’

Creation Of An Ideal Database

Ideal Database

Character Recognition

Magnification

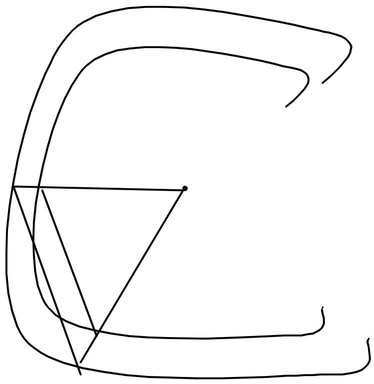


FIG 9: MAGNIFICATION OF CHARACTER ‘C’

Here “O” is the centroid of both the figures. In the figure , we have ΔOAB ≡ ΔODE,

′ = ′ = ′.............................. ′

′ are a particular characteristic value of the smaller and the larger character

Comparison Of The Characteristic Vector And Identification Of The Right Match

∨ j ϵ J ,

β

β.

‡

∅

∅ **ignorable symbol**

In our comparison here, we compare the unknown character β with those ideal characters only for

β

β β

$\epsilon\ E_\beta$

β

β

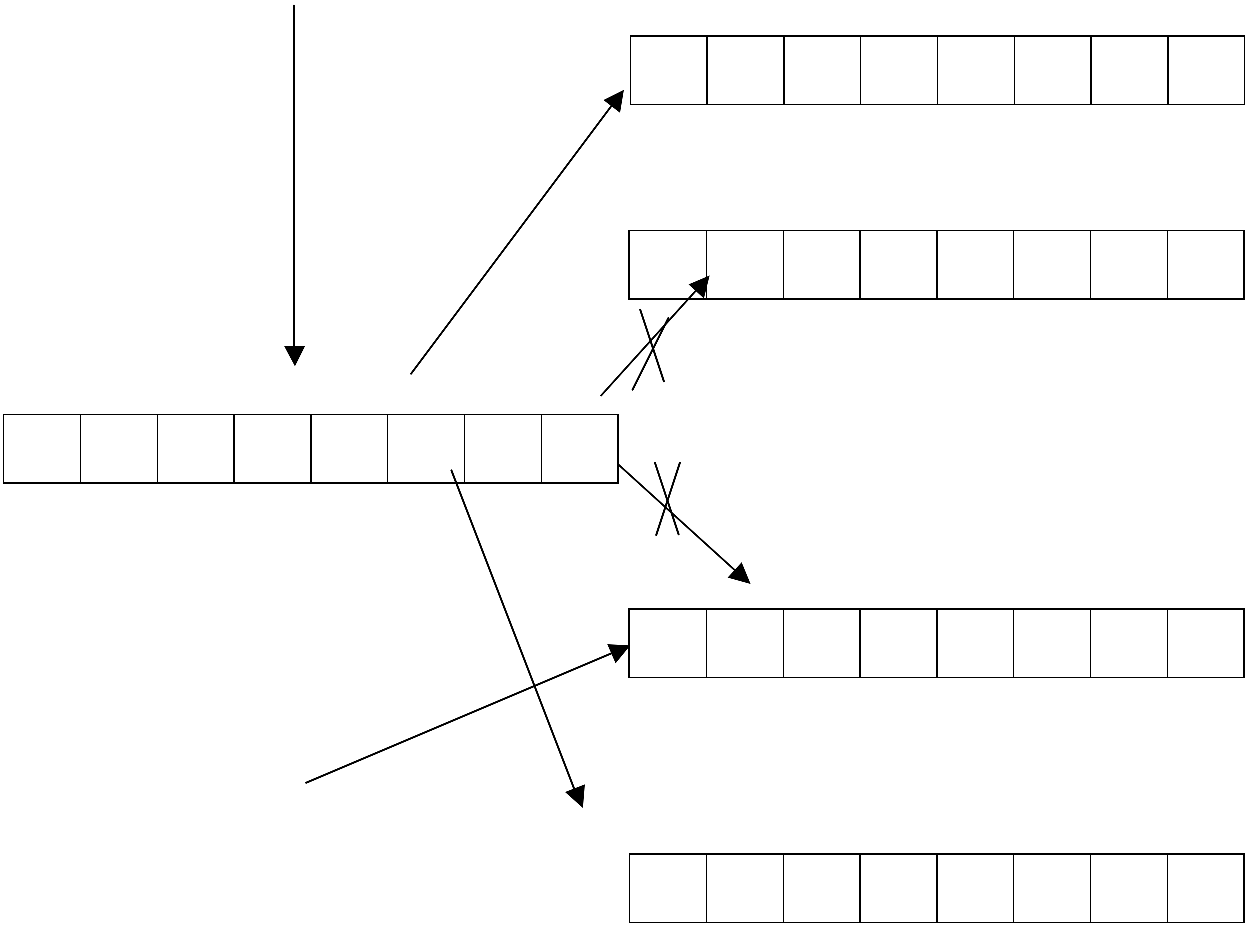


FIG 10: THE 2ND AND 3RD CHARACTERISTIC VECTORS ARE NOT CONSIDERED FOR COMPARISON

2. SOFTWARE REQUIREMENT ANALYSIS

2.1 INTRODUCTION (PROBLEM DISCUSSION AND ANALYSIS)

2.1.1 PURPOSE

“Character Recognition System Using Template Matching”

“Character Recognition System Using Template Matching”

- ✓
- ✓
- ✓

2.1.2 GOALS AND OBJECTIVES

“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

- ✓
- ✓
- ✓
- ✓
- ✓

2.2.2 GENERAL CONSTRAINTS

- ✓

-
  - ✓
    -

2.3 REQUIREMENTS

3. SYSTEM DESIGN

It provides the specification and design for system design showing flow of work, program and user functions

3.0 DESIGN OBJECTIVES

✓

✓

✓

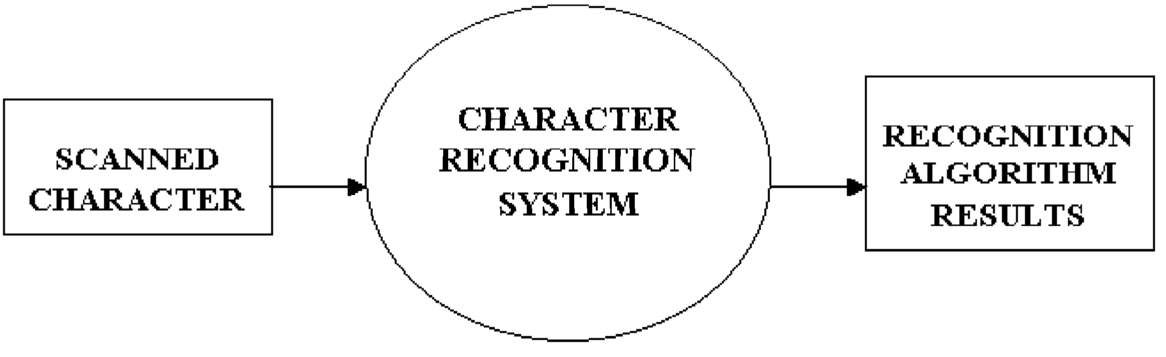
3.1 DESIGN DECISION

3.2 DATA FLOW DIAGRAM

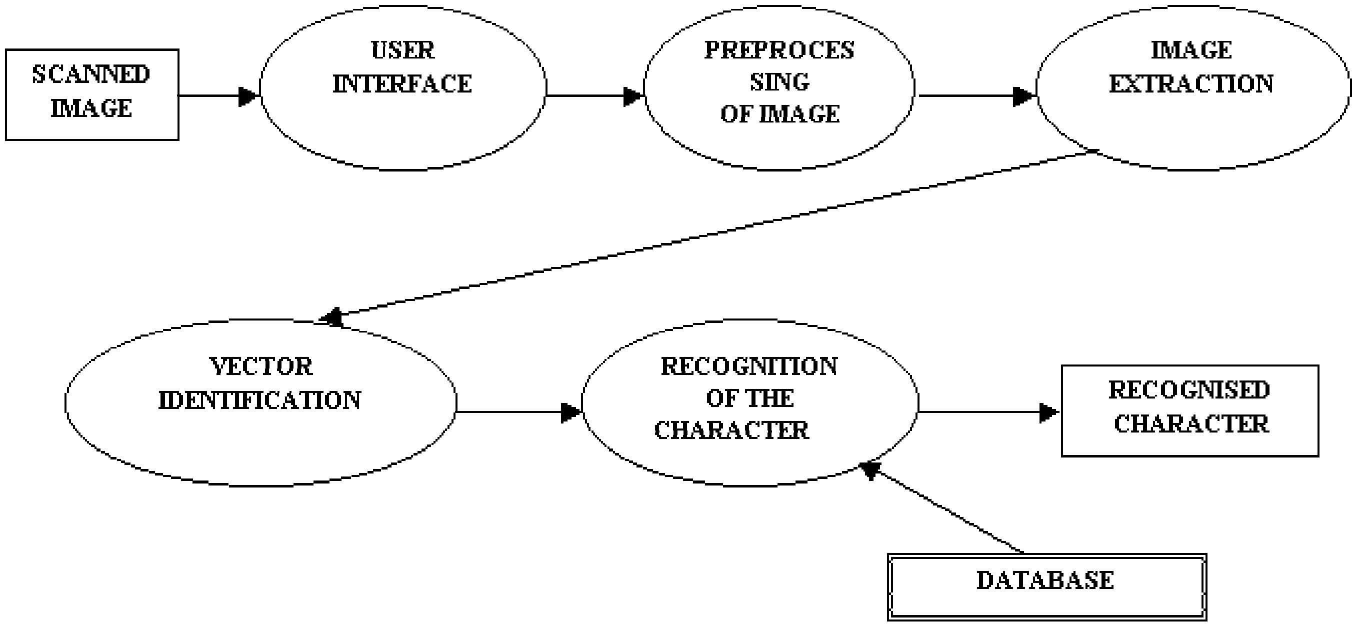
✓

✓

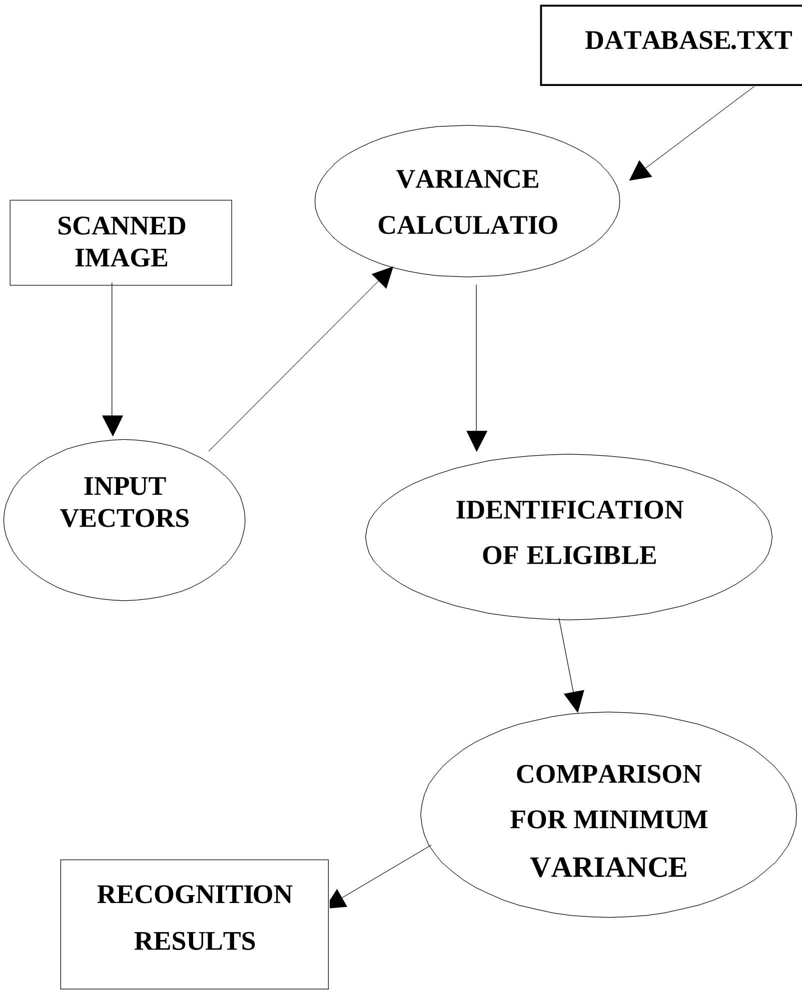
3.2.1 CONTEXT LEVEL DFD FOR THE SYSTEM



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

PSPEC: Preprocessing Of Image

PSPEC: Image Extraction

PSPEC: Vectors

PSPEC: Recognition

3.3 ARCHITECTURAL DESIGN

3.3.1 Software Components

3.3.2 Properties of Components

3.3.3 Summary

Architectural Design For Our System

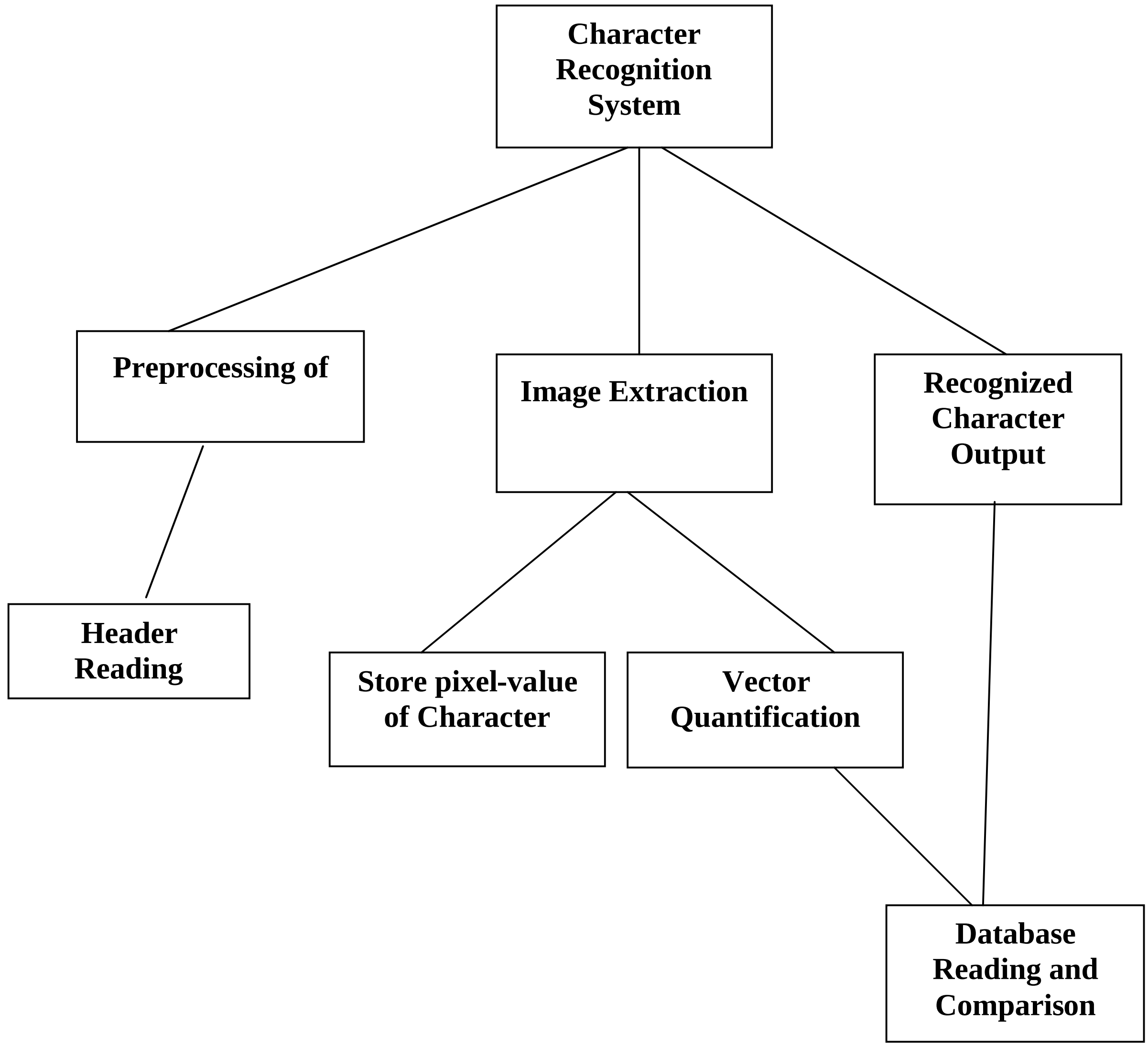


FIG 11 ARCHITECTURAL DESIGN

3.4 IMPLEMENTATION

3.4.1 CREATION OF DATABASE

A. Saving the image

B. Image Calling

C. BMP Header Reading

D. Image Reading

E. Identification of dominant color

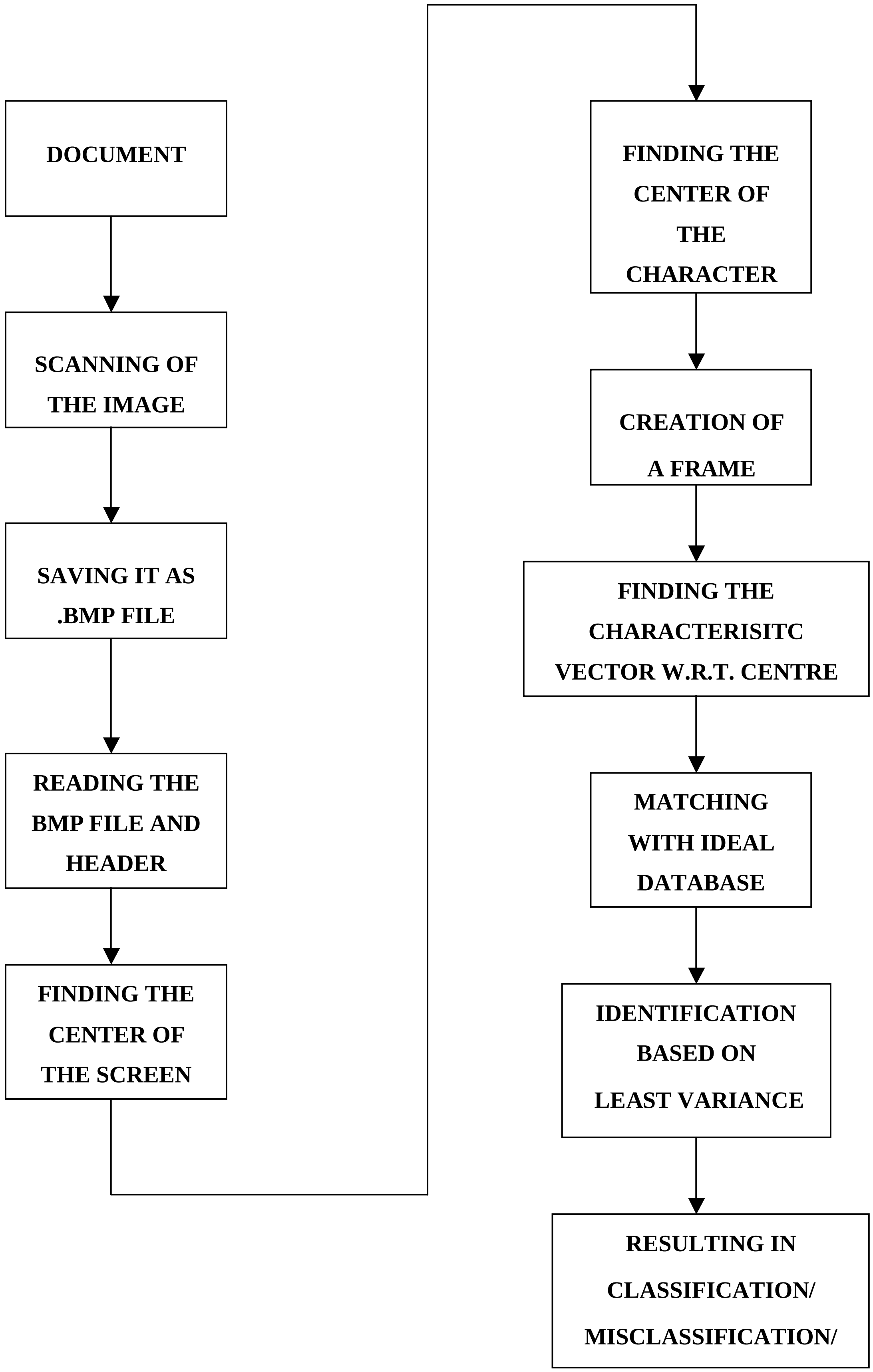


FIG 12 PROGRAM FLOW

F. Formation of Initial and Final X-Y table

G. Quantification of Character

θ ° °,

° ° ° ° ° ° °

H. Creation of a database file

database.txt

dataNUM.txt

3.4.2 RECOGNITION OF A CHARACTER

A. **A-H**

B.

C.

3.4.3 EXAMPLE

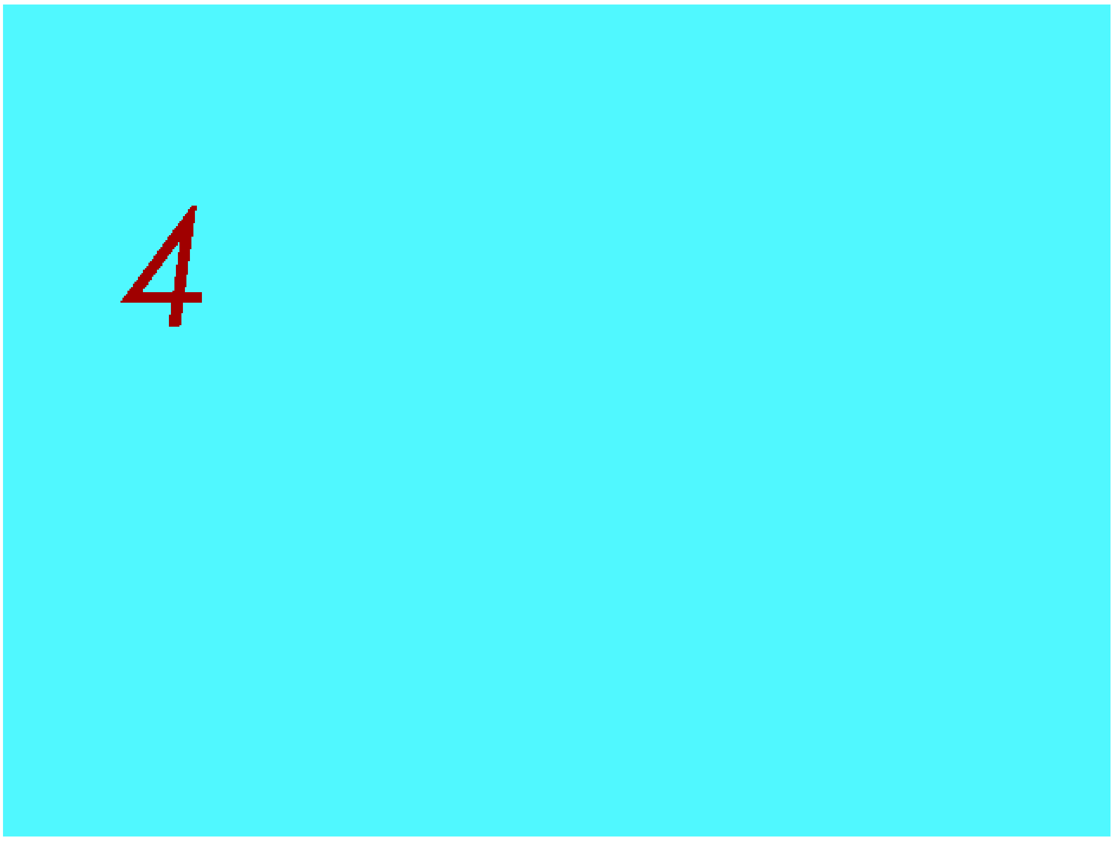


FIG 13: THE DIGIT 4 STORED IN THE MEMORY IS ACCESSED AND OPENED BY THE C ROUTINE IN A PROGRAM WINDOW.

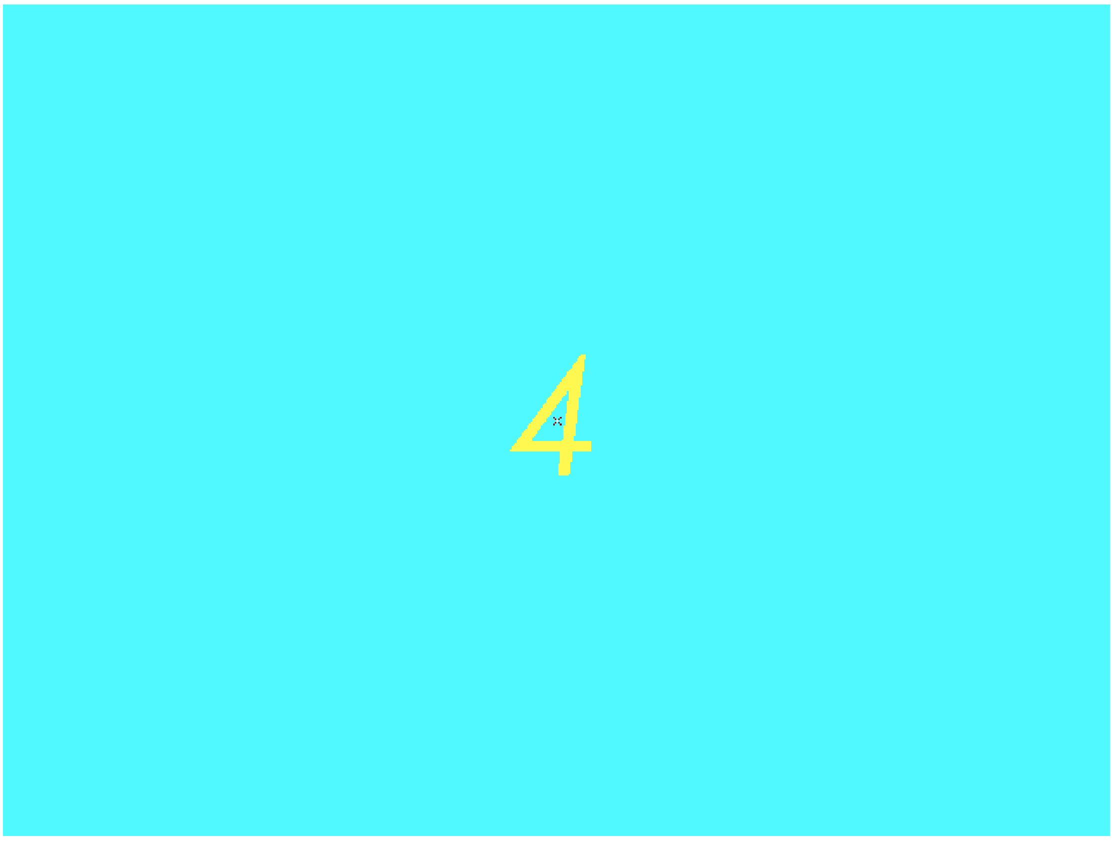


FIG 14: THE DIGIT IS LOGICALLY SHIFTED TO THE CENTER OF THE SCREEN.

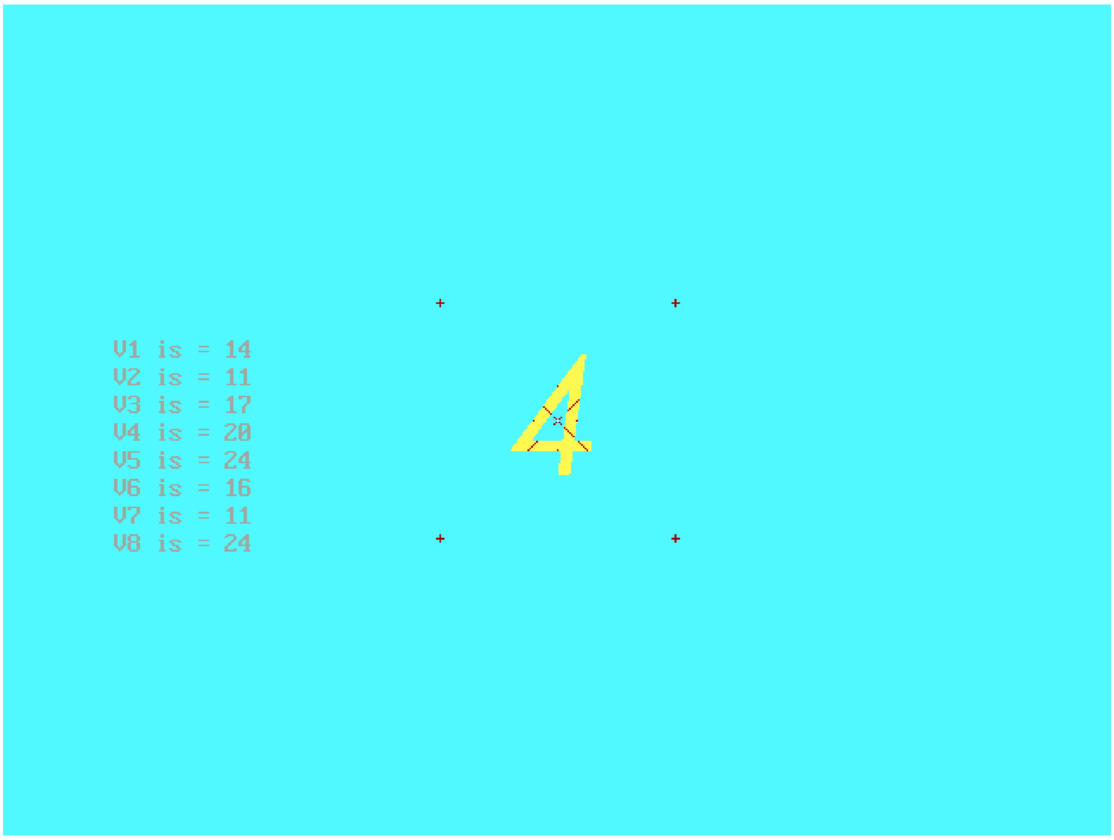


FIG 15: CREATION OF SQUARE FRAME AND QUANTIFICATION OF DIGIT IN THE FORM OF 8 VECTORS.

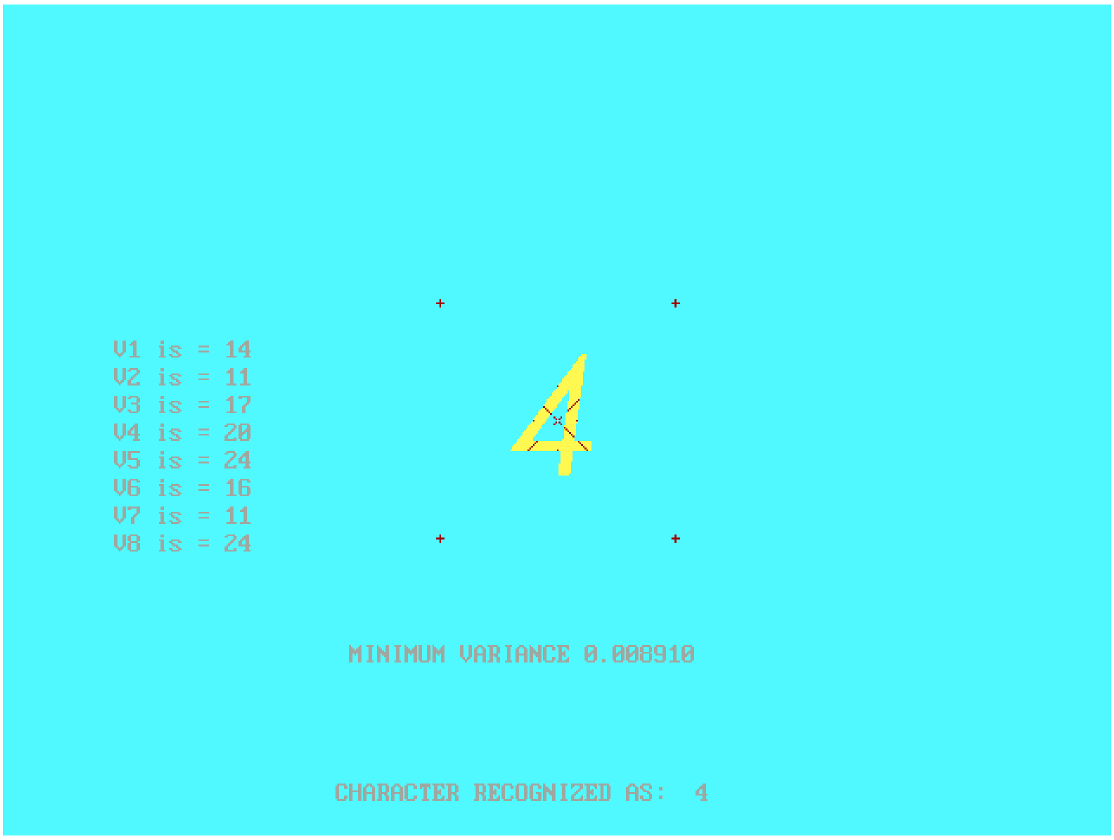


FIG 16 :CALCULATING VARIANCE AND DISPLAYING RECOGNITION RESULTS.

3.4.4 SOME SAMPLE RECOGNITION RESULTS

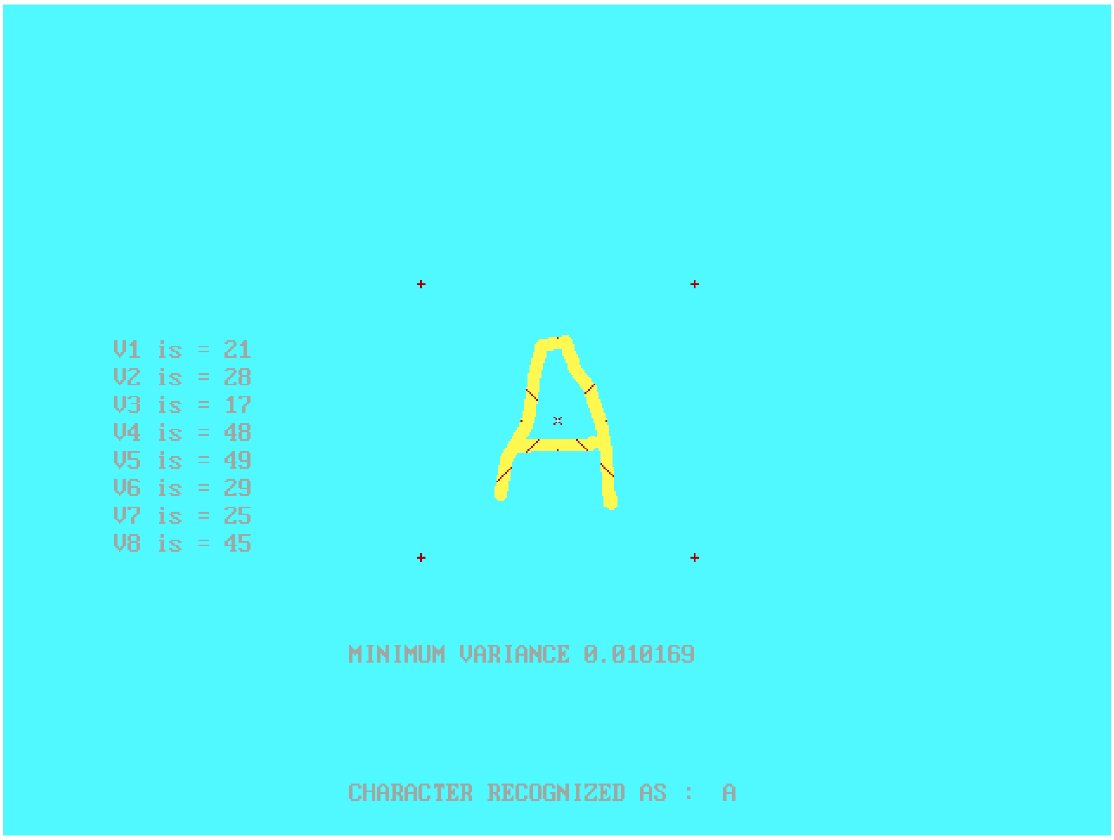


FIG 17 : SAMPLE OF HANDWRITTEN CHARACTER 'A'

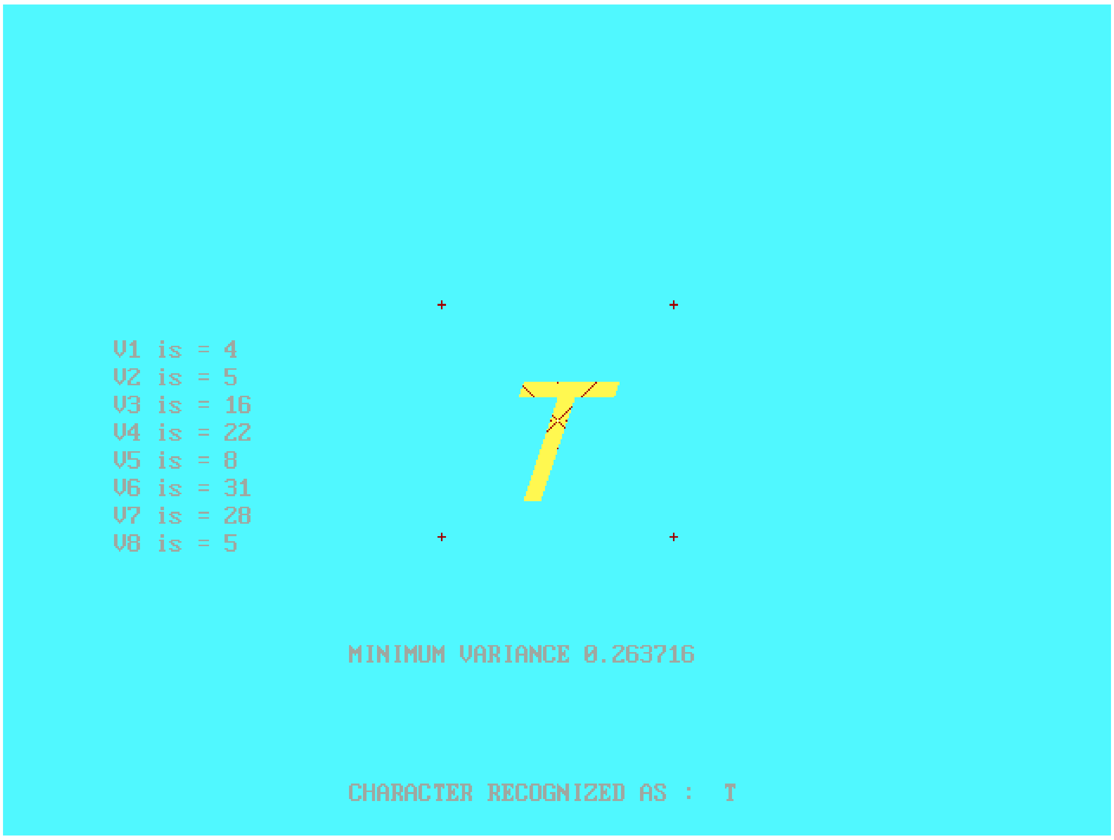


FIG 18 : SAMPLE OF AN ALPHABET IN ITALICS

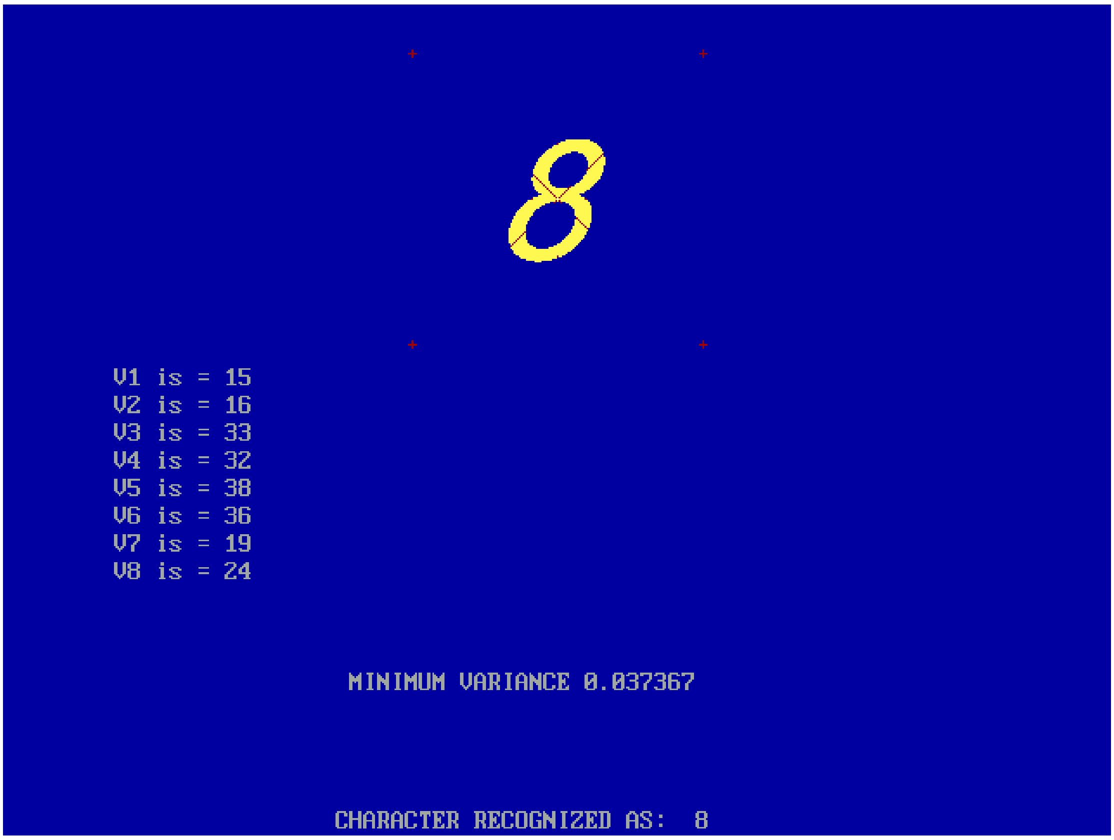


FIG 19 SAMPLE OF A DIGIT IN ITALICS

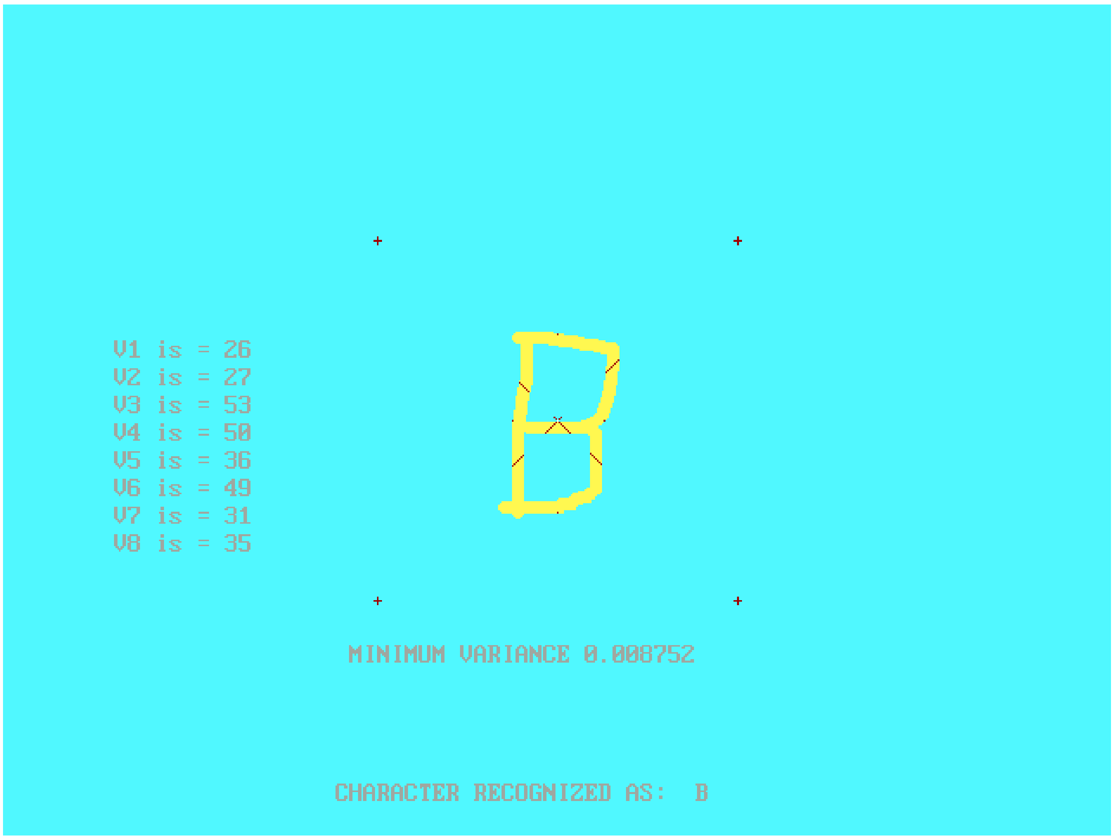


FIG 20 SAMPLE OF HANDWRITTEN ALPHABET ‘B’

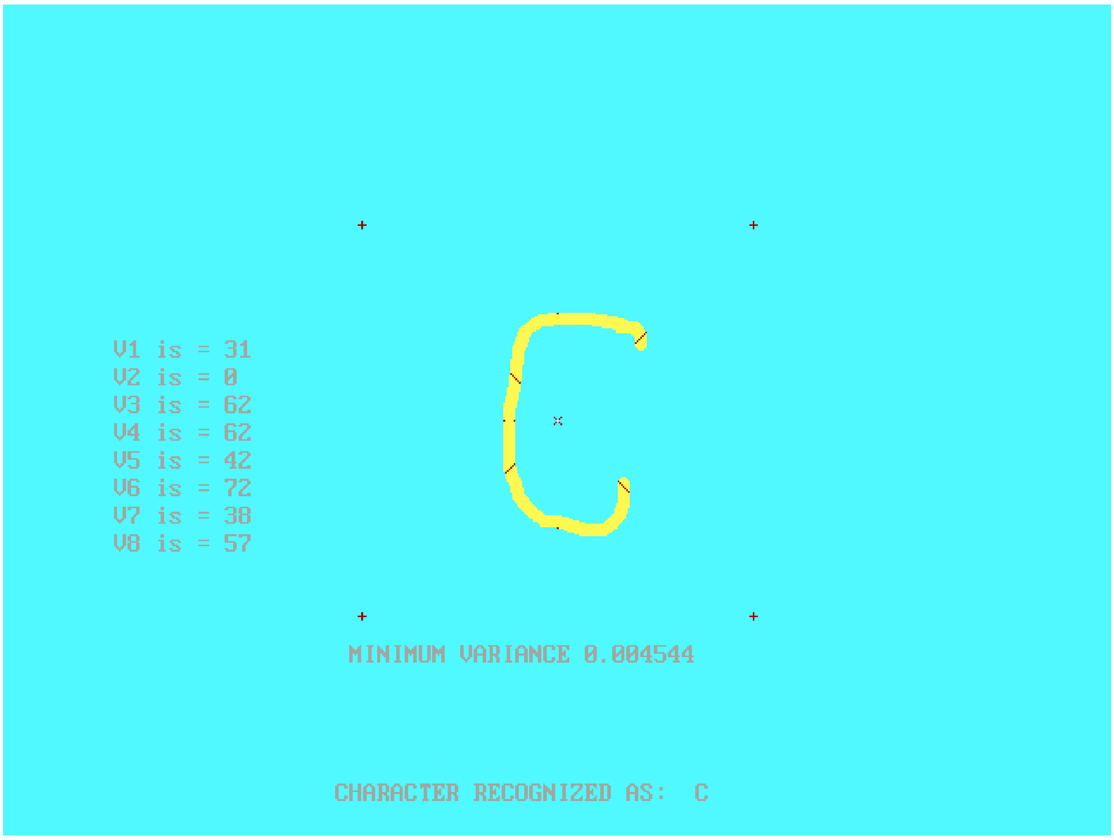


FIG 21 : SAMPLE OF HANDWRITTEN ALPHABET 'C'

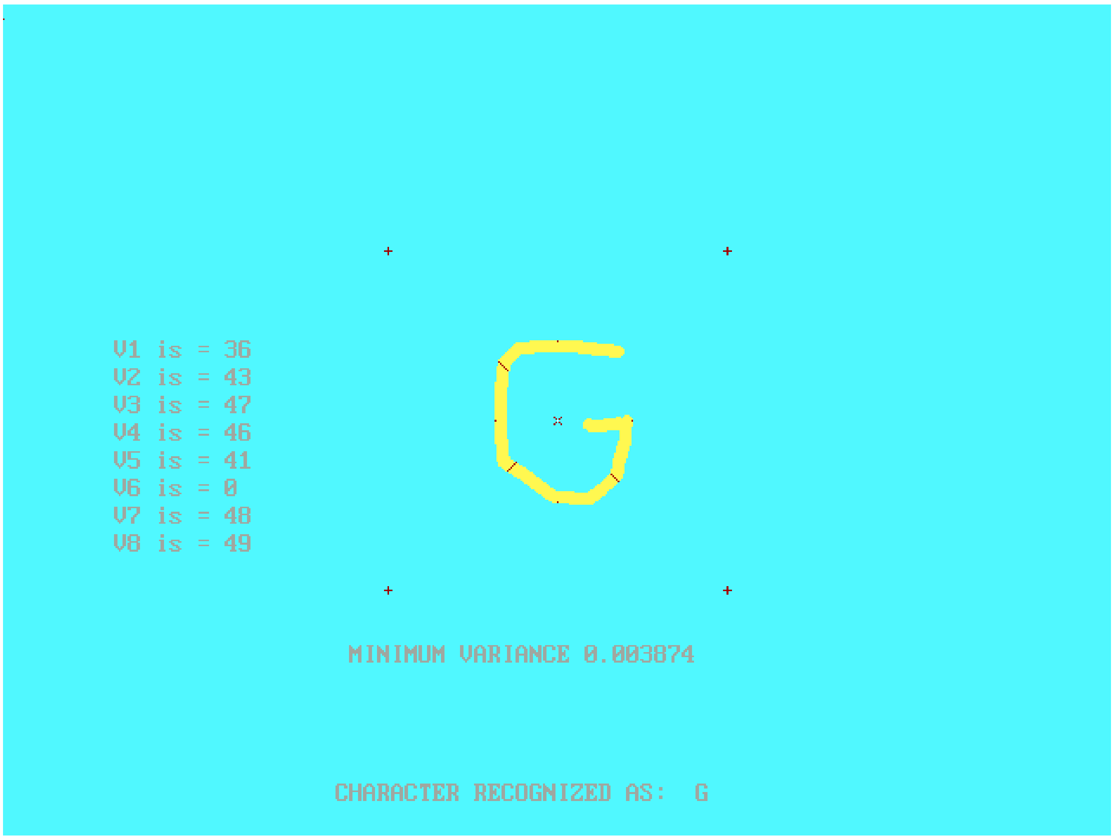


FIG 22 : SAMPLE OF HANDWRITTEN ALPHABET 'G'

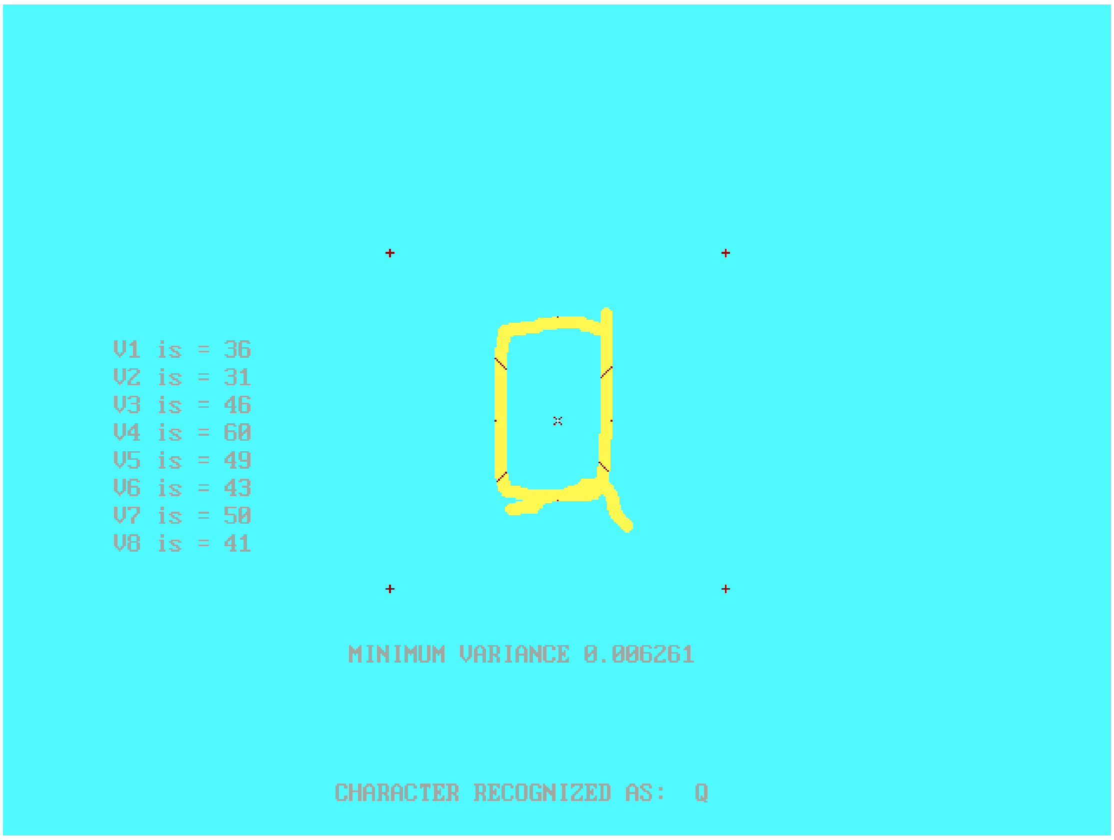


FIG 23 : SAMPLE OF HANDWRITTEN ALPHABET 'Q'

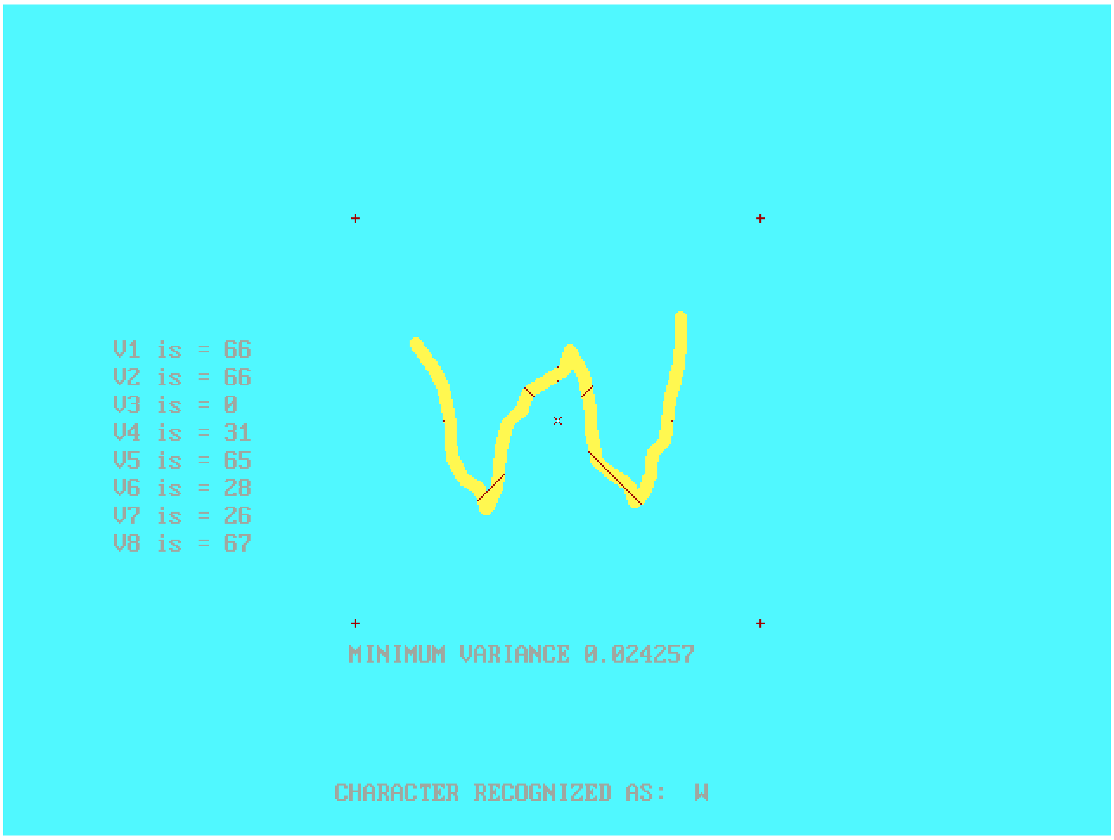


FIG 24 SAMPLE OF HANDWRITTEN ALPHABET 'W'

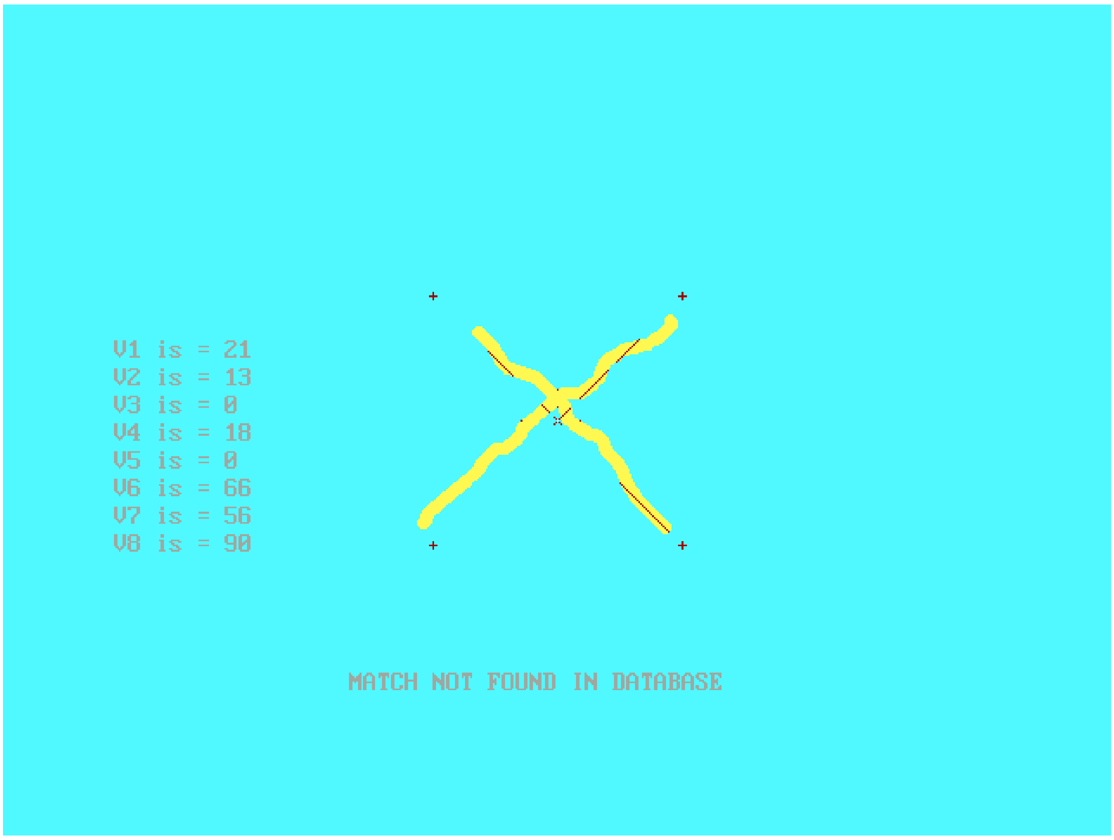


FIG 25: SAMPLE OF HANDWRITTEN ALPHABET 'X'.

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);
```

```
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;
     }
```

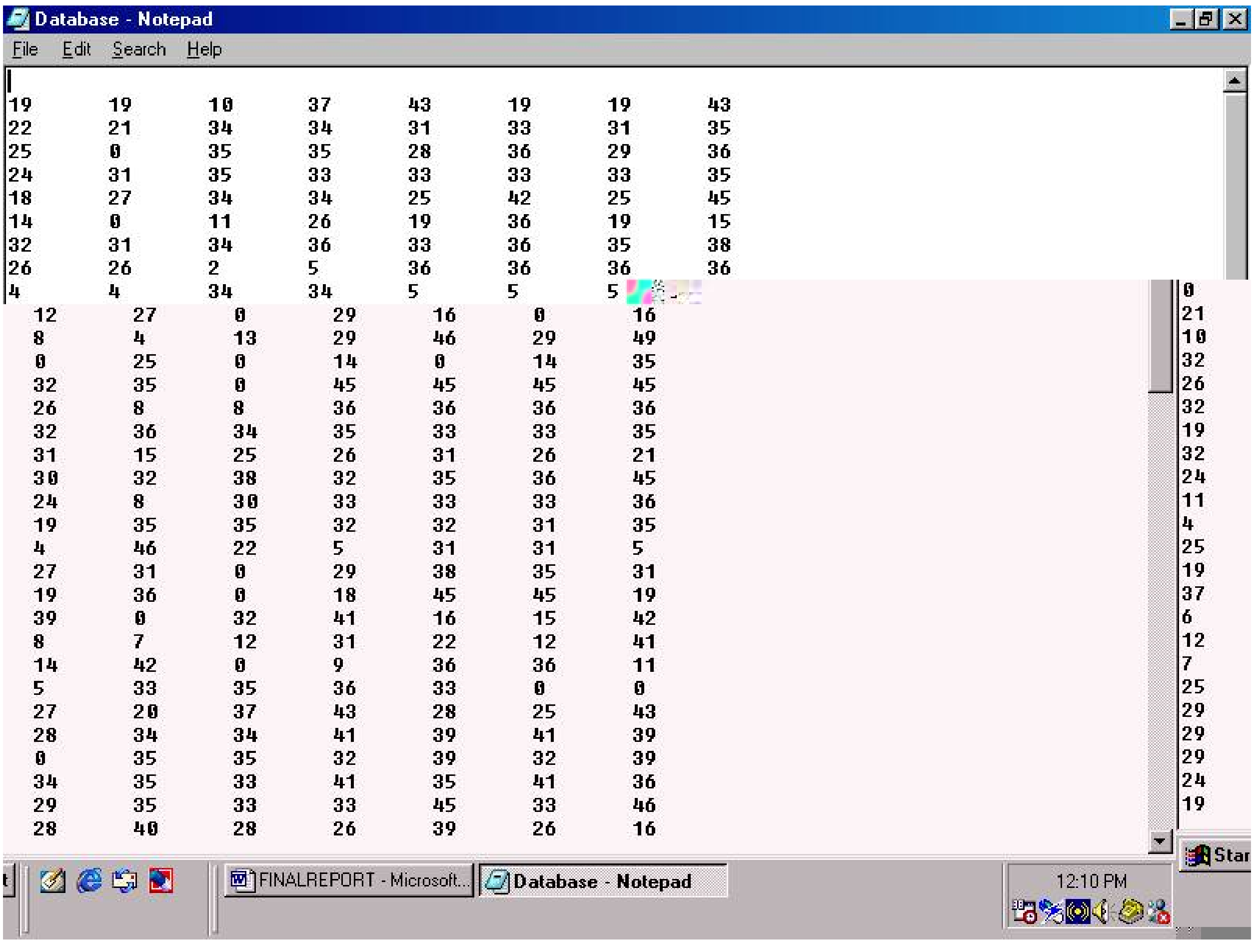
```
}
float comp1[150][9];
int I1[8];
for(i=0;i<=8;i++)
I1[i]=0;
for(l=0;l<=t-1;l++)
{
comp1[l][0]=comp[l][0];
a=1;
for(n=1;n<=8;n++)
    {   if(I[n]!=0)
       {
        comp1[l][a]=comp[l][n];
        I1[a]=I[n];
         a=a+1;
   }
     }
}
for(l=0;l<=t-1;l++)
{
for(n=1;n<=a-1;n++)
    { comp1[l][n]=comp1[l][n]/(float)I1[n];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

- ✓
- ✓




```
19	19	10	37	43	19	19	43
22	21	34	34	31	33	31	35
25	0	35	35	28	36	29	36
24	31	35	33	33	33	33	35
18	27	34	34	25	42	25	45
14	0	11	26	19	36	19	15
32	31	34	36	33	36	35	38
26	26	2	5	36	36	36	36
4	4	34	34	5	5	5		0
 12	27	0	29	16	0	16		21
 8	4	13	29	46	29	49		10
 0	25	0	14	0	14	35		32
 32	35	0	45	45	45	45		26
 26	8	8	36	36	36	36		32
 32	36	34	35	33	33	35		19
 31	15	25	26	31	26	21		32
 30	32	38	32	35	36	45		24
 24	8	30	33	33	33	36		11
 19	35	35	32	32	31	35		4
 4	46	22	5	31	31	5		25
 27	31	0	29	38	35	31		19
 19	36	0	18	45	45	19		37
 39	0	32	41	16	15	42		6
 8	7	12	31	22	12	41		12
 14	42	0	9	36	36	11		7
 5	33	35	36	33	0	0		25
 27	20	37	43	28	25	43		29
 28	34	34	41	39	41	39		29
 0	35	35	32	39	32	39		29
 34	35	33	41	35	41	36		24
 29	35	33	33	45	33	46		19
 28	40	28	26	39	26	16
```

FIG 27 : THE DATABASE FILE

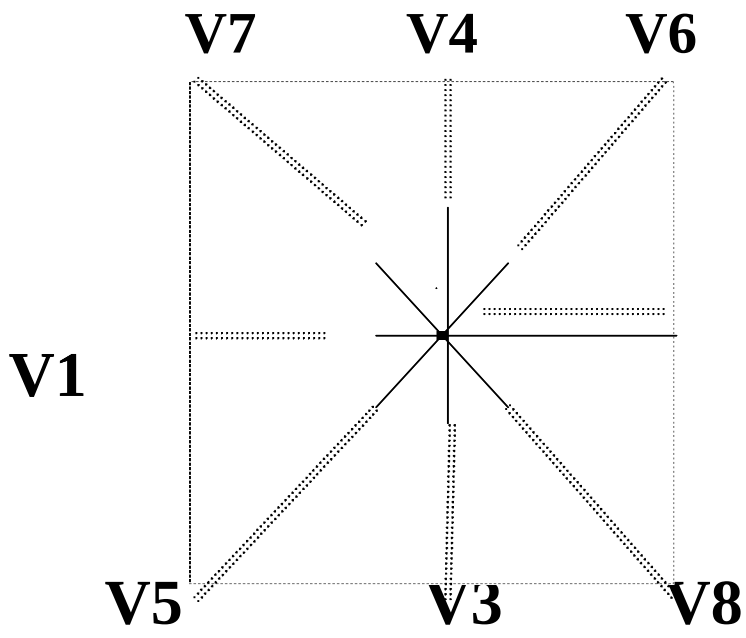


FIG 28 : THE ORDER OF VECTORS AS STORED IN THE DATABASE

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				
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				
L				
M				
N				
O				
P				
Q				
R				
S				
T				
U				
V				
W				
X				
Y				
Z				

94.30%

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

	NUMBER OF SAMPLES	CORRECTLY RECOGNIZED	MIS-MATCHED	UNMATCHED
J				
K				
L				
M				
N				
O				
P				
Q				
R				
S				
T				
U				
V				
W				
X				
Y				
Z				

88.02%

6.3 RESULTS OF SOME HAND-WRITTEN ALPHABET

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

75.42%

6.4 RESULTS OF SOME TYPE-WRITTEN NUMERALS OF STANDARD FONT

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

98.00%

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

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

82.00%

6.6 SUMMARY OF RESULT

6.6.1 ALPHABETS

94.30 %

88.02 %

75.42 %

6.6.2 NUMERALS

98.00 %

82.00 %

7. SCOPE FOR FURTHER IMPROVEMENTS

8. TYPES OF CHARACTER RECOGNITION SYSTEMS AND THEIR POTENTIAL APPLICATIONS

-
-

8.0 TASK SPECIFIC READERS

-
-
-
-
-
-

8.0.1 FORM READERS

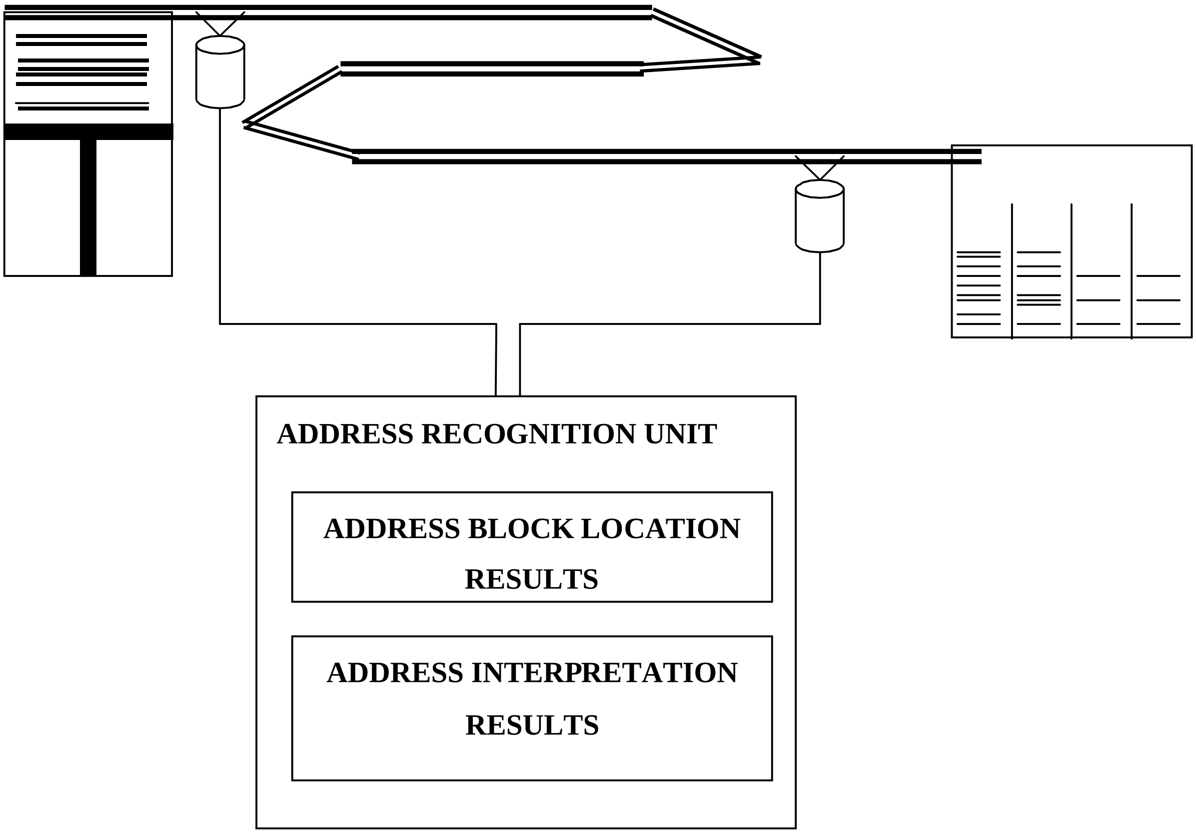
8.0.2 CHECK READERS

8.0.3 BILL PROCESSING SYSTEMS

8.0.4 AIRLINE TICKET READERS

8.0.5 PASSPORT READERS

8.0.6 ADDRESS READERS



8.1 GENERAL PURPOSE PAGE READERS

9. APPENDIX

9.0 BACKGROUND (FURTHER DETAILS)

9.0.1 COLOR MODELS

Additive (RGB)

Subtractive (CMY)

9.0.2 LUMINANCE

9.0.3 COMPRESSION TYPES

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

Run-Length Encoding (RLE)

Huffman Codes

Huffman

Codes

Dictionary Systems

9.0.4 PBM APPROACH (Detailed)

PBM Format (Single Bit)

PGM Format (Grayscale)

PPM Format (Color)

9.1 CASE STUDIES OF DIFFERENT ONLINE CR SYSTEMS

9.1.1 VISUAL INPUT FOR PEN BASED COMPUTERS [7]

Initialization And Preprocessing

Pen Tip Detector

Filter / Predictor

Ballpoint Detector

Pen Up/Down Classifier

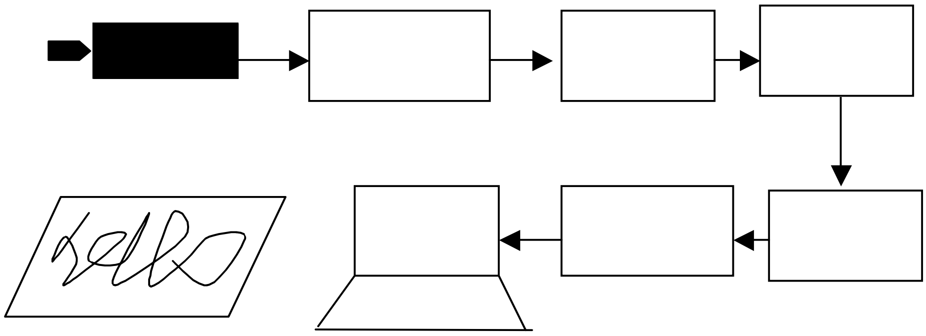


FIG 30 DIAGRAMMATIC REPRESENTATION OF THE METHOD

Implementation

9.1.2 "ONLINE RECOGNITION OF HANDWRITTEN SYMBOLS"[8]

Implementation

9.2 SOME MORE SAMPLE SCREENS

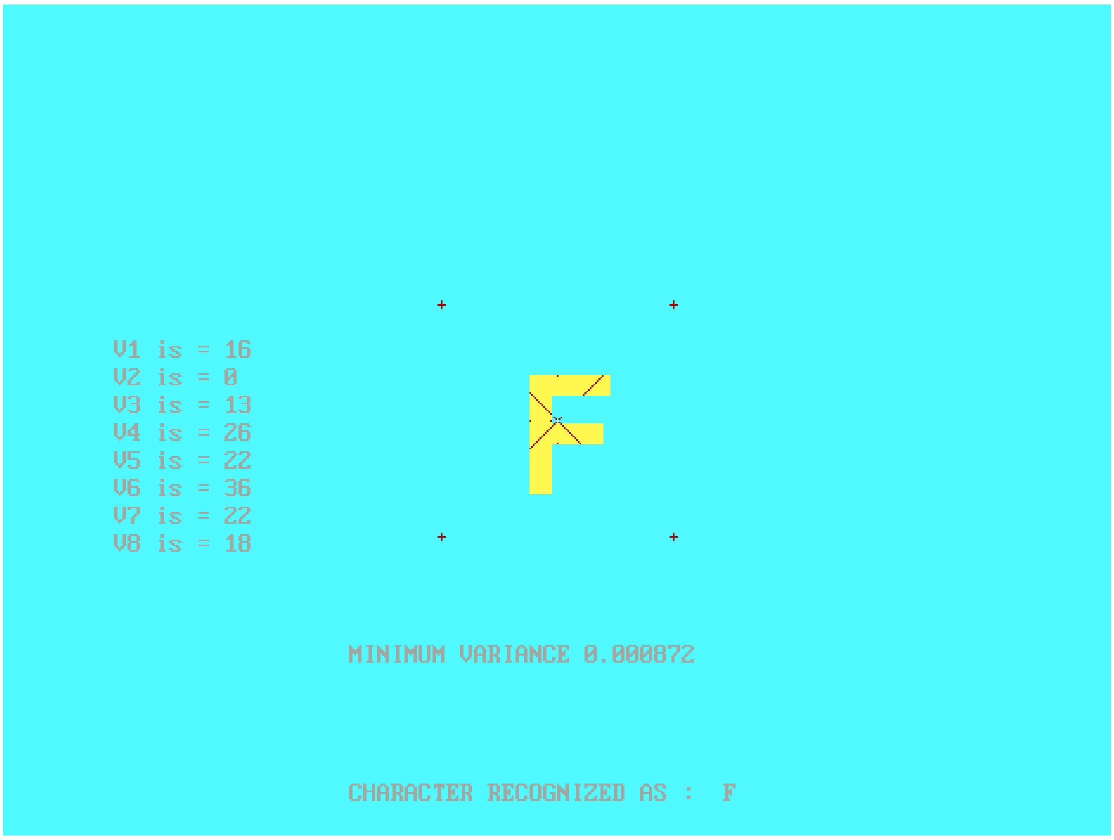


FIG 30 SAMPLE OF A TYPEWRITTEN CHARACTER 'F'

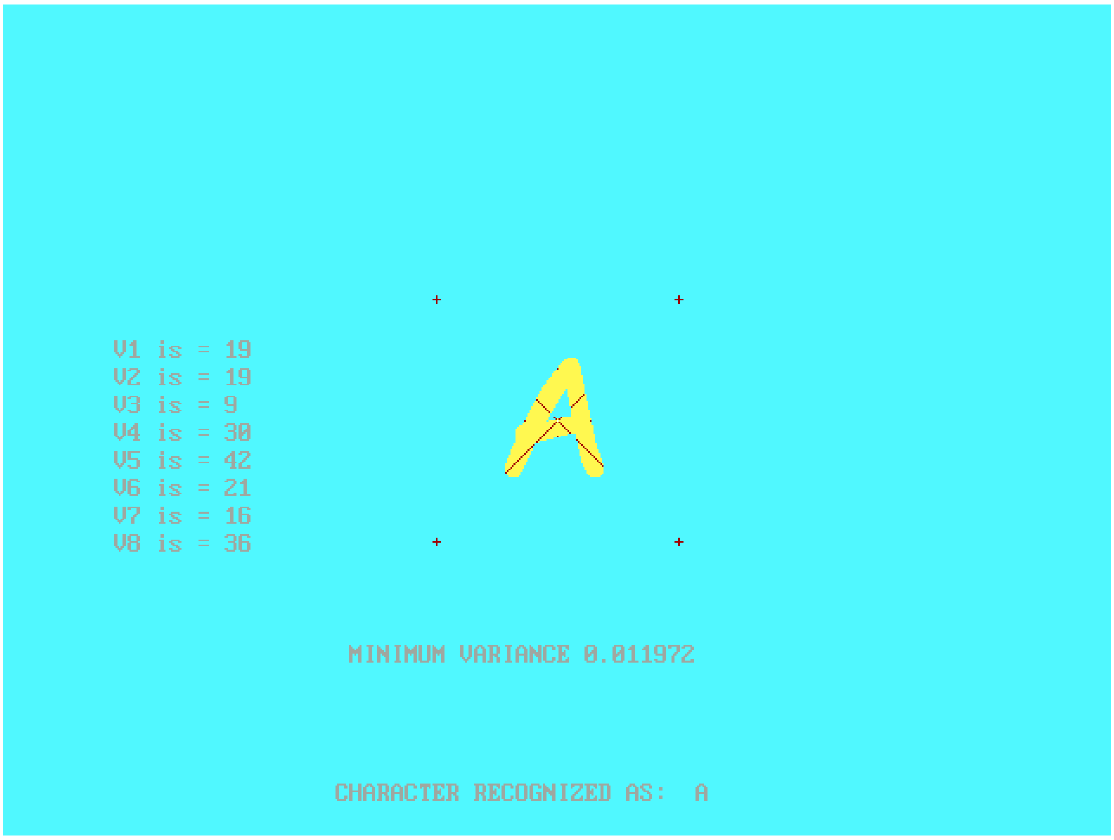


FIG 31 SAMPLE OF ALPHABET 'A'

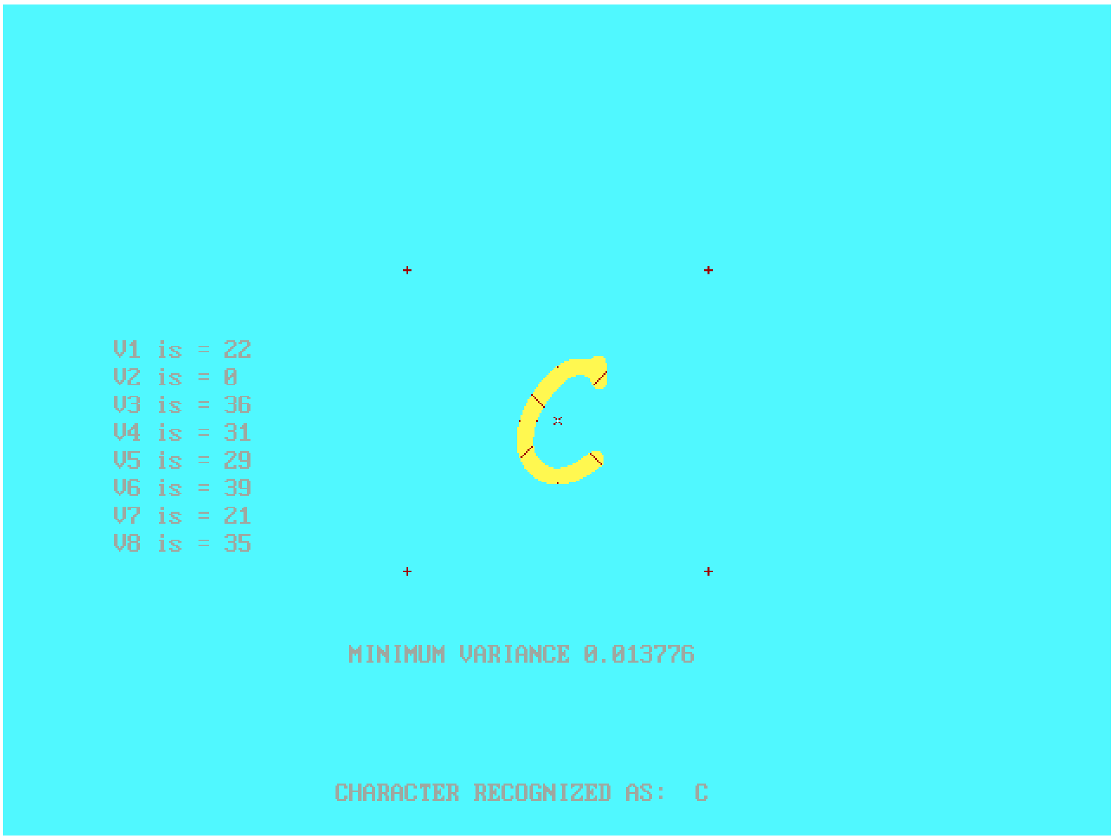


FIG 32 SAMPLE OF ALPHABET 'C'

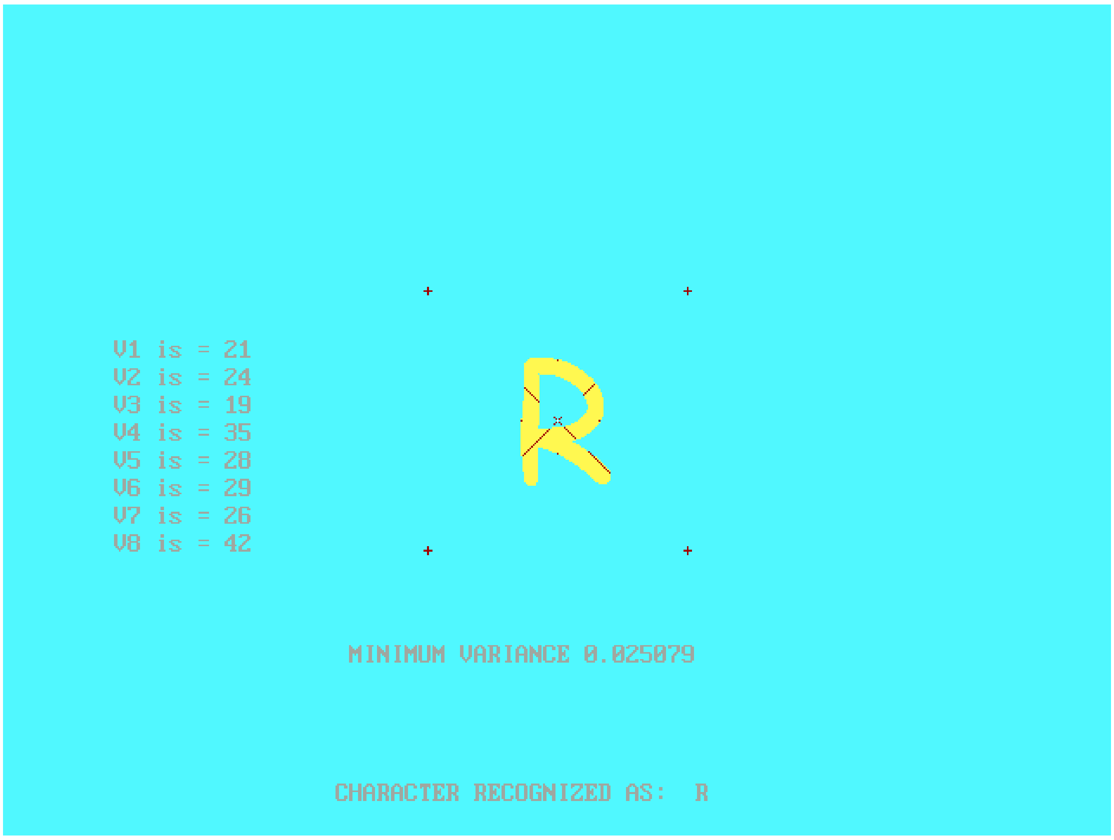


FIG 33 SAMPLE OF ALPHABET 'R'

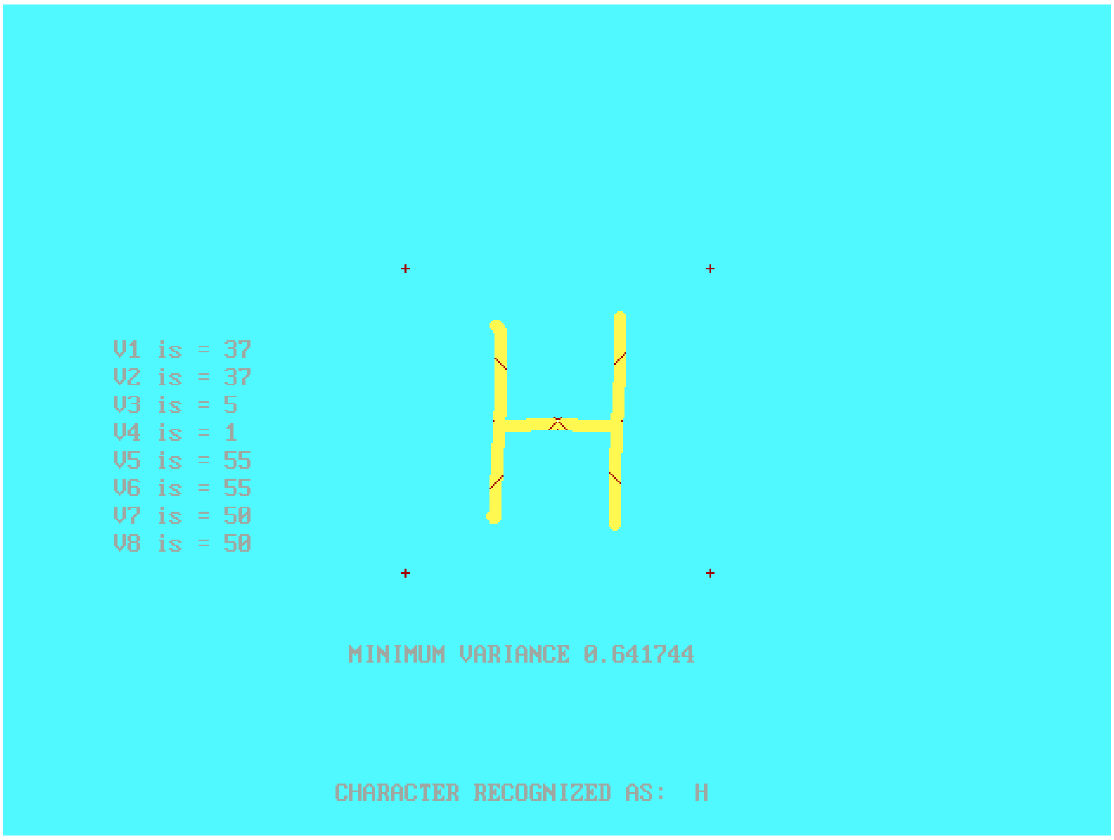


FIG 34 SAMPLE OF HANDWRITTEN ALPHABET 'H'

10. REFERENCES