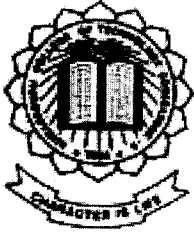


HANDWRITING RECOGNITION SYSTEM



P-1175



Project Report

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

Bachelor of Engineering in Information Technology
Bharathiyar University, Coimbatore.

Submitted by

Arun Kumaran M.
0027S0066

Sreenidhi V.
0027S0111

Under the guidance of

Mr. M. Nageswara Gupta B.E.,
Lecturer

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

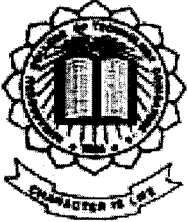
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE – 641006.

MARCH 2004.

DEPARTMENT OF INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University, Coimbatore)



CERTIFICATE

This is to certify that the project entitled

HANDWRITING RECOGNITION SYSTEM

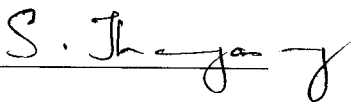
done by

Arun Kumaran M.
0027S0066

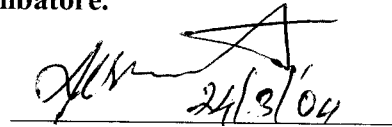
Sreenidhi V.
0027S0111

Submitted in partial fulfillment of the
Requirement for the award of the degree of the

**Bachelor of Engineering in Information Technology of
Bharathiar University, Coimbatore.**

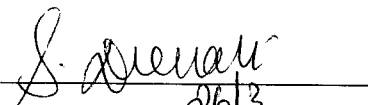



**Professor & Head of the department
(Dr. S. Thangasamy)**



**Project Guide
(Mr. M. Nageswara Gupta)**

Certified that the candidates were examined by us in the project work
Viva voce examination held on 26.3.2004.


26/3
Internal Examiner


External Examiner

]

Declaration

Declaration

We,

Arun Kumaran M.
Sreenidhi V.


0027S0066
0027S0111


declare that we did the project entitled “**Handwriting Recognition System**”, and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted in partial fulfillment of the requirements for the Award of the degree of Bachelor of Engineering In Information Technology of Bharathiar university.

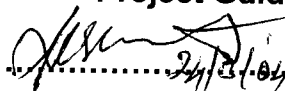
Place: Coimbatore.

Date: 24/03/04


[Arun Kumaran M.]


[Sreenidhi V.]

Project Guided by


.....24/3/04.....

Mr. M. Nageswara Guptha B.E.

Acknowledgement

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr .K. K. Padmanabhan Ph.D., for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department, .Dr. S. Thangasamy Ph.D., whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide Mr. M. Nageswara Guptha B.E., Lecturer, Department of Computer Science and Engineering for his excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honor him for his assistance towards this cause.

We also thank our project coordinator Mrs. S. Devaki M.S., and our beloved class advisor Ms. P. Sudha B.E., for their invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

Handwriting Recognition System is developed to recognize and to convert non-cursive handwritten text into electronic form for word processing purpose.

Here the input is given as an image file that contains the text to be recognized. The system processes this image to extract the characters in it, recognizes the characters and writes them in to a text file. Various graphical transformations such as scaling, translation, and cropping are used to process the image files.

Already existing systems have too many constraints which have been tried to overcome in this system. Pattern Recognition is the area which is of prime importance to researchers now and in this system we have implemented research algorithms in this area to recognize the characters.

CONTENTS

1.	Introduction	1
	1.1. Existing System and Limitations	2
	1.2. Proposed System and Advantages	2
2.	System Requirement Analysis	3
	2.1. Product Definition	4
3.	Software Requirements Specification	5
	3.1. Introduction	6
	3.2. General Description	7
	3.3. Functional Requirements	9
	3.4. External Interface Requirements	12
	3.5. Design Constraints	12
4.	System Design Specification	14
	4.1. Overall Design	15
	4.2. Design Diagram	15
	4.3. The Extraction Algorithm	16
	4.4. The Comparison Algorithm	21
	4.5. The Matching Algorithm	22
5.	Product Testing	24
	5.1. Unit Testing	25
	5.2. Integrity Testing	25
	5.3. Output Testing	26
6.	Future Enhancements	27
7.	Conclusion	29
8.	References	31
9.	Annexure	33
	9.1. Source Code	34

1. Introduction

Pattern Recognition is one area into which tremendous amount of research has gone in. We in this project have aimed at producing a Handwriting Recognition System by using these researches, which would recognize and convert handwritten text into electronic text which could be manipulated on the computer. This is an effort to produce a Handwriting Recognition System that would produce better results than those which currently exist.

1.1. Existing system and limitations

The limitation of the existing systems is that input can be given as character in specific area of the input form called as grids. Also characters in specific areas need to be of fixed size. These constraints make the system all the more difficult for the user to use it.

1.2. Proposed system and advantages

Our software lets the user to give characters at any point in the image also only specifying an upper limit on the size of the character. Also the minor constraint is that there is a finite gap between two letters. These limited constraints make the system all the more effective and opens new doors in this fixed field of character recognition.

2. System Requirement Analysis

2.1. Product Definition

Handwriting Recognition System extracts and recognizes handwritten text and then writes them into a text file for effective word processing purpose. Here the input is taken in the form of an image file that contains the text. The letters of the text are then taken into individual image file. Research algorithm is used on these image files to find the appropriate character. The recognized characters are then written into a text file.

3. Software Requirements Specification

3.1. Introduction

3.1.1. Purpose

The purpose of this system is to convert non-cursive handwritten text into editable form (a text file). The input is taken in the form of an image file that represents the scanned form of text written on a paper. The letters are then split and then processed to find what letter their pattern matches and then write the letter in a text file. This project is aimed at easy conversion of text from the paper into electronic form. If successful this project will stay as a harbinger to the projects that are done to minimize the manual work in converting hardcopies of text in to softcopies.

3.1.2. Scope

The handwriting recognition system, when completed will be useful for any user who would need to convert text on paper into electronic form. To enable this, the project incorporates the following features:

A easy to use Graphical User Interface.

A robust system that could recognize and convert handwriting of various forms.

3.1.3. Definitions, Acronyms and Abbreviations

Bmp image Bitmap image.

GUI Graphical User Interface.

SDD Software Design Description.

SRS Software Requirements Specification.

Scanning The process of traversing each pixel of the image to find its value.

Scaling The process of altering the size of a graphical entity without deforming it.

Translation The process of repositioning an entity along a straight line path from one coordinate location to another.

3.2. General description

3.2.1. Product Perspective

The main feature of this project will be the implementation of research algorithms to recognize and convert handwritten text into electronic form. These algorithms will be capable of recognizing the letters of various forms that maybe influenced by the inherent handwriting qualities of each user.

3.2.2. Product Functions

The function of this system is to scan the image file (bitmap form) given as the input by the user perform various graphical operations on the image file such as cropping, translation and scaling to split the individual letters of the file. Then the research algorithms help in recognizing each letter of the file and then the output is produced into a text file that will contain the text that is present in the initial bitmap file

3.2.3. User Characteristics

The user need not be familiar with any programming languages. He should be able to give the input in the form of a bitmap form and have sufficient knowledge to use the GUI.

3.2.4. General Constraints

The basic constraint is that the user should enter a valid bitmap file of size 64x64 resolution for the system, which is the image size capable of being handled by the system.

The other important constraint is that the user enters input without cursive writing. The system uses the gaps between the letters to recognize them and hence cursive writing may not be suitable for the system to process.

3.2.5. Assumptions and Dependencies

The user can competently read and follow the GUI's instructions.

The user enters the image file in which the letters are in English language and are well defined. Half letters and crooked writing may be processed incorrectly.

3.3. Functional Requirements

3.3.1. Functional Requirements-1

3.3.1.1. Introduction

This module splits the letters of the image file and writes each of the letters into individual image files for further processing.

3.3.1.2. Inputs

Image file with the text to be converted, in bitmap form with resolution 64x64.

3.3.1.3. Processing

Various graphical functions like scaling, translation and cropping form the core of this module. The bitmap image is scanned to detect the edges of the letters and then the detected letter is cropped to and translated to the top of the page. It is then scaled after being written into a new bitmap file to the size 64x64.

3.3.1.4. Outputs

Images files with individual letters in the previous file, in bitmap form with resolution 64x64.

3.3.2. Functional Requirements-2

3.3.2.1. Introduction

This module is designed to process the individual letters to compare them and recognize them.

3.3.2.2. Inputs

Image files that contain that contain the individual letters of the text.

3.3.2.3. Processing

The image file is scanned by drawing lines and the number of intersections with the letter are found. Then the lines are tilted by 1 degree and again the scanning is performed to find the number of intersections until the line is tilted for 180 degrees. The number of intersections for each angle of the line is stored in a text file.

3.3.2.4. Outputs

A text file that contains the number of intersections of each letter for various angles of the line.

3.3.3. Functional Requirements-3

3.3.3.1. Introduction

This module is to recognize the letters using the information from the previous module and to write the letter in to the output text file.

3.3.3.2. Inputs

A text file that contains the number of intersections of each letter for various angles of the line.

3.3.3.3. Processing

The information of the number of intersections of a given letter for various angles of the line is compared with the previously stored values for each letter. The letters that match the most are appended in to a text file.

3.3.3.4. Outputs

A text file that contains the final recognized letters.

3.4. External Interface Requirements

3.4.1. User Interface

The user interface is a GUI that will enable the user to enter a image file as the input (location of the file). A button is provided for the user to prompt the system to convert the bitmap image in to text file.

3.4.2. S/W Interfaces

A database is used to store the information for each letter in the English grammar. The system accesses this information to match the patter for any particular letter. The software interface provides access to this database.

3.5. Design Constraints

3.5.1. H/W Limitations

The minimum hardware requirements for this system are

Pentium III Processor

128 MB RAM

10 GB Hard Disk

104 keys keyboard

3.5.2. Fault Tolerance and Reliability

The system developed will be able to scan and recognize bitmap images that contain well-defined letters of the English grammar. Letters that are not complete or crooked may not be recognized properly and may be reproduced improperly.

4. System Design Specification

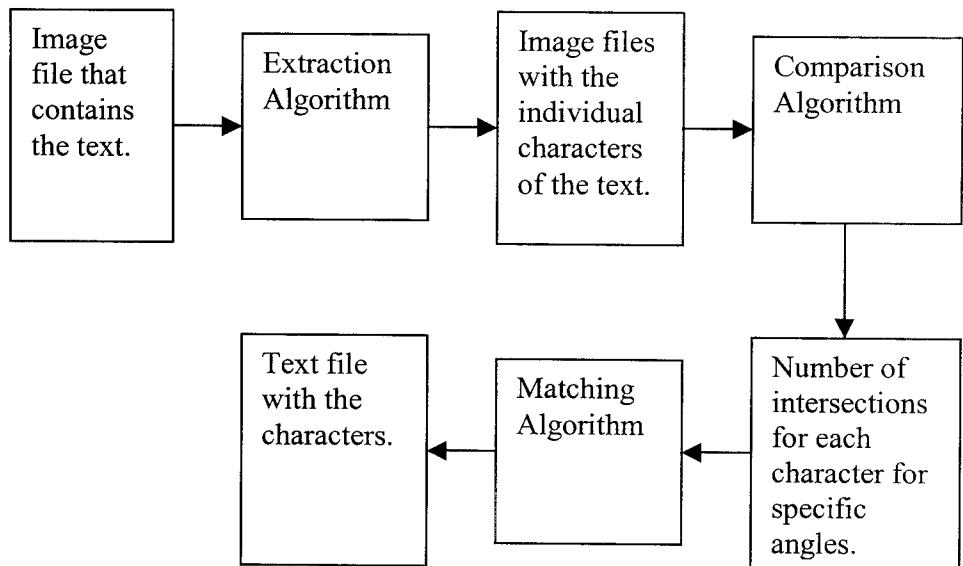
4. System Design Specification

4.1. Overall Design:

Our system implements algorithms to extract the characters from the image file, to recognize them and to write them into text file. The extraction algorithm splits the letters from the input bitmap file that contains the whole of the text and then writes each character into an individual image file. Edge detection technique is used in this module to extract the characters. These characters are then scaled into a standard size. Basic graphic transformations such as scaling and translation are used in this extraction module.

In the next module which is the recognition algorithm, the standard size image files that were formed in the previous stage are processed using research algorithms. This algorithm draws lines through the individual image files at various angles. The number of intersections of, these lines with the character are determined. These are then compared with those of the standard images stored previously. The character that matches the most (the one with the minimum difference) is written into the text file.

4.2. Design Diagram:



4.3. The Extraction Algorithm:

This section explains in detail the extraction algorithm. This is the very first phase of the system's function. The input given here is a bitmap file of any size. This bitmap file contains the text to be recognized. The letters can be present anywhere in the file. These can be letters of the English alphabet and have to be in proportionate size. Cursive writing cannot be recognized and hence the letters should have a finite distance between each other.

smile

Sample Input File

Each pixel position in the input bitmap file is scanned and its RGB value is checked. The first pixel that is a black pixel is identified. This point is copied into a new image file. The color of this pixel is changed (grey) in the source file. This point is called the edge of the character. From this point a few pixels to the left and top are traced back. Now the new point lays a few pixels to the left and top of the pixel whose color was changed. This point is considered as the first pixel now. Starting from this pixel horizontal scan is started. The maximum pixel that is scanned lies n pixels away horizontally and vertically. Here n is the maximum size of the character. For instance if n is 64 then the maximum character size can be 64x64. Again each pixel's RGB value is determined.



smile

Red pixel shows the edge of the character, while the pink pixels form the rectangle which will be scanned for continuous pixels to extract the character.

The first black pixel is identified. Now the 8 pixels that surround the current black pixel are checked. If there is any grey pixel then it means that the pixels are continuous and are a part of a single character. The black pixel is then copied in to the new image file and its color in the source file is changed (to grey).



smile

The source image after an edge is detected, and set to grey.

The scan is again started from the next pixel and continued till the end of the rectangle under consideration is reached. Each black pixel with a grey pixel in its surrounding 8 pixels is transferred in to the new image file and its color in the source file is changed. This is called a pass.



smile

The source image after a pass.

The next pass is started from the bottom of the rectangle and continued till the top (bottom up scanning). The letter can be extracted in these two passes and when the two passes are over, all the grey pixels in the source image that belong to the character are converted into white. Therefore a character extracted will not be present in the source file.

smi e

The source image file after a character is extracted.

The process is started from the beginning and continued till no more black pixels are found in the source file. Thus all the characters are extracted and saved in individual image files. The size of the image in the source file and the file in which it is written into are the same.

l

The image filed formed after extraction

The next phase in the extraction algorithm is scaling. The individual image files are opened and the pixels are scanned. The minimum and maximum x and y positions of nonwhite pixels are determined. Then with those as the vertices, the rectangle formed is scaled into a standard size (64x64). The scaled rectangle is saved as a new image file. This process is done for all the character images formed in the previous stage.



The extracted character image after scaling.

By the edge detection technique the tallest character is recognized first and hence the first image will be that of the tallest character. Based on the x value of these characters in the source file, the individual character image files are renamed so that they occur in the order they were written.

4.4. The Comparison Algorithm:

In this phase each character image formed in the previous phase is processed by using research algorithms to recognize a given character. Lines are drawn across the character image files and the number of intersections for various angles is determined.

Each character image file formed in the previous phase is processed here. Once an image file is opened, lines starting at 0 degrees (horizontal) are drawn. Here drawing lines refers to scanning the pixels from the start of the image till the end in any particular angle. Initially the image is scanned horizontally and the number of pixels that form the character is determined. This is the number of intersection points for 0 degrees. Now the line is tilted by 5 degrees and again

the number of intersection points is determined. Similarly the number of intersection points for every 5 degree tilt is determined for a total tilt of 180 degrees of the line. To find the next point in a given line the equation $Y = m X + C$ is used, where 'm' is the slope of the line. 'm' is found out by using the tilt in angle with the normal. Thus the number of intersection points for 0, 5, 10, 15, ..., 180 degrees are found. The lines are tilted only by 45 degrees, but changing the x and y coordinates results in an effective tilt of the scanning lines by 180 degrees.

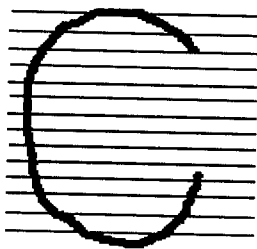


Figure shows the scanning lines at 0 degrees and the intersection points.

Similarly for every 5 degrees the scanning lines are considered and the number of intersection points is determined.

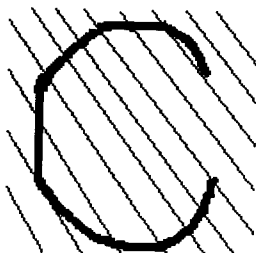


Figure shows the scanning lines at 'n' degrees and the intersection points

The output of this phase is the number of intersection points for every 5 degree scanning lines for all the character image files.

4.5. The matching algorithm:

In this algorithm the characters in the image files are deduced. This algorithm compares the number of intersection points of the current character with that of previously stored values of all standard character images. The character that has the least difference is selected and is written in to the text file.

Here the number of intersection points for every 5 degrees of the current character image is subtracted from the corresponding values of angles of the previously stored character images. The differences for each angle are added for each character. The character that has the minimum cumulative difference is chosen and is written in to the text file. This procedure is repeated for all the characters from the image file given as input.

5. Product Testing

5. Product Testing

Here, the various test strategies adopted in testing this system are outlined. The strategies include Unit Testing, Integrity Testing, Sub-System Testing and System Testing.

5.1. Unit Testing

In this testing step, each module was found to be working satisfactory as per the expected output of the module. In the package development, each module was tested separately after it had been completed and checked with valid data. Unit testing exercises specific paths in the modules control structure to ensure complete coverage and maximum error detection.

The extraction module takes input as the bitmap file that contains the text and produces images files of individual characters of 64x64 sizes as output. The comparison module takes each of these individual image files processes them by drawing scanning lines at various angles and produces the number of intersections for each angle as the output. The matching algorithm compares this intersection data to compare with those of previously stored image files and produces output as a text file containing the characters recognized. Each module was given its specific input and its output was found to be correct.

5.2. Integrity Testing

The individual modules are integrated to form the complete system. This system is then tested to find if the output of each module reaches the next module correctly. Various errors such as accessing of non-existing files were detected and corrected. A status field was included to let the user to know the current status of the system (which module of the system is being executed).

5.3. Output Testing

Here various inputs are given to the system and the output is checked. The handwriting of different users may be different and the system was enhanced so that each user can configure his handwriting into the system before using it. This way the system was made more effective in recognizing the handwriting of different users.

6. Future Enhancements

6. Future Enhancements

This project could be in future extended to recognize cursive writing and the input could be directly given in paper form which can be scanned and converted into bitmap file. Further enhancements can be made such that the system becomes less sensitive to discrepancies in writing of various users. Also this system could be extended to a shorthand recognition system wherein the shorthand symbols could be recognized their corresponding meanings looked up in the dictionary and written in to the text file.

7. Conclusion

7. Conclusion

This system developed has been able to extract and recognize characters at a very good standard. The system's performance has been very impressive and further enhancements when considered, shows promise of an effective handwriting recognition system that could be of immense use to anyone who wishes to edit handwritten text electronically. Also the system could be changed a bit to support shorthand recognition.

The techniques applied in the design of the programs provide a scope for expansion and implementation of changes, which may be required in future, all programs have been tested and have found to execute correctly. All the programs have been documented and can be easily understood.

8. References

- 8.1.** Kwok-Wai Cheung, Dit-Yan Yeung, Roland T. Chin, Bidirectional Deformable Matching With Application to handwritten Character Extraction, 2002.
- 8.2.** Burton Harvey, Simon Robinson, Julian Templeman, Karli Watson, Programming C#, Shroff Publishers and Distributors, 2000, First Edition.
- 8.3.** Matt Telles, C# Black Book, Dreamtech Press, 2002, First Edition.
- 8.4.** www.citeseer.com

9. Annexure

9.1. Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Drawing.Imaging;
using System.IO;

namespace mybitmap
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        double xmin=1000, ymin=1000, xmax=0, ymax=0;
        double minx1,maxx1,miny1,maxy1;

        Bitmap bmp11;
        Bitmap bm1;
        Bitmap bm2;
        StreamWriter sr1=new StreamWriter("c:\\images\\output.txt",false);

        String strpath;
        Graphics gra,gra1;
        Font font;
```

```

        Color col=new Color();
        int i=0,j=0, m_x, m_y, m_i, m_j, m_offset, m_minusx, m_minusy,
m_plusx, m_plusy,cnt=0;
        public int x=0,y=0;
        public int[][] pts=new int [60][];
        int [] pts1=new int[36];
        int mindif,curdif,foundindex;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox t_filename;
        private System.Windows.Forms.Label status;
        private System.Windows.Forms.MainMenu mainMenu1;
        private System.Windows.Forms.Label label2;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent
            //
            //
            for (int i=0;i<60;i++)
                pts[i]=new int[36];
            for(int i=0;i<60;i++)
            {
                for(int i=0;i<36;i++)

```

call

```

        pts[i][j]=0;
    }
    for (int i=0;i<36;i++)
        pts1[i]=0;
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

```

```

#region Windows Form Designer generated code

```

```

/// <summary>

```

```

/// Required method for Designer support - do not modify

```

```

/// the contents of this method with the code editor.

```

```

/// </summary>

```

```

private void InitializeComponent()

```

```

{

```

```

    this.t_filename = new System.Windows.Forms.TextBox();

```

```

    this.label1 = new System.Windows.Forms.Label();

```

```

    this.button1 = new System.Windows.Forms.Button();

```

```

    this.status = new System.Windows.Forms.Label();

```

```

    this.mainMenu1 = new System.Windows.Forms.MainMenu();

```

```

    this.label2 = new System.Windows.Forms.Label();

```

```
this.SuspendLayout();
//
// t_filename
//
this.t_filename.Location = new System.Drawing.Point(136, 40);
this.t_filename.Name = "t_filename";
this.t_filename.Size = new System.Drawing.Size(144, 20);
this.t_filename.TabIndex = 0;
this.t_filename.Text = "c:\\samp.bmp";
//
// label1
//
this.label1.Location = new System.Drawing.Point(8, 40);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(112, 24);
this.label1.TabIndex = 1;
this.label1.Text = "File Name";
//
// button1
//
this.button1.Location = new System.Drawing.Point(88, 80);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(112, 32);
this.button1.TabIndex = 2;
this.button1.Text = "Extract";
this.button1.Click += new
System.EventHandler(this.button1_Click);
//
// status
//
this.status.Location = new System.Drawing.Point(88, 136);
this.status.Name = "status";
this.status.Size = new System.Drawing.Size(112, 24);
this.status.TabIndex = 3;
```



```

        this.status.Text = "Halted";
        //
        // label2
        //
        this.label2.Location = new System.Drawing.Point(96, 192);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(88, 24);
        this.label2.TabIndex = 4;
        //
        // Form1
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
        this.ClientSize = new System.Drawing.Size(336, 266);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.status);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.t_filename);
        this.Menu = this.mainMenu1;
        this.Name = "Form1";
        this.Text = "Form1";
        this.ResumeLayout(false);
    }
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

```

```
private void button1_Click(object sender, System.EventArgs e)
```

```
{
```

```
    status.Text ="Extracting...";
```

```
    status.Update ();
```

```
    bmp11=new Bitmap (t_filename.Text );
```

```
    bm1=new Bitmap (bmp11.Width ,bmp11.Width );
```

```
    gra=Graphics.FromImage(bm1);
```

```
    bm2=new Bitmap (bmp11.Width ,bmp11.Height );
```

```
    gra1=Graphics.FromImage (bm2);
```

```
    for(int t1=0;t1<bmp11.Width ;t1++)
```

```
    {
```

```
        for(int t2=0;t2<bmp11.Height ;t2++)
```

```
        {
```

```
            col=bmp11.GetPixel (t1,t2);
```

```
            bm2.SetPixel (t1,t2,col);
```

```
        }
```

```
    }
```

```
    bm2.Save("C:\\images\\notrecg.bmp");
```

```
    try
```

```
    {
```

```
        getxy();
```

```
        bm1.Save ("c:\\images\\sam1.bmp");
```

```
        bm2.Save ("c:\\images\\notrecg.bmp");
```

```
        gra.Dispose ();
```

```
        bm1.Dispose ();
```

```
        bm2.Dispose ();
```

```
        gra1.Dispose ();
```

```
        bmp11.Dispose ();
```

```
    }
```

```
    catch(Exception e1)
```

```

        {
            cnt--;
            //label1.Text ="Number of Characters Extracted : "+
cnt.ToString();
            //label2.Text =j.ToString () ;
        }
        status.Text ="Scaling....";
        status.Update ();
        for (int count=1;count<=cnt;count++)
        {
            bm2=new Bitmap (64,64);
            bm1= new Bitmap ("c:\\images\\"+count.ToString
)+"out.bmp");

            Color col;
            for (i=1;i<bm1.Width;i++)
            {
                for(j=1;j<bm1.Height;j++)
                {
                    col=bm1.GetPixel (i,j);

                    if((col.R!=255)||((col.B!=255)||((col.G!=255))
                    {
                        minx1=i;
                        goto label1;
                    }
                }
            }

            label1:
            Color col1;
            for (i=bm1.Width-1;i>0;i--)
            {
                for (i=bm1.Height-1;i>0;i--)

```

```

        {
            col1=bm1.GetPixel (i,j);

if((col1.R!=255)||(col1.B!=255)||(col1.G!=255))
            {
                maxx1=i;
                goto label2;
            }
        }
label2:
        Color col2;
        for (i=1;i<bm1.Height;i++)
        {
            for(j=1;j<bm1.Width;j++)
            {
                col2=bm1.GetPixel (j,i);

if((col2.R!=255)||(col2.B!=255)||(col2.G!=255))
                {
                    miny1=i;
                    goto label3;
                }
            }
        }
label3:
        Color col3;
        for (i=bm1.Height-1;i>0;i--)
        {
            for(j=bm1.Width-1;j>0;j--)
            {
                col3=bm1.GetPixel (j,i);

```

```

if((col3.R!=255)||(col3.B!=255)||(col3.G!=255))

```

```

        {
            maxy1=i;
            goto label4;
        }
    }
}
label4:

    double scalex=64.0/(double)(maxx1-minx1),
scaley=64.0/(double)(maxy1-miny1), scale;
    if (scalex<scaley) scale=scalex; else scale=scaley;
    int neww=(int)(bm1.Width*scale),
newh=(int)(bm1.Height*scale);
    Rectangle expansionRect=new Rectangle ((int)(-
minx1*scale), (int)(-miny1*scale), neww, newh);
    Graphics myGraphics=Graphics.FromImage (bm2);
    myGraphics.DrawImage(bm1, expansionRect);
    bm2.Save ("c:\\images\\alpha"+count.ToString
()+".bmp");

    try
    {
        bm1.Dispose();
        bm2.Dispose();
        myGraphics.Dispose();
    }
    catch(Exception e1)
    {

    }
    status.Text ="Finished";
}

```

```

        double rad;
        int flg=0;
        StreamWriter sr=new
StreamWriter("C:\\images\\refinfo.txt",false);

        Color cl;
        Bitmap bp;
        for (int ct=1;ct<=52;ct++)
        {
            status.Text ="Extracting From Reference: "+ct.ToString
());

            status.Update ();
            bp= new Bitmap("C:\\images\\reference\\"+ct.ToString
()+".bmp",false);

            x=0;y=0;
            int ref1=0;
            for(int deg=0;deg<=45;deg+=5)
            {
                rad=Math.Tan(Convert.ToDouble(
(Convert.ToDouble(deg))*((22.0/7.0)/180.0) ));

                for(int i=-63;i<63;i+=4)
                {
                    x=0;
                    for(int j=0;j<63;j++)
                    {
                        x=x+1;

                        y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
                        if(y>=0 && y<=63 && x<=63 &&
x>=0)

                        {

```

```

cl.B<50 )
if(cl.R>250 && cl.G<50 &&
{
    if(flag==0)
    {
        pts[ct][ref1]=pts[ct][ref1]+1;
        flag=1;
    }
}
else flag=0;
}
else break;
}
}

```

```

//91 to 180
for(int i=-63;i<63;i+=4)
{
    x=0;
    for(int j=0;j<63;j++)
    {
        x=x+1;
        y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
        if(y>=0 && y<=63 && x<=63 &&
x>=0)
        {
            cl=bp.GetPixel(y,63-x);

```

```

        if(cl.R>250 && cl.G<50 &&
cl.B<50 )
        {
            if(flag==0)
            {
                pts[ct][35-
ref1]=pts[ct][35-ref1]+1;
                flag=1;
            }
            else flag=0;
        }
        else break;
    }
}
/* the other angle */

for(int i=-63;i<63;i+=4)
{
    x=0;
    for(int j=0;j<63;j++)
    {
        x=x+1;

        y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
        if(y>=0 && y<=63 && x<=63 &&
x>=0)
        {
            cl=bp.GetPixel(x,y);
            if(cl.R>250 && cl.B<50 &&
cl.G<50 )
            {

```



```

ref1]=pts[ct][17-ref1]+1;
        {
            pts[ct][17-
            flg=1;
        }
    }
    else flg=0;
}
else break;

}
}
// 91-180

for(int i=0;i<63;i+=4)
{
    x=0;
    for(int j=0;j<63;j++)
    {
        x=x+1;

        y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
        if(y>=0 && y<=63 && x<=63 &&
x>=0)
        {
            cl=bp.GetPixel(63-x,y);
            if(cl.R>250 && cl.G<50 &&
cl.B<50 )
            {
                if(flg==0)
                {

```

```

17+ref1]=pts[ct][35-17+ref1]+1;
pts[ct][35-
flg=1;
    }
    }
    else flg=0;
    }
    else break;
    }
    }
    ref1+=1;
}
bt[0]=Convert.ToByte('a');
bt[1]=Convert.ToByte(':');
sr.Write(ct.ToString()+"::");
for(int i=0;i<=35;i++)
{
    sr.Write((i+1)*5);
    sr.Write(":");
    sr.Write(pts[ct][i]);
    sr.Write(" ");
}

sr.WriteLine(" ");
sr.Flush();
bp.Dispose ();
}
status.Text ="Finished";
sr.Close ();
//alphabets of the text
sr=new StreamWriter("C:\\images\\datainfo.txt",false);

```

```
for (int ct=1;ct<=entiret;ct++)
```



```

                                                    flg=1;
                                                    }
                                                }
                                            else flg=0;
                                        }
                                    else break;

                                }
                            }

//91 to 180
for(int i=-63;i<63;i+=4)
{
    x=0;
    for(int j=0;j<63;j++)
    {
        x=x+1;

        y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
        if(y>=0 && y<=63 && x<=63 &&
x>=0)
        {
            cl=bp.GetPixel(y,63-x);
            if(cl.R>250 && cl.G<50 &&
cl.B<50 )
            {
                if(flg==0)
                {
                    pts1[35-
ref1]=pts1[35-ref1]+1;

```



```

else break;

    }
}
// 91-180

for(int i=0;i<63;i+=4)
{
    x=0;
    for(int j=0;j<63;j++)
    {
        x=x+1;

y=Convert.ToInt32(rad*Convert.ToDouble(x)+i);
        if(y>=0 && y<=63 && x<=63 &&
x>=0)
        {
            cl=bp.GetPixel(63-x,y);
            if(cl.R>250 && cl.G<50 &&
cl.B<50 )
            {
                if(flag==0)
                {
                    pts1[35-
17+ref1]=pts1[35-17+ref1]+1;
                    flag=1;
                }
            }
            else flag=0;
        }
    }
}
else break;
}

```

```

        }
        ref1+=1;
    }
    bt[0]=Convert.ToByte('a');
    bt[1]=Convert.ToByte(':');
    sr.Write(ct.ToString ()+":");
    curdif=0;
    for(int i=0;i<=35;i++)
    {
        sr.Write((i+1)*5);
        sr.Write(":");

        sr.Write(pts1[i]);
        sr.Write(" ");
    }
    sr.WriteLine(" ");
    sr.Flush();
    bp.Dispose ();
    int k;
    mindif=0;
    for (k=1;k<=52;k++)
    {
        curdif=0;
        for(int i=0;i<=35;i++)
        {
            int temp=pts[k][i]-pts1[i];
            if (temp>0)
                curdif=curdif+temp;
            else
                curdif=curdif-temp;
        }
        if (k==1)
        {

```

```
        foundindex=k;
    }
    if (curdif<mindif)
    {
        foundindex=k;
        mindif=curdif;
    }
}
label2.Text =label2.Text+" " +foundindex.ToString ();
string c;
switch(foundindex)
{
    case 1:
    case 27:
        sr1.Write("a");
        break;
    case 2:
    case 28:
        sr1.Write("b");
        break;
    case 3:
    case 29:
        sr1.Write("c");
        break;
    case 4:
    case 30:
        sr1.Write("d");
        break;
    case 5:
    case 31:
        sr1.Write("e");
        break;
    case 6:
```



```
case 32:  
    sr1.Write("f");  
    break;
```

```
case 7:
```

```
case 33:  
    sr1.Write("g");  
    break;
```

```
case 8:
```

```
case 34:  
    sr1.Write("h");  
    break;
```

```
case 9:
```

```
case 35:  
    sr1.Write("i");  
    break;
```

```
case 10:
```

```
case 36:  
    sr1.Write("j");  
    break;
```

```
case 11:
```

```
case 37:  
    sr1.Write("k");  
    break;
```

```
case 12:
```

```
case 38:  
    sr1.Write("l");  
    break;
```

```
case 13:
```

```
case 39:  
    sr1.Write("m");  
    break;
```

```
case 14:
```

```
case 40:
```

```
        break;
case 15:
case 41:
    sr1.Write("o");
    break;
case 16:
case 42:
    sr1.Write("p");
    break;
case 17:
case 43:
    sr1.Write("q");
    break;
case 18:
case 44:
    sr1.Write("r");
    break;
case 19:
case 45:
    sr1.Write("s");
    break;
case 20:
case 46:
    sr1.Write("t");
    break;
case 21:
case 47:
    sr1.Write("u");
    break;
case 22:
case 48:
    sr1.Write("v");
    break;
case 23:
```

```

        case 49:
            sr1.Write("w");
            break;
        case 24:
        case 50:
            sr1.Write("x");
            break;
        case 25:
        case 51:
            sr1.Write("y");
            break;
        case 26:
        case 52:
            sr1.Write("z");
            break;
    }
    for (int t=0;t<36;t++)
        pts1[t]=0;

        //sr1.WriteLine(" ");
        sr1.Flush();
    }

}

private bool getpt()
{
    strpath=null;
    strpath+="c:\\images\\" + cnt.ToString () + "out.bmp";
    bm1.Save (strpath);
    cnt++;
}

```

```

        gra.Clear (Color.White );
        bm2.Save ("c:\\images\\notrecg.bmp");
        col=bm2.GetPixel (i,j);
        while((j < bm2.Height) && ((col.R !=0) || (col.G !=0) || (col.B
!=0)))
        {
            i++;
            if (i==bm2.Width)
            {
                j++; i=0;
            }
            col=bm2.GetPixel (i,j);
        }
        if (j==bm2.Height) return false;
        else
        {
            minmax(i, j);
            return true;
        }
    }
}

```

```

private void minmax(int x, int y)
{
    if (x<xmin) xmin=x;
    if (x>xmax) xmax=x;
    if (y<ymin) ymin=y;
    if (y>ymax) ymax=y;
}

```

```

private bool checkall(int x, int y)
{
    bool flag=false;
    Color colt=new Color();
    for (int p=-m_offset; p<=m_offset; p++)
        for (int q=-m_offset; q<=m_offset; q++)

```

```

        {
            colt=bm1.GetPixel (p+x,q+y);
            if ((colt.R==255)&&(colt.G==0)&&(colt.B==0))
            {
                flag=true;
            }
        }
    }
    return flag;
}

```

```

private void getxy()
{
    // constants

    m_offset=4;           // checks for x^4 pixels

    m_minusx=10;         // range i-x
    m_minusy=10;         // range i-y
    m_plusx=50;          // range i+x
    m_plusy=50;          // range i+y

    while (getpt())
    {
        first=true;
        for (int z=0; z<2; z++)
        {
            m_x=minx(); m_y=miny();
            m_i=mini(); m_j=minj();

            pass1();
            pass2();
            // ...

```

pass1();

pass2();

// ...

```

        //pass2());
    }

}

private void pass1()
{
    for (int m=m_x; m<m_j; m++)
        for (int l=m_y; l<m_j;l++)
        {
            col=bm2.GetPixel (m,l);
            if((col.R!=255)||col.G!=255)||col.B!=255)
            {
                if (first)
                {
                    bm1.SetPixel (m-m_x+m_offset,l-
m_y+m_offset,Color.Red );
                    bm2.SetPixel (m,l,Color.White );
                    first=false;
                }
                else
                if (checkall(m-m_x+m_offset,l-
m_y+m_offset))
                {
                    minmax(m,l);
                    bm1.SetPixel (m-m_x+m_offset,l-
m_y+m_offset,Color.Red );
                    bm2.SetPixel (m,l,Color.White );
                }
            }
        }
}

```

```

    }

    private void pass2()
    {
        for (int m=m_j; m>m_x; m--)
            for (int l=m_j; l>m_x;l--)
                {
                    col=bm2.GetPixel (m,l);
                    if((col.R!=255)||col.G!=255)||col.B!=255)
                        {
                            if (checkall(m-m_x+m_offset,l-
m_y+m_offset))
                                {
                                    bm1.SetPixel (m-m_x+m_offset,l-
m_y+m_offset,Color.Red );
                                    bm2.SetPixel (m,l,Color.White );
                                }
                        }
                }
    }

```

```

    private int mini()
    {
        if ((i+m_plusx)< bm2.Height)
            return (i+m_plusx);
        else
            return bm2.Height ;
    }

```

```

    private int minj()
    {
        if ((j+m_plusy)< bm2.Width )
            return (j+m_plusy) ;
    }

```

```
        else
            return bm2.Width ;
    }

private int minx()
{
    if ((i-m_minusx)>0)
        return (i-m_minusx);
    else
        return 0 ;
}

private int miny()
{
    if ((j-m_minusx)>0)
        return (j-m_minusx);
    else
        return 0 ;
}
}
```