HANDWRITING RECOGNITION

A PROJECT REPORT

Submitted by

POOJASREE.S.J
VIGNESH.C
RAMANI.N

in partial fulfillment for the award of the degree
of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS, MADURAI

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY REGIONAL CAMPUS MADURAI

BONAFIDE CERTIFICATE

Certified that this project report

"**HANDWRITING RECOGNITION**"

is the bonafide work of

"**POOJASREE S J**
RAMANI N
VIGNESH C "

who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of CSE	**Department of CSE**
Anna University Regional campus	**Anna University Regional campus**
Madurai - 625019	**Madurai - 625019**

ACKNOWDGEMENT

We would like to thanks our Dean, Dr.V.Malathi for providing infrastructure facilities and whole hearted encouragement for completing our project successfully.

For mostly, We pay our grateful acknowledgement and extend our sincere gratitude to Dr.E.Srie Vidhya Janani, Head of the Department, Computer Science and Engineering, Anna University Regional Campus Madurai, Anna University, Chennai, for extending the facilities of the department towards our project and for her unstinting support.

We express our thanks to our guide, Ms.R.Subhasini , Department of Computer Science and Engineering, Anna University Regional Campus Madurai, Anna University, Chennai, for guiding us through every phase of the project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us. We thank her for being easily approachable and quite thoughtful. We owe for harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent.

We would like to thanks our Class Advisor Mr.Rajkumar SC, and all our teaching and non-teaching faculty members of the Department and also our fellow friends for helping us in providing valuable suggestions and timely ideas for the successful completion of the project. We extend our thanks to our family members who have been a great source of inspiration and strength to us during the course of this Project work. We sincerely thank to all of them.

Poojasree.S.J

Ramani.N

Vignesh.C

ABSTRACT

In todays' world advancement in sophisticated scientific techniques is pushing further the limits of human outreach in various fields of technology. One such field is the field of character recognition commonly known as OCR (Optical Character Recognition).

In this fast paced world there is an immense urge for the digitalization of printed documents and documentation of information directly in digital form. And there is still some gap in this area even today. OCR techniques and their continuous improvisation from time to time is trying to fill this gap. This project is about devising an algorithm for recognition of handwriting.

The handwriting recognition is the ability of computers to recognize human handwritten characters. It is a hard task for the machine because handwriting are not perfect and can be made with many different f lavors. The handwriting recognition is the solution to this problem which uses the image of a handwriting and recognizes the handwriting present in the image.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SVM	:	Support Vector Machine
OCR	:	Optical Character Recognition

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

Handwriting is one of the most important ways of communication. It was used since the Stone Age where symbols were drawn on stones in order to express or convey some meaningful information. Later, handwriting was done using pen and paper. Handwriting was used for personal benefits like writing reminders and notes for ourselves or for business purposes such as writing letters, statements and filling up forms. Thus handwriting then was by human to human for conveying information. The handwriting of each individual is unique because the process of handwriting is a physical process, which involves the mind, skeleton and muscles, controlled by the brain. Even so, individual handwriting could also differ, based on the mood and the state of mind of the person writing. The handwriting among the different stock of people (Europeans vs. Asian or French vs. Malaysian) are normally different, due to the conditioning and training during the period of growing up. However, even though the same stock of people has similar handwriting, it is an accepted fact that no two people have the same style of handwriting.

Initially, in a modern computer, the most important device used to interface them to human is a keyboard. As computers are becoming ubiquitous and more people are using it, a more natural interface is needed. The most likely candidates could be voice or handwriting. Voice or speech recognition capability and handwriting recognition capability built into a computer can simplify a lot of data entry, which was handled before by using keyboards. Handwriting recognition seems to be more practical than speech recognition because of the fact that in crowded rooms or public places one might not wish to speak to his or her computer due to the confidentiality or personal nature of the data. Another reason is that it might be annoying to others if someone keeps speaking to his or her machine. It is also already possible to have handwriting recognition in very small hand-held devices, while a speech recognition system is not yet suitable for use as a hand-held machine. However, on the contrary, in term of speed of data entry, speech system is apparently faster and it is much easier to dictate something than to write it

Pen-based interfaces in digital devices are popular lately and will play a more important role in human computer interfaces in the future. In personal digital assistant (PDA) which is a small handheld device, built-in pen-based handwriting recognition system is already used as an input method. The input method is interfaced to the applications in the PDA, such as personal agenda, address book and communication facilities. In personal computers, pen-based input device (pen or stylus and a pad) is sometimes used to replace the cumbersome mouse for handwriting capability and its small footprint.

Automatic handwriting recognition is the transcription of handwritten data into text in digital format, for use by the computer. The area has been under investigation since the 1950's. Since then there has been steady research effort into the area. Two categorizations are possible; first, in term of processing domains, second, in term of usage categories. Handwriting recognition can be categorized into two domains; online recognition, used in the pen-based interface or offline, used in automated recognition system for processing cheques, forms and the like. Figure 1.1 shows the difference between the two domains. In online handwriting recognition, handwriting signals are captured from the pen traces on the surface of a writing pad. The signals are the input to the recognizer, which then gives out the text of the handwritten input. In off-line handwriting recognition, static images of words written are used instead in the process. A difference between the two is that on-line handwriting recognition requires fast and immediate processing while off-line recognition can be performed within quite a relaxed time constrain. However, recently, this might not always be the case because it is possible to collect forms containing online handwriting and then to process them in a batch system.

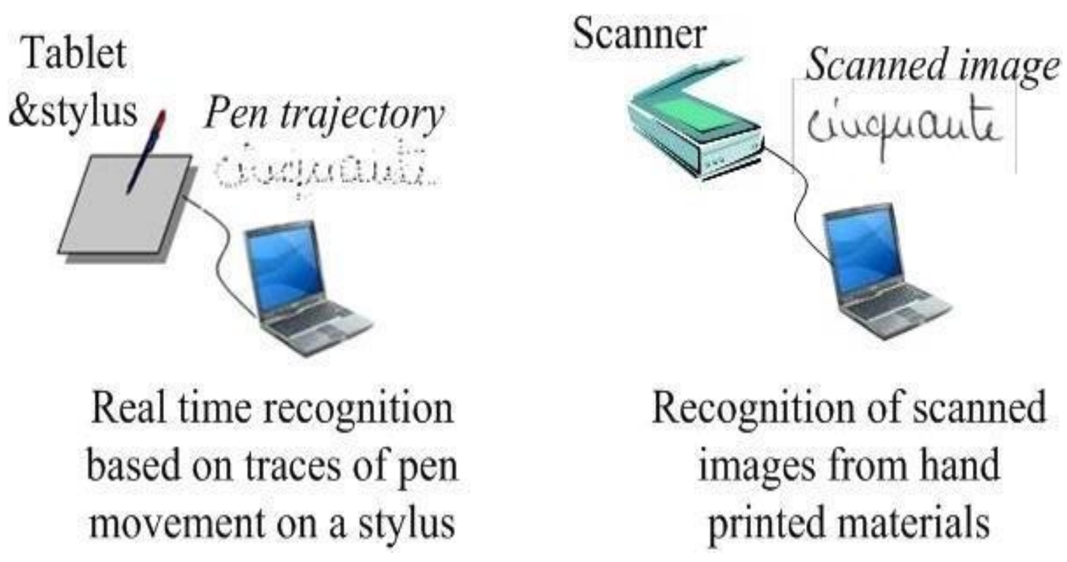

Tablet &stylus
Pen trajectory
Scanner
Scanned image
cinquante
Real time recognition based on traces of pen movement on a stylus
Recognition of scanned images from hand printed materials

Figure 1.1

In term of signals, online signals are normally the pen trajectories, recorded as the x and y coordinates of each point together with eventually the pressure and the time at each point, while offline signals are the image files recorded in a particular image format such as tiff or jpeg.

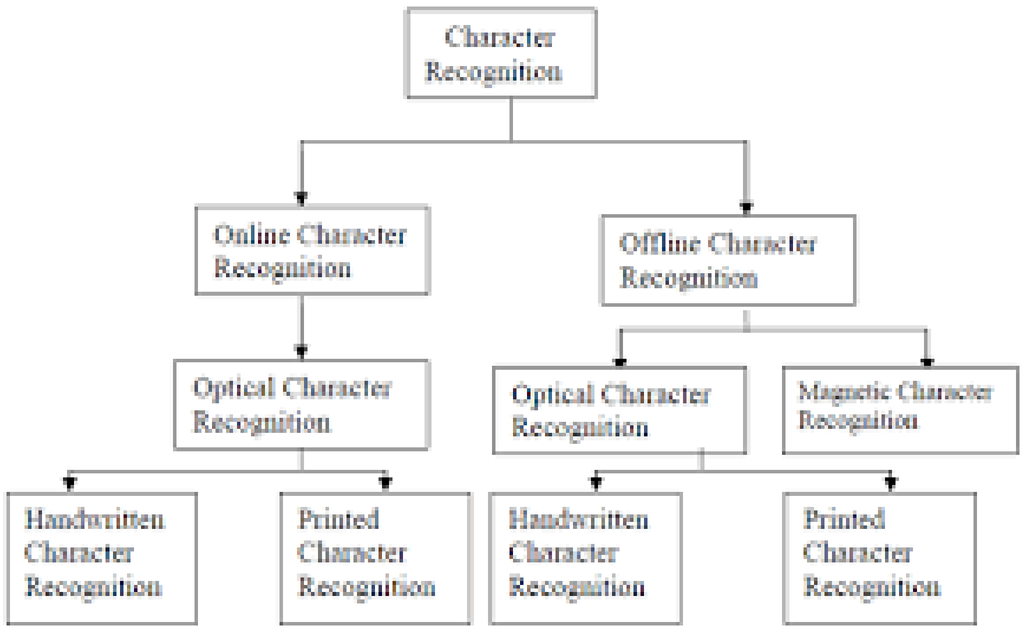

Character Recognition
Online Character Recognition
Offline Character Recognition
Optical Character Recognition
Optical Character Recognition
Magnetic Character Recognition
Handwritten Character Recognition
Printed Character Recognition
Handwritten Character Recognition
Printed Character Recognition

Figure 1.2

1.2 OBJECTIVE OF THE PROJECT

The objective of this project is to create own dataset to recognize and predict handwriting with the use of Support Vector Machine (SVM) . We have to construct Support Vector Classifier and fit the model . The program should be able to extract the characters one by one and map the target output for training purpose. The program code has supported with the usage of Graphical User Interface (GUI) for Prediction.

1.3 PROPOSED SYSTEM

To solve the defined handwriting recognition problem of classification we used Support Vector Classifier algorithm.

The computation code is divided into the next categories:

- Dataset Creation
- Generate dataset
- Preprocessing of the image
- Feature extraction
- Load dataset
- Training and testing
- Fit the model using SVC
- Prediction

1.4 LIMITATIONS AND PROBLEMS OF HANDWRITING RECOGNITION SYSTEM

Although there are many applications of handwriting recognition in both online and offline domain, the technology is not fully matured. There are many improvements that can still be made to make handwriting recognition more widely accepted in computer based applications.

Handwriting recognition is still considered as an open research problem mainly due to its substantial individual variation in appearance, consequently the challenges include the distortion of **handwritten** characters, since different people may use different style of **handwriting**, direction etc.

The issue is that there's a wide range of handwriting – good and bad. This makes it tricky for programmers to provide enough examples of how every character might look. Plus, sometimes, characters look very similar, making it hard for a computer to recognise accurately.

Joined-up handwriting is another challenge for computers. When your letters all connect, it makes it hard for computers to recognise individual characters. Consider, for instance, an 'r' and an 'n'. Joined up, these letters could be mistaken for an 'm'.

As constraints in the handwriting are reduced, the problem will become more complex because the recognition system needs to handle various limitations, thus, this will affect the recognition accuracy.

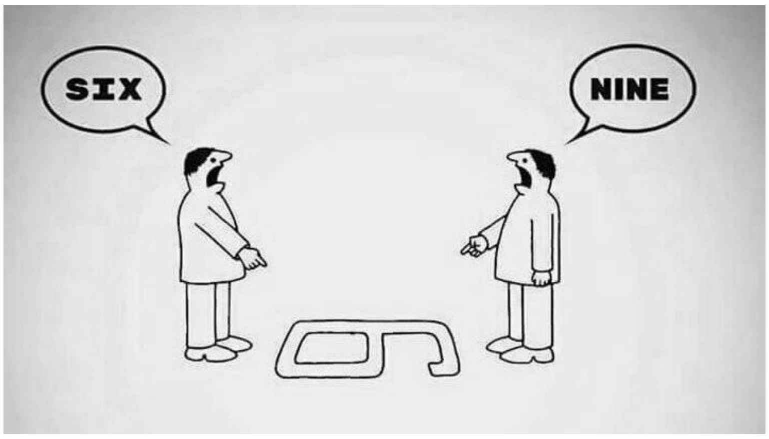


Figure 1.3

Many researchers have conducted research in handwriting recognition in the last years. Although many problems have been solved, there are still many problems at hand. Despite the availability of computing power and progress made so far, the capability of handwriting recognition system is still incomparable to human recognition. As mentioned earlier, no two humans have exactly the same handwriting and even no two sets of handwritings of the same person for the same word are exactly the same. Between people, the variability can include the slant, the size of characters, the shape and how cursive or disjoint the characters in the handwriting are. Variations in handwriting can also be in term of the applications, even if for off-line handwriting applications such as form processing, handwritings are normally guided by boxes.

Handwriting recognition can be performed by taking a word itself as a whole entity for recognition. This method has been used by a number of researchers. The model for recognition is the whole word model, which are trained to cater for variations and similarities within word such as co-articulation. Because the whole word is taken in training the system, segmentation is avoided.

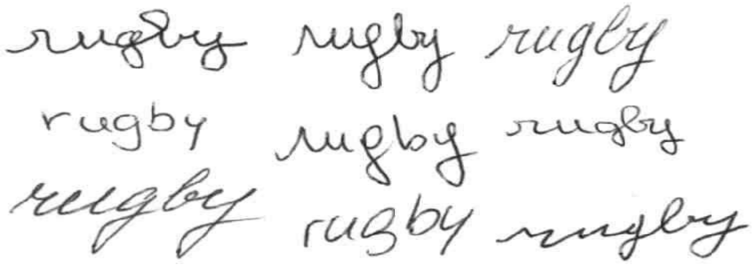


Figure 1.4

CHAPTER 2

LITERATURE SURVEY AND RELATED WORK

2.1 INTRODUCTION

The second chapter presents an in-depth overview of the domain of handwritten character recognition. It outlines the characteristics and the results of the different methods in machine learning and offers opinions about possible conclusions. This Chapter will provide an indepth and detailed overview of the recent literature, corresponding to this study.

2.2 LITERATURE REVIEW

An early notable attempt in the area of character recognition research is by Grimsdale in 1959. The origin of a great deal of research work in the early sixties was based on an approach known as analysis-by-synthesis method suggested by Eden in 1968. The great importance of Eden's work was that he formally proved that all handwritten characters are formed by a finite number of schematic features, a point that was implicitly included in previous works. This notion was later used in all methods in syntactic (structural) approaches of character recognition. K. Gaurav, Bhatia P. K. [5] Et al, this paper deals with the various pre-processing techniques involved in the character recognition with different kind of images ranges from a simple handwritten form based documents and documents containing colored and complex background and varied intensities. In this, different preprocessing techniques like skew detection and correction, image enhancement techniques of contrast stretching, binarization, noise removal techniques, normalization and segmentation, morphological processing techniques are discussed. It was concluded that using a single technique for preprocessing, we can’t completely process the image.

However, even after applying all the said techniques might not possible to achieve the full accuracy in a preprocessing system.

Digital recognition is one of the most indispensable applications in pattern recognition. Many researchers have studied and identified different datasets. For example, the US Postal Code on the letter was collected into the CEDAR digital database and used as a standard database for researchers to analyze (Sarkhel et al., 2016). Since there is no criterion database available at the moment for Marathi, a dataset of 2000 images containing Marathi numerals from 0-9, has been collected from different age groups (Mane& Kulkarni, 2018). Furthermore, the SD19 database provided by the National Institute of Standards and Technology (NIST) was also viral among researchers (Hochuli et al., 2018). The CMATERdb 3.1.3.3 is a database including 171 unique categories of isolated grayscale images of Bangla compound characters. However, images in the dataset are neither of central nor uniform proportions, resulting in difficult pattern recognition problems (Roy et al., 2017).

The performance of a classifier can depend on the quality of the features of the classifier itself (Elleuch, Maalej & Kherallah 2016). However, many classifiers such as SVM and RF cannot process raw images or data efficiently, because extracting appropriate structural features from complex shapes is a considerable challenge (Pramanik & Bag, 2018). Therefore, how to use the combination of sophisticated features extraction and classifier is the main problem of OCR in handwritten digit recognition. A 2017 paper by Phangtriastu, Harefa, and Tanoto compared the most commonly used classifiers SVM and ANN, while this experiment achieved the highest accuracy of 94.43% using the SVM classifier with the combination of feature extraction algorithms which are projection histogram and HOG.

In order to improve the high reliability in handwritten digit recognition, a new feature combination, including of MPCA based statistical features and QTLR features has been evaluated on handwritten digits of five prevalent scripts of Indian, viz., Arabic, Bangla, Devanagari, Latin, and Telugu with SVM based on OCR (Winkler, 1980). Winkler observed that only the features extracted by the PCA algorithm are not sufficient to solve the variability of the handwritten digit mode, while the QTLR-based topology features also have limitations in classifying digital patterns into individual scripts. Consequently, the combination of MPCA + QTLR is applied to increase the recognition accuracies significantly to 98.7%.

Many researches have been carried out on feature extraction and classifier algorithms for handwritten digit recognition. Most of them got good recognition accuracy. For instance, Mane and Kulkarni (2018) proposed a Customized Convolutional Neural Network (CCNN) that can automatically learn features and predict the categories of numerals in extensive ranged data set such as Marathi which is one of the most diffusely spoken regional languages in India. Besides, the CCNN's performance reached an average of 94.93% accuracy by using K-fold crossvalidation. The proposed CCNN model does not impose any restrictions on the count of layers, but instead optimizes the number to satisfy the demand of the issue. Also, the different filter sizes have been applied for the intermediate convolutional layer

In 2007, Sadri, Suen and Bui indicated that the correct use of context knowledge in segmentation, evaluation, and the search could remarkably improve the overall performance of the handwritten digit recognition system. As a result, the recognition system was able to get 95.28% and 96.42% recognition accuracy on handwritten numeric strings using NN and SVM classifiers, respectively. Hochuli et al. (2018) stated the CNN classifier could handle the complexities of touch numbers more efficiently than all the segmentation algorithms provided in the literature. The experiments on two famous databases consisting of Touching Pairs Dataset and NIST SD19 highlight the proposed method by achieving an accuracy level of 97% recognition accuracy

Recently, Mahto, Bahtia and Sharma (2015) designed a combination of horizontal and vertical projection feature extraction to identify Gurmukhi's handwritten characters. The experiment applied linear SVM and k-NN (k = 1, k = 3, k = 5, k = 7) to classify handwritten characters to a maximum accuracy of 98.06%. However, in some other reports, Lauer, Suen, and Bloch (2007) replaced the last layer of the LeNet4 network with a K-NN classifier to process the extracted features. But compared with the regular LeNet4 network, this approach does not improve the results.

A 2007 paper by Hanmandlu and Murthy proposed the recognition of handwritten Hindi and English digits in the form of exponential membership functions as fuzzy models. Furthermore, the defuzzification parameters have been optimized by the double layer perceptron, and the fuzzy rules have been generated based on the ID-3 method. This technique overcame the difficulties of traditional handwritten character recognition syntactic methods and achieved the accuracy of 95% for Hindi digits and 98.4% for English digits.

In the field of pattern recognition, researchers have paid more attention to multi-classifier systems in recent years, especially Bagging, Boosting. Bernard, Adam and Heutte (2007) researched a conventional feature extraction technique based on a greyscale multi-resolution pyramid to find out the effect of the parameter values on the performance of the RF. They have experimented with the Forest-RI algorithm, which is considered as the Random Forest reference method, on the MNIST handwritten digital database and reached a level of accuracy in handwritten digit recognition to greater than 93%.

A 2017 paper by Roy et al. studied the innovative analysis of handwritten Bangla that isolated compound character recognition using a novel deep learning technique. The researchers performed layered training on deep convolutional neural networks (DCNN) and used the RMSProp algorithm to enhance the training process to achieve faster convergence. On experimentation, the proposed DCNN showed significant improvement in recognition rates compared with the standard shallow learning model, reaching 90.33%.

In 2018, Shamim et al. completed the comparative analysis of the performance of SVM with polynomial kernel and radial basis function kernel (RBF) kernel for classifying students with or without handwriting difficulties. Also, cross-validation which is a statistical method for assessing and comparing ML algorithms has been widely used for evaluating the performance of NN and other applications such as SVM and K-NN. Shamim et al. (2018) adopted the ten cross-validation method to select the parameter to gain the highest recognition rates. While, this experiment illustrated that the performance of SVM with RBF is better than with polynomial kernel, reaches more than 93%.

2.3 COMPARISION BETWEEN DIFFERENT TECHNIQUES

S.NO	TITLE	AUTHOR(S)	SOURCE (DETAILS OF PUBLICATION)	FINDINGS
1	Using Random Forests for Handwritten Digit Recognition	1.S.Bernard 2.S.Adam 3.L.Heutte	IEEE (Ninth International Conference on Document Analysis and Recognition (ICDAR 2007))	This aims at studying those methods in a strictly pragmatic approach, in order to provide rules on parameter settings for practitioners. For that purpose we have experimented the forest-RI algorithm, considered as the random forest reference method, on the MNIST handwritten digits database.
2	On-line handwritten digit recognition based on trajectory and velocity modeling	1.MonjiKheralla 2.LobnalHaddad 3.Adel M.Alimi 4.AmarMitiche	Elsevier-- The General Direction of Scientific Research and Technological Renovation (DGRSRT), Tunisia, under the ARUB program , 2008	A system and associated methodology recognizes an Arabic like alphanumeric character using fuzzy modeling. The method receives a handwritten Arabic like

				alphanumeric character, stores fuzzy models of a plurality of Arabic like alphanumeric characters…
3	Handwritten Digit Recognition Using Chemical Reaction Optimization	1. Pritam Khan Boni 2. Bappy Shagnir Abir 3. Md Rafiqul Islam 4.H.M.Mehedi Hasan	IEEE 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)	This paper propose a new methodology to recognize handwritten Bengali numerals using a recently established metaheuristic algorithm known as Chemical Reaction Optimization (CRO) in order to increase the recognition accuracy. The proposed method produces a higher accuracy rate, 98.96% which is higher than the outcome of any other proposed method.
4	Effective handwritten digit recognition based on multi-feature extraction and deep analysis	1.Caiyun Ma 2.Hong Zhang	IEEE 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)	It propose specific feature definitions, including structure features, distribution features and projection features It results on benchmark database of MNIST handwritten digit images show that the performance of

				our algorithm is remarkable and demonstrate its superiority over several existing algorithms.

Table 2.1

2.4 SUMMARY

This chapter has reviewed the existing literature relevant to the research. These factors should be addressed in the acquisition of data resources and the preparation of the plan to achieve the highest handwritten recognition accuracy.

CHAPTER 3

SUPPORT VECTOR MACHINE

3.1 INTRODUCTION

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

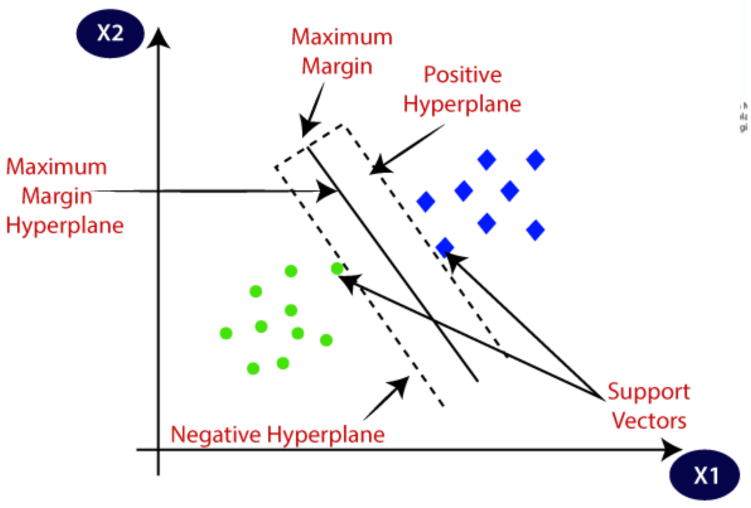


Figure 3.1

3.2 TYPES OF SVM

SVM can be of two types:

- **Linear SVM:**

  Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

- **Non-linear SVM:**

  Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

3.3 WORKING OF SVM

3.3.1 LINEAR SVM

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:

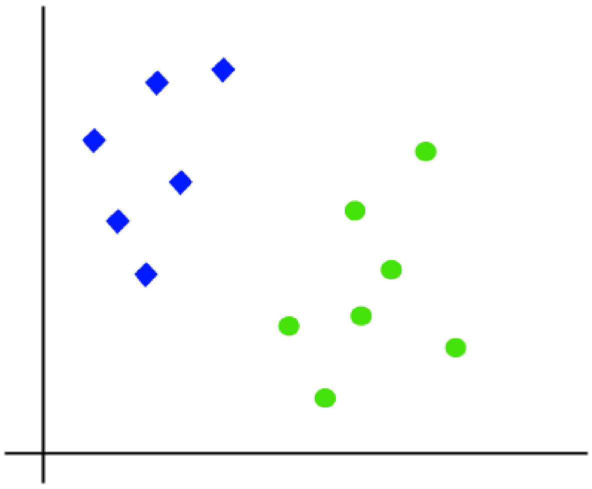


Figure 3.2

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

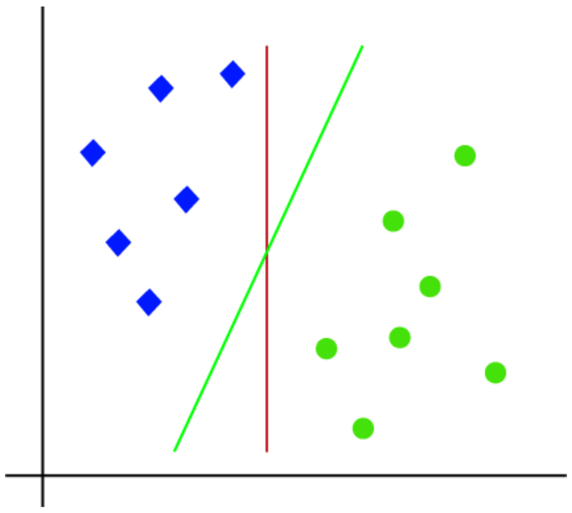


Figure 3.3

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.

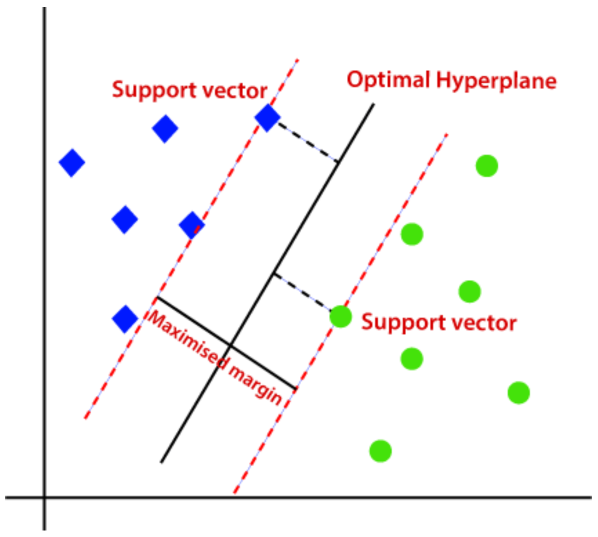


Figure 3.4

3.3.2 NON-LINEAR SVM

If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line.

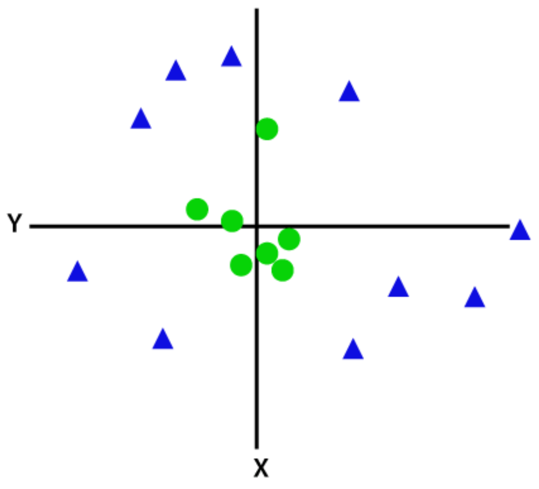


Figure 3.5

So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 + y^2$

By adding the third dimension, the sample space will become as below image:

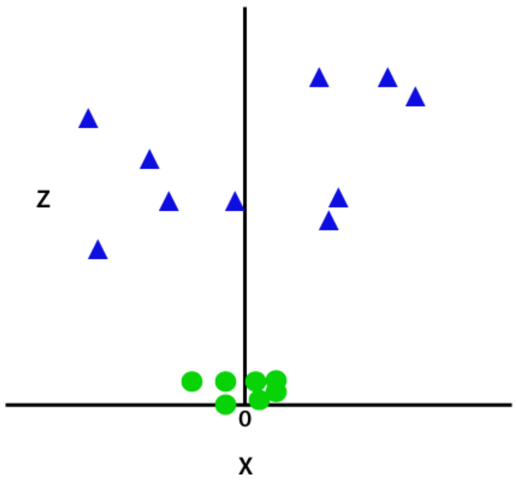


Figure 3.6

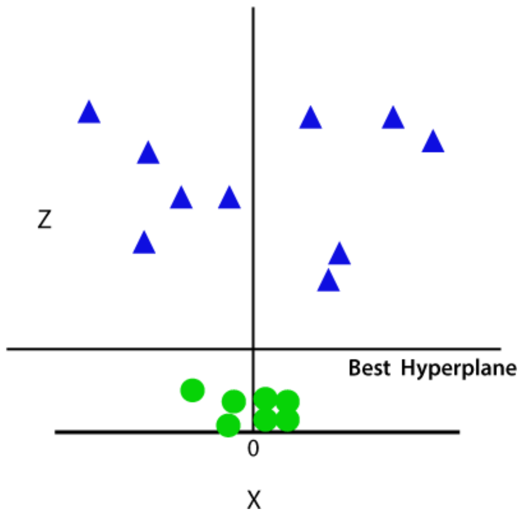


Figure 3.7

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

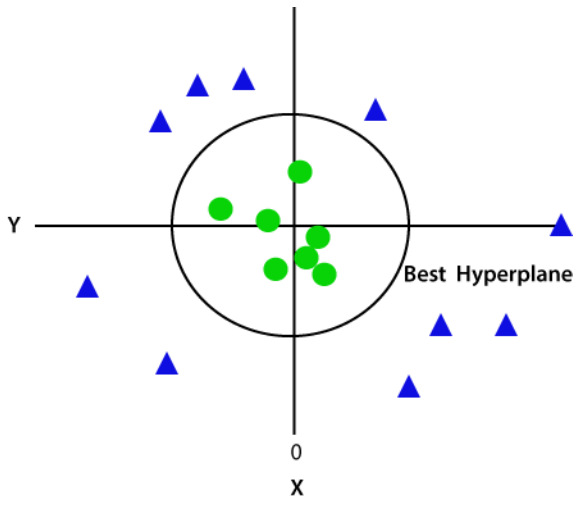


Figure 3.8

Hence we get a circumference of radius 1 in case of non-linear data.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 HARDWARE REQUIREMENTS

- PC/Laptop
- RAM - Minimum 2GB

4.2 SOFTWARE REQUIREMENTS

- Operating System – Windows 10
- Platform – Anaconda Navigator
- IDE – Jupyter Notebook
- Coding Language – Python

4.2.1 PLATFORM – ANACONDA NAVIGATOR

The Platform used in this Project is Anaconda Navigator. Anaconda Navigator is a desktop graphical user interface (GUI) included in **Anaconda** distribution that allows you to launch **applications** and easily manage **conda** packages, environments, and channels without using command-line commands.

It has more than 1500 Python/R data science packages. **Anaconda** simplifies package management and deployment. It has tools to easily collect data from sources **using** machine learning and AI. It creates an environment that is easily manageable for deploying any project.

4.2.2 IDE – JUPYTER NOTEBOOK

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. **Jupyter** is a free, open-source, interactive web tool known as a computational **notebook**, which researchers can **use** to combine software code, computational output, explanatory text and multimedia resources in a single document.

4.2.3 CODING LANGUAGE – PYTHON

Python is an interpreted, object-oriented, high-level **programming language** with dynamic semantics. **Python's** simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. **Python** supports modules and packages, which encourages program modularity and code reuse..

Python is undoubtedly the best choice for **machine learning**. It's easy to understand, which makes data validation quick and practically error-free. By having access to a widely developed library ecosystem, developers can perform complex tasks without extensive coding.

4.2.4 MODULES DESCRIPTION

4.2.4.1 PYSCREENSHOT

The **pyscreenshot module** can be used to copy the contents of the screen to a PIL or Pillow image memory. Replacement for the ImageGrab **Module**, which works on Windows only.

4.2.4.2 SKLEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

4.2.4.3 TIME

The **Python time module** provides many ways of representing **time** in code, such as objects, numbers, and strings. It also provides functionality other than representing **time**, like waiting during code execution and measuring the efficiency of your code.

4.2.4.4 CV2

OpenCV releases two types of Python interfaces, cv and cv2. Cv2 is the latest one. In this, everything is returned as numpy objects.

4.2.4.5 CSV

Python provides a CSV module to handle CSV files. To read/write data, we need to loop through rows of the CSV. We need to use the split method to get data from specified columns.

4.2.4.6 GLOB

Glob is a general term used to define techniques to match specified patterns according to rules related to Unix shell. Linux and Unix systems and shells also support **glob** and also provide function **glob**() in system **libraries**. In **Python**, the **glob module** is used to retrieve files/pathnames matching a specified pattern.

4.2.4.7 PANDAS

Pandas is an open source **Python** package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named Numpy, which provides support for multi-dimensional arrays.

Pandas is used to analyse data. Pandas ia a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

4.2.4.8 MATPLOTLIB

Matplotlib is an amazing visualization library in **Python** for 2D plots of arrays. **Matplotlib** is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

4.2.4.9 JOBLIB

Joblib is such an pacakge that can simply turn our **Python** code into parallel computing mode and of course increase the computing speed. **Joblib** is optimized to be fast and robust in particular on large data and has specific optimizations for numpy arrays.

4.2.4.10 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

CHAPTER 5

DATASETS

5.1 DATASET CREATION

In this Project we created dataset by our own using Pyscreenshot method. In our dataset , first we created 10 folders [0-9] for single digits, and then created 13 folders for double digits, 12 folders for Characters, 6 folders for words, in total we created 41 folders. Each one has an 100 images and totally there are 4100 images. For training we used 80% of dataset and for testing we used 20% of dataset.

5.1.1 FOR SINGLE DIGITS

Sample datasets for Single digits :

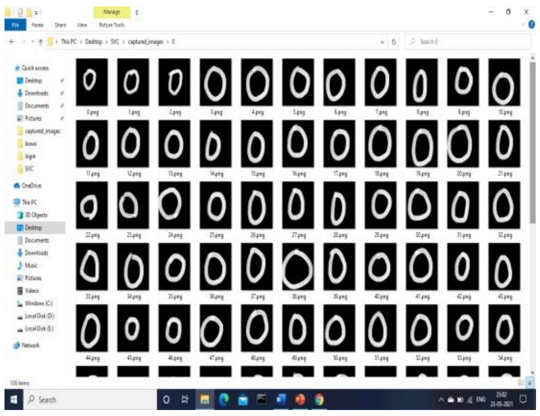


Figure 5.1

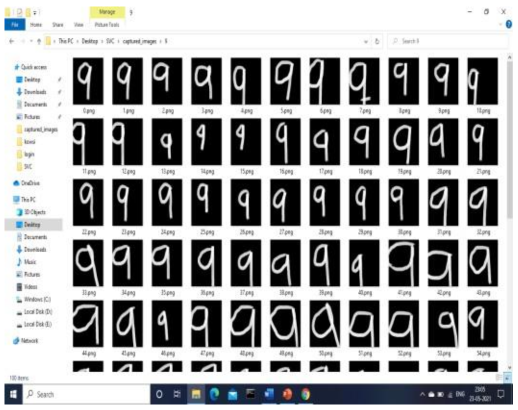


Figure 5.2

5.1.2 FOR DOUBLE DIGITS

Sample datasets for double digits :

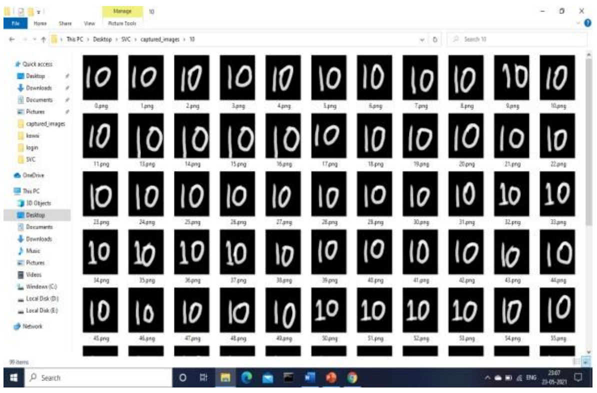


Figure 5.3

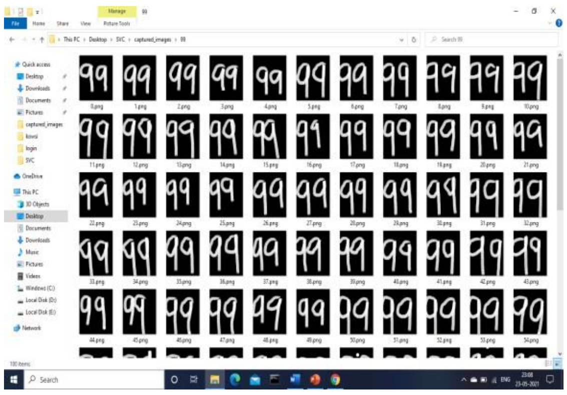


Figure 5.4

5.1.3 FOR ALPHABETS

Sample datasets for Alphabets :

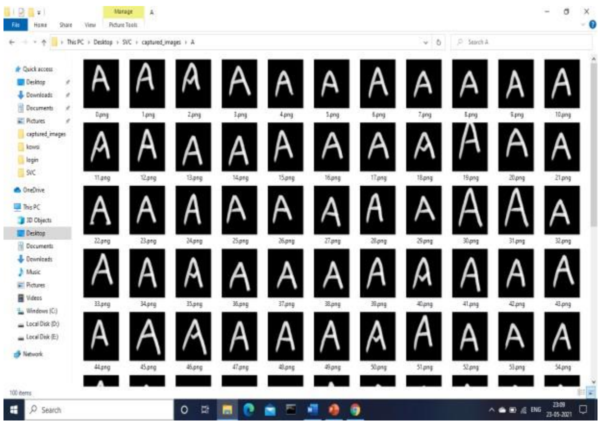


Figure 5.5

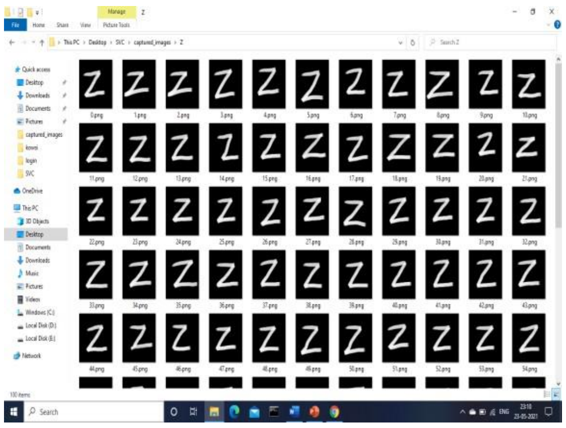


Figure 5.6

5.1.4 FOR WORDS

Sample datasets for Words :

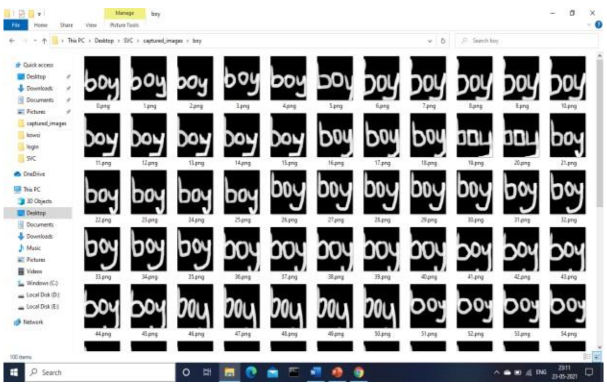


Figure 5.7

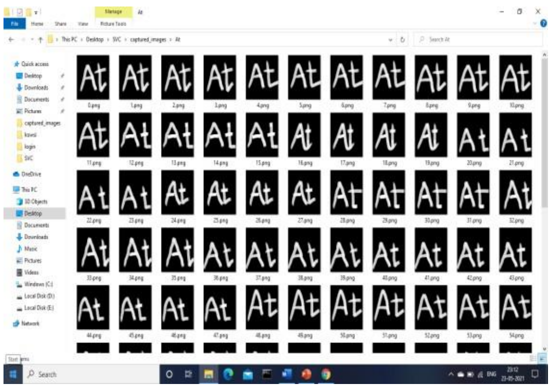


Figure 5.8

CHAPTER 6

IMAGE PROCESSING INVOLVED IN HANDWRITING RECOGNITION

SVM MODEL :

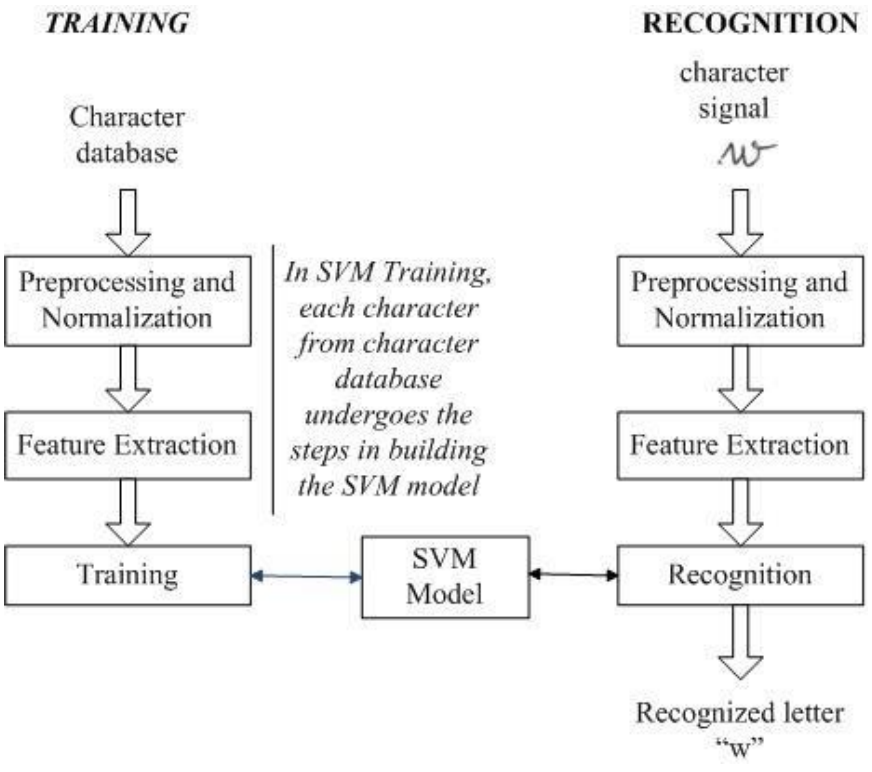

TRAINING
RECOGNITION
Character database
character signal
Preprocessing and Normalization
Feature Extraction
Training
In SVM Training, each character from character database undergoes the steps in building the SVM model
SVM Model
Preprocessing and Normalization
Feature Extraction
Recognition
Recognized letter "w"

Figure 6.1

6.1 PREPROCESSING OF SAMPLE IMAGE

Pre-processing of the sample image involves few steps that are mentioned as follows:

Grey-scaling of RGB image

Grey-scaling of an image is a process by which an RGB image is converted into a black and white image. This process is important for Binarization as after greyscaling of the image, only shades of grey remains in the image, binarization of such image is efficient

Binarization

Binarisation of an image converts it into an image which only have pure black and pure white pixel values in it. Basically during binarization of a grey-scale image, pixels with intensity lower than half of the full intensity value gets a zero value converting them into black ones. And the remaining pixels get a full intensity value converting it into white pixels.

Inversion

Inversion is a process in which each pixel of the image gets a colour which is the inverted colour of the previous one. This process is the most important one because any character on a sample image can only be extracted efficiently if it contains only one colour which is distinct from the background colour. Note that it is only required if the objects we have to identify if of darker intensity on a lighter background.

The flow chart shown below illustrates the physical meaning of the processes that are mentioned above:

RGB => Grey-scaling => Binarization => Inversion

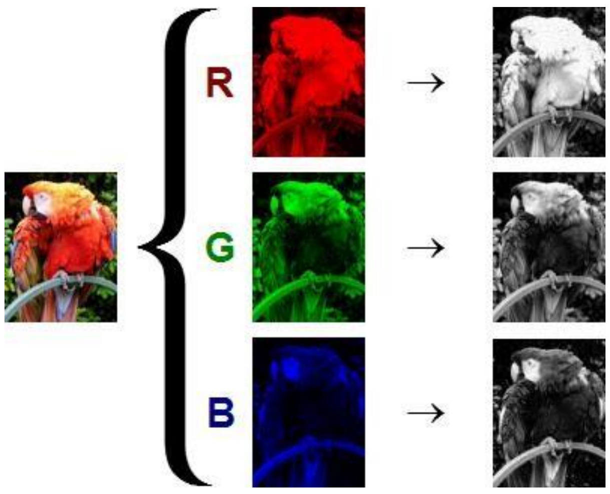


Figure 6.2

6.2 FEATURE EXTRACTION

Features of a character depicts the morphological and spatial characteristics in the image. Feature extraction is a method of extracting of features of characters from the sample image. There are basically two types of feature extraction:

- ♣ Statistical feature extraction
- ♣ Structural feature extraction

<u>Statistical feature extraction</u>

In this type of extraction the extracted feature vector is the combination of all the features extracted from each character. The associated feature in feature vector of this type of extraction is due to the relative positions of features in character image matrix.

<u>Structural feature extraction</u>

This is a primitive method of feature extraction which extracts morphological features of a character from image matrix. It takes into account the edges, curvature, regions, etc. This method extracts the features of the way character are written on image matrix.

The different methods used for feature extraction are

- ➢ Piecewise –linear regression
- ➢ Curve-fitting
- ➢ Zoning
- ➢ Chain code, etc.

The functions that are used in feature extraction are:

<u>Indexing and labelling</u>

This is a process by which distinct characters in an image are indexed and labelled in an image. Thus helps in classification of characters in image and makes feature extraction of characters simple.

Boxing and Cropping

This is a process of creating a boundary around the characters identified in an image. This helps by making cropping of characters easier. After boxing the characters are cropped out for storing them as input variables for recognition.

Reshaping and Resizing

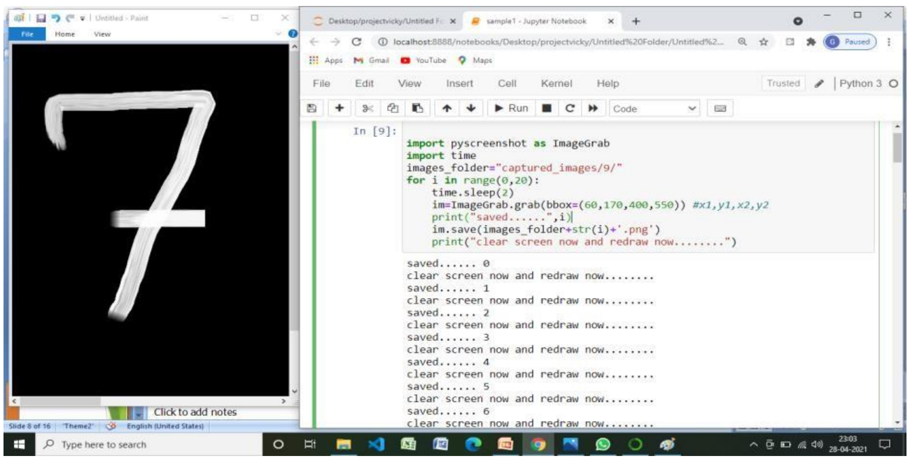
Reshaping is done to change the dimensions of the acquired character in desired shape. Resizing is done to reduce the size of characters to a particular minimum level.

CHAPTER 7
IMPLEMENTATION AND RESULTS

7.1 IMPLEMENTATION

7.1.1 DATASET CREATION

- Using pyscreenshot we have created dataset by our own .



```
In [9]:
import pyscreenshot as ImageGrab
import time
images_folder="captured_images/9/"
for i in range(0,20):
    time.sleep(2)
    im=ImageGrab.grab(bbox=(60,170,400,550)) #x1,y1,x2,y2
    print("saved......",i)
    im.save(images_folder+str(i)+'.png')
    print("clear screen now and redraw now........")
```

```
saved...... 0
clear screen now and redraw now........
saved...... 1
clear screen now and redraw now........
saved...... 2
clear screen now and redraw now........
saved...... 3
clear screen now and redraw now........
saved...... 4
clear screen now and redraw now........
saved...... 5
clear screen now and redraw now........
saved...... 6
clear screen now and redraw now........
```

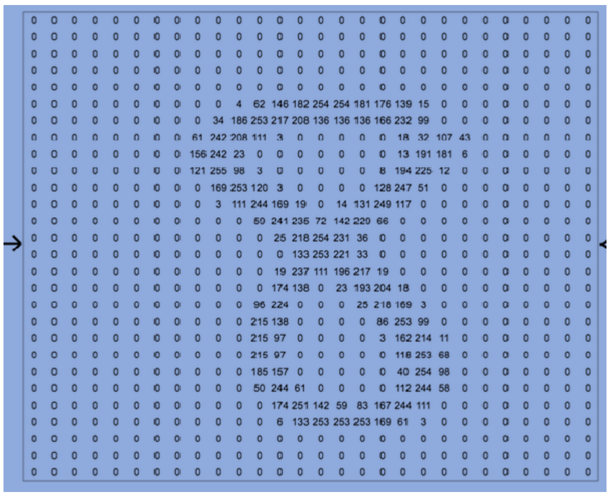
Figure 7.1

- Pyscreen shot will capture the entire screen ,so we have used bbox
  im=ImageGrab.grab(bbox=(60,170,400,550)) #x1,y1,x2,y2
- bbox means an bounding box it returns xy co ordinates all items in image
- And then we imported time module in it
  time.sleep(1)
- So it takes screenshot after that given time
  for i in range(0,100):
- Like this it will loops for n times that we have given.

7.1.2 GENERATE DATASET

- Using for loop we appended pixel value
  from 0 to 783
  for i in range(0,784):
  header.append("pixel"+str(i))
- Using glob module we extracted captured_images folder
- In this captured_images folder ,we have stored our dataset
- Using cv2 module it loads an image from captured_images folder and it will read all the images
  im_gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
- And then we have to change RGB colour to grayscale image ,because RGB has lot of complexities and it is harder to proceses the colour images ,so we converted into grayscale image
  im_gray = cv2.GaussianBlur(im_gray,(15,15), 0)

  Using Gaussian blur method we blur the images,this will remove unwanted noises.



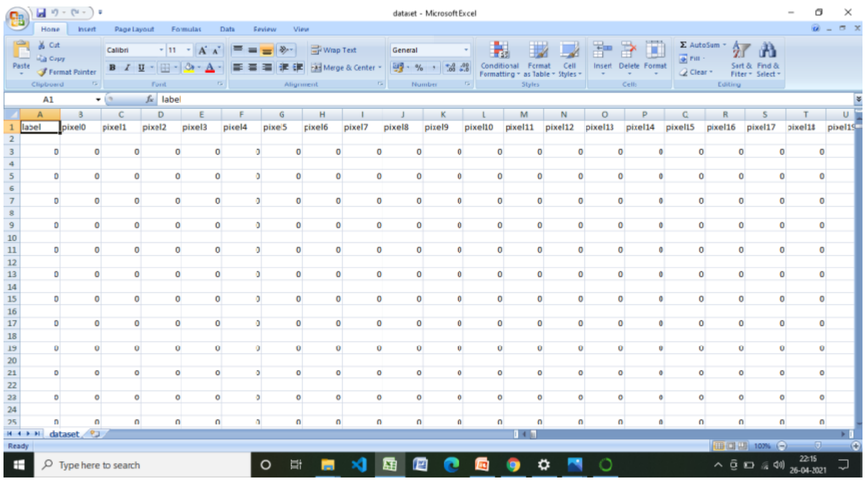
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.2

roi= cv2.resize(im_gray,(28,28),
interpolation=cv2.INTER_AREA)

- Using interpolation function resize the image into 28*28 pixel, it fills the missing values .

7.1.3 ADD PIXEL INTO DATA ARRAY



dataset - Microsoft Excel

A1 | label

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17	pixel18	pixel1
2																					
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4																					
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6																					
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8																					
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10																					
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12																					
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14																					
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16																					
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18																					
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20																					
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22																					
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24																					
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

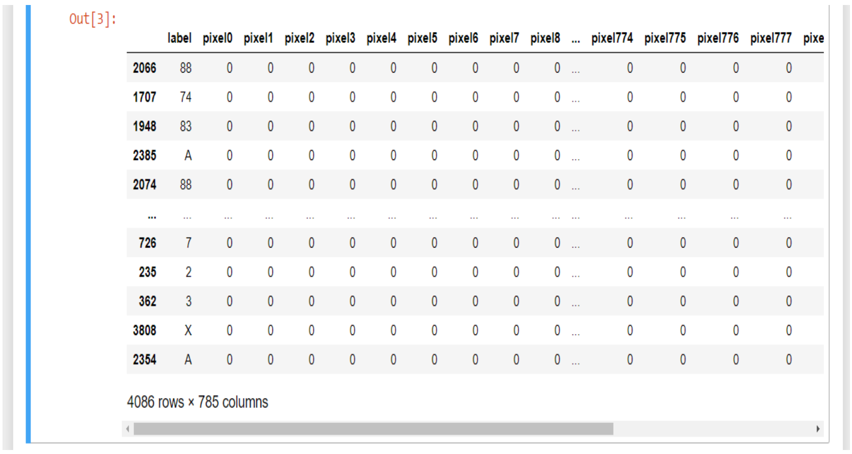
dataset

Ready

Figure 7.3

- In this part we appended the data ie..0 & 1 value from pixel 0 to pixel 783

7.1.4 LOAD THE DATASET



```
Out[3]:
      label pixel0 pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 ... pixel774 pixel775 pixel776 pixel777 pixe
2066     88      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
1707     74      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
1948     83      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
2385      A      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
2074     88      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
 ...    ...    ...    ...    ...    ...    ...    ...    ...    ...    ... ...      ...      ...      ...      ...
 726      7      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
 235      2      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
 362      3      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
3808      X      0      0      0      0      0      0      0      0      0 ...        0        0        0        0
2354      A      0      0      0      0      0      0      0      0      0 ...        0        0        0        0

4086 rows × 785 columns
```

Figure 7.4

```
data=pd.read_csv('dataset.csv',low_memory=False)
```

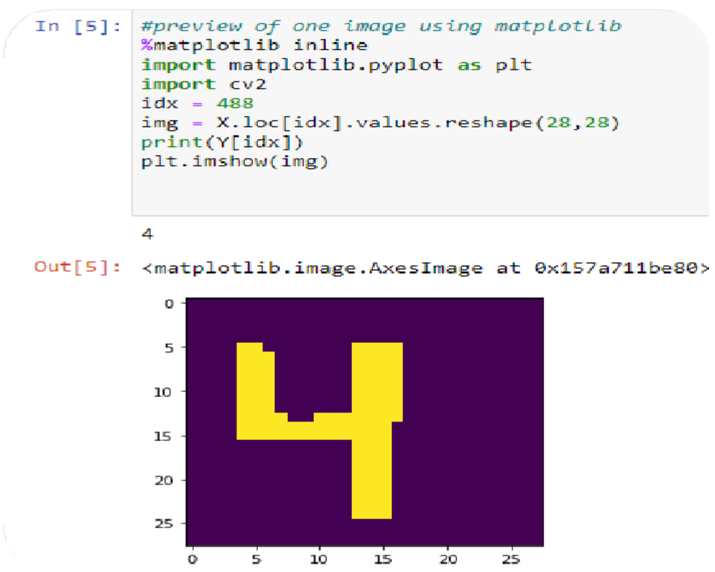
- Using pandas library ,we have loaded the dataset (ie..,created before)

7.1.5 SEPARATION OF DEPENDENT AND INDEPENDENT VARIABLE

- We have splitted dataset into two types one is training dataset and another one is testing dataset
- Training dataset is stored in the form of x train and y train
- Testing dataset is stored in the form of x test and y test

  X = data.drop(["label"],axis=1)
  Y= data["label"]
- X is an independent variable
- Y is an dependent variable
- Using this x independent variable only we can predict dependent variable

7.1.6 PREVIEW OF IMAGE



```
In [5]: #preview of one image using matplotlib
        %matplotlib inline
        import matplotlib.pyplot as plt
        import cv2
        idx = 488
        img = X.loc[idx].values.reshape(28,28)
        print(Y[idx])
        plt.imshow(img)

        4
Out[5]: <matplotlib.image.AxesImage at 0x157a711be80>
```

Figure 7.5

idx = 34
img = X.loc[idx].values.reshape(28,28)

- In this part using matplotlib we have previewed one image by index value from the loaded dataset

7.1.7 TRAIN-TEST SPLIT

- We have splitted ,80% of dataset for training the model and 20% of dataset for testing the model
  train_x,test_x,train_y,test_y = train_test_split(X,Y, test_size = 0.2)
- Always training dataset should be greater than the testing dataset

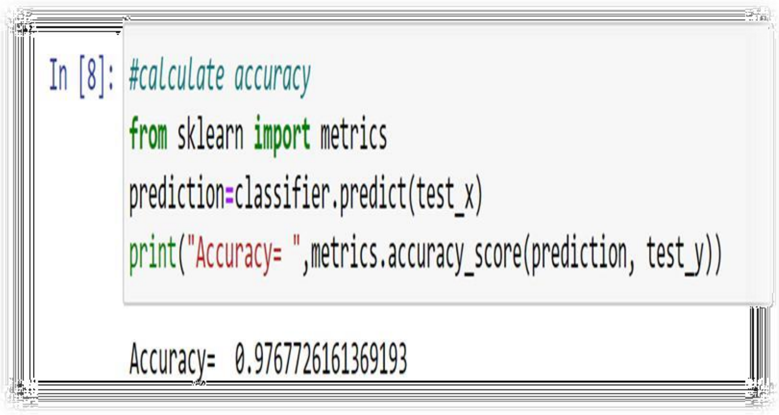
7.1.8 FIT AND SAVE THE MODEL

- In this part ,we have fitted the model using svc
  classifier=SVC(kernel="linear", random_state=6)
- Saved the model using joblib
  joblib.dump(classifier, "model/digit_recognizer")

7.1.9 ACCURACY CALCULATION

- In this testing dataset x test and y test is used to predict the model and calculate its accurancy

  prediction=classifier.predict(test_x)
  print("Accuracy=",metrics.accuracy_score(prediction, test_y))



```
In [8]: #calculate accuracy
        from sklearn import metrics
        prediction=classifier.predict(test_x)
        print("Accuracy= ",metrics.accuracy_score(prediction, test_y))

        Accuracy=  0.9767726161369193
```

Figure 7.6

- Accurancy for this Project is 0.9767(97%).

7.2 RESULTS

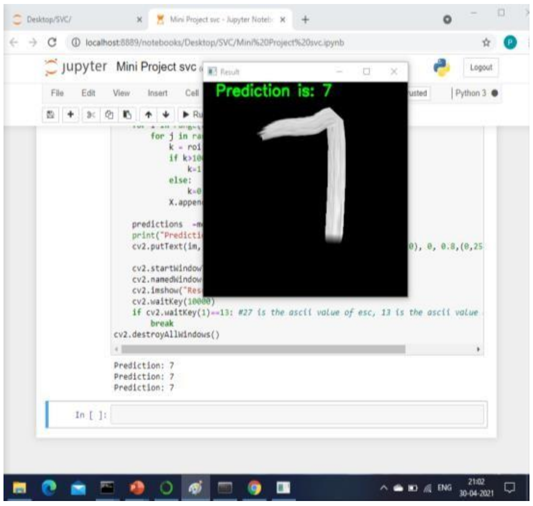
7.2.1 PREDICTION OF IMAGE

- In this we have threshold the image ,it seperates black and white pixel image and it will resize the image into 28*28 pixel

```
#Threshold the image
ret, im_th = cv2.threshold(im_gray,100, 255, cv2.THRESH_BINARY)
roi = cv2.resize(im_th, (28,28), interpolation  =cv2.INTER_AREA)
```

- Add pixel one by one into data array as same as we have done in before

```
## Add pixel one by one into data array
for i in range(rows):
for j in range(cols):
    o  k = roi[i,j]
    o  if k>100:
           ▪  k=1
    o  else:
           ▪  k=0
    o  X.append(k)
```



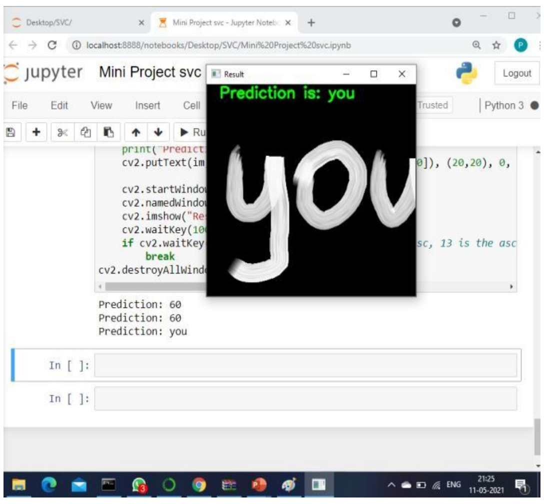
```
        for j in ra
            k = roi
            if k>10
                k=1
            else:
                k=0
            X.appen

    predictions  =m
    print("Predicti
    cv2.putText(im,                                    0), 0, 0.8,(0,25

    cv2.startWindow
    cv2.namedWindow
    cv2.imshow("Res
    cv2.waitKey(10000)
    if cv2.waitKey(1)==13: #27 is the ascii value of esc, 13 is the ascii value
        break
cv2.destroyAllWindows()

Prediction: 7
Prediction: 7
Prediction: 7
```

Figure 7.7



```
    print("Predict
    cv2.putText(im                                0]), (20,20), 0,

    cv2.startWindo
    cv2.namedWindo
    cv2.imshow("Re
    cv2.waitKey(10
    if cv2.waitKey                                sc, 13 is the asc
        break
cv2.destroyAllWind

Prediction: 60
Prediction: 60
Prediction: you
```

Figure 7.8

- We have predicted the image using paint and pyscreenshot
- And then using time module it will predict after the completion of that given time .It loop continuously until we break it.

CHAPTER 8

CONCLUSION AND FUTURE IMPROVEMENTS

8.1 CONCLUSIONS

Classification of characters and learning of image processing techniques is done in this project. Also the scheme through which project is achieved is Support Vector Classifier. The result which was got was correct up to more than 90% of the cases, but it would be improved at the end. This work was basically focused on envisaging methods that can efficiently extract feature vectors from each individual character. The method I came up with gave efficient and effective result both for feature extraction as well as recognition. There are also different methods through which ‘handwriting recognition’ is achieved.

8.2 FUTURE SCOPE OF THIS PROJECT

The application of this HCR algorithm is extensive. Now-a-days recent advancement in technologies has pushed the limits further for man to get rid of older equipment which posed inconvenience in using. In our case that equipment is a keyboard. There are many situations when using a keyboard is cumbersome like,

- We don’t get fluency with keyboard as real word writing
- When any key on keyboard is damaged
- Keyboard have scripts on its keys in only one language
- We have to find each character on keyboard which takes time
- In touch-enabled portable devices it is difficult to add a keyboard with much ease

On the other hand if we use an HCR software in any device, we can get benefits like,

- Multiple language support
- No keyboard required
- Real world writing style support
- Convenient for touch enabled devices
- Previously hand written record can be documented easily
- Extensive features can also be added to the software like,

    1. Translation
    2. Voice reading

REFERENCES

1. G.Vamvakas, B.Gatos, N. Stamatopoulos, and S.J.Perantonis, "A Complete Optical Character Recognition Methodology for Historical Documents", the Eighth IAPR Workshop on Document Analysis Systems

2. Debasish Basa ,Sukadev Meher , "Handwritten Odia Character Recognition", Presented in the National Conference on Recent Advances in Microwave Tubes, Devices and Communication Systems, Jagannath Gupta Institute of Engineering and Technology, Jaipur, March 4-5,2011

3. Rajean Plamondon, Fellow IEEE and Sargur N. Srihari, Fellow IEEE , "On-Line and Off-Line Handwriting character Recognition: A Comprehensive Survey", 1EEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. VOL. 22, NO. 1. JANUARY 2000

4. Sanghamitra Mohanty, Hemanta Kumar Behera , "A Complete OCR Development System For Oriya Script"

5. B.B. Chaudhuri U. Pal M. Mitra, "Automatic Recognition of Printed Oriya Script"

6. Fumitaka KIMURA, Tetsushi WAKABAYASHI and Yasuji MIYAKE, "On Feature Extraction for Limited Class Problem"

7. Supriya Deshmukh, Leena Ragha, "Analysis of Directional Features - Stroke and Contour for Handwritten Character Recognition", 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009

8. M. Blumenstein, B. K. Verma and H. Basli, "A Novel Feature ExtractionTechnique for the Recognition of Segmented Handwritten Characters", 7th International Conference on Document Analysis and Recognition (ICDAR '03) Eddinburgh, Scotland: pp.137-141, 2003.

9. Tapan K Bhowmik Swapan K Parui Ujjwal Bhattacharya Bikash Shaw, " An HMM Based Recognition Scheme for Handwritten Oriya Numerals".

10. Sung-Bae Cho, “Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals”, IEEE Transactions on Neural Networks, Vol. 8, No. 1, January 1997

11.N. Tripathy and U. Pal, “Handwriting Segmentation of Unconstrained Oriya Text”

12. BAHL, L. R., BROWN, P.F., SOUZE, P.V.DE, AND MERCER, R.L. (1986) Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. ICASSP 86.

13. BAHL, L. R., BROWN, P.F., SOUZE, P.V.DE, AND MERCER, R.L. (1992) Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition. ICASSP 86.

14. BAHLMANN, C. B., H. (2004) The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. Ieee Transactions on Pattern Analysis and Machine Intelligence, 26, 299-310.

15. BAHLMANN, C. H., B. BURKHARDT, H. (2002) Online handwriting recognition with support vector machines - a kernel approach. Eighth International Workshop on Frontiers in Handwriting Recognition, 2002.

16. BAKER, J. (1975) Stochastic Modeling as a Means of Automatic Speech Recognition. Carnegie Mellon University.

17. BAUM, L. E. P., T. , SOULES, G. AND WEISS N. (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann. Math. Statist, 41, 164 -171.

18. BEIGI, H. M. S., NATHAN, K., SUBRAHMONIA, J. (1995) Online Uncostraint Handwriting Recognition Based on Probabilistic Techniques.

19. BEIGI, H. S. M. N., K. CLARY, G.J. SUBRAHMONIA, J. (1994) Size normalization in on-line unconstrained handwriting recognition. IEEE International Conference Image Processing, 1994. ICIP-94., . Austin, TX, USA.

20. BELLEGARDA, E. J. B., J.R. NAHAMOO, D. NATHAN, K.S. (1994) A fast statistical mixture algorithm for on-line handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16, 1227-1233.