Online Handwritten Character Recognition Using Wavelet-Based Features

Dissertation submitted in partial fulfillment of requirement for award of degree of

Master of Technology

in

Computer Science and Applications

Submitted By:

Avneet Kaur

(Roll No. 601203005)

Supervised By:

Dr. R. K. Sharma

SMCA, Thapar University, Patiala

SCHOOL OF MATHEMATICS AND COMPUTER APPLICATIONS

THAPAR UNIVERSITY

PATIALA – 147004

July, 2014

Certificate

I hereby certify that the work which is presented in the dissertation entitled, "Online Handwritten Character Recognition using Wavelet-based Features", in partial fulfillment of the requirements for the award of degree of the Master of Technology in **Computer Science and Applications**, submitted in School of Mathematics and Computer Applications, Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. R. K. Sharma and refers others researcher's work which are duly listed in the reference section.

The matter presented in this dissertation has not been submitted for award of any other degree or in any other university.

(Avneet Kaur)

601203005

This is to certify that above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. R. K. Sharma)

Professor

School of Mathematics and Computer Applications

Countersigned by:

(Dr. Rajesh Kumar)

Head

School of Mathematics and Computer Applications

Thapar University

Patiala

(Dr. S. K. Mohapatra)

Dean Academic Affairs

Thapar University

Patiala

Acknowledgement

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life.

This work would not have been possible without the encouragement and able guidance of my supervisor Dr. R. K. Sharma. I thank my supervisor for her time, patience, discussions and valuable comments. Her enthusiasm and optimism made this experience both rewarding and enjoyable.

I am equally grateful to Dr. Rajesh Kumar, Associate Professor and Head, School of Mathematics and Computer Applications, for motivation and inspiration that triggered me for the dissertation work.

I will be failing in my duty if I don't express my gratitude to Dr. S. K. Mohapatra, Senior Professor and Dean of Academic Affairs of University, for making provisions of infrastructure such as library facilities, immensely useful for the learners to equip themselves with the latest field.

I am also thankful to the entire faculty and staff members of School of Mathematics and Computer Applications Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable.

Last but not least, I would like to thank my parents for their wonderful love and encouragement, without their blessings none of this would have been possible. I would also like to thank my close friends for their constant support.

(Avneet Kaur)
601203005

Abstract

In today's era, everyone is busy to the fullest and with increasing digitization everyone wants one's work to be completed easily and as fast as possible. If it was possible to give input to a digital machine in our own handwriting and in our own native language, it would have been easier and probably less time consuming. Researchers are continuously working on handwriting recognizers. One of the basic steps in building a handwriting recognizer is recognition of handwritten characters. Handwritten Character Recognition can be done in two ways, namely, Offline and Online. This dissertation deals with the recognition of Online Handwritten Punjabi Characters.

The steps involved in this process are capturing the inputs, preprocessing the data, extracting the features and finally the classification or recognition of the character captured. Feature extraction is a very important step in this process. A number of extraction methods have been proposed in literature for this process.

This dissertation targets this field of research in the Punjabi language. Inputs were collected from various users in their handwritings using digital input. The Punjabi Character or Numeral taken as input was then converted to an image. This image was preprocessed to bring it to a standard size. Three different sizes have been tested in this work. The features have been extracted through wavelet decomposition of the preprocessed images. The features thus obtained are the wavelet coefficients. These coefficients have been used to train the SVM classifier to get the accuracy of the overall recognition system, *i.e.*, percentage of correct classification or recognition of the character or numeral given as input. Complete methodology of the system and its overall performance and effectiveness has been demonstrated with the help of suitable examples.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Online Handwritten Character Recognition, as the name suggests, refers to recognizing the input characters that are written by the user instead of typing it in. It is similar to the Optical Character Recognition where instead of recognizing the printed characters, the handwritten characters are recognized.

1.1 Use of online handwritten character recognition

With advancements in digital technology field so far technologists have continuously been making efforts to help the world go digital by digitizing each and every possible thing around us from voice to image. As handwriting is the widely used way to communicate digitization could not stay far from handwritten characters too. For years researchers have been trying to build an efficient machine that can recognize the handwritten characters written by human beings by using the information contained in those characters by converting them into digital form to as much extent as possible. They have been trying to build devices that can take handwritten characters as input from users as it is the easiest way of communication for the users especially if they can write in their native language.

In today's era one tries to capture his work as easily as possible. Therefore, the research in this field can be very fruitful. If people can use their native language digitally like in their mobile phones, internet, personal computers their task will be easy and will involve less time as whatever task they do in their native language will be easier for them to execute. Many of the researchers have worked in this field and have tried to build the concept of recognizing handwritten character or numerals that gives a good amount of accuracy, *i.e.,* a machine that can recognize handwritten characters almost every time correctly. Researches are being carried out to improve this accuracy upto 100% to get perfect recognition of the character or numerals that are given as input by users in the handwritten form. Various researchers have contributed to this by building the whole new concepts giving a huge number of techniques to recognize the character/numeral that builds a huge spectrum of recognition methods.

1.2 Steps to recognize a handwritten character or numeral

Figure 1.1 represents the steps involved in recognition of a character entered by a user by hands, *i.e.*, handwritten characters [18].

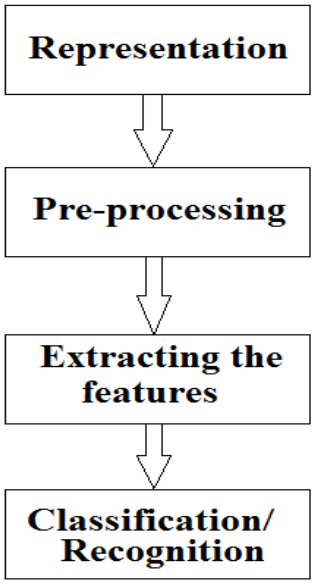


Figure 1.1: Steps to recognize a handwritten character

1.2.1 Representation of the character

The handwritten character that is taken as an input from a user is to be represented in digital format so that the machine or the software can recognize it, *i.e.*, to start the recognition process the character or numeral has to be represented digitally.

There are many ways of representing a character digitally like representing it in the form of strokes where stroke is nothing but the part of the character or numeral written without lifting the pen. The other ways include capturing it as an image, *etc*.

1.2.2 Preprocessing

Pre-processing, as the name expresses, involves the processing that is done before the character/numeral is recognized, *i.e.*, preparing the character in the form suitable for recognition process [28]. It includes techniques like smoothening, normalization, binarization, controlling the skewness and removal of the slant.

These techniques are discussed below, in brief.

- Normalization is one of the widely used technique for pre-processing. It involves resizing the image to a particular size. As the character written by the user can vary in size, it can be very difficult to recognize the character/numeral if it is to be recognized as it is. Therefore, a standardization is done, *i.e.*, a standard size is decided and all the characters that are taken as input from users are normalized or resized to that size before recognition process is carried out.

- Smoothening involves removing the noise or extra information that may arise along with the input to make the input better.

- Removing the slant involves changing the slanting level of the handwritten character to a particular pre-decided level to ease the recognition as the user can write at any slant angle.

- Binarization involves developing the binary images from the gray-scale images as binary data is easy for the machine to understand.

- Controlling the skewness involves aligning the direction of the character/numeral written according to the co-ordinate system of the digital pad used for input.

1.2.3 Extracting the features

1.2.3.1 What is a feature ?

Feature also known as attribute is nothing but anything that can describe an object (character in our case) perfectly [7]. It can be a temporal property, physical property, geometrical property, spectral property, spatial property, *etc*. For example, size of characters, shape of character, width of character, pixels of character, loops in the character, *etc.*

1.2.3.2 Analysis of the features

As discussed above, there is a large variety of the features available that can represent the character uniquely.

Deciding what feature will be included in the feature vector for a character is a major step in the whole recognition process. The process of deciding on the property to be chosen to describe the object is called *feature selection process* [7].

For this purpose, an analysis of features is done according to the application domain in order to find the best suited features for that particular application.

1.2.3.3 Feature Vector

Once the feature is decided, it's corresponding values that represent the character efficiently are kept in a vector known as *feature vector* [11].

It may contain *n* number of values corresponding to the feature.

1.2.3.4 Extraction of the features

Every information collected in the representation step to represent the character is not necessary, *i.e.,* there is redundant information to a large extent in that representation. The information that is not required or that is duplicate/redundant has to be cleaned up.

Extracting the feature majorly involves building a vector that can represent the character properly with minimal information in it. Thus, reducing any redundant information included at first, *i.e.,* building a minimal feature vector [28].

Therefore, the major goal of this step is to

- compress data for the vector.
- ensuring that data describes the character properly [11].

1.2.3.5 Feature Space

A two dimensional or three dimensional space diagram of the values in the feature vector that helps in building an approximate view of the character and comparing it with its neighbor classes (feature vectors for other characters) [7].

1.2.3.6 Types of features

Features can be divided into the categories namely, high-level features and low-level features; global and local features; statistical and structural features.

- **High-level features and Low-level features**

High-level features:

High-level feature set includes only the significant features. High-level features are the features that are extracted in a particular order. Thus high-level feature extraction involves reduction in the redundancy but in a way that the important information required to differentiate the character from other characters is retained [17]. For example, global features, transition histograms, segmentation features, *etc*.

Low-level features:

Low-level features are the features that we can directly extract from the character's or numeral's image. These are the features that are basic in sense they do not require any information regarding the shape or space of the character [17]. For example, features through edge detection, curvature detection, profile and projection histograms, corner detection, image motion, *etc.*

- **Global features and Local features**

Global features:

Global features are the features that describe anything on general terms. It does not describe the minute details, instead it gives the overall description of the character in general [17]. For example, features represented by transition histograms, segmentation features, *etc.*

Local features:

Local features are the features that are extracted with minute details in a way that the character or numeral represented by a particular feature set can easily be differentiated from their neighbor feature sets representing different characters efficiently [41]. For example, points, intensity, regions, edges, textures, corners, color features *etc*.

- **Structural and Statistical features**

Structural features:

Structural features that are based on the structure of the character represented in the image, *i.e.*, geometry of the character or its topology. That is why they are also called topological features [1]. These kinds of features are based on finding what kind of structure the character is forming like information about loops, edges, corners, *etc*.

These kinds of features can work well even with high amount of distortions and vast differences in writing style [11].

Statistical features:

Structural features are the features that involve statistics and numbers, *i.e.,* using quantitative properties stored in the character image. It involves analyzing the distribution formed by the image of the character/numeral [7]. For example, area, aspect ratio, center of mass, *etc*.

1.2.3.7 Feature filter metrics

Feature filter metrics describe the feature that is filtered among vast variety of metrics available to describe the character [7]. Some of them are discussed below.

- **Horizontal, vertical and diagonal lines**

Number of vertical lines, diagonal lines and horizontal lines and their placement in the image representing the character from the feature vector [39].

- **Aspect ratio**

Aspect ratio is defined the ratio of the broadness to the length/height of the image representing the character or numeral. It is also known as *elongation* [36].

$$Elongation = \frac{broadness\ of\ image}{height\ of\ image} = \frac{Xmax - Xmin + 1}{Ymax - Ymin + 1} \quad \ldots (1.1)$$

It tells about the extent of stretch of the character in both the directions. Low aspect ratio indicates more stretch of the character towards the vertical direction.

- **Open and closed curves**

Finding whether the curves in the image representing the character are open or are closed [48]. Figure 1.2 illustrates this in detail.

Figure 1.2: [a] closed curve, [b] open curve

- **Area of the character**

Area of the character in the image that represents the character can also describe its shape and size [18].

- **Concavities and convexities in characters**

Concavity is the property of being inward and convexity being outward [43].

As shown in Figure 1.3, the characters have their own concavities and convexities and these can be utilized as a metric.

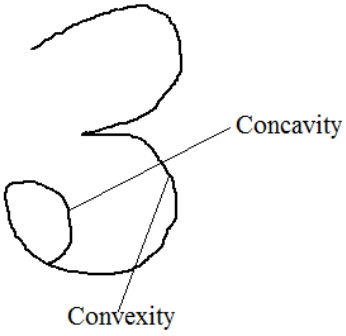


Figure 1.3: Example of convexity and concavity in a character

- **Center of the character**

The center (x, y) of the character relative to the image representing it can also be considered as a filter metric [18].

- **Ascenders/descenders**

Ascender is the portion of the character that moves above the peak line of the zone whereas the descender moves beneath the baseline of the zone [17]. Figure 1.4 shows the illustration.

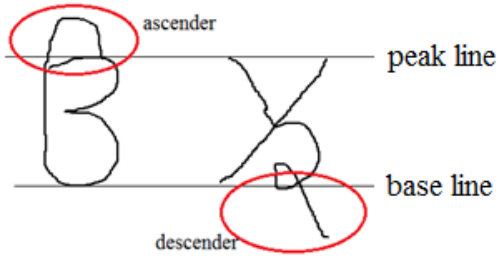


Figure 1.4: An illustration of ascenders and descenders

- **Cross points and end points**

Cross point is a point where two segments of a character cross each other [39] and end points are the ending points of the character [43] as shown in Figure 1.5.

- **Holes in the character**

The number of holes in the character can also be one of the metrics. This measure is known as *Euler number*.

Holes are the blocked curves within the character [43]. Figure 1.6 illustrates this concept in detail.

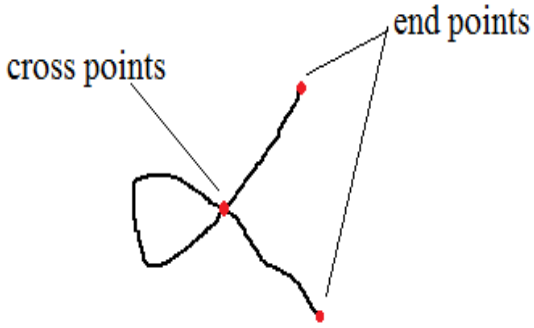


Figure 1.5: cross and end points

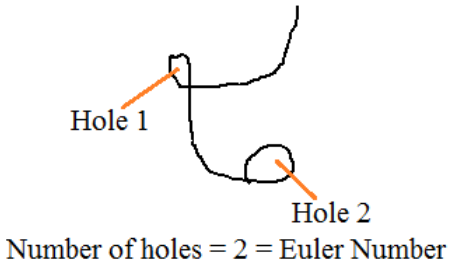


Figure 1.6: Number of holes

- **Perimeter of the character**

The perimeter of a character is the sum of number of pixels in the every segment of the character [18].

It can help in finding the character's location in the image that is representing the character.

- **Loops**

Loop is anything that has its end point same as its starting point. For example, if A, B, C and D represents 4 different points, then the path ABCDA will be a loop as starting and end point are same, *i.e.,* A. Figure 1.7 illustrates a loop.

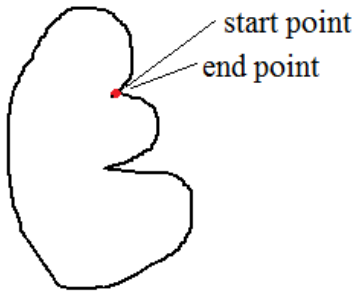


Figure 1.7: An illustration of a loop in the character

- **Thinness ratio**

It combines two features, first the area of the character and second the perimeter of the character.

It is the ratio of the area to the perimeter square [18].

- **Distribution of pixels or dots**

It uses the pixels of the character to define the distribution they are following [36], *e.g.,* normal distribution, uniform distribution, *etc.*

- **Irregularity ratio**

Irregularity ratio also known as compactness ratio is the reciprocal of the thinness ratio [18].

It helps in finding out how regular the character is.

- **Radial profiles**

Radial means along the radius. Thus, a radial profile is the profile of the character taken along the radius if the character image is considered to be circular rather than being rectangular [28].

- **Strokes and their directions: vertical, horizontal and diagonal**

Strokes as discussed before are the portion of the character that is written without lifting the writing instrument, *i.e.*, pen, pencil, digital pen, *etc*. above the writing surface [36].

The strokes and their directions can also be one of the metrics. Figure 1.8 illustrates the strokes.

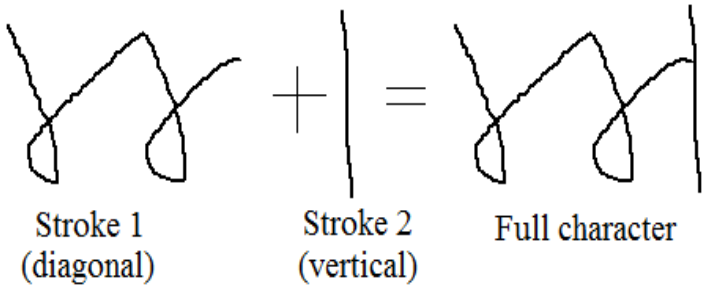


Figure 1.8: Strokes example

  - **Discrete Wavelet Transform measures**

    DTW is a new method that measures the distance between two strokes [12].

    This distance tells us how apart the different strokes are from each other and for this every feasible positioning between them is considered.

- **Inflection points**

These are points where the character changes its concavity to convexity or vice versa [28]. Figure 1.9 illustrates this in detail.

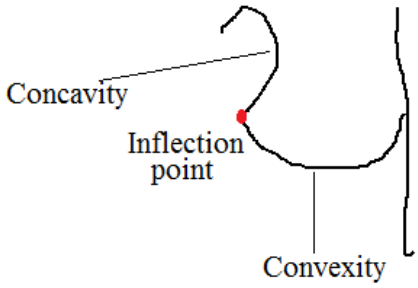


Figure 1.9: Inflection point illustration

- **Location of maximum and minimum**

The location where the character reaches maximum and where it reaches the minimum in both the directions, *i.e.,* horizontal and vertical is also a metric [33].

- **Junctions**

As shown in Figure 1.10, junctions are the points where two strokes join each other [12].

- **Corners**

Corner is any point where two segments of the character meet each other [11]. Figure 1.11 shows an example.

- **Edges**

Edges are the segments forming the character as shown in Figure 1.11 [18].

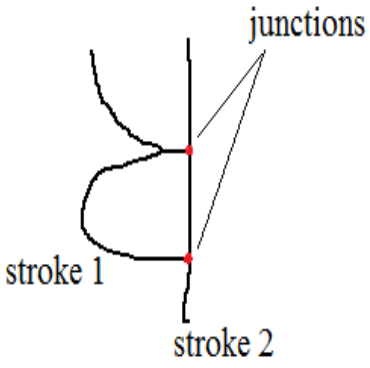


Figure 1.10: An illustration of junctions

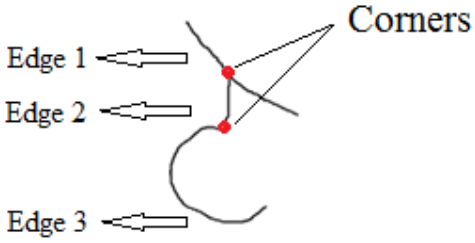


Figure 1.11: Corners and edges example

- **North, east, west and south curves**

Finding the angles from north, east, west and south directions of the curves forming the character gives one of the metric [36]. This is found by drawing tangent on every edge and finding the angles formed between two tangents from all the four directions.

- **Center of mass**

Center of mass as we know is the portion where the whole mass of the object resides. In terms of a character, the center of mass is the location where maximum pixels of the character are present [42].

- **Regular and irregular regions inside the character**

Regular regions are the regions that have a particular pattern in their formation that is repeating itself periodically [39].

Contrary to these are the regions that does not form any such pattern in their formation. They are known as irregular regions. Figure 1.12 represents the regular and irregular regions of the characters in details along with the pattern that repeats periodically.

- **Segment information**

Segment information includes

- co-ordinates of every segment
- length of the segments
- direction of connectivity between segments
- slope of the segments
- shape of the segments
- their alignment, *i.e.,* angles formed by the tangents from the different adjacent segments [41].

All this information helps in knowing the character formation and ultimately helps in recognizing the character [9].

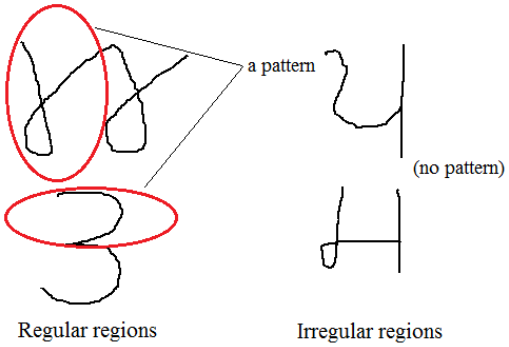


Figure 1.12: Regular and irregular regions

- **Crossings and distances**

Crossing is the number of times the image representing the character changes its pixels to foreground from background pixels along both the directions, *i.e.*, horizontal and vertical [1].

Distances are the measures of the character's first pixel inside the image from the left, right, top and bottom [16].

- **Character width and height**

Width of the character is how broad the character is, *i.e.*, to which extent it extends in horizontal direction and height of the character is the extent to which it extends in the vertical direction [3]. Figure 1.13 represents this metric.

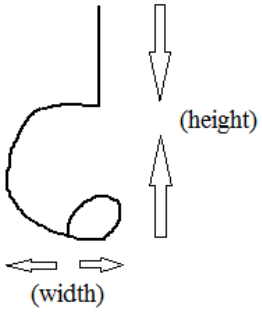


Figure 1.13: Character height and width

- **Weight of the character below and above the baseline**

This is the measure of the extent to which the character is extended above the baseline or below the baseline [18].

1.2.3.8 Feature Extraction methods

Major feature extraction techniques are discussed below:

- **Zoning**

Zoning is a global feature extraction technique and it is performed by dividing the image representing the character into P×Q zones [30]. Every zone gives a feature vector. Figure 1.14 shows an example of zoning.

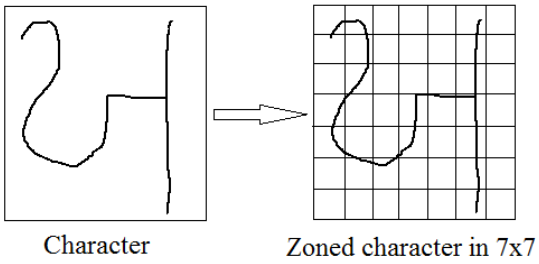


Figure 1.14: Zoning example

- **Density features:**

Pixels with higher density forms the feature vector and pixels with higher density are the ones that are darker than their neighbour pixels [39]. Figure 1.15 illustrates the concept.

- **Directional features:**

Feature sets are constructed using the directional information of the characters' image [32]. Figure 1.16 shows the directions of a pixel.

Directional information includes

- vertical and horizontal lines
- right and left diagonal lines
- length and number of various segments

$$\text{length of a segment} = \frac{\text{number of pixels in the segment}}{2 \times (\text{image height or width})} \quad \ldots (1.2)$$

- angles between various segments
- each segment's and each pixel's direction slope [32].

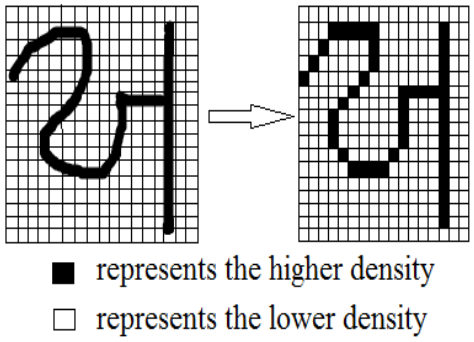


Figure 1.15: Density features

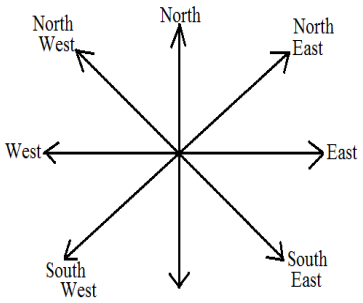


Figure 1.16: A pixel's directions

- **Profiles and projection analysis**

Profiles:

Profiles are the outlines of the characters from various possible directions [8].

- **Distance profile:**

  Distance profile represents the distance in the form of pixels of the boundary box of the image representing the character from any edge of the character [32]. It is measured in number of term of pixels. These distances measured from different edges form different profiles as shown in Figure 1.17.

  These profiles are analysed for every character and they form the feature set.

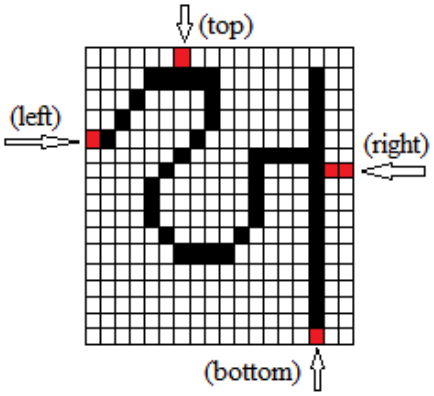


Figure 1.17: Distance profile

- ***n*-tuples:**

This is a profile in which we consider the dark or light pixels in every tuple of the character image from front and they are considered as a feature [43].

Projections:

Projection is a representation of the character. Projection of the image representing the character helps us understand the character's geometry, *i.e.*, its shape and size [12].

- **Horizontal projection**

It is the representation of the rows of the character's image. It is found by the summation of all the pixels along all the rows [11].

- **Vertical projection**

It is the representation of the columns of character's image. It is found by the summation of all the pixels along all the columns [11].

Projection is the summation of the horizontal and vertical projections [28].

An illustration is shown in Figure 1.18.

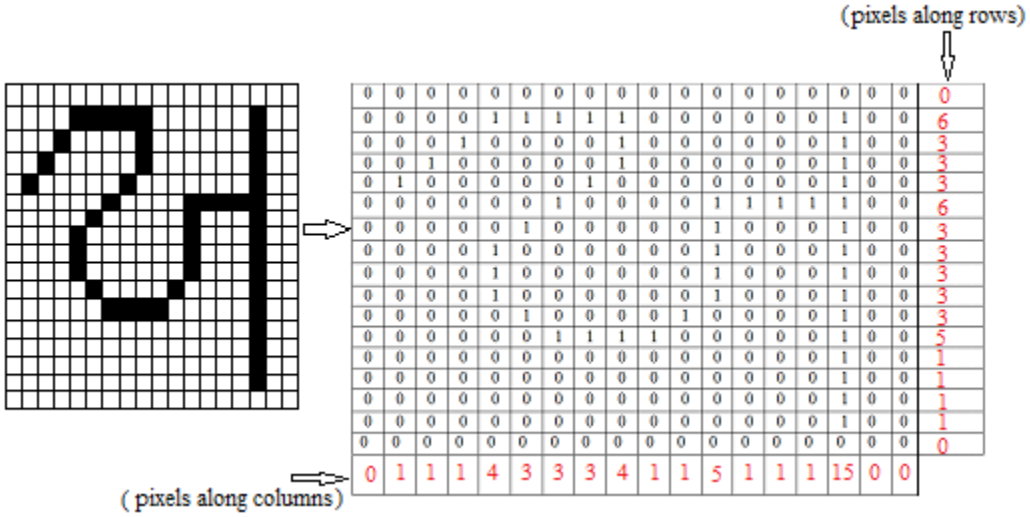


Figure 1.18: Projection example

The process shown above for finding the projections is called *celled projections* [16].

- **Texture analysis**

Texture analysis involves comparing the natural surfaces or texture of the images. Using this we can compare the homogeneity amongst the various feature sets [18].

Texture analysis involves comparing

- pixels with higher density
- how coarse the two objects are
- contrast of the two images
- entropy
- direction information of the images
- how homogeneous the image is
- comparing the lines-horizontal and vertical
- the regularity among the images
- how rough the two images are [6].

There are two types of texture analysis:

- **Spatial texture analysis**

  In spatial texture analysis, a function is used to remove irregularities in the image representing the character or numeral by replacing every pixel with a new value for the pixel using any mathematical function [38].

- **Spectral texture analysis**

  In this type of texture analysis power is analyzed at different portions of the image representing the character [18].

  - Low power in a portion of the character with small radius represents coarse textures.
  - Low power in a portion of the character with large radius represents smooth textures.

There are various feature extraction methods based on texture analysis like SIFT (Scale Invariant Feature Transform), F-SIFT (Feast-SIFT), PCA-SIFT (Principal Component Analysis-SIFT), SURF (Speed-Up Robust Feature detector), *etc.*

- **Histograms**

It is a plot of the values of the pixels of the image representing the character in the gray-scale format on one-axis and amount of dots at that value [18].

This plot's shape tells about the character's formation and thus helps in knowing about the character's shape that helps in recognizing the character well. The shape of the plot widely depends upon the image's contrast and brightness.

The probability for histogram is defined as:

$$P(h) = \frac{Number\ of\ pixels\ at\ a\ particular\ gray\ level}{Overall\ amount\ of\ pixels\ in\ character\ image} \quad \ldots (1.3)$$

The probability is for gray-scale image in first-order [18].

Features based on this probability are:

– **Mean**

Mean is nothing but the average of the pixels. It helps us in knowing how bright the image is.

- Image that is dark has a lesser value of mean.
- Image that is bright has a greater value of mean.

– **Standard deviation**

It is a value for divergence of the pixel values from the mean value calculated in the above step. It helps in knowing the contrast of the image.

- Image with low contrast has lower value for it.

– **Skew**

It helps in knowing how symmetrical the pixels are around their mean.

– **Energy**

It helps in knowing the distribution of the pixels. It also tells about the gray-levels in the image.

- An image with a consistent pixel value has 1 as its value.
- With increase in gray-scale stages its value decreases.

– **Entropy**

It tells us about the number of bits required to accommodate the overall pixel value for the image.

Types of histograms:

- **Intensity histogram**

It visualizes the contrast and tells us how bright the image is using the above mentioned measures [26].

- **Projection histograms**

It helps in projecting two dimensional images in one dimensional vector. Different projections can be formed by rotating the image with different angles [40]. Figure 1.19 shows an example.

Histograms find the overall amount of pixels horizontally and vertically and in each tuple and each column [32]. There are four types of projection histograms with four different directions for projections as mentioned below.

- Horizontal
- Vertical
- Left diagonal
- Right diagonal

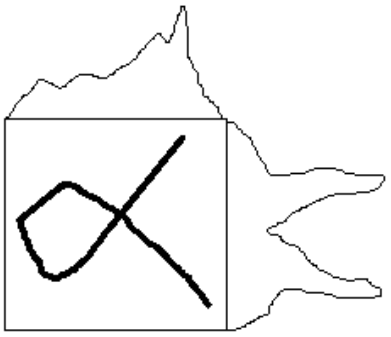


Figure 1.19: An example of projection

- **Profile histograms**

They tell us the amount of dots (in terms of number) between the first segment of the character and the boundary of the image representing the character [39].

Various profiles exist like

- horizontal
- vertical
- left diagonal
- right diagonal

based on the direction from which the number of pixels are calculated.

- **Radial histograms**

If the plot is taken from the radial profile, radial histograms are obtained. They give us the information about the distribution along the radius. Thus, helps in visualizing the radial profile [28].

- **Chain code histograms**

They are built for the character's snakes. Snakes will be discussed in detail later [4]. For now to build these histograms all the character's snakes must first be known.

According to the snakes the chain codes are built depending on the directions of connectivity of the character's snakes.

These histogram provide us with smooth character's image. Thus, recognition of the character becomes easier.

- **Color histograms**

They give the information about the distribution of various colored pixels in the character's image [6]. It tells about their color, their location in the image, *etc*.

They are of two types.

- **Global**

  It gives the information about colored pixels in general.

- **Local**

  It gives a more detailed information with every minute details that can help in knowing about every pixel's color information.

- **Horizontal and vertical histograms**

It tells about the distribution of pixels around horizontal and vertical direction [11]. It helps in knowing the extent of spread of the character in both the directions.

- **Searching for junctions**

To find junctions, the image is divided into *n* equal sub-portions. Open points and intersection points are found in every sub-portions [4]. Figure 1.20 shows an example.

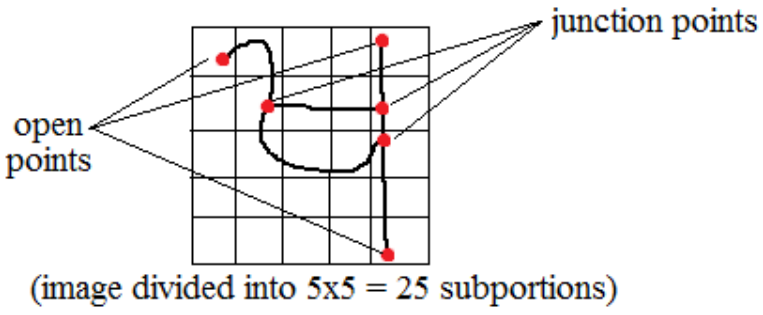


Figure 1.20: Searching for junctions

- **Using the shadow features**

Features computed from the shadows of the character in various directions are known as shading features [4]. For computing these features, the image representing the character is divided into eight regions by cutting it from middle in horizontal, vertical, left and right diagonal directions. Figure 1.21 illustrates shadow features.

All the perpendicular sides are considered as the base for the shadow of the edges of the character or numeral.

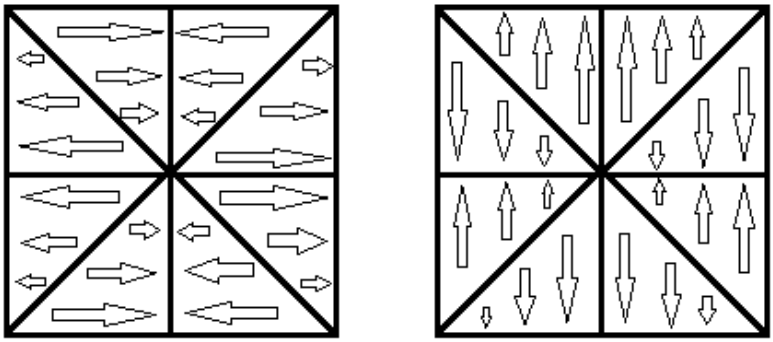


Figure 1.21: Shadow features

- **Using mathematical operators**

Using mathematical operators, certain mathematical functions and calculations are applied on the image's pixels and a new image is obtained with different pixel values.

They can be divided into two categories.

  - **Point operators**

    These are the ones that apply on every pixel one by one [26]. Few of them are discussed below.

      - **Multiplication operator**

        To increase the contrast, every pixel is multiplied by some value according to the magnitude of increase required.

- **Division operator**

To decrease the contrast, every pixel is divided by some value according to the magnitude of reduction required.

- **Saw-tooth operator**

It changes the pixel values using the operators mentioned above continuously. The domain of values is pre-decided.

- **Log operator**

It changes back the image to original image. Thus, removing the effect of any changes made by applying any operators.

- **Histogram normalization**

It is used to increase the range of values used for changing the brightness of the image.

- **Histogram equalization**

It converts the histogram according to the intensity level of human viewing it.

- **Thresholding**

It is used to select specific pixels out of the whole image based on the particular domain of values specified.

- **Group operators**

These operators work by using the nearby pixels to calculate new values [26]. Few of them are discussed below.

- **Template convolution operator**

A template is developed that contains different factors needed for the calculations to be done on the neighboring pixels.

- **Average operator**

It averages the entire nearby pixels to get new value for the pixel. Thus, for this operator the template developed in the above procedure has values as 'one' for multiplicative or division factor.

- **Gaussian average operator**

In this kind of averaging, the template contain the values for averaging using Gaussian mathematical function.

- **Median operator**

This takes the middle value of the template in order to perform calculations required. The process of building the template and choosing the median is described in Figure 1.22.

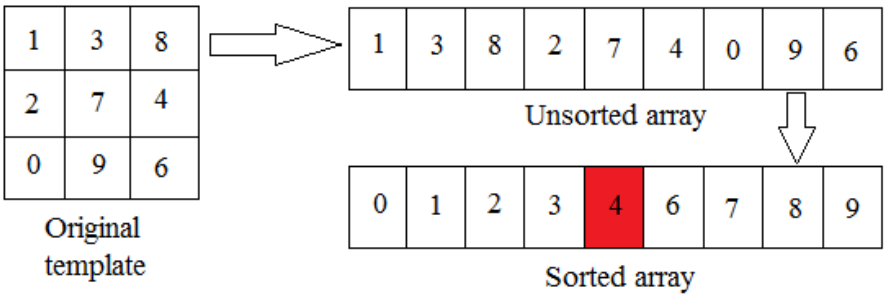


Figure 1.22: Median operator

- **Using mathematical morphology**

It uses morphological operators instead of normal mathematical operators to change values for the pixels of character's image [17]. These are the operators used for the set theory like union, difference, intersection, *etc*.

In this process a set P contains various pixels and set Q contains the measures required for operations. The set P undergoes operations and is divided into P1 and P2.

In the basic form either of the two sets P1 or P2 will be empty.

– empty P1 represents decrease and is called erosion.

- empty P2 represents increase and is called dilation.

- **Edge detection**

As the name suggests, it involves detecting the edges in the image of the character [26]. The image is divided into sub-images in order to detect the edges.

These are of two types.

- **First-order detection**

  It uses first-order differentiation of the images to detect the edges in it [18].

  Types of first-order edge detection techniques are:

  - Prewitt edge detection operator
  - Sobel edge detection operator
  - Canny edge detection operator

- **Second order edge detection operators**

  It involves using the second order differentiation techniques in the process of detection of edges [26].

  The operators used are:

  - Mexican hat operator
  - Laplace operator
  - Marr–Hildreth operator

  They help in building the template for the process of differentiation.

- **Phase congruency**

It is a special kind of feature extractor with following features.

- It can extract a vast variety of features.
- It can overcome or work efficiently with small changes.

Frequency distribution is used to collect features from the image [40].

- **Curvatures (corner) detection**

Curvatures are the transformations in the path of the segments forming the character. The point where these transformations occur is known as a corner.

There are following types of curvature detection techniques.

  - **Difference in the direction of segment**

    This is computed by calculating the change in the angle or slope of the segment [41].

  - **Intensity changes**

    It represents a curvature as brightness function. It measures changes in the brightness or the intensity of the character's image to recognize the corners or curvatures [31].

There are various types of detectors available for corner detections like FAST (Features from Accelerated Segment Test), Harris affine, Harris laplace, SUSAN (Smallest Univalue Segment Assimilating Nucleus), Harris detector, *etc*.

- **Blob detection**

It involves detecting the dense regions in the image of the character [18]. The character can be recognized by knowing where the pixels are present in vast amount, *i.e.,* the regions that are dense.

- **Region analysis**

Region is a particular portion of the image to be analyzed [41]. This can help to know the formation of the character. The image is analysed for various regions by partitioning it into sub images. To describe the shape and formation of a particular region, region descriptors are available [25].

There are various techniques available for this purpose like SIFT (Scale Invariant Feature Transform), Saliency, *etc*.

The various types of regions that can be detected are:

- Intensity based
- Maximally stable extremal regions [39]

- **Image motion analysis**

In order to define motion in images, a number of similar images with slightly changed segments are considered [26].

This concept can be used to recognize a character by considering the same character's various images as series of images for a motion. All the same character's images will also have a slight difference in their formation.

There are two approaches for this that are shown below:

- Area-based approach
- Differential approach

- **Binarization based feature extraction**

Binarization involves replacing all the colors in the image with two basic colors [8,] *i.e.*, black and white. Gray-scale image is the one that take gray and black as its basic colors. Figure 1.23 illustrates an example.

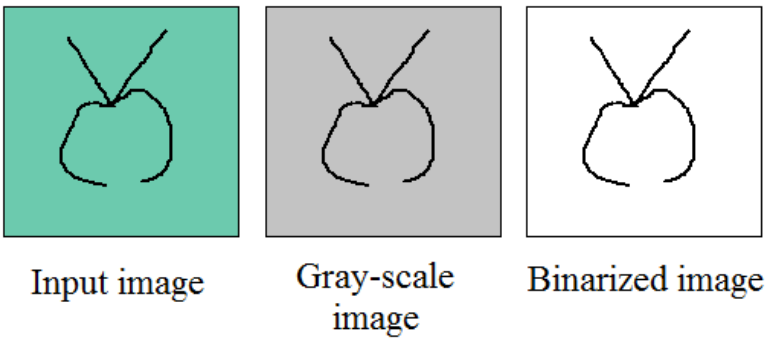


Figure 1.23: An example of binarization and gray-scale conversion

There are a variety of techniques available that can extract features from the binary images.

- **Convolutional feature extraction**

  It extracts features from the characters with pixels having nearby pixels correlated to a great extent [13].

Its motive is to get a minimal pattern that can recognize the character in an efficient way [48].

- **Feature extraction by shape analysis**

This involves analyzing the shape of the character and comparing it with the database of character's shapes [38].

Comparing two character's shapes involves,

– converting image into binary form

It involves changing the image into two basic colors, *i.e.,* black and white.

– shape skeletonization of the image

Shape skeletonization involves removing the filled portion in the character's image in order to generate the skeleton of the character [13]. This eases the comparison of the two characters, *i.e.*, two shapes. Figure 1.24 illustrates an example.

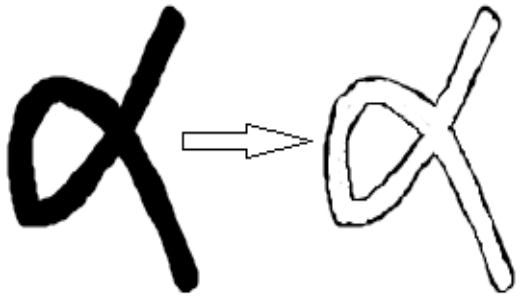


Figure 1.24: Shape skeletonization example

Shape features that can be extracted are:

- Shape transform domain
- Shape-index
- Polygonal features
- Scalable vector templates
- Spatial interrelation features
- Scale-space

Methods for shape analysis include following:

- **Thresholding and subtraction**

  Subtraction operator is used in order to change the intensity of the character's image to the level of human's vision [26].

- **Template matching**

  It matches the image representing the character with a template having information about the formation of the character. It represents the shape of the character in a smaller sized template [40].

  – **Deformable templates**

  These templates cannot be changed. Thus, removing the chances of templates getting modified accidently.

- **Using mathematical transformations**

These are applied on images to obtain information from it that will not be available from the image itself [20]. The image that has to be transformed is considered to be in time domain. For understanding the character's formation, the image representing it has to be converted into frequency-domain. This helps us knowing the frequency spectrum of the image. This is equivalent to finding the frequencies available in the character's image.

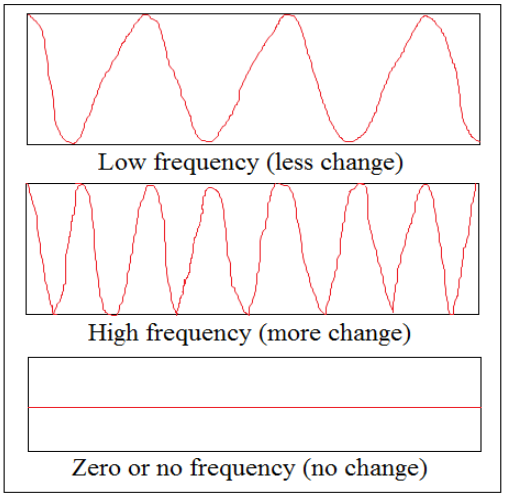


Figure 1.25: An illustration of low, high and zero frequency

Finding the frequency components contained in the image is equivalent to finding the amount of variations in it and the locations where these variations occur. No variation represents zero frequency. Similarly less variations correspond to less frequency and more variations compared to high frequency as shown in Figure 1.25.

There are vast variety of transformations available. Some of them are discussed below.

- **Fourier transform**

This transformation finds the frequencies contained in the character's image. It converts the time domain representation of the character to a amplitude versus frequency graph like shown in Figure 1.26 [18].

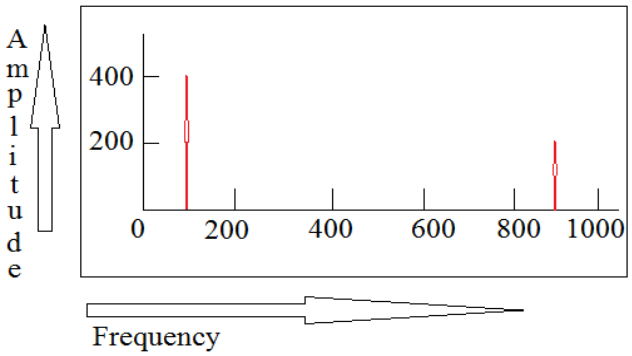


Figure 1.26: Amplitude versus frequency plot

It expresses that the frequencies contained in the image are 100 and 900 with amplitudes 400 and 200 respectively.

- **Wavelet transform**

With fourier transformations only frequency components available can be viewed along with their amplitude [27]. There can be two cases.

- either all the frequencies are available at every time
- or different frequencies are available at different times

Thus, that transform does not give any information about the time.

Fourier transform give us the information about frequencies present and the time at which they are present. Thus, a frequency versus time graph is obtained.

There is a vast variety of wavelets that can be applied on the image to decompose it into frequency versus time plot [38].

- **Discrete cosine transform [20]**

It also gives the frequency information contained in the image representing the character [20].

- **Discrete Hartley transform**

This transformation is similar to the fourier transform. But it involves easier calculations [26].

- **Unitary image transforms**

This transformation tells about the shape information of the character represented by the image [40].

The main purpose is to lessen the feature set but not losing any information contained in the image about the character's shape.

- **Hough transform**

This transformation finds out the specific shapes in the image. The shape is the input to the transformation [2]. The available transformations are:

  - lines
  - ellipses
  - circles

- **Distance transforms**

This is used to recognize the formation of the character. They are used to find the shape of the character by skeletonization [26]. Then, the distances of the skeleton of the character are measured from every profile in order to recognize the character.

- **Flexible feature extraction**

It involves detecting contours. It is the change in the pixels from being white to being black. Thus, it only applies on the binary image.

Contour array:

All the pixels are analyzed one by one and if a transition in pixel color is encountered it is stored in an array [11].

Contour profiles:

Instead of analyzing pixel by pixel, profiles of the characters are checked from different angles to detect contours [40].

- **Moments**

Moments give the information that describe the formation of the character in a general way, more like global feature extraction [16]. It gives the description about the shape of the character. Based on the way they describe the shape they are divided into following types [33].

  - Geometric moments
  - Invariant moments
  - Zernike moments
  - Pseudo Zenrike moments
  - Global transformation moments

- **Segmentation features**

These are the features extracted by dividing the image into various sub images [17]. A number of times till the sub image can be categorized in either of the following ways.

– dark

– bright

- **Graph matching**

The character's image is considered as a graph and finding out the connected components in it corresponds to graph matching [12].

Graph descriptors

The information obtained from the graph is called graph descriptor [1].

Connected components

These are the components, *i.e.,* segments that are inter-related to each other. This involves finding whether the segments have path to reach each other [9]. If the two segments can be reached from one another they are connected else not.

- **Content based feature extraction 0or color features**

Color features give information about the colors in the image [41]. Every pixel is scanned to know its color information.

Techniques to obtain color information are:

Average RGB

This involves averaging all the three components of the image, *i.e.,* red, blue and green components [6].

Color moments

These are used to find the distribution of the pixels of the character. Types of moments are discussed below.

- Mean
- Variance
- Skewness

Color features are of following types:

- **Primitive features** like RGB, average color, *etc.*

- **Semantic features** like texture, color outline, *etc.*

- **Discriminant feature extraction**

It involves building a matrix called discriminant matrix [21]. This is done using a discriminant function that is defined initially.

- **Other important statistical techniques**

There are many other important statistical techniques like Isomap, Dimensionality reduction techniques, SC (Sparse Coding), Linear Semantic Analysis (LSA), MDR (Multifactor Dimensionality Reduction), Regression analysis, Partial least squares (PLS), Centroid based distance metric feature extraction, methods using parameters of polynomials, *etc*.

All the approaches discussed above have been widely used by many scholars and researchers in the recognition of the Punjabi Character Recognition.

1.2.4 Classifying the character into a specific class

The feature set or vector developed in the feature extraction step is used to classify the character written by the user in a particular class, *i.e.*, finding what can be the class of the character written.

In order to classify the character properly into the right class, the recognizer is first trained with a number of such feature vectors for every character so that the classifier is able to put the character fed as input to it in the right class using the vectors by which the classifier is trained. Every character represents a different class.

For this purpose there are a number of classifiers available like support vector classifiers, k-nearest neighbors, k-means, c4.5, page rank, *etc*.

In the future implementation, SVM Classifier has been used.

1.2.4.1 SVM classifier

Support vector machine is widely used recognizer in this field.

SVM classifies a feature vector into particular class by obtaining a hyper plane that can differentiate between two feature vectors representing two different characters or numerals properly by putting them into their classes respectively.

Similarly, a multi class classification is done to differentiate between more than two classes.

A good illustration on SVM is given by Thorsten [37].

Chapter 2

Review of the Literature

Various scholars and researchers have proposed different works that use different feature extraction techniques. Few of them are discussed below.

- Aouadi *et al.* [3] proposed a system for Farsi characters based on segmentation methods. It used matching the shape for recognition purposes. Template matching is done to get the best matching template for the components that are connected. This system also worked for the components that are connected at number of places. Segmentation methods and shape matching was used to compare the character's template with templates stored by finding the similarity between the components and to get features. Distances given by Manhattan and Euclidean formulae were used to compare. Manhattan formula gave the maximum accuracy of 95.7% for touching measure of the segmentation and for Euclidean formula also touching measure for segmentation gave a maximum accuracy of 94.5%. It was shown that the proposed system was robust.

- Arora *et al.* [4] combined various features capturing methods all together in order to recognize Devnagari handwritten character. The experiment was performed on 4900 images. One of the feature extracted was junction points. For this the image representing the character was segmented into 16 regions of size 25 × 25 and for all these regions end points that were open and junctions were used. The next method used was chain code histogram. For this the image was first converted into binary image. Contours were taken in the feature set that were captured by taking every 3 × 3 pixels nearby every object point. Shadow features were also captured. For that image was segmented into 8 portions cutting the image from middle horizontally, vertically and diagonally from right and left directions. Multi layered Perceptron was used as a recognizer. Majority voting was used to combine the results obtained from every method. Overall accuracy achieved was 92.80%.

- Bouda *et al.* [5] gave a new approach for the implementation of detection of corners that used electric field that was not real, *i.e.*, that was virtual. Using this approach, images were processed and it is even applicable in the vision functions for computers. An image is considered to be a virtual lattice with electrical charges. The process contained two steps. Firstly, gradients were calculated directionally that allowed detection of edges by structuring an operator of 3 × 3 size (in form of pixels). Secondly, the obtained operator was utilized to find the corners by finding the derivatives of the images. Gaussian function was used to get images without any noise. After this, calculation of the output function given by Harris and Stephens is done for every pixel. This was done using the automatic correlation matrix. In the end local maximum is calculated. The performance was good enough and proposed algorithm was very simple to implement. The performance was satisfactorily better than the Harris detection method.

- Chadha *et al.* [6] analysed and optimized the approaches to obtain the features for image retrieval based on the content. The user types in or selects the descriptor that can describe the type of image required. Various features are extracted from it like Co-occurrence, Average RGB, Color Moments, Geometric Moment, Local and Global color histograms. If taken alone none of these give good results. Therefore, all the above mentioned approaches were together considered and were used as one approach. A new idea of image cropping was used. It was found that the results became better by the use of cropping.

- Cheung *et al.* [44] gives a system that uses texture analysis to recognize the bark. Four methods were analysed namely Histogram technique, Run Length approach, Auto Correlation technique and Co occurrence technique to get the feature vectors from the barks by analyzing their textures. Specifically three recognizers were used for the training and prediction of the feature vectors obtained by above mentioned techniques. They were Moving Median Centers, Nearest Neighbor and KNN. In order to improve the efficiency the color related information was also collected and analysed and it actually gave better results. After the whole analysis it was found that out of the four above mentioned

methods used to obtain the feature vectors, COMM gave the best results, *i.e.*, the maximum accuracy.

- Choudhary *et al.* [8] proposed an off line system for recognition of handwritten characters. Features were obtained by binarization based techniques. This system was proposed for the classification of English characters. The preprocessing involved normalization, smoothening and thinning. Images were processed in two ways. They were converted in gray-scale format and the binary format too. Every image representing any of the 26 characters of English grammar were processed to give a feature vector of sixe 180 × 1. Multi layered feed forward ANN was used for the classification purposes. At the entry layer 180 neurons were used and at the exit layer 26. To get the best results 80 neurons were present in the hidden layer. For every character 50 vectors developed from 50 images were used for training purposes. On average 85.62% of accuracy was achieved.

- Choudhary *et al.* [9] proposed a different approach for character extraction. Characters were taken out from Images using the approach presented. The words contained in the images were divided into portions using segmentation based approach. Features were found from the various connected components that represent the segmented characters. This method was applied to the variable sized text represented by images. The texts collected were of different shape, size, height, width and fonts. It was seen that this method gave quite a good accuracy and thus, this approach can be widely used but it has an issue with disconnected components.

- Dabbaghchian *et al.* [10] proposed a system to recognize faces using DCT to obtain the features. Few statistical calculations were done to get the coefficients as feature vectors by applying the Discrete Cosine Transform on the face image captured. It is a new approach and differs from the zonal and zigzag masking. It considers the coefficients that can have better recognition powers, *i.e.*, that can easily differentiate between various classes. For this purpose quite a few approaches were tested to find the best results. Euclidian distance was used as the classifier. The experiments were done on the famous

databases of face data. The data was randomly selected for training and prediction. The proposed system gives the good performance in the field of recognition of faces.

- Gatos *et al.* [13] gave a novel approach for extraction of features and recognition of the characters. Convolutional approach was used for obtaining the features. Natural as well as artificial sets were used for training and prediction. 1500 characters of Greek language were used to collect the feature vectors of images for natural sets. Images that captured 10 different fonts of 32 characters were used to obtain the artificial sets. ANN was used for classification purposes with 5 different variations and the best result obtained was the accuracy of 99.87%.

- Ghazali *et al.* [14] proposed a system that classified the image given to it as input into one of the pre decided classes using the Discrete wavelet Transform. Classification of images involved reducing the image to the lowest possible matrix or array size for its representation. Low level features were obtained with no pre processing. To get the high level features, Discrete Wavelet Transform was used in two dimensional. The size of the image produced after decomposition using wavelets was same as the input image before the decomposition process. Therefore, to get the coefficients as feature vectors further processing was required. The extracted feature vector was utilized to classify the image into broad and narrow weed. In the end it was concluded that the approach tested to obtain features gave good performance accuracy and classified broad and narrow weed easily and effectively.

- Hao [15] proposed an approach to obtain feature vectors from the images based on their color information, *i.e.*, content based retrieval. He used a new approach namely MPEG-7 with its corresponding framework MPEG-21 to capture the color information contained in the features in the feature vectors. After obtaining the features in the feature vector, it was compared with the images present in the database. Principal Component Analysis, Moment Invariant technique and Wavelet Decomposition were used to capture the features as feature vectors from the image. Then the correlated images were found using those feature vectors. Different accuracies were achieved on changing the weight for the

extraction coefficients and changing the threshold level. The results obtained finally proved that a good performance was achieved by the proposed system.

- Hossain *et al.* [16] gave a rapid method for obtaining the features for OCR. The technique is titled Celled Projections. It segmented the image capturing the character into various regions and the projection of all the regions developed was taken. In this system various methods for obtaining the features were tested and compared with the above mentioned technique to find the best performance. Other techniques used were moments, zoning, checking the crossings, fourier transform and the projection histograms. Methods used for classification were *k* Nearest Neighbor, FBPN and Probabilistic Neural Network. It was shown that the maximum accuracy achieved was for four horizontal and four vertical celled projections using PNN classifier with spread factor between one and two. The accuracy achieved was of 94.12%.

- Koerich *et al.* [17] extracted low level and high level features and fused them to classify the handwritten words. Word was captured and given as input in the form of an image. To capture high level feature vector words having loosely segmented images were used. For the recognition purposes HMM was used. The character hypotheses were used to obtain the low level features. SNN was used to recognize the characters from these vectors for features. 84,760 characters were used for training purposes and for prediction around 36,170 words were used. The performance for the combination of both level features was found to be around 71%.

- Lahmiri *et al.* [19] proposed a new approach to capture features in the feature vectors using Hybrid DWT. It automatically obtained the vectors from the mammograms. This approach uses Gabor Filter along with DWT. It has a good scope in the screening system of cancer. Two dimensional DWT was used to get coefficients as features in the feature vector from the mammogram. After that Gabor was used to calculate the standard deviation and average from the image obtained.

- Lawgali *et al.* [20] proposed a recognition system for Arabic Handwritten Character. In it Discrete Wavelet Transform was used to get the features for Arabic characters. These

features were compared with the features extracted by Discrete Cosine Transform to find out which of them gave the better results. For analysis, 5600 characters were checked in three sizes namely 64 × 64, 128 × 128 and 256 × 256 that included almost all the ways to write Arabic characters. Artificial Neural Networks were used for the recognition purposes. It achieved maximum accuracy of 79.87% with Discrete Cosine Transform for 64 × 64 sized images.

- Liu *et al.* [22] proposed a system to detect boundaries by using Scale Invariant Feature Transform. SIFT finds out the break points in the videos or visuals. After that a new proposed method namely LDT-SBD (Local Double Threshold-Shot Boundary Detection). After that keyframes were obtained using two different methods. The proposed approach gave the acceptable results for obtaining the keyframe and for the detection of the bounadaries.

- Liu *et al.* [23] proposed a system based on histograms to obtain the features. The approach is novel and is used for recognition of faces. LBP histograms were utilized to get the feature vectors. The image was segmented into rectangular portions that were non overlapping. Histograms were developed for every portion. The differences were computed between the histograms to recognize the facial features. The proposed system was used to check the JAFFE database to get the effectiveness and performance of the system proposed. The system gave the maximum accuracy of 90%.

- Markov *et al.* [24] gave a method for high level extraction of features using a self taught method for classification. The method gave overcomes the problem of taking labeled and unlabeled samples with same distribution only. The data is represented by dividing it into sub matrices, *i.e.*, activation and bases matrices. For this purpose Principal Component Analysis, Sparse Coding and Non negative Matrix Factorization were used and their performances were compared. The vectors obtained from any of these methods were the input to self-taught learning technique that was used for the recognition. The Sparse Coding gave the best results as compared with other two approaches. It was concluded that with self-taught technique, the best result was obtained when the data that is labeled

is less in quantity. Consistent increase in accuracy was observed for different sizes of data.

- Melo *et al.* [25] proposed a system to guess the protein structure. For this purpose Principal Component Analysis matrix calculation was used. PCA gave the feature vectors to compare the structure with already stored structure using the training and prediction of Neural Networks. Various components obtained from the protein taken as input were considered as input for the three different Neural Networks (artificial). The three networks had 30, 35 and 40 points in the hidden layer. The recognizers were trained using RPROP approach. With this system even if the components are reduced to a vast amount (from 260 to 80), performance was atleast 1% more than the best possible structure prediction algorithm.

- Ruohong *et al.* [29] discussed a system that obtained feature vectors using the Non negative Matrix Factorization. SAR images were the input and they were processed and using NMF, feature vectors were obtained out of them. For data MASTAR database was used. SVM was used as the classifier. Using Bayesian fusion approach, the results obtained were fused. The results concluded that if NMF was used, recognition was significantly correct.

- Saha *et al.* [30] gave a system to classify a handwritten character with a new approach namely forty points to obtain the feature vector. The characters worked upon were the English alphabets. This method was used to recognize the handwritten names. 1040 images were developed of different English alphabets of different font, shape and size. The feature vectors obtained from these images using the above mentioned approach, were used for training the Artificial Neural Network that was used for recognition purposes.

- Siddiqi *et al.* [31] suggested a new approach to recognize the handwriting by using the chain code based features that were obtained by developing chain code histograms from the images that captured the character or words written by the writer. In this feature vectors were obtained from the curvatures of the images containing the written

characters. The contours or curvatures were checked from various profiles, directions and angles to get different views of the character. Global features were obtained from the chain code histograms. Local features were captured by segmentation of the image into smaller regions. For every region the eight profiles were checked to get the different views of the curvatures from eight different directions. To compare two handwritings the distance measures are used between the images. The proposed method was tried on 650 and 225 writers' handwriting respectively in two sets. The performance of the overall system was satisfactory.

- Singh *et al.* [33] compared various methods for obtaining the features. They tested various scripts to check the performance of various approaches for the extraction of features to recognize the digits and characters of those scripts. The results using zoning, histograms, chessboard, gradient, chain codes, profiles, distances, chamfer and crossings were compared. It was found the maximum accuracies were 84.45% for Chinese characters using Statistical Discriminant Analysis, 94.3% for hand printed characters using template matching, 98.37% using SVC and 84.37% using Ensemble system for English characters, 87.32% using NLP for Thai characters, 92% using KNN and 91.3% using SVC for Persian digits and 94.3% using NLP and SVC for Devanagari characters.

- Sun *et al.* [35] proposed a system to obtain features in the form of shape characteristics from the images of the fruits. This was done using the chain code algorithm. It extracted or detected the contour or curvature of the fruits represented in the image. Area of the fruit inside the image, its height, width, circumference, complexity in the graph, degree of circular portion, parameters of the graph, rate of being concave and radius of the inscribed circle were considered as the shape features in this technique. The results obtained were enough to describe the effectiveness of the method. It was effectively shown that the method was very simple to implement and gave almost accurate results. The results were obtained fast by using this technique.

- Supriana *et al.* [36] proposed a system for the recognition of Farsi characters. During pre-processing, binarization was performed and skewness was processed by two approaches, *i.e.*, by moments and by triangle. Hilditch algorithm using templates was used for

thinning. Features extracted were loops, distribution of points, perimeter, elongation, energy and compactness ratio. C4.5 classifier was used for the recognition purposes. The result concluded was that algorithm triangle resulted with better skewness. It gave maximum accuracy of 99.9% for segmentation and 82% for classification. The overall accuracy achieved was unfortunately 48.3%.

- Vamvakas *et al.* [42] gave a novel approach for obtaining the features and for classification of the historical papers. It is an offline approach for character recognition that can handle the characters in any shape, font and size. To check the robustness of the system a number of different historical papers were collected. To obtain the feature vector the image was segmented into various regions recursively. The center of mass was calculated for every region. Each region is divided recursively till it reaches at the pixel level. Hierarchical recognition approach was used for classification. Granularity was used to implement it. This system gave maximum accuracy of 80.19% that was higher than other modern techniques like Yamada, Gader, *etc*.

- Yang *et al.* [47] discussed an approach to detect contours and obtain feature vectors based on them for surface or shape matching. Features were obtained from the point clouds using this approach. They are nothing but the surfaces with high complexity. They contained surfaces with branch points, blends and contours that are open. The features to match the shapes were angle of rotation and the length of the arc. The analysis in the end showed that the method proposed gave optimal results in segmentation of the surfaces with huge complexity.

- Zahedi *et al.* [48] proposed a recognition system for Farsi characters. It recognized the Arabic characters in documents that were not in cursive format. Scale Invariant Feature Transform (SIFT) was used to extract features for the characters. The features obtained were neutral to variations in height and width of character, shape of character and direction of writing. Pre-processing was not required for this system apart from smoothening that was done only for the poor quality characters entered. In total 1400

characters were captured as images. The recognition was almost correct every time giving an accuracy of approximately 100%.

After having a review of all these methods for extraction of features, Discrete Wavelet Transform was decided for extraction of features. The next Chapters discuss all the four steps for recognition of character in detail using Discrete Wavelet Transform in the third step and using SVM as the classifier for recognition of the Punjabi characters or numerals.

Chapter 3

Pre-processing and Wavelet Based Feature Extraction

Punjabi characters and Punjabi numerals were taken as an input from various random users. These inputs were captured using digital pen and pads. The strokes (the part of character written without lifting the pen up) made by a user while writing were captured as series of *x* and *y* co-ordinates. Those co-ordinates were used to draw an image of the character or numeral written by the user on digital pad. All the co-ordinates of a character or numeral were stored together in a text file along with its ID (used for recognition purposes discussed later in Chapter 4) and all the characters and numerals written by a single user were kept in the same file. The example of sample file is shown in Figure 3.1. Algorithm 3.1 uses Figure 3.1 to draw images of characters an numerals.

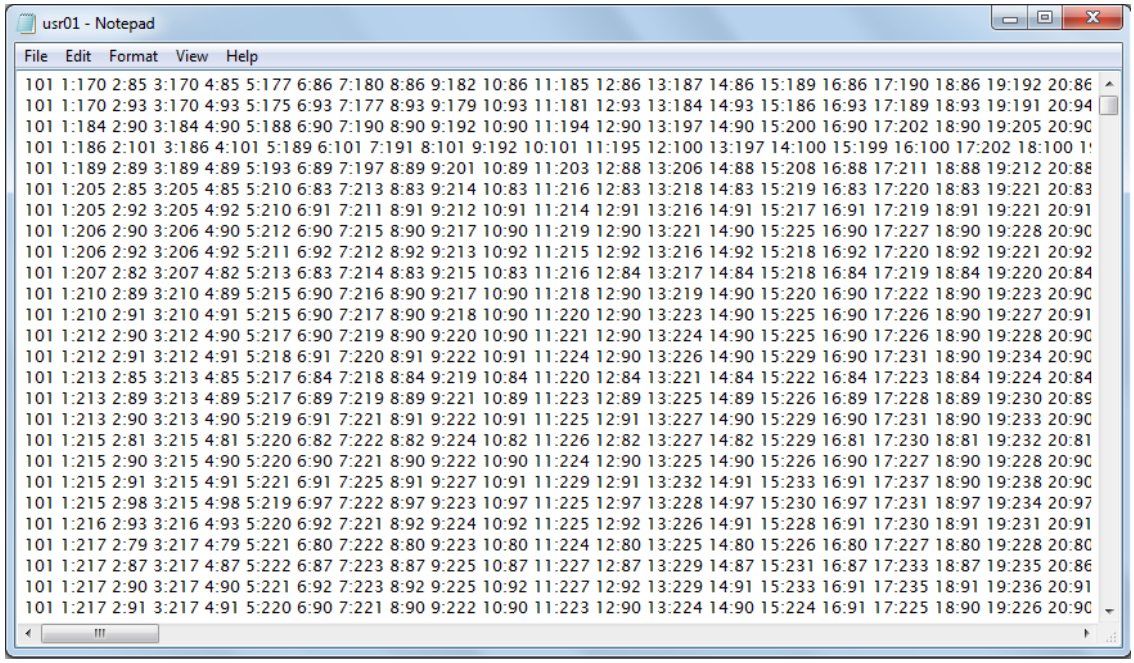


Figure 3.1: A sample file used to draw images of characters and numerals

Algorithm 3.1: Drawing images of characters and numerals from *x* and *y* co-ordinates captured in text file

1. Save the ID encountered in a file.
2. Make a 1000 × 1000 blank white image.

3. Save *x* and *y* co-ordinates in *X* and *Y* files.
4. Set (*x*, *y*) = 1 of the blank image developed where *x* and *y* are the values stored in *X* and *Y* files correspondingly and 1 represents black color.
5. Repeat 3 and 4 until a line feed is encountered.
6. Save the image using the name saved in step 1.
7. Repeat step 1 to 6 until EOF is encountered.

The images developed by the above procedure were all of 1000 × 1000 sizes and they all represented a character or numeral in some portion of it.

Figure 3.2 represents sample of 1000 × 1000 images developed by the above algorithm.

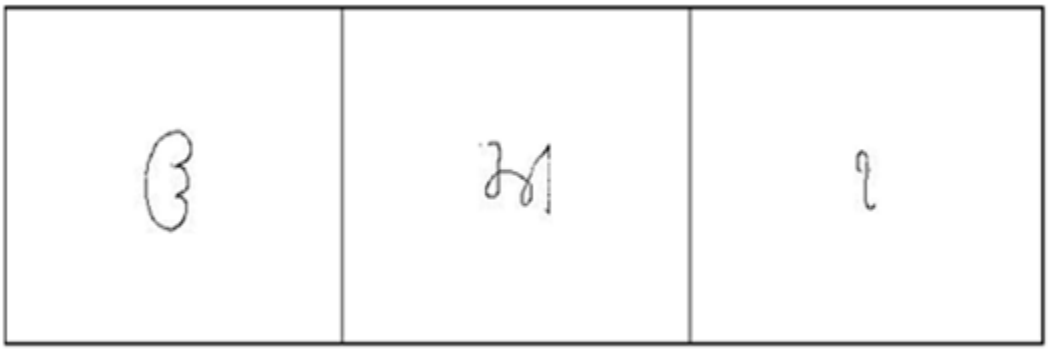


Figure 3.2 Sample of 1000 × 1000 images developed : [a] represents Urah (Punjabi Character), [b] represents Aarah (Punjabi Character), [c] represents ikk (Punjabi Numeral)

3.1 Pre-processing

The images developed by the above procedure represented the character or numeral written by the users. As everyone's writing style, font and size is different, it would have been very difficult to recognize the character or numeral if we were to extract features directly from these images. Therefore, all the images were normalized to a specific size in order to ease the recognition process.

3.1.1 Windowing

Before normalization images were windowed to the sizes they were originally written. Windowing involved removing the extra white region from 1000 × 1000 image developed by

Algorithm 3.1 as that extra information contained in those images was not required to recognize the character or numeral captured in that image. After windowing, the images developed contained only the character with its original shape and size without any extra information. Algorithm 3.2 illustrates wndowing.

Algorithm 3.2: Windowing of characters and numerals captured as 1000 × 1000 images

1. Set *xmin* = 1000, *xmax* = 0, *ymin* = 1000 and *ymax* = 0.
2. Read image's pixels one by one into a variable.
3. For every pixel (*x*, *y*) check if pixel value is <= 200
    - If *xmin* > *x*,

      Set *xmin* = *x*

      End if
    - If *xmax* < *x*,

      Set *xmax* = *x*

      End if
    - If *ymin* > *y*,

      Set *ymin* = *y*

      End if
    - If *ymax* < *y*,

      Set *ymax* = *y*

      End if

    End if
4. Make a new blank white image of size (*xmax* – *xmin* + 3, *ymax* – *ymin* + 3)
5. For every pixel (*i*, *j*) of the original image,
    - Set *x* = *xmin* + *i* where *i* moves from 0 to *xmax* – *xmin* + 1
    - Set *y*=ymin + j where *j* moves from 0 to *ymax* - *ymin* + 1

      where *i* and *j* represents the i^{th} row and j^{th} column of the original image that together represents the $(i, j)^{th}$ pixel of the original image.
    - If 1 <= *x* <= 1000 and 1 <=*y* <= 1000

Set $(i + 2, j + 2)^{th}$ pixel of the new image developed in step 4 to $(x, y)^{th}$ pixel's value of the original image.

Else

Set it to 0 where 0 represents white color.

End if

The image developed after applying this algorithm was the windowed image having size exactly same as the character or numeral written by the user. All the images developed by applying the above algorithm were of different sizes and shapes depending upon the way writer wrote the character or numeral.

Figure 3.3 represents the windowed images developed from the images represented in Figure 3.2. [a], [b], [c] in Figure 3.3 represents the corresponding windowed images of the images represented in Figure 3.2.

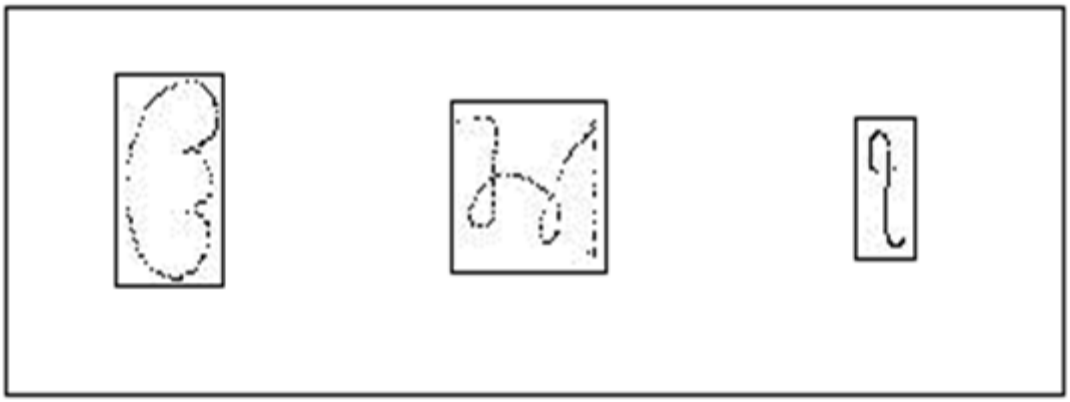


Figure 3.3 Windowed images from images in Figure 3.2: [a] Windowed Urah, [b] Windowed Aarah, [c] Windowed ikk

3.1.2 Normalization

All the windowed images developed by the above mentioned algorithm were then normalized to a specific size in order to maintain a standard for the recognition purposes. All the images to a particular size were resized to ease the recognition process.

These resized/normalized images were then used for the feature extraction purposes as recognition process was done using the feature set developed from these normalized images

during the feature extraction process. For every single image, feature set was developed from three different normalized images developed from that image.

Images were normalized to three different sizes shown in Figure 3.4.

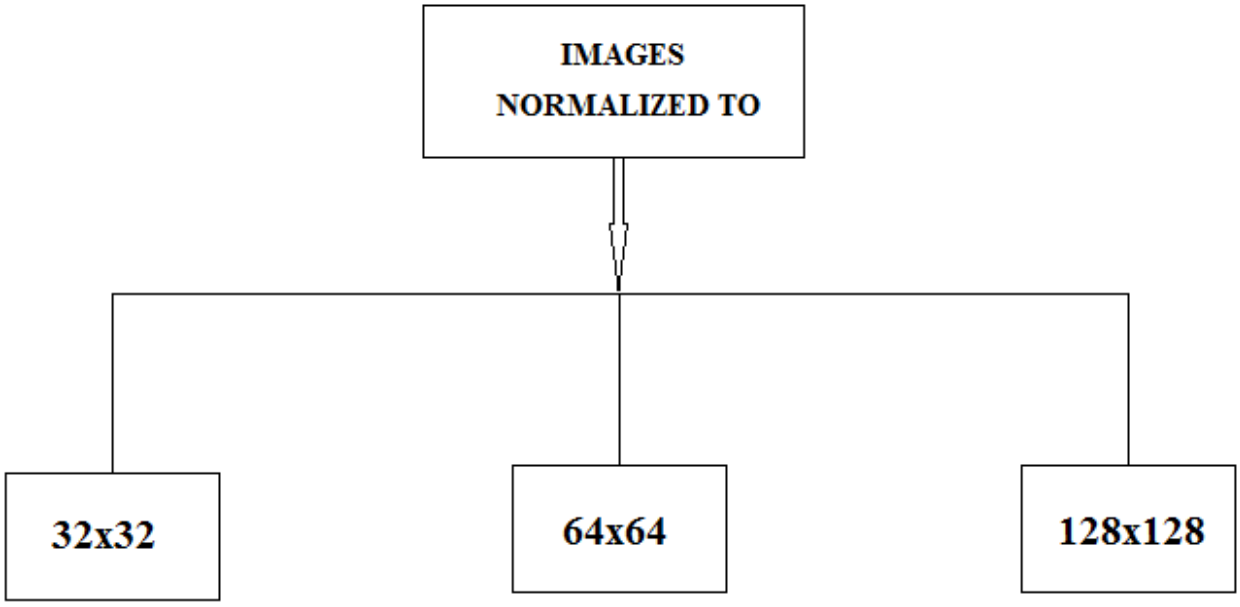


Figure 3.4: Normalization sizes tried for feature extraction

Algorithm 3.3 illustrates developing normalized images from windowed images.

Algorithm 3.3: Resizing the windowed images to size *j* × *k*

1. Save all the IDs to be resized in an array *A*.
2. Loop '*p*' from 1 to length of the array *A*.
    - Loop '*i*' from 1 to 100 where 100 is the number of images per ID
        - Set *id*=*A*(*p*)
        - Set *a*=int2str(*id*)
        - Set *b*='win'
        - Set *c*=int2str(*i*)
        - Set *d*='.jpg'
        - Set *abc*=[*a b c d*] where *abc* represents one of the windowed images.

- Set *X*=imread(*abc*)
- Set *Y*=imresize(*X*,[*j k*])
- Set *a2* to the folder where to store the resized images.
- Set *b*='_'
- Set *d*='.jpg'
- Set *abc*=[*a2 a b c d*]
- Set imwrite(*Y*,*abc*)

- End loop

3. End loop

The above algorithm was applied on every windowed image to get for every image three resized images as shown in Figure 3.5, 3.6 and 3.7.

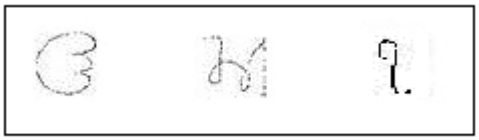


Figure 3.5 Resized images to 32 × 32: [a] Resized Urah, [b] Resized Aarah, [c] Resized ikk

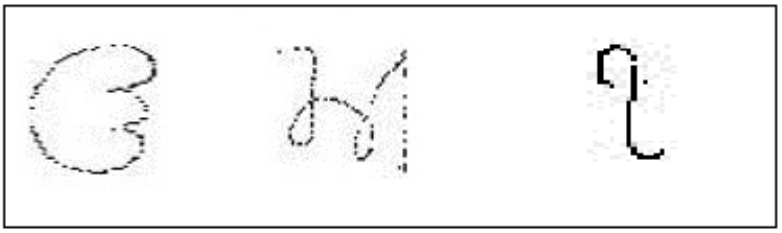


Figure 3.6 Resized images to 64 × 64: [a] Resized Urah, [b] Resized Aarah, [c] Resized ikk

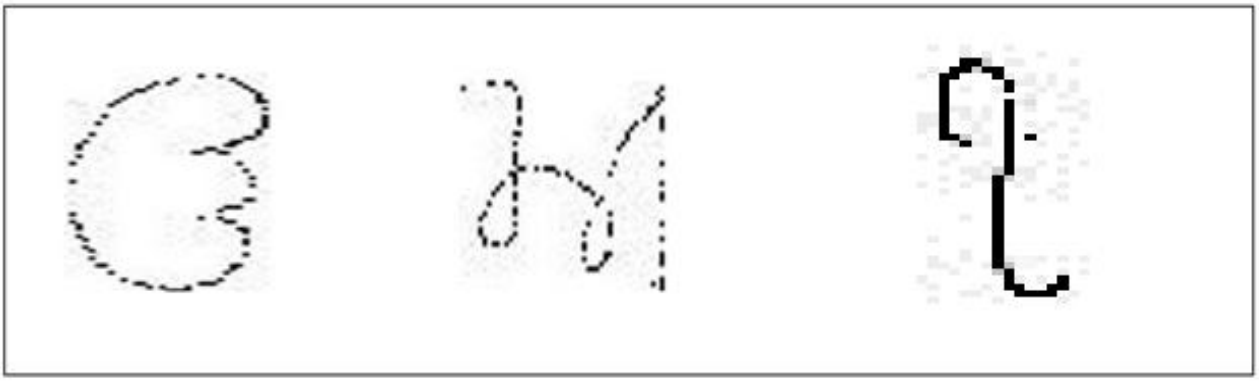


Figure 3.7 Resized images to 128 × 128: [a] Resized Urah, [b] Resized Aarah, [c] Resized ikk

Figure 3.5 [a], [b] and [c] represents the resized images to 32 × 32 of [a], [b] and [c] windowed images represented in Figure 3.3 correspondingly. Similarly, Figure 3.6 [a], [b], [c] and Figure 3.7 [a], [b], [c] represents the resized images to 64 × 64 and 128 × 128 of [a], [b] and [c] windowed images represented in Figure 3.3 correspondingly. These resized images were then used to get the 3 different feature sets for every character or numeral written by the user13s that was captured as an image.

3.2 Feature Extraction

Feature Extraction forms the building base for the recognition process as every feature set developed from the resized images during the feature extraction process is fed to the Classifier to recognize the character or numeral. In the present work, from every image three resized images were constructed. On every resized image wavelet transformation was applied to get a feature set that contained wavelet coefficients as the features.

The wavelet transformation performed on every image was done using 54 different wavelets which gave 54 different feature sets from a single resized image. Similarly, 54 different wavelet transformations were done on the other two resized images obtained from a single windowed image using 54 different wavelets.

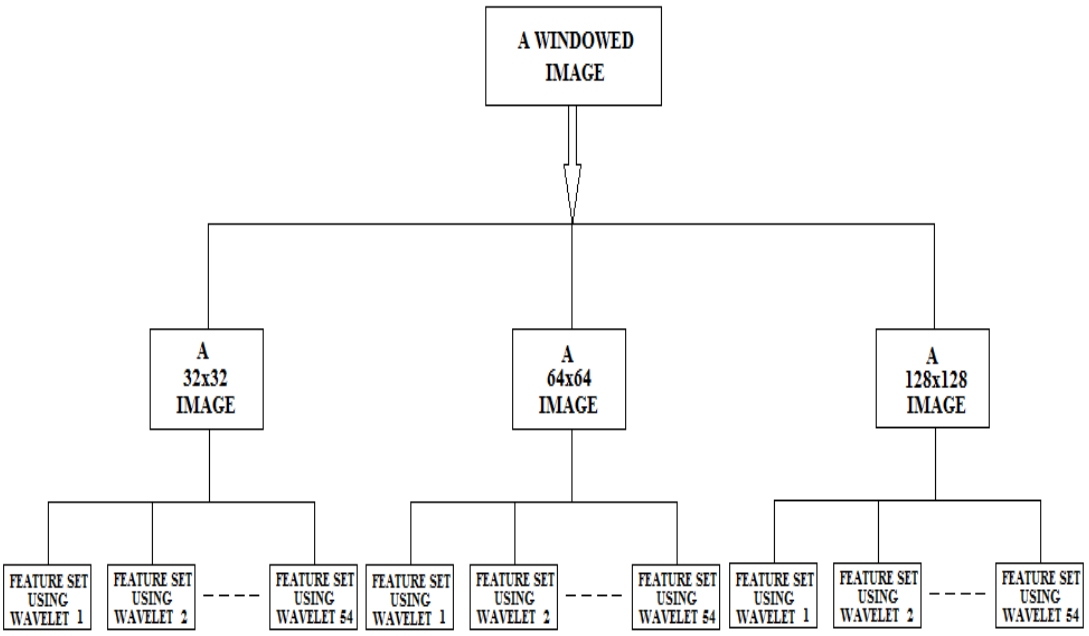


Figure 3.8: Feature sets developed from a single windowed image

Figure 3.8 represents the complete feature extraction process representing the exact number of feature sets developed from a windowed image. For every windowed image, $3 \times 54 = 162$ feature sets were developed. Table 3.1 contains a list of the wavelets used [46] and Table 3.2 shows the wavelet families.

Table 3.1: Various wavelets used for wavelet decomposition

bior1.1	db8
bior1.3	db9
bior1.5	db10
bior2.2	dmey
bior2.4	haar
bior2.6	sym2
bior2.8	sym3
bior3.1	sym4
bior3.3	sym5
bior3.5	sym6
bior3.7	sym7
bior3.9	sym8
bior4.4	rbio1.1
bior5.5	rbio1.3
bior6.8	rbio1.5
coif1	rbio2.2
coif2	rbio2.4
coif3	rbio2.6
coif4	rbio2.8
coif5	rbio3.1
db1	rbio3.3
db2	rbio3.5
db3	rbio3.7
db4	rbio3.9

db5	rbio4.4
db6	rbio5.5
db7	rbio6.8

Table 3.2: Wavelet families

Wavelet family	**Wavelet keyword**	**Wavelets belonging to the family**
Bior Splines	bior	bior1.1, bior1.3, bior1.5, bior2.2, bior2.4, bior2.6, bior2.8, bior3.1, bior3.3, bior3.5, bior3.7, bior3.9, bior4.4, bior5.5, bior6.8
Coiflets	coif	coif1, coif2, coif3, coif4, coif5
Daubechies	db	db1, db2, db3, db4, db5, db6, db7, db8, db9, db10
DMeyer	dmey	dmey
Haar	haar	haar
Symlets	sym	sym2, sym3, sym4, sym5, sym6, sym7, sym8
Reverse Bior	rbio	rbio1.1, rbio1.3, rbio1.5, rbio2.2, rbio2.4, rbio2.6, rbio2.8, rbio3.1, rbio3.3, rbio3.5, rbio3.7, rbio3.9, rbio4.4, rbio5.5, rbio6.8

For a single character/numeral 100 different samples were taken from different users and all these samples were captured as an image from which windowed images were developed for all the samples. From all those images 3 resized images were developed and for all those 3 images per sample 54 feature sets were constructed using the 54 wavelets mentioned in Table 3.1.

Therefore, for each character/numeral $100 \times 3 \times 54 = 16{,}200$ feature sets were developed out of which $80 \times 3 \times 54 = 12{,}960$ feature sets were used to train the lib-svm classifier to recognize the character/numeral where all these feature sets were given the same ID in order to express that all these feature sets belong to a single character and if provided with any matching feature set, the

output should be this particular character/numeral only. The rest 20 × 3 × 54 = 3,240 feature sets were used for the testing purposes in order to find out that lib-svm is able to recognize this particular ID correctly to which extent and this process formed the basis of the process of finding the accuracy of recognition which will be discussed in detail in next chapter.

3.2.1 Samples developed for the upper zone characters

11 characters from the upper zone were tested. The characters tested are shown in Table 3.3 along with their IDs.

Table 3.3: Upper zone strokes tested

ID	CHARACTER	DETAILS
121		Upper bar
122		Laawan
123		Dalaawan
124		Hora
125		Ghanoura bar
126		Adhak
127		Bindi
128		Tippi
130		Dalaawan
131		Dalaawan
132		Laawan

For all the above mentioned IDs from the upper zone 100 samples each were developed = 100 × 11 = 1,100 samples and from each of these samples, three normalized samples were developed = 1,100 × 3 = 3,300 samples from which 3,300 × 54 = 178,200 feature sets were obtained by wavelet decomposition of these 3,300 samples using 54 different wavelets (if only 1 level of decomposition is considered). Any of the above mentioned feature sets contained the wavelet coefficients as features obtained after wavelet decomposition of a sample by using one of the wavelets shown in Table 3.1.

3.2.2 Samples developed for the middle zone characters

33 characters from the middle zone were tested. The characters tested are shown in Table 3.4 along with their IDs.

Table 3.4: Middle zone strokes tested

ID	CHARACTER	DETAILS
141		Urah
144		Aarah
150		Eedi
152		Sassa
155		Haaha
157		Kakka
159		Khakha, Pappa
160		Khakha
162		Vertical bar
164		Gagga, Rara

166		Ghagga
169		Aeyaa
170		Chachaa
172		Chhachha
175		Jajja
176		Jhajja
180		Neyaa
181		Vaawa
182		Tenqa
183		Thatha
185		Dadda
187		Dhadhaa
190		Nahnaa
191		Tatta
194		Dadhaa
195		Dhadda
196		Nanhaa

200		Faffa
203		Babba
205		Bhabhaa
206		Mamma
208		Yaeya
210		Lalla

For all the above mentioned IDs from the middle zone 100 samples each were developed = 100 × 33 = 3,300 samples and from each of these samples, three normalized samples were developed = 3,300 × 3 = 9,900 samples from which 9,900 × 54 = 534,600 feature sets were obtained by wavelet decomposition of these 9,900 samples using 54 different wavelets (if only 1 level of decomposition is considered). Any of the above mentioned feature sets contained the wavelet coefficients as features obtained after wavelet decomposition of a sample by using one of the wavelets shown in Table 3.1.

3.2.3 Samples developed for the lower zone characters

7 characters from the middle zone were tested. The characters tested are shown in Table 3.5 along with their IDs.

Table 3.5: Lower zone strokes tested

ID	CHARACTER	DETAILS
101		Onkaar
102		Bindi
103		Raa

104		Haa
105		Vaa
106		Dulenkar
107		Halant

For all the above mentioned IDs from the middle zone 100 samples each were developed = 100 × 7 = 700 samples and from each of these samples, three normalized samples were developed = 700 × 3 = 2,100 samples from which 2,100 × 54 = 113,400 feature sets were obtained by wavelet decomposition of these 2,100 samples using 54 different wavelets (if only 1 level of decomposition is considered). Any of the above mentioned feature sets contained the wavelet coefficients as features obtained after wavelet decomposition of a sample by using one of the wavelets shown in Table 3.1.

3.2.4 Samples developed for numerals

40 characters from the numerals were tested. The characters tested are shown in Table 3.6 along with their IDs.

Table 3.6: Numerals strokes tested

ID	CHARACTER	DETAILS
351		One
352		Two
353		Three
354		Four

355		Five
356		Six
357		Seven
358		Eight
359		Nine
360		Zero
361		One
362		Two
363		Three
364		Four
365		Five
366		Six
367		Seven
368		Eight

369		Nine
370		Roman One
371		Roman Five
372		Roman Ten
373		Roman Fifty
374		Roman Hundred
375		Roman Five Hundred
376		Roman Thousand
377		Plus
378		Minus
379		Back Slash
380		Forward Slash
381		Open Square Bracket
382		Close Square Bracket

383		Open Round Bracket
384		Close Round Bracket
385		Open Curly Bracket
386		Close Curly Bracket
387		Apostrophe
388		At the Rate
389		Comma
390		Four

For all the above mentioned IDs from the numerals 22 samples each were developed = 22 × 40 = 880 samples and from each of these samples, three normalized samples were developed = 880 × 3 = 2,640 samples from which 2,640 × 54 = 142,560 feature sets were obtained by wavelet decomposition of these 2,640 samples using 54 different wavelets (if only 1 level of decomposition is considered). Any of the above mentioned feature sets contained the wavelet coefficients as features obtained after wavelet decomposition of a sample by using one of the wavelets shown in Table 3.1.

3.2.5 Feature sets development from a sample

A single sample was normalized three times to get three normalized images of 32 × 32, 64 × 64 and 128 × 128.

32 × 32 sample was wavelet decomposed (using any one of the wavelets) two times to get 16 × 16 image that gives 16 × 16 = 256 coefficients that formed one feature set and 8 × 8 image that

gives 8 × 8 = 64 coefficients that formed the other feature set. Similarly, by wavelet decomposition of the same sample by other 53 wavelets, 53 × 2 = 106 feature sets were obtained making a total of 108 feature sets, half of which had 256 coefficients and the other half contained 64 coefficients.

Similarly, 64 × 64 sample was wavelet decomposed (using any one of the wavelets) two times to get 32×32 image that gives 32 × 32 = 1,024 coefficients that formed one feature set and 16×16 image that gives 16 × 16 = 256 coefficients that formed the other feature set. Similarly, by wavelet decomposition of the same sample by other 53 wavelets, 53 × 2 = 106 feature sets were obtained making a total of 108 feature sets, half of which had 1,024 coefficients and the other half contained 256 coefficients.

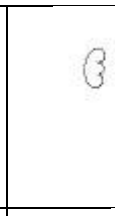
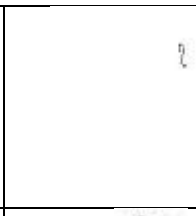
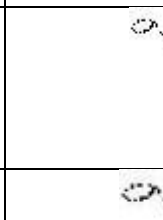
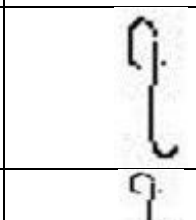
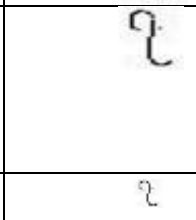
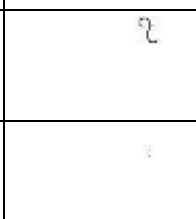
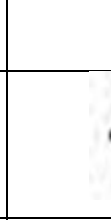
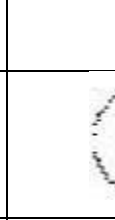
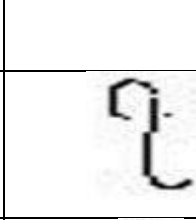
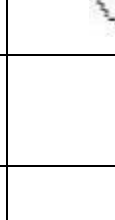
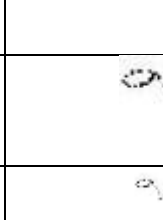
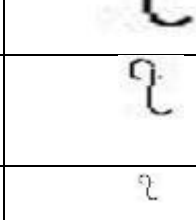
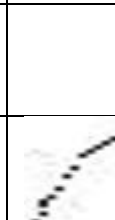
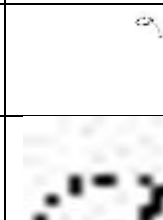
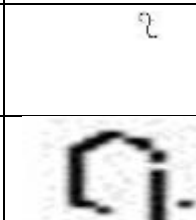
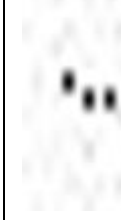
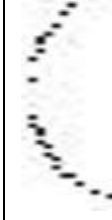
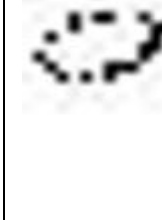
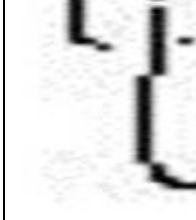
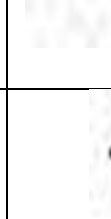
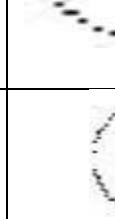
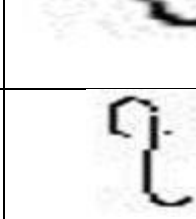
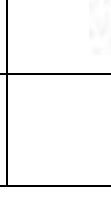
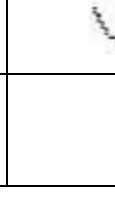
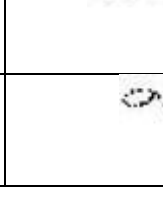
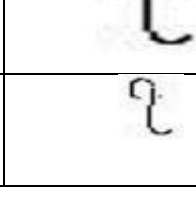
128 × 128 sample was wavelet decomposed (using any one of the wavelets) three times to get 64 × 64 image that gives 64 × 64 = 4,096 coefficients that formed one feature set, 32 × 32 image that gives 32 × 32 = 1,024 coefficients that formed second feature set and 16 × 16 image that gives 16 × 16 = 256 coefficients that formed the third feature set. Similarly, by wavelet decomposition of the same sample by other 53 wavelets, 53 × 3 = 159 feature sets were obtained making a total of 162 feature sets, one-third of which were 4,096 coefficients, other one-third contained 1,024 coefficients and the remaining one-third had 256 coefficients.

Therefore, in total 108 + 108 + 162 = 378 feature sets were developed for a single sample. In total (3,300 + 9,900 + 2,100 + 2,640) × 378 = 6,781,320 feature sets were developed.

Table 3.7 represents the complete procedure of development of all feature sets from a single sample by representing the all levels of decomposition.

Table 3.7: Levels of decomposition used

Description	**Upper Zone Character**	**Middle Zone Character**	**Lower Zone Character**	**Numeral**

1000 × 1000 developed image				
Windowed image				
Normalized image to 32 × 32				
Decomposed to 16 × 16				
Decomposed to 8 × 8				
Normalized to 64 × 64				
Decomposed to 32 × 32				
Decomposed to 16 × 16				
Normalized to 128 × 128				
Decomposed to 64 × 64				
Decomposed to 32 × 32				

Decomposed to 16 × 16				

Algorithm 3.4 represents a sample how to develop the feature set 1 from sample 128 × 128 image shown in Figure 3.9 by performing level 1 decomposition.

Figure 3.9: A sample 128 × 128 image

Algorithm 3.4: Developing all the feature sets from a 128 × 128 image

1. Read the image in *X*.
2. Apply 3 levels of wavelet decomposition on the image using any one of the 54 wavelets shown in Table 3.1.
3. Store the approximation coefficients in *A1*, *A2* and *A3* corresponding to level 1, level 2 and level 3 decomposition.
4. Store the size limits of level 1, level 2 and level3 in *Y* in the same sequence.
5. Set '*name*' array to [*a*,*b*,*c*];
6. Loop *i* from 1 to 3
   - Set *n*=0
   - Open a text file named *name*(*i*)
   - Loop *j* from 1 to *Y*(*i*)
     - Loop *k* from 1 to *Y*(*i*)
       - Increment *n*
       - If *i*==1

         Coeff=*A1*(*j*, *k*)

         Else if *i*==2

         Coeff=*A2*(*j*, *k*)

Else

Coeff=A3(j, k)

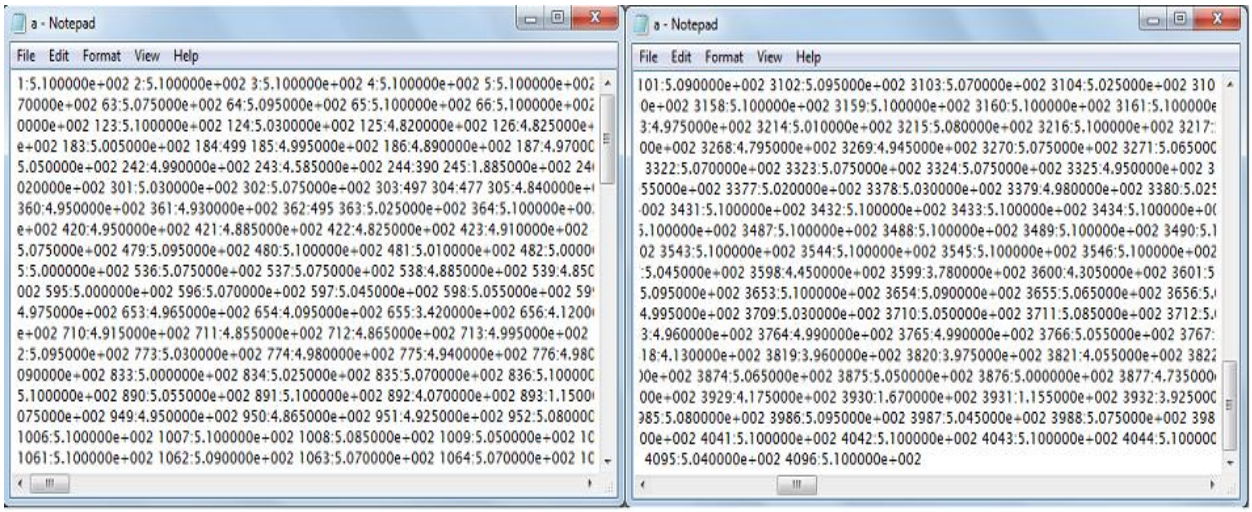
End if

- Write *n* followed by colon followed by the *coeff* in file opened.

  - End loop

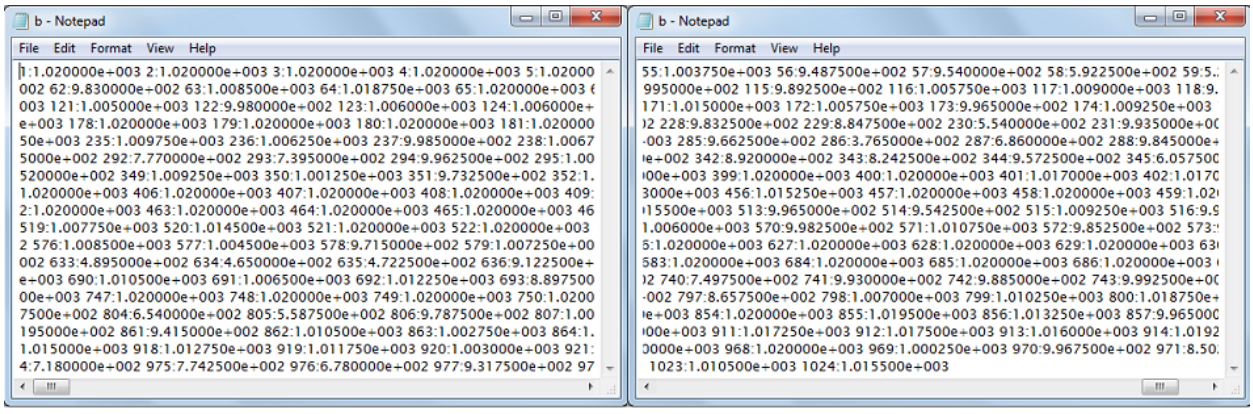
- End loop

7. Close the file opened
8. End loop



[a] Start of file [b] End of file

Figure 3.10: Feature set 1 (4,096 coefficients)



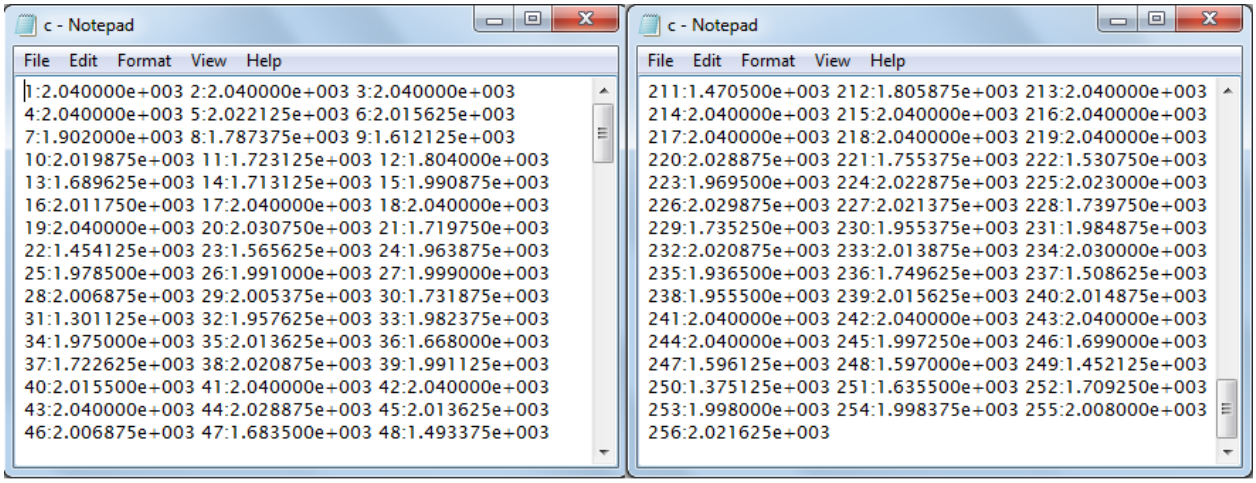
[a] Start of file [b] End of file

Figure3.11: Feature set 2 (1,024 coefficients)

Figure 3.10 represents the output feature set in a text file. As the sample image obtained after wavelet decomposition is of size 64 × 64, therefore, the feature set 1 contains 4,096 coefficients.

Figure 3.11 represents the output feature set 2 developed by the level 2 wavelet decomposition of the image represented by the Figure 3.9 or level 1 decomposition of the decomposed sample obtained by the level 1 decomposition. As the sample image obtained after level 2 wavelet decomposition is of size 32 × 32, therefore, the feature set 2 contains 1,024 coefficients.

Figure 3.12 represents the output feature set 3 in a text file obtained by the wavelet decomposition of the decomposed sample obtained by the level 2 decomposition. This process is called level 3 decomposition. This feature set can also be obtained by 3 level decomposition of the image shown in Figure 3.9. As the sample image obtained after level 3 wavelet decomposition is of size 16 × 16, therefore, the feature set 3 contains 256 coefficients.



[a] Start of file [b] End of file

Figure 3.12: Feature set 3 (256 coefficients)

Similarly, a 64 × 64 image is decomposed to level 2 of wavelet decomposition giving two feature sets of 32 × 32 and 16 × 16 sizes containing 1,024 and 256 coefficients correspondingly and a 32 × 32 image is also decomposed to level 2 of decomposition giving two feature sets of 16 × 16 and 8 × 8 sizes containing 256 and 64 coefficients correspondingly.

Chapter 4

Recognition of Punjabi Numerals and Zone-wise Recognition of Punjabi Characters Using Wavelet Based Features

Recognition is the major step of the whole online handwritten character recognition process. As discussed before it involves recognizing a particular character/numeral written by the user online that is captured as an image, by classifying the feature set (developed from any of the normalized image by wavelet decomposition of the image using any of the 54 wavelets discussed in previous Chapter). Classification of the feature set involves providing a class to the character/numeral to which it may belong depending on its feature set where a class is any ID from it's zone if it's a character else any ID from numerals. Every ID represents a different class.

In order to classify a character/numeral depending upon its feature set, the classifier must be trained first with a number of feature sets of the same character/numeral obtained from various samples of that character/numeral so that it classifies the feature sets provided from the same ID's sample to the same ID.

The classifier used was lib-svm and for recognition, out of 100 samples of each ID of a particular size 80 samples were used for training which gave 80 feature sets using one wavelet which were kept as one training file and the rest 20 samples were used for testing from which the feature sets obtained using one wavelet were kept in one testing file. Similarly, for every 54 wavelets applied, 54 different training and 54 different testing files were obtained.

But instead of keeping every ID's testing feature sets in different files, testing feature sets of all IDs from same zone or all numeral IDs using same wavelet were kept in the same file called testing file and similarly, all training feature sets from same wavelet of all similar zone IDs were kept in the same file called training file. Similarly 53 other training and testing files were obtained from all the same ID's samples but of same size.

Figure 4.1 represents the total number of training and testing files and the number of feature sets per training/testing file.

$$\text{Total number of prediction files} = \sum_{i=1}^{\text{number of normalizations}} \text{number of decomposition levels} \times 54(\text{number of wavelets})$$

$$= \text{total number of training files}$$

$$\text{Number of feature sets per prediction file} = \text{number of Ids in zone/number of numerals} \times 20$$

$$\text{Number of feature sets per training file} = \text{number of Ids in zone/number of numerals} \times 80$$

Figure 4.1: Total number of training/testing files and number of feature sets per training/testing file

4.1 Lib-svm format for the training and testing files

The lib-svm classifier uses a special format for the training and testing files. The classifier takes as input the testing and training files to give the accuracy of recognition only in this format. The format goes like,

ID 1:1st coefficient 2:2nd coefficient … n:n^{th} coefficient (a line break)

ID 1:1st coefficient 2:2nd coefficient ... n:n^{th} coefficient (a line break)

.

.

.

ID 1:1st coefficient 2:2nd coefficient … n:n^{th} coefficient

Therefore, we conclude that every line's first few characters before a space is taken as the ID and then before the line break a single feature set goes in the format, the coefficient number (m) followed by colon followed by the coefficient (m^{th}). n for every ID in single file whether it is the training or the testing file should be same and even all the feature sets in a single training/testing file were developed by wavelet decomposition of same sized samples using the same wavelet.

A sample is shown in Figure 4.2.

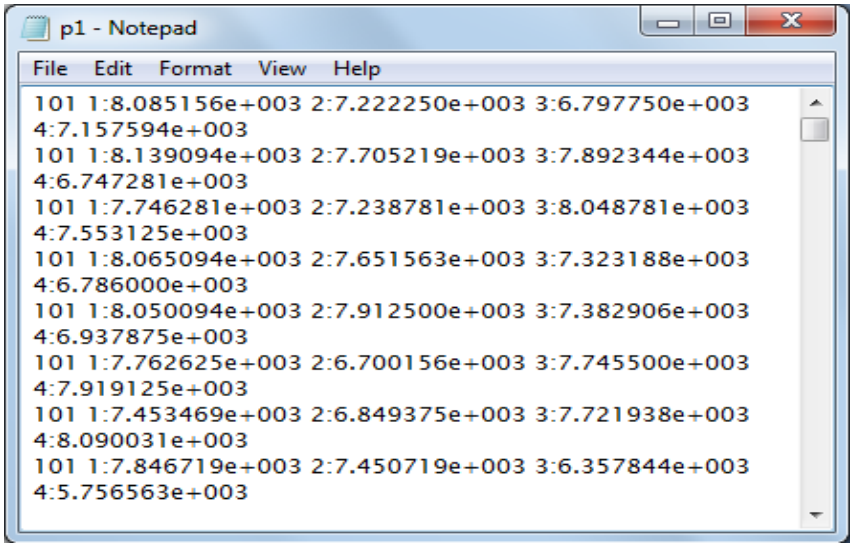


Figure 4.2: A sample testing file with $n = 4$

4.2 Statistics on training and testing files from upper zone

11 IDs were tested from the upper zone for as discussed before.

From every 100 samples of each ID of $32 \times 32 = 11 \times 100 = 1{,}100$ samples, $11 \times 80 = 880$ samples were used for training which gave 880 feature sets using one wavelet which were kept as one training file and the rest $11 \times 20 = 220$ samples were used for testing from which the feature sets obtained using one wavelet were kept in one testing file. Similarly, for every 54 wavelets applied, 54 different training and 54 different testing files were obtained. But these 54 training/testing files were developed by level one decomposition of the samples. Therefore, they had 256 coefficients in every feature set. Similarly, 54 training and 54 testing files were developed from the same samples by level 2 decomposition of the samples in which every feature set had 64 coefficients.

Similarly, $54 \times 2 = 108$ training files and 108 testing files were developed from 1,100 samples of 64×64 size, out of which 54 training and 54 testing files had 1,024 coefficients in each feature set and the rest all had 256 coefficients per feature set.

Similarly, $54 \times 3 = 162$ training files and 162 testing files were developed from 1,100 samples of 128×128 size, out of which 54 training and 54 testing files had 4,096 coefficients, other 54

training and 54 testing files had 1,024 coefficients in each feature set and the rest all had 256 coefficients per feature set.

Therefore, in total the number of testing files = 108 + 108 + 162 = 378 files = number of training files.

Using the formula shown in Figure 4.1,

Number of testing files = 2 × 54 (for 32 × 32) + 2 × 54 (for 64 × 64) + 3 × 54 (for 128 × 128) = 108 + 108 + 162 = 378 files = number of training files that is equivalent to the above calculated files and the number of feature sets per testing file = 11 × 20 = 220 and the number of feature sets per training file = 11 × 80 = 880.

4.3 Statistics on training and testing files from middle zone

33 IDs were tested in from the middle zone. According to the formula in Figure 4.1,

Number of testing files = 2 × 54 (for 32 × 32) + 2 × 54 (for 64 × 64) + 3 × 54 (for 128 × 128) = 108 + 108 + 162 = 378 files = number of training files that is equivalent to the above calculated files and the number of feature sets per testing file = 33 × 20 = 660 and the number of feature sets per training file = 33 × 80 = 2,640 sets.

4.4 Statistics on training and testing files from lower zone

7 IDs were tested in from the lower zone as discussed in previous chapter. Using the formula in Figure 4.1,

Number of testing files = 2 × 54 (for 32 × 32) + 2 × 54 (for 64 × 64) + 3 × 54 (for 128 × 128) = 108 + 108 + 162 = 378 files = number of training files that is equivalent to the above calculated files and the number of feature sets per testing file = 7 × 20 = 140 and the number of feature sets per training file = 7 × 80 = 560.

4.5 Statistics on training and testing files from numerals

40 IDs were tested in from the numerals. According to the formula in Figure 4.1,

Number of testing files = 2 × 54 (for 32 × 32) + 2 × 54 (for 64 × 64) + 3 × 54 (for 128 × 128) = 108 + 108 + 162 = 378 files = number of training files that is equivalent to the above calculated files and the number of feature sets per testing file = 40 × 8 = 320 and the number of feature sets per training file = 40 × 16 = 640 sets.

4.6 Creation of training and testing files

Algorithm 4.1 represents the complete testing and training files creation process for all 100 samples of same size of all IDs belonging to same zone/numerals.

Algorithm 4.1: Creation of training and testing files from all same sized samples of same zone/of numerals

1. Set array *coeff*'s values to the number of coefficients wanted from the images (size of array is either 2 or 3 depending on the size of the samples taken).
2. Set array *limit* to the number of rows acc to the number of coefficients required corresponding to the array *coeff*.
3. Set array *wave* to 54 wavelets to be used for decomposition.
4. Set array *Ids* to all the IDs from the zone or numerals accordingly.
5. Loop *pt* from 1 to 2
    - Loop *L* from 1 to length of array *coeff*
        - Loop *w* from 1 to length of array *wave*
            - Loop *i* from 1 to length of array *Ids*
                - If pt==1

                  L1=1 and *L2*=20

                  Else

                  L1=21 and *L2*=80

                  End if
                - Loop *j* from *L1* to *L2*
                    - Set name to file represented by *j*
                    - Apply wavelet decomposition to 3 levels on the file represented by name using wavelet represented by *wave*(*w*) and

store the approximation coefficients from these levels in *A1, A2, A3*

- If *pt*==1

  Set filename to concatenation of '*p*' and number represented by *w*

  Else

  Set filename to concatenation of '*t*' and number represented by *w*

  End if
- Open the file represented by filename in writing mode
- Set *n*=0
- Loop *k* from 1 to *Limit*(*L*)
  - Loop *m* from 1 to *Limit*(*L*)
    - Increment '*n*'
    - If *L*==1

      Put *n* followed by colon followed by *A1*(*k*, m) followed by a line break or a space depending on whether *n*=*coeff*(*L*) or not in file filename

      Else if *L*==2

      Put *n* followed by colon followed by *A2*(*k*, *m*) followed by a line break or a space depending on whether *n*=*coeff*(*L*) or not in file filename

Else

Put *n* followed by colon followed by *A3*(*k*, *m*) followed by a line break or a space depending on whether *n*=*coeff*(*L*) or not in file filename

End if

- End loop
- End loop
- Close file represented by filename
- End loop
- End loop
- End loop
- End loop

6. End loop

For 32 × 32 samples, array *coeff*'s values are 256 and 64 and correspondingly the array limit contains values 16 and 8. For any set of IDs represented by IDs array the algorithm will develop 54 × 2 = 108 testing files and 108 training files out of which 54 testing and 54 training files will contain feature sets with 256 coefficients and the rest will contain feature sets with 64 coefficients.

Similarly, for 64×64 samples, array *coeff*'s values are 1,024 and 256 and correspondingly the array limit contains values 32 and 16. 108 testing files and 108 training files will be developed out of which 54 testing and 54 training files will contain feature sets with 1,024 coefficients and the rest will contain feature sets with 256 coefficients.

Similarly, for 128 × 128 samples, array *coeff*'s values are 4,096, 1,024 and 256 and correspondingly the array limit contains values 64, 32 and 16. 162 testing files and 162 training files will be developed out of which 54 testing and 54 training files will contain feature sets with

4,096 coefficients and other 54 testing and 54 training files will contain feature sets with 1,024 coefficients and the rest will contain feature sets with 256 coefficients.

Figure 4.3 represents testing files from a 64 × 64 image that was decomposed using wavelet bior1.1.

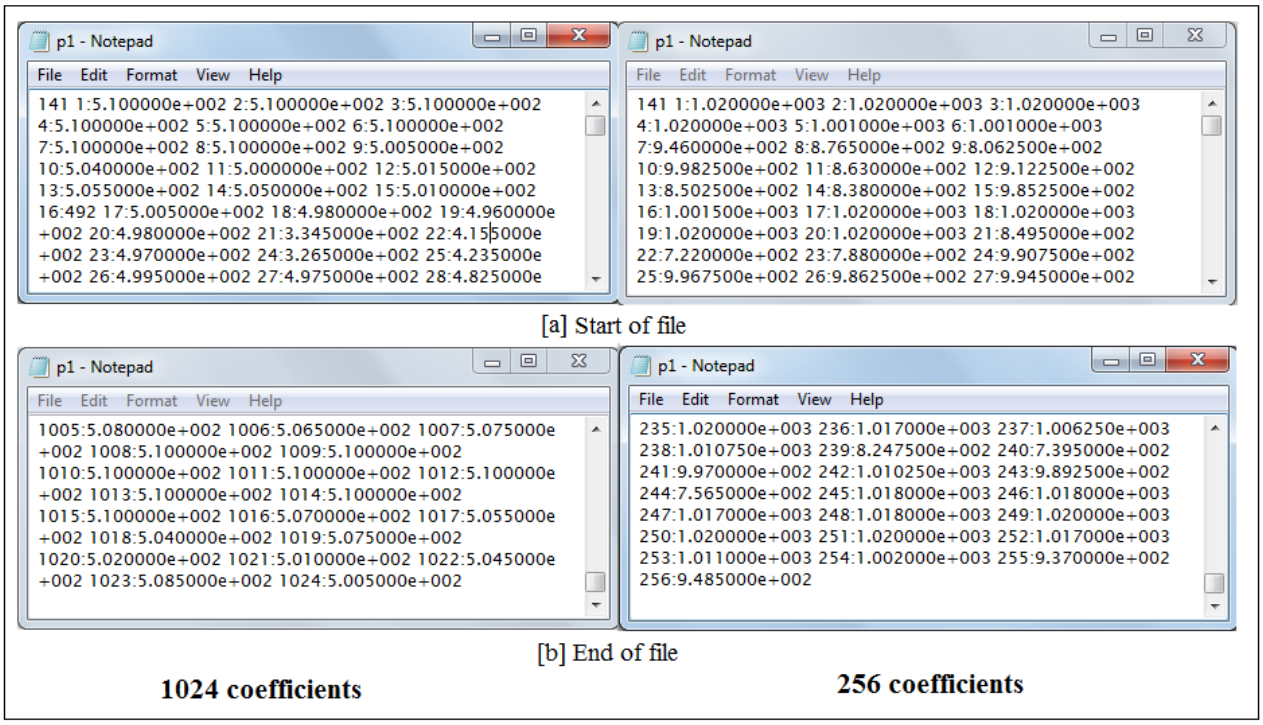


Figure 4.3: Testing files of 64 × 64 image decomposed using bior1.1

4.7 Recognition process

Figure 4.4 represents the process of recognition using the testing and training files developed above.

4.7.1 Scaling

The scaling was done using the same parameters for all the testing or training files used. The parameters taken were,

-l 1 -u 9

where ‘l’ represents the minimum value for scaling and ‘u’ represents the maximum limit for scaling.

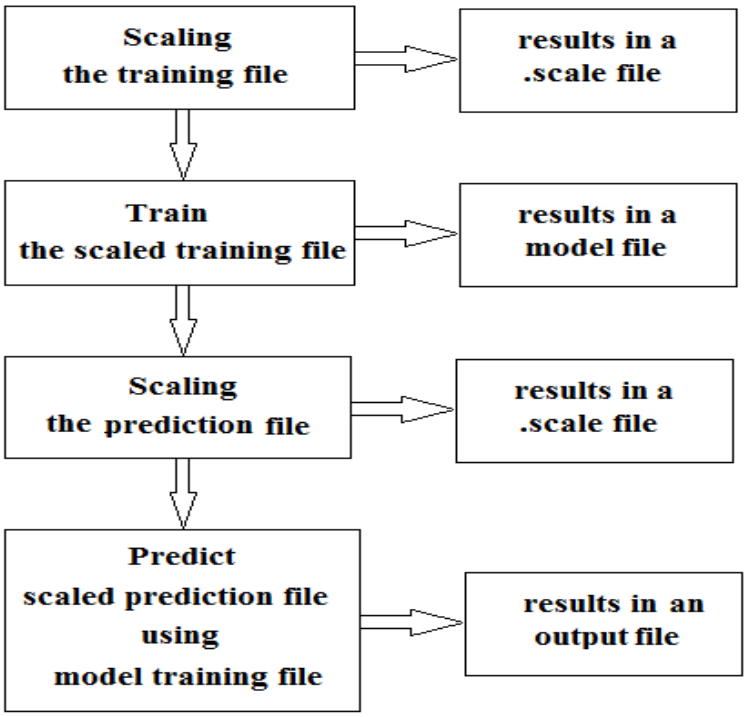


Figure 4.4: The recognition process

4.7.2 Training

For training firstly, the general parameters were taken for all the 54 training files of same number of coefficients that are described below that gave 54 model files correspondingly

-s 0 –t 2 –d 3 –g 1 –r 0 –c 1

and then out of the 54 output files obtained using these 54 model files correspondingly, the best output that corresponds to the maximum accuracy achieved for that number of coefficients is found and then all combinations are tried for the various parameters on that particular wavelet training-testing file.

Various parameters combinations [45] tried are shown in Table 4.1 and 4.2.

Table 4.1: Parameters tried for training

Parameter name	**Parameter description**	**Parameter symbol**	**Values tried**
svm type	type of the	*-s*	0, 1, 2, 3, 4

	support vector machine		
cost	value for the parameter C for C-SVC, epsilon-SVR, and nu-SVR classifiers	*-t*	0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000
kernel type	type of the kernel	*-c*	0, 1, 2,
nu	Value for the parameter nu for nu-SVC, one-class SVM, and nu-SVR classifiers	*-n*	0.1, 0.2, 0.3,…0.9
epsilon	tolerance value for the termination criterion	*-e*	0.01, 0.05, 0.1, 0.5, 1
epsilon	epsilon value for the loss function of epsilon-SVR	*-p*	0.1
gamma	gamma value for the kernel function	*-g*	0, 0.01, 0.05, 0.1, 0.5, 1
coef0	coef0 value for the kernel function	*-r*	0, 1, 5, 10, 50, 75, 100
degree	degree value for the kernel function	*-d*	0, 1, 2, 3

cross validation	k-fold cross validation	*-v*	1, 5, 10, 20, 50, 80

Table 4.2: SVM type and kernel type parameters in detail

Parameter name	**Parameter symbol**	**Value**	**Description**
type of support vector machine	*-s*	0	*C*-SVC
		1	*nu*-SVC
		2	One class SVM
		3	Epsilon SVR
		4	*nu*-SVR
kernel type	*-t*	0	linear kernel = $u \times v$
		1	polynomial = (gamma × $u \times v$ + coef0)^{degree}
		2	radial basis = $e^{(-gamma \times |u-v|^2)}$
		3	sigmoid = tanh(gamma × $u \times v$ + coef0)

All the parameters with all the possible combinations were not tried together. There are parameters combinations that are tried together and rest are not required at that time. Table 4.3 represents the various parameters that are tested together for various combinations.

Table 4.3: Various combinations tried

Parameter	**Value**	**Parameters used along with it**
-s	0	*c*
	1	*n*
	2	*n*
	3	*c, e, p*
	4	*c, n*
-t	0	*v*
	1	*g, r, d, v*
	2	*g, v*

	3	*g, r, v*

4.7.3 Testing

The testing step takes as input a model training file and a scaled testing file and gives two output files, one gives the testing analyzing which tells us which feature set was predicted with correct ID and which was not and the other file tells us the accuracy achieved in this process with that particular model file that was achieved with particular values for various parameters used. Algorithm 4.2 shows creation of batch files.

Algorithm 4.2: Creation of batch files to try all the parameter combinations on a testing-training files pair

1. Set range of values for '*n*', '*d*', '*g*', '*v*', '*c*', '*p*', '*e*', '*r*' in *nr*, *dr*, *gr*, *vr*, *cr*, *pr*, *er* and *rr* correspondingly.
2. Loop *s* from 0 to 4
    - If *s*==0

      sr='*c*'

      Else if *s*==1 or *s*==2

      sr='*n*'

      Else if *s*==3

      sr=['*c*','*e*','*p*']

      Else

      sr=['*c*','*n*']

      End if
    - Loop *t* from 0 to 3
        - If *t*==0

          tr='*v*'

          Else if *t*==1

          tr=['*g*','*r*','*d*','*v*']

          Else if *t*==2

          tr=['*g*','*v*']

          Else

tr=[‘*g*’,’*r*’,’*v*’]

End if

- Set *A* to combination of arrays *sr* and *tr*.
- Set *a* to the range arrays of parameters in *sr* and *tr*.
- Set *len* to the number of columns in *A*.
- Open file named *SvmTrain* in append mode.
- Write ‘(\n’ in *SvmTrain*.
- Loop *k* from 1 to *len*
    - Loop *j* from 1 to length(*A*)
        - Set string *aa* to ‘–*A*(*j*) *a*(*j*, *k*)’.
    - End loop
    - Write a string ‘svm-scale.exe -l 1 -u 9 -s range t.txt > t.txt.scale' into *SvmTrain* followed by a line break.
    - Write string 'svm-train.exe –s ’ in *SvmTrain*.
    - Write value of *s* in *SvmTrain*.
    - Write string ‘ –t ‘ in *SvmTrain*.
    - Write value of *t* in *SvmTrain*.
    - Write string ‘t.txt.scale' in *SvmTrain* followed by a line break.
    - Write string ‘svm-scale.exe -l 1 -u 9 -s range p.txt > p.txt.scale' into *SvmTrain* followed by a line break.
    - Write string 'svm-predict.exe p.txt.scale t.txt.scale.model o.txt >> output.txt’ in *SvmTrain* followed by a line break.
    - Write a string ‘svm-scale.exe -l 1 -u 9 -s range t.txt > t.txt.scale' into *SvmTrain* followed by a line break.
- End loop
- Write ‘)’ in *SvmTrain*.
- Close *SvmTrain*.
- End loop

3. End loop

`

Chapter 5

Results and Discussions

In order to find out the best accuracy, every 54 files were first tested with the general parameters in order to find out the wavelet that gives the maximum accuracy for that size and that number of coefficients combination.

In all, for any zone or for numerals 2 (for images of 32 × 32) + 2 (for images of 64 × 64) + 3 (for images of 128 × 128) = 7 sets of 54 training files were tested to find out their set's wavelet that gave the best accuracy with general parameters for svm-train.

A single batch file was created with all the four commands of scaling the training file, training the training file, scaling the prediction file and prediction of the scaled prediction file using the model training file developed, for all 54 prediction-training files (54 × 4 = 216 commands). This batch file was executed 7 times for 7 pairs of 54 training-prediction files for one zone and for all zones including numerals 7 × 4 = 28 times.

Table 5.1 represents the wavelets found for all 54 training-prediction files that gave the best accuracy with general (default) parameters.

Table 5.1: Wavelet giving maximum accuracy with default/general parameters for training

Size	Coefficients	Wavelet	Accuracy
UPPER ZONE			
32 × 32	256	db2	91.3%
32 × 32	64	bior1.1	90.4%
64 × 64	1,024	sym4	90.4%
64 × 64	256	db1	90.4%
128 × 128	4,096	bior1.1	88.6%
128 × 128	1,024	bior1.1	90.0%
128 × 128	256	bior1.1	90.0%
MIDDLE ZONE			
32 × 32	256	rbio3.1	85.6%

`

32 × 32	64	coif5	59.7%
64 × 64	1,024	rbio3.1	86.3%
64 × 64	256	rbio3.1	86.0%
128 × 128	4,096	rbio3.3	84.8%
128 × 128	1,024	bior1.1	61.8%
128 × 128	256	bior1.1	38.4%
LOWER ZONE			
32 × 32	256	rbio3.3	81.4%
32 × 32	64	db2	82.1%
64 × 64	1,024	bior3.1	80.7%
64 × 64	256	rbio3.1	83.5%
128 × 128	4,096	bior3.1	75.7%
128 × 128	1,024	bior2.8	78.5%
128 × 128	256	rbio3.3	82.8%
NUMERALS			
32 × 32	256	rbio3.1	61.8%
32 × 32	64	rbio3.1	65.6%
64 × 64	1,024	rbio3.1	55.9%
64 × 64	256	rbio3.1	60.9%
128 × 128	4,096	bior1.1	52.8%
128 × 128	1,024	rbio3.1	54.0%
128 × 128	256	rbio3.1	61.5%

After finding the wavelets, all the parameter combinations discussed in chapter 4 were applied on the training-prediction pair obtained from the specific wavelets that gave maximum accuracy for that zone or numerals.

Table 5.2 represents the results obtained after running the batch file of all combinations on the particular pair of training-prediction files obtained using the particular wavelets determined earlier shown in Table 5.1.

`

Table 5.2: Maximum accuracy achieved for all sizes of character image and number of wavelet coefficients tried

Size	Coefficients	Accuracy
UPPER ZONE		
32 × 32	256	91.8%
32 × 32	64	95.0%
64 × 64	1,024	93.6%
64 × 64	256	93.2%
128 × 128	4,096	90.0%
128 × 128	1,024	92.7%
128 × 128	256	93.6%
MIDDLE ZONE		
32 × 32	256	90.7%
32 × 32	64	69.3%
64 × 64	1,024	90.3%
64 × 64	256	90.6%
128 × 128	4,096	70.3%
128 × 128	1,024	65.6%
128 × 128	256	65.6%
LOWER ZONE		
32 × 32	256	89.2%
32 × 32	64	88.5%
64 × 64	1,024	87.8%
64 × 64	256	89.2%
128 × 128	4,096	88.5%
128 × 128	1,024	84.6%
128 × 128	256	83.5%
NUMERALS		
32 × 32	256	71.8%
32 × 32	64	74.3%

`

64 × 64	1,024	67.1%
64 × 64	256	72.8%
128 × 128	4,096	61.5%
128 × 128	1,024	65.9%
128 × 128	256	70.6%

`

Chapter 6

Conclusion and the Future Scope

6.1 Conclusion

After trying out all the combinations on 32 × 32 (64 coefficients), 32 × 32 (256 coefficients), 64 × 64 (256 coefficients), 64 × 64 (1,024 coefficients), 128 × 128 (256 coefficients), 128 × 128 (1,024 coefficients) and 128 × 128 (4,096 coefficients) images of upper zone, middle zone, lower zone and numerals it was concluded that the maximum accuracies in all the zones and numerals were those shown in Table 6.1.

Table 6.1: Overall maximum accuracy achieved

Zone	Size	Coefficients	Wavelet	Maximum Accuracy
Upper Zone	32 × 32	64	bior1.1	95.0%
Middle Zone	32 × 32	256	rbio3.1	90.7%
Lower Zone	32 × 32, 64 × 64	256	rbio3.3, rbio3.1	89.2%
Numerals	32 × 32	64	rbio3.1	74.3%

From above table it can be concluded that for upper zone maximum accuracy of 95.0% was achieved by feature sets containing 64 coefficients that were obtained by wavelet decomposition by bior1.1 wavelet of images from this zone of size 32 × 32. For middle zone maximum accuracy of 90.7% was achieved by feature sets containing 256 coefficients that were obtained by wavelet decomposition by rbio3.1 wavelet of images from this zone of size 32 × 32. For lower zone maximum accuracy of 89.2% was achieved by feature sets containing 256 coefficients that were obtained by wavelet decomposition by rbio3.3 and rbio3.1 wavelet of images from this zone of size 32 × 32 and 64 × 64 respectively. For numerals maximum accuracy of 74.3% was achieved by feature sets containing 64 coefficients that were obtained by wavelet decomposition by rbio3.1 wavelet of images from this zone of size 32 × 32.

6.2 Future scope

In future, one can try to improve the accuracies achieved by following experiments:

`

- 100 samples per ID were considered out of which 80 per Id were used to train and the rest 20 were used to predict. One can try and increase or decrease the number of samples per Id and even different divisions of samples into training file and prediction file can also be considered.
- For all the images three normalization sizes were considered; 32 × 32, 64 × 64 and 128 × 128. Various other sizes can also be experimented on.
- 32 × 32 image was decomposed to two levels giving 256 and 64 coefficients respectively and 64 × 64 image was also decomposed to two levels giving 1,024 and 256 coefficients. Similarly, 128 × 128 image was also decomposed to three levels of decomposition giving 4,096, 1,024 and 256 coefficients.
- Along with various sizes, one can also try out various levels of decomposition giving different number of coefficients.
- From middle zone all the strokes were not considered. The left strokes should also be considered for testing for accuracy.
- The parameters for various exe files related to the scaling, training and prediction can be tested for other different values.

`

References

[1]	P. Ahmed and Y. Al-Ohali, "Arabic Character Recognition: Progress and Challenges," J. King Saud Univ., Dept. of Comp. Sci., Technical Report Comp. and Info. Sci., pp. 85-116, March 16, 1999.
[2]	A. Aichert, "Feature extraction techniques," Technical Report Camp Medical Seminar WS0708, Jan. 9, 2008.
[3]	N. Aouadi, S. Amiri and A. K. Echi, "Segmentation of Connected Components in Arabic Handwritten Documents," in *Proc. Int. Conf Computational Intelligence: Modeling Techniques and Applications*, pp. 738-746, 2013.
[4]	S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu and M. Kundu, "Combining Multiple Feature Extraction Techniques for Handwritten Devnagari Character Recognition," *IEEE Region 10 Colloquium*, Dec. 8-10, 2008.
[5]	B. Bouda, Lh. Masmoudi, B. Chaouki and D. Aboutajdine, "Gray-level Corner Detection by Virtual Electric Field Model," *IEEE Int. Symposium on Computational Intell. and Informatics*, March 28-30, 2007.
[6]	A. Chadha, S. Mallik and R. Johar, "Comparative Study and Optimization of Feature-Extraction Techniques for Content based Image," *Int. Journal Comp. Applications*, vol. 52, no. 20, pp. 975-8887, Aug. 2012.
[7]	S. Cho, "Feature Extraction for Lifelog Management," Technical Report, Sept. 25, 2008.
[8]	A. Choudhary, R. Rishi and S. Ahlawat, "Off-Line Handwritten Character Recognition using Features Extracted from Binarization Technique," in *Proc. AASRI Conf Intelligent Systems and Control*, pp. 306-312, 2013.
[9]	A. Choudhary, R. Rishi and S. Ahlawat, "A New Approach to Detect and Extract Characters from Off-Line Printed Images and Text," in *Proc. Int. Conf Info. Tech. and Quantitative Management*, pp. 434-440, 2013.
[10]	S. Dabbaghchian, A. Aghagolzadeh and M. S. Moin, "Feature Extraction using Discrete Cosine Transform for Face Recognition," *IEEE*, 2007.
[11]	S. Dalal and L. Malik, "A survey for Feature Extraction Methods in Handwritten Script Identification," *IJSSST*, vol. 10, no. 3.

`

[12]	S. G. Dedgaonkar, A. A. Chandavale and A. M. Sapkal, "Survey of Methods for Character Recognition," *Int. Journal of Engineering and Innovative Tech.*, vol. 1, May 5, 2012.
[13]	B. Gatos, D. Karras and S. Perantonis, "Optical Character Recognition using Novel Feature Extraction and Neural Network Classification Techniques," *IEEE*, 1994.
[14]	K. H. Ghazali, M. F. Mansor, M. M. Mustafa and A. Hussain, "Feature Extraction Technique using Discrete Wavelet Transform for Image Classification," in *Proc. Student Conf Research and Development*, Dec. 11-12, 2007.
[15]	S. Hao, "Image Features Extraction for Multimedia Database Content Description," in *Proc. Int. Conf Comp. and Info. Sci.*, 2006.
[16]	M. Z. Hossain, M. A. Amin and H. Yan, "Rapid Feature Extraction for Optical Character Recognition," North South Univ., Dept. of Comp. Sci., Technical Report, June 1, 2012.
[17]	A. L. Koerich, A. S. Britto, L. E. S. de Oliveira and R. Sabourin, "Fusing High- and Low-Level Features for Handwritten Word Recognition," Potential Catholic University of Parana, Technical Report.
[18]	O. M. Kruse, "Feature extraction techniques to use in cereal classification," Norwegian Univ. of Life Sciences, Dept. of Mathematical Sci. and Tech., Technical Report.
[19]	S. Lahmiri and M. Boukadoum, "Hybrid Discrete Wavelet Transform and Gabor Filter Banks Processing for Mammogram Features Extraction," *IEEE*, 2011.
[20]	A. Lawgali, A. Bouridane, M. Angelova and Z. Ghassemlooy, "Handwritten Arabic Character Recognition: Which Feature Extraction Method?," *Int. Journal Advanced Sci. and Tech.*, vol. 34, Sept. 2011.
[21]	C. Lee and D. Landgrebe, "Feature Extraction and Classification Algorithms for High Dimensional Data," Purde Univ., School of Elect. Engineering, Technical Report, January, 1993.
[22]	G. Liu, X. Wen, W. Zheng and P. He, "Shot Boundary Detection and Keyframe Extraction based on Scale Invariant Feature Transform," in *Proc. Eighth IEEE/ACIS Int. Conf Comp. and Info. Sci.*, 2009.
[23]	W. Liu and Y. Wang, "Expression Feature Extraction Based on Difference of Local

`

	Binary Pattern Histogram Sequences," in *Proc. ICSP*, 2008.
[24]	K. Markov and T. Matsui, "High level feature extraction for the self-taught learning algorithm," Univ. of Aizu, Dept. of Info. Systems, Technical Report, 2013.
[25]	J. C. B. Melo, G. D. C. Cavalcanti and K. S. Guimaraes, "PCA Feature Extraction for Protein Structure Prediction," *IEEE*, 2003.
[26]	M. Nixon and A. Aguado, *Feature Extraction and Image Processing*, 2nd ed. Hungary: Academic Press, 2008.
[27]	R. Polikar, "The Wavelet Tutorial," Rowan Univ., College of Engineering Web Servers, Technical Report, Jan. 21, 2001.
[28]	Y. Rui, T. Huang and S. Chang, "Image Retrieval: Current Techniques, Promising Directions, and Open Issues," *Journal Visual Communication and Image Representation*.
[29]	H. Ruohong, P. Yun and M. Keji, "SAR Image Target Recognition Based on NMF Feature Extraction and Bayesian Decision Fusion," in *Proc. Second IITA Int. Conf Geoscience and Remote Sensing*, 2010.
[30]	S. Saha, N. Paul, S. K. Das and S. Kundu, "Optical Character Recognition using 40-point Feature Extraction and Artificial Neural Network," *Int. Journal Advanced Research in Comp. Sci. and Software Engineering*, vol. 3, no. 4, April 2013.
[31]	I. Siddiqi and N. Vincent, "A Set of Chain Code Based Features for Writer Recognition," in *Proc. Int. Conf Document Analysis and Recognition*, 2009.
[32]	P. Singh and S. Budhiraja, "Feature Extraction and Classification Techniques in O.C.R. Systems for Handwritten Gurmukhi Script – A Survey," *Int. Journal of Engineering Research and Applications*, vol. 1, no. 4, pp. 1736-1739.
[33]	V. Singh, B. Kumar and T. Patnaik, "Feature Extraction Techniques for Handwritten Text in Various Scripts: a Survey," *Int. Journal Soft Computing and Engineering*, vol. 3, no. 1, March 2013.
[34]	H. Soliman, M. M. El-gayar and N. Meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informatics Journal*, vol. 14, pp. 175-181, 2013.
[35]	Y. Sun, C. Zhang, P. Liu and H. Zhu, "Shape Feature Extraction of Fruit Image based

`

	on Chain Code," in *Proc. Int. Conf Wavelet Analysis and Pattern Recognition*, Nov. 2-4, 2007.
[36]	I. Supriana and A. Nasution, "Arabic Character Recognition System Development," in *Proc. Fourth Int. Conf Electrical Engineering and Informatics*, pp. 3324-341, 2013.
[37]	J. Thorsten, *Learning to Classify Text Using Support Vector Machines Methods, Theory and Algorithm*, 1st ed. Springer, 2002.
[38]	D. P. Tian, "A Review on Image Feature Extraction and Representation Techniques," *Int. Journal Multimedia and Ubiquitous Engineering*, vol. 8, no. 4, July 2013.
[39]	R. Tokas and A. Bhadu, "A Comparative Analysis of Feature Extraction Techniques for Handwritten Character Recognition," *Int. journal Advanced Tech. and Engineering Research*, vol. 2, no. 4, July 2012.
[40]	Q. D. Trier, A. K. Jain and T. Taxt, "Feature Extraction Methods for Character Recognition-A survey," *Journal Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.
[41]	T. Tuytelaars and K. Mikolajczyk, "Local Invariant Feature Detectors: A Survey," *Journal Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177-280, 2007.
[42]	G. Vamvakas, B. Gatos and S. J. Perantonis, "A Novel Feature Extraction and Classification Methodology for the Recognition of Historical Documents," in *Proc. Tenth Int. Conf Document Analysis and Recognition*, 2009.
[43]	R. Verma and J. Ali, "A-Survey of Feature Extraction and Classification Techniques in OCR Systems," *Int. Journal Computer Applications and Info. Tech.*, vol. 1, no. 3, Nov. 2012.
[44]	Y. Wan, J. Du, D. Huang and Z. Chi, "Bark Texture Feature Extraction based on Statistical Texture Analysis," in *Proc. Int. Symposium on Intelligent Multimedia, Video and Speech Processing*, Oct. 20-22, 2004.
[45]	"Watch parameters for lib-svm," www.csie.ntu.edu.tw/~r94100/libsvm-2.8/README
[46]	"Wavelet Families," http://www.mathworks.in/help/wavelet/ref/waveletfamilies.html
[47]	H. Yang, J. Chen and Y. Zhou, "Similarity Measures of Sectional Contour based Surface Feature Extraction from Point Clouds," in *Proc. Ninth Int. Conf. Electrnic*

`

	Measurement and Instruments, 2009.
[48]	M. Zahedi and S. Eslami, "Farsi/Arabic Optical Font Recognition Using SIFT Features," Shahrood Univ. of Tech., School of Comp. Engineering and Info. Tech., Technical Report WCIT, 2010.