HAND-WRITTEN DEVNAGARI CHARACTER RECOGNITION

A THESIS SUBMITTED IN PARTIAL FUFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND INSTRUMENTATION ENGINERING
BY

SWAMY SARAN ATUL
SWAPNEEL PRASANTH MISHRA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

2007

HAND-WRITTEN DEVNAGARI CHARACTER RECOGNITION

A THESIS SUBMITTED IN PARTIAL FUFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND INSTRUMENTATION ENGINERING
BY

SWAMY SARAN ATUL
SWAPNEEL PRASANTH MISHRA

UNDER THE GUIDANCE OF: Dr SUKADEV MEHER

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

2007

National Institute of technology

ROURKELA

CERTIFICATE

This is to certify that the thesis entitled , “ Hand Written Devnagari character recognition” , submitted by Sri Swamy Saran Atul and Sri Swapneel Prasanth Mishra in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in ‘ ELECTRONICS AND INSTRUMENTATION ‘ Engineering at the National Institute of Technology , Rourkela (Deemed University) is an authentic work carried out by him under my supervision.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university / institute for the award of any Degree or Diploma.

Date

Dr. Sukadev Meher
Dept. of Electronics and Communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENT

Under the esteem guidance of our project guide Dr SUKADEV MEHER detail study of HAND WRITTEN DEVANAGARI CHARACTER RECOGNITION and its applications to various models has been studied as well as simulation has been done. We are very thankful for his whole-hearted co-operation without which this project could not have been completed.

CONTENTS

Abstract

Chapter 1: INTRODUCTION

Chapter 2: OBJECTIVE OF THE PROJECT

Chapter 3: OVERVIEW OF ALGORITHM

Chapter 4: IMPLEMENTATION DETAILS

Chapter 5: RESULTS

Chapter 6: CONLUSION

Chapter 7: REFERENCES

CHAPTER 1

INTRODUCTION

INTRODUCTION

Artificial Intelligence, giving machines the human like abilities, has remained one of the most challenging areas in Electronic sciences in last few decades. Giving machine the power to see, interpret and the ability to read text is one of the major tasks of AI.A lot of work has been done in this field, but still the problem is not solved in its full complexity. A good text recognizer has many commercial and practical applications, e.g. from searching data in scanned book to automation of any organization, like post office, which involve manual task of interpreting text.

The problem of text recognition has been attempted by many different approaches. Template matching is one of the simplest approaches. In this many templates of each word are maintained for a input image, error or difference with each template is computed. The symbol corresponding to minimum error is output. The technique works effectively for recognition of standard fonts, but gives poor performance with hand written characters. Feature extraction is another approach, in which statistical distribution of points is analyzed and orthogonal properties extracted. For each symbol a feature vector is calculated and stored in database. And recognition is done by finding distance of feature vector of input image to that of stored in the database, and outputting the symbol with minimum deviation. Though this technique gives lot better results on handwritten characters ,but is very sensitive to noise and edge thickness. Features extracted in last approach tend to be very mathematical (most of them are results of some transforms etc) and mostly cannot be interpreted by simple logic. In geometric approach, on the other hand, attempt is made to extract features that are quite explicit and can be very easily interpreted. These features depend on physical properties, such as number of joints, relative positions, number of end points, length to width ratio etc. Classes forced on basis of these geometric features are quite distinct with not much overlapping. The main draw back of this approach however is that this approach depends heavily on the character set, Hindi alphabets or English alphabets or Numbers etc. Features extracted for one set are very unlikely to work for other character set. In contrast to this ,Neural networks offer complete independence of recognition process and character set. In this neural network is first trained by multiple sample images of each alphabets .Then in recognition process ,the input image is directly given to neural network and recognized symbol is outputted.

The advantage of neural networks is that domain of the character set can be very easily extended,one just needs to train the network over new set.Another advantage is that neural networks are very robust to noise.The disadvantage is that a lot of training is required which is very time consuming .None of the above approaches thus used in their pure form yields good result on handwritten text.So generally hybrid approaches are taken to achieve desired recognition rates.

Although even best of the recognition approaches are able to recognize all those texts that human can.Major reason for this has been wide variation in writing practices and styles of different people and lot of context based information that human brain uses to interpret any text sample.

CHAPTER 2
OBJECTIVE OF THE PROJECT

OBJECTIVE OF THE PROJECT

The objective of the project is to develop an OCR for Hindi characters .The input to the system would be a pure hindi text and output would be a recognized Hindi characters .
Following were the main objectives of the project:

2.1 Recognition of only Devnagari Script alphabets

The input image should contain only and only Devnagari script alphabets .The reason for this constraint is that the algorithm is based on geometrical and topological properties and distributional properties of the varius hindi characters.

2.2 Characters or Matras should not be overlapping

The characters should not be touching each other .Though segmentation of touching characters has been one of the toughest jobs in text recognition and this alone has remained one of the toughest area of research.

2.3 Inputs image should not contain ardhakshers

Ardhakshers were excluded from the recognition domain because of the following two main reasons

- There is a huge variation in writing styles of writing ardhakshers.Relative positions of ardhakshers, such as ‘l’ and ‘v’ with respect to the following character vary drastically .So the identification of these ardhakshers become quite difficult job.
- In addition to uncetainity in the relative positon of ardhakshers,most of these are written by joining them to the alphabet following them and it becomes difficult to segment them out. Since the problem of segmentation of characters is not beng addressed in this project,this assumption was taken.

2.4 Text lines should be almost straight

The sentences inputed to the OCR should not have large slopes .They should be almost straight.Though the system is robust enough to handle lines with slopes of 10-20 degrees.

2.5 Image should be noise free

Another assumption about the input image of Hindi text should be noise free .This assumption does not reduce the complexity of the problem as this is just a part of preprocessing module.
A noisy image can be made noise free by applying standard functions and techniques.Thes were skipped due to time constraints involved.

CHAPTER 3

OVERVIEW OF ALGORITHM

OVERVIEW OF ALGORITHM

The algorithm is based on geometrical features of hindi characters .Input image is parsed into many subimages based on these features .Then other properties such as distribution of points /pixels and edges within each subimages are feature used to recognize parsd symbol.

Two major properties used to segment input word (image) into various sub symbols are as follows:

- Horizontal Bars
- Vertical Bars

Both Horizontal and vertical bars form an integral part of most Hindi characters .

These are exploited as follows:

3.1 Exploiting Horizontal Bars

This bar separates out the matra zone and the consonant zone .Removal of this bar has following advantages:

- Separation of matra zone and consonant zone
- Segmentation of characters In a word

Since the horizontal bar separates out the upper zone or the matra zone of words ,its removal leads to segmentation of word into two subimages ,matra zone image and consonant zone image. Now both of these images can be recognized using special properties of symbols in each of these zones.

Removal of Horizontal bar also facilitates the segmentation of characters . Since the assumption was that the characters are not touching each other ,once the horizontal bar is removed all the characters get separated by the spaces ,and hence can be easily extracted out.

3.2 Exploiting Vertical Bars

Vertical bar is also an integral part of most of the hindi characters. Removal of this bar has following advantage:

- Reduce the complexity of character image
- In dividing the possible symbols into various classes

Since this bar occurs so frequently in hindi characters ,this bar can be easily removed .This would significantly reduce the complexity of character image .

Another observation is that there are following types of hindi characters:

- Characters with vertical bar some where in the middle of the character
- Characters with vertical bar at the right side of the character
- Characters with no vertical bar

Once we remove vertical bars ,we can derive another folowng three classes of symbols:

- Symbols that occur on the left of the vertical bar in first two classes defined above -class I
- Symbols that occur on the right side of vertical bar in the second class defined above – class II
- Characters without vertical bar –class III

In the above classes ,symbols are resultant sub symbols after removal of horizontal and vertical bar .one may note that these symbols are not same as original Hindi characters ,these are lot simpler symbols. A hindi alphabet ,such as 'K',is divided into two sub symbols ,one of which is in class I and other in class II. Symbols in each of these classes can be recognized with relative case, by using their geometric properties.

3.3 Overview of the algorithm

- Detect Horizontal Bar
- Remove Horizontal Bar, hence divide image into two subimages-matra zone and consonant zone
- Segment out isolated symbols in both sub images
- Crop all isolated symbols in both sub-images
- In the image corresponding to consonant zone
  - detect Vertical Bar
  - remove Vertical Bar ,hence divide image into sub-images,each belonging to classI ,II or III,defined in previous subselection
- For each symbol extracted from matra zone
  -thin the symbol image
  -identify thids symbol with possible set of symbols
- Compute following attributes ,and generate a property vector for each symbol extracted
  -class of the symbol image ,e.g. the top,left right and bottom sections of the image
  -Centre of gravity of the symbol image
  - 4 zone classification of the sub image on left, giving the distribution of pixels in each zone ,this forms the basis for another type of classification.
- Using the above attributes ,find the set of candidate symbols with similar characteristics
- Resolve in the candidate symbols using Heuristics
  -analyze subsection of image, compute slopes of each point on the image from the Center of gravity.
  -compute the distance between the obtained slopes with that of the slopes of the Templates, the template giving minimum distance gives the subsymbol
- Integrating the sub symbols so identified the letter is reconstructed

CHAPTER 4

IMPLEMENTATION DETAILS

IMPLEMENTATION DETAILS

Following are the implementation details of the various steps in the algorithm.

4.1 Input to the OCR

The implemented OCR expects the input image to be in .png format. The image should be a binary one. The text should be written with black color and the background should be white. That is, the image should have only two types of pixel values, 1, for background(white) and 1, for the foreground(black).

4.2 Binarization

For testing purposes, the letters were written on 200x200 size background using MS Paint. Since writing using a mouse itself is a relatively difficult task, the letters appear skewed. This provides an excellent substrate for testing the algorithm. The images stored via Paint are RGB images. They have to be converted to Black and White image with thresholding at a value of 0.5 to get a true binary image.

4.3 Inversion

The binary image consists of a black foreground in front of a large white background. The number of pixels in the background far exceeds that of those in the foreground. This means the numbers of 1s will always be atleast 10 times the number of 0s. Now, conventionally, we prefer to work with 1s and leave the 0s aside. Moreover, smaller number 1s will mean lesser calculations in correlation. Hence the image is inverted such that the background is black and the foreground is white, just like the appearance of the familiar DOS screen. This is done by subtracting the binary image from a matrix of 1s of the same size.

4.4 Thinning the Image

Since the algorithm is based on the geometrical and structural properties of the Hindi characters, we thin the image to single-pixel width so the contours are brought out more vividly. In this way, the attributes to be studied later will not be affected by the uneven thickness of edges or lines in the symbol. Thinning is a morphological operation that is used to remove selected foreground pixels from binary images. The key here is the selection of the right pixels. For this, we classify the pixels into three categories:

Critical Pixels – Pixels whose removal damages the connectivity of the image. Any pixel which is the lone link between a boundary pixel and the rest image is a Critical Pixel. It's removal will isolate the boundary pixel. Hence it should not be removed.

End Pixels – Pixels whose removal shortens the length of the image. An end pixel is connected to two or less pixels. Remember that we are talking about 8-connectivity here. Different considerations have to be taken for 4-connectivity.

Simple Pixels – Pixels which are neither Critical nor End pixels. These are the ones that can be removed for thinning.

A smarter method than manually looking for suitable pixels is to use the Hit And Miss transform. Like other morphological operators, the behavior of the thinning operation is determined by a Structuring Element. The binary structuring elements used for thinning are of the extended type described under the hit-and-miss transform(*i.e.* they can contain both ones and zeros).

The thinning operation is related to the hit-and-miss transform and can be expressed quite simply in terms of it. The thinning of an image *I* by a structuring element ***J*** is:

$$\mathbf{thin}(I, J) = I - \mathbf{hit\text{-}and\text{-}miss}(I, J)$$

where the subtraction is a logical subtraction defined by

$$X - Y = X \cap \mathrm{NOT}\, Y$$

In everyday terms, the thinning operation is calculated by translating the origin of the structuring element to each possible pixel position in the image, and at each such position comparing it with the underlying image pixels. If the foreground and background pixels in the structuring element exactly match foreground and background pixels in the image, then the image pixel underneath the origin of the structuring element is set to background (zero). Otherwise it is left unchanged. Note that the structuring element must always have a one or a blank at its origin if it is to have any effect.

The choice of structuring element determines under what situations a foreground pixel will be set to background, and hence it determines the application for the thinning operation. In fact, the operator is normally applied repeatedly until it causes no further changes to the image (*i.e.* until convergence).

HIT AND MISS TRANSFORM

The hit-and-miss transform is a general binary morphological operation that can be used to look for particular patterns of foreground and background pixels in an image. It is actually the basic operation of binary morphology since almost all the other binary morphological operators can be derived from it. As with other binary morphological operators it takes as input a binary image and a structuring element, and produces another binary image as output.

The hit-and-miss operation is performed in much the same way as other morphological operators, by translating the origin of the structuring element to all points in the image, and then comparing the structuring element with the underlying image pixels. If the foreground and background pixels in the structuring element exactly match foreground and background pixels in the image, then the pixel underneath the origin of the structuring element is set to the foreground color. If it doesn't match, then that pixel is set to the background color.

For instance, the structuring element shown in the figure below can be used to find right angle convex corner points in images.

×	1	×
0	$\underline{1}$	1
0	0	×

Notice that the pixels in the element form the shape of a bottom-left convex corner. We assume that the origin of the element is at the center of the 3×3 element. In order to find all the corners in a binary image we need to run the hit-and-miss transform four times with four different elements representing the four kinds of right angle corners found in binary images. The next figure shows the four different elements used in this operation.

X	1	X
0	1	1
0	0	X

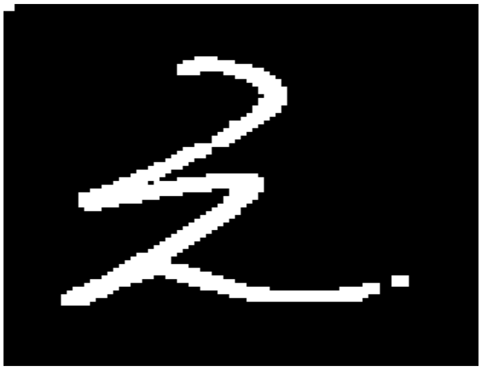
X	1	X
1	1	0
X	0	0

X	0	0
1	1	0
X	1	X

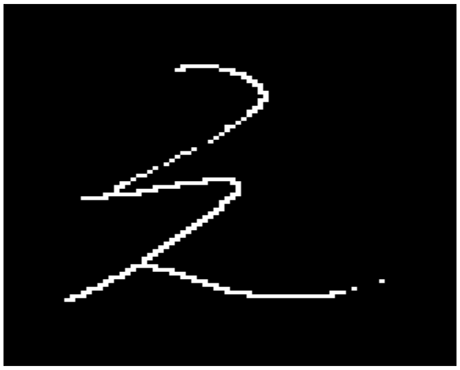
0	0	X
0	1	1
X	1	X

As an illustration of the thinning procedure, we look at the following figure:

Before Thinning



After Thinning



4.5 Detection of The Horizontal Bar

Since the head-line is an integral part of all Hindi alphabets, it can be used for segmentation of the letter. The horizontal histogram of the image is taken, counting the number of 1s in each row. The maxima indicates the position of the horizontal bar. This works under the assumption that the horizontal bar IS sufficiently long.

When the horizontal bar is inclined, it does not give a distinct maxima. To overcome such cases, for any row i, we always take the average of the number of 1s for the i-1th, ith and the i+1th rows. This smoothes out the summation and gives a clearer picture of where the maxima lies. Once the maxima is detected, the part of the image above that row is clipped off and stored. This portion will come in handy when we study the upper *matra*s and the characteristics of letters *bha, tha* and *dha.*

4.6 Detection of The Vertical bar

The vertical bar is found in a majority of the Devnagari alphabet. The method of detection is similar to that for the Horizontal Bar. A Vertical Histogram of the binary image is taken, and the maxima gives the position of the Vertical Bar.

The case of the inclined Vertical bar is tackled in the same way as was done for the Horizontal bar. However, another complexity in this case is the possible absence of the Vertical Bar itself. There are letters that do not contain a vertical bar, viz, *da, ha* etc. If we go purely by the maxima, the local maxima of these letters will also indicate the presence of a vertical bar. Hence, a check is put for such cases. After locating the maxima, we check if it is atleast 75% of the total image height, taking in to account the inclined vertical bars. If the value at maxima is less than this threshold, it is not taken as vertical bar. Then the letter is taken be one without a vertical bar.

After detection, the regions to the left and right of the vertical bar are separated out. Moreover, since we already have the height of the letter, we separate out the area below the height to get the lower *matra* zone. All these zones help in classification of the image into possible groups.

4.7 Classification of Image

The emphasis of the project is more on the classification of the image than the recognition itself. There are poignant reasons behind this line of thinking. There are many possible approaches that can be used to recognize each individual letter. The technique will, of course, be that of pattern recognition and matching. We may use simple correlation, attribute vectors, neural networks, or transform the image to a new domain and compare the coefficients in that domain. While there are many approaches possible, to attack the image up-front with a recognition procedure will be hara-kiri. There are 44 letter in the Devnagari character set. That means, for each input, we have to compare it to atleast 44 standard characters. If we take a greater number of standard templates, which IS the norm, for greater accuracy, the processing time increases manifold.

An optimization to the direct-hit approach will be if the number of standard templates, with which the image is to be compared, is relatively less. We should be somehow able to reduce the probable number of characters that will match with the input image. The image can be studied during the pre-processing stage itself to zero down to a fewer number of possible candidates. The 44 letters of the Devnagari script are so varied in their structural attributes that they can be easily classified into certain distinct classes. The input image can then be studied for clues which assign it to a particular class in the scheme of the algorithm.

The figure shown in the following page is a list of all the Hindi characters in present usage. It can be seen that the geometries of some characters are similar, while being diametrically opposite to other characters. For instance, some letters like *ka* are characterized by the presence of the Vertical Bar, while some, like *da* by the absence of it. This may be defined as a broad category of classification, in which characters are broken down into two sets, those with Vertical Bar and those without Vertical bar. As will be shown in the next sub-section, we further divide the letters into more classes depending on their structural attributes.

As a result of this classification, the probable matching outputs for the input image is reduced to smaller set of characters, mostly to three or four. The pattern matching and recognition technique can be applied to these three-four probable candidates to zero down to the most probable match. This approach shifts the focus from the recognition technique, which is an active field of research, to classification of the image. In this particular case, it results in an efficient system design even if the recognition algorithm may not be very robust.

The Devnagari Script

Roman	Devanāgarī	Roman	Devanāgarī
a	अ	ṭh	ठ
ā	आ	ḍ	ड
i	इ	ḍh	ढ
ī	ई	ṇ	ण
u	उ	t	त
ū	ऊ	th	थ
ṛ	ऋ	d	द
e	ए	dh	ध
ai	ऐ	n	न
o	ओ	p	प
au	औ	ph	फ
ṁ (anusvāra)	ं	b	ब
ḥ (visarga)	ः	bh	भ
k	क	m	म
kh	ख	y	य
g	ग	r	र
gh	घ	i	ल
ṅ	ङ	v	व
c	च	ś	श
ch	छ	ṣ	ष
j	ज	s	स
jh	झ	h	ह
ñ	ञ	jñ	ज्ञ
ṭ	ट		

The visual scrutiny of the characters in the figure led us to define certain image attributes that may be used to classify the Devnagari characters into broad categories:

- Presence or absence of Vertical Bar.
- Presence or absence of curves on the right side of the Vertical Bar.
- Density of foreground pixels in upper, middle or lower parts of the image.
- Nature of the curve to the left of the Vertical Bar, i.e., it is closed or not, which side of the curve is open, etc.

4.7.1 Stage I Classification

The first categorization leads to well compartmentalized classification of characters with or without the Vertical bar. Those without the vertical bar will be studied later. The characters with the vertical bar are of more interest. We notice that only *ka* and *fa* have curves on either side of the vertical bar. These two are separated out in this account. This is followed by the recognition procedure, in which the curve to the left of the vertical bar is studied to make out whether it is *ka* or *fa*.

4.7.2 Stage II Classification

For the next classification, we study the contour properties of the image to the left of the vertical bar. We have already separated out the left part of the image, let's call it L.

- First, we find out the centroid of the curve in L. This is done by taking the average of all the x(or y) coordinates of pixels that are valued 1.
- The whole of L is divided into four quadrants, with the centroid as the origin. This similar to the quadrants in general mathematics, except that here the origin will be different for different characters, depending on the location of the centroid.
- The number of foreground pixels in each quadrant is calculated. This sum is stored in a 2x2 matrix.
- There will characters in which one of these quadrants will be empty. For example, the letters *pa, ya, tha, ma* and *bha* will have very low density of pixels in their upper-right

quadrant. Similarly, *ta, la* and *na* will have negligible foreground-pixel density in their lower right quadrant. These categories are marked.

- Now that there are less than five characters in these categories, the suitable recognition techniques may be applied to find the best possible match. There's no need to check for the other letters if a match is found here.

4.7.3 Stage III Classification

Letters which do not fall in the aforementioned categories have to be analyzed in greater detail. This is done by dividing L into smaller zones and analyzing them for clues.

- L is divided into 9 uniform zones, i.e., a uniform grid of 3-by-3. This will be studied for pixel density. To remove redundant parts of L, the portion to the left of the first foreground-pixel containing column in L is clipped off. Similarly, the portion below the last foreground-pixel containing row is also clipped off.
- There are letters in which the upper three zones will be less dense in terms of foreground pixels. Examples are *ja, cha va, ba* and *gya*. These fall under one category and can be identified separately.
- There are letters in which the lower three zones will be less dense in terms of foreground pixels. Examples are *nda* and *ga*. These two can be easily identified because of large difference in curvature.

4.7.4 Stage IV Classification

Letters which have uniform distribution of foreground pixels around the centroid and in all of the nine zones are filtered out into the last category of those possessing the vertical bar. This final category can not be further broken down. But since the number of possible characters at this stage has already been reduced to six, we do not go for further classification. Instead, we directly try to identify these characters using the recognition technique. The characters in this category are *dha, gha, jha, kha*, *talabya sha* and *murdhanya sha*.

4.7.5 Stage V Classification

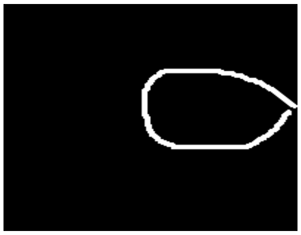
The final category of characters has already been separated out at the start, i.e., those without vertical bar. The letters in this category are *chha, da, dda, ddha, tta, ttha, ra* and *ha*. Of these, da, ddha and tta are very similar. They can be classified into one class, but that will be of not much use because, firstly, the basis of classification is mathematically complicated, and secondly, they can always be easily separated from the other characters even without this classification. Hence, further classification is discarded. Instead, we straightaway go for identification of these eight characters. The results here are not very promising due to close resemblance of the characters. The identification efficiency will be very low. The focus is now on optimizing the recognition algorithm, which is left as a further scope for improvement.

4.8 Identification of Characters

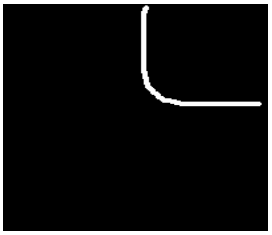
The method adopted for recognition is a rather simplistic approach to image analysis. Since the number of probable candidate has already been reduced to a small value, the recognition algorithm need not be a very complex one. As in the case of Stage I classified characters ka and fa, we only have to analyze the left part of the letters to determine their identity. We compare the attributes of the portion to the left of the vertical bar to a two standard templates, one representing the left portion of ka and the other representing the left portion of fa. The one that gives a better match is selected. To take into account the variations possible, five standard templates of each character are taken, and their mean is calculated to give the ideal template.

Shown below are the standard templates for *ka* and *fa* :

Left of Ka



Left of Fa



Similar standard templates of letters are used for each character. Now we introduce the recognition technique.

4.8.1 Slope Error Technique

The Slope Error technique takes into account the variations in the curve of the foreground with respect to a reference point. The reference point is taken to be the centroid of the curve. The slope of each foreground pixel in the curve is calculated with respect to the centroid. Since the slope is a directional property, it is independent of the image size. Hence, this method may be used to compare two similar curves of different sizes. The procedure is, thus, size-proof.

The slope m(i) of the i^{th} pixel is given by:

$$\frac{y(i) - y(c)}{x(i) - x(c)}$$

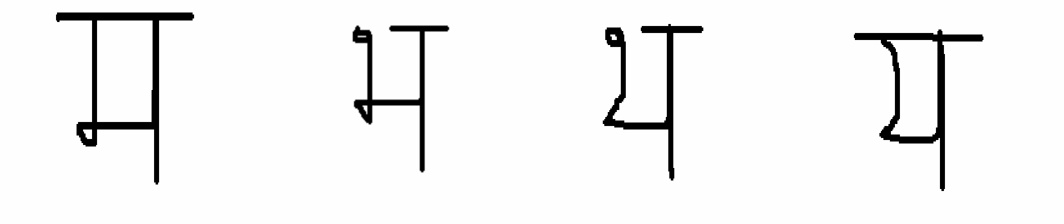
When x(i) = x(c), we assign a large value like 1000 to m(i). Since the slopes here may be negative or positive, we convert them to radians by using the formula

$$m(i) = \begin{cases} atan(m(i)) & \text{for } m(i) > 0 \\ pie + atan(m(i)) & \text{for } m(i) < 0 \end{cases}$$

This slope vector m contains the slopes of all the points in the curve, which is a characteristic of the curve. Similarly, the slope vector m1 of the standard template is also calculated. Since the images are of different sizes, the number points will differ, so will the size of the slope vectors. We equalize the slope vectors by using the MATLAB function imresize which extrapolates the smaller vector to the size of the larger one while retaining the characteristic of the curve.

The Slope Error is calculated by simply subtracting the vector m1 from vector m, resulting in the Error Vector ‘err’. The Error vector is normalized with respect to its largest value and then it’s mean is calculated. This value will be different for different curves. The minimum error values correspond to the best match. Using a switch-case construct, the corresponding matching standard letter is displayed.

The above method does have a shortcoming though. For letter which are similar in structure, variations in the hand-written input may give the wrong identification. The letters da, tta and ddha, for example are often confused with each other. Moreover, if the top portion of bha is not detailed, it may be confused with ma or tha. Similarly dha and gha may be confused. This similarity between letters is illustrated in the following figure :



The confusion is countered by detailed analysis of overlapping categories. Whenever the error percentage is very small as compared to the maximum error, the image is again studied for further clarification. The following characteristics are used to identify various similar characters:

- *bha* and *tha* are separated from *ma* and *ya* using the discontinuity in their horizontal bar.
- Similarly, *dha* and *gha* are separated.
- The closed curve at the bottom of *ma* is detected using histograms, which separates it from *tha*.
- Similarly, *ma* and *ya* are separated.
- *tta* and *da* are distinguished by the presence of loop at the end of *da.*

These methods are elementary methods and may not be very efficient. Better algorithms may be designed to distinguish such similar characters.

CHAPTER 5

RESULTS

RESULTS

The program was rigorously tested on approximately sample images ,hand written on Microsoft Paint ,interface.Since the samples were hand written the experimental results provide a good estimate of the performance of the program .

An analysis of the results have been tabulated as below. Each of the characters were tested for 10 samples.

INPUT CHARACTER IMAGE	% SUCCESS IN RECOGNITION
KA	90
TA	60
NA	80
LA	50
PA	40
FA	80
MA	70
BHA	80
THA	70
DHA	40
GHA	60
KHA	50

CHAPTER 6

CONCLUSION

CONCLUSION

In this paper, we have described a system for OCR of printed Hindi script material. The recognition accuracy of the prototype implementation is promising, but more work needs to be done. In particular, no fine-tuning of the system has been done so far. Our character segmentation method also needs to be improved so that it can handle a larger variety of touching characters, which occur fairly often in images obtained from inferior-quality printed material. In general, the system needs to be tested on a wider variety of images containing characters in diverse fonts and sizes. This will enable us to identify the major weaknesses in the system and implement remedies for them.

6.1 SCOPE OF IMPROVEMENT

- Segmentation of characters can be improved to include overlapping and joined characters . New algorithms can be developed to help the segmentation of joined hand written letters.
- The current thinning algorithm used produces short spurs which make computation of various attributes like pixel densities ,or checking continuity of an image at times faulty and corruptive, though reverse thinning or dilation has been done as when required ,an improved thinning algorithm can be developed to counter the problem of short spurs.
- The algorithm used in this project though very efficient in classifying an image, but lacks pinpoint accuracy in identifying a singular perfect match. In order to do so the algorithm has to depend on the distributional properties of the image, this requires a rigorous analysis and study of particular image and still rigorous programming .This results at times the thresholds and limiting conditions being set by hit and trial approach, this can be avoided by more experimentation and further careful study of complex characters

REFERENCES

1. Image processing in C By Dwayne Phillips (BPB Publications)
2. Digital Image Processing By Gonzalez and Woods (Prentice Hall)
3. DIP and Analysis By B.Chandra and D.Dutta Majumdar
4. Hyper Media Image Processing Reference (HIPR) By Department of Artificial Intelligence in the University of Edinbourgh