Fuzzy Stroke Analysis of Devnagari Handwritten Characters

PRACHI MUKHERJI[1], PRITI P. REGE[2]

[1]Electronics and Telecommunication Department,
Smt. Kashibai Navle College of Engg., Pune 411041,
INDIA
[2]Electronics and Telecommunication Department,
College of Engineering Pune, Shivajinagar, Pune, 411005.
INDIA
prachimukherji@rediffmail.com, ppr@coep.extc

Abstract: - Devnagari script is a major script of India widely used for various languages. In this work, we propose a fuzzy stroke-based technique for analyzing handwritten Devnagari characters. After preprocessing, the character is segmented in strokes using our thinning and segmentation algorithm. We propose Average Compressed Direction Codes (ACDC) for shape description of segmented strokes. The strokes are classified as left curve, right curve, horizontal stroke, vertical stroke and slanted lines etc. We assign fuzzy weight to the strokes according to their circularity to find similarity between over segmented strokes and model strokes. The character is divided into nine zones and the occurrences of strokes in each zone and combinations of zones are found to contribute to Zonal Stroke Frequency (ZSF) and Regional Stroke Frequency (RSF) respectively. The classification space is partitioned on the basis of number of strokes, Zonal Stroke Frequency and Regional Stroke Frequency. The knowledge of script grammar is applied to classify characters using features like ACDC based stroke shape, relative strength, circularity and relative area. Euclidean distance classifier is applied for unordered stroke matching. The system tolerates slant of about 10^o left and right and a skew of 5^o up and down. The system proves to be fast and efficient with regard to space and time and gives high discrimination between similar characters and gives a recognition accuracy of 92.8%.

Key-Words: - Devnagari Script, Segmentation, Strokes, Average Compressed Direction Code, Zonal and Regional Stroke Frequency, Euclidean Classifier.

1 Introduction

Over 500 million people all over the world use Devnagari Script. It provides written form to over forty languages [1] including Hindi, Konkani and Marathi. It is a logical composition of its constituent symbols in two dimensions [2]. It has a horizontal line drawn on top of all characters [2]. The character set consists of 35 consonants and 13 vowels. A marked distinction in Devnagari script from those scripts of Roman genre is the fact that a character represents a syllabic sound, complete in itself. There has been intense research work done on the English, Latin, Chinese, Persian, Tamil, and Bangla scripts on both handwritten and machine printed texts. There are two main approaches for feature extraction of handwritten characters: statistical and structural approach. Structural approach has been successfully used for printed [3] and broken [4] Thai character recognition. While most work has been published for printed Devnagari text, very little is reported for handwritten Devnagari script. One of the first attempts for handprinted characters has been by Sethi [5] and for typed Devnagari script by Sinha and Mahabala [6]. V. Bansal and Sinha in [7] divided the typed word in three strips and separated it in top strip, core strip and bottom strip and achieved 93% performance on individual characters. Pal and Chaudhuri have attempted OCR for two scripts, Bangla and Devnagari in [8]. Machine recognition of online handwritten Devnagari characters has been reported in [2] with 82-85% accuracy. In [9], a survey of different structural techniques used for feature extraction in OCR of different scripts and status of other Indian scripts is given. In [10], Devnagari handwritten character recognition is achieved using connected segments and neural network. U. Pal et al in [11] extract directional chain code of contour points of the characters. The length of feature is 64 and they achieved an accuracy of 80.36% using quadratic classifier. More recently in [12], U. Pal et al, extracted a 400 feature vector based on Roberts filter on isolated normalized characters and achieved an accuracy of 94%. They consider characters without top-modifiers.

In this proposed work, attempt has been made to recognize 45 handwritten characters in Devnagari script using a structural approach. This system allows variations that occur in individual handwriting. The handwriting should be legible and adhering to structural syntax of the Devnagari script. The processing steps of our OCR system can be summarized concisely. The documents are scanned and filtered. The filtered images are subjected to binarization as the text is printed as dark points on light background (or vice versa) and can be mapped as a binary image. The characters are extracted from the form sheet and enclosed in a component box. Skeletonization, topline detection and top modifier segmentation are the next steps. Skew removal and slant removal are integral part of any optical character recognition (OCR) system as it increases accuracy by removing the variability of handwritten data. Average Compressed Direction Coding (ACDC) algorithm codes the strokes of the character and suppresses the variations due to slope and skew thereby reducing the need for slope and skew normalization. Thus the algorithm also suppresses shape variations. Cognitive scientists report that humans base their thinking on conceptual patterns and mental images rather than on any numerical quantities [10]. Keeping this view and the construction of Devnagari script, fifteen basic shapes are defined with a fuzzy membership function for better classification. The classification space is partitioned on the basis of number of segments, Zonal and Regional Stroke Frequency (ZSF and RSF). The characters are finally classified using Euclidean classifier on a feature vector for unordered stroke matching. The features used are shape codes, circularity, relative area and relative strength of the segmented strokes. The accuracy of recognition achieved is 92.8%.

The paper is organized as follows: Section 2 gives the characteristics of Devnagari script. Section 3 describes pre-processing of the character, section 4 explains feature extraction and ACDC algorithm in detail. Section 5 explains stroke and character classification. In Section 6 segmentation, feature extraction and recognition results are discussed. In Section 7, conclusions and directions for future research are given

2 The Devnagari Character Set

The basic character set of Devnagri script is of 48 characters. The character set for experimentation is based on the present-day usage and is shown in Fig. 1(a) for typed data. Fig.1 (b) shows the handwritten character set for this work. It consists of 12 vowels (first two lines) and 33 consonants. As can be observed the characters have circular strokes in different directions. This feature demands a specific curve detection to recognize them as set methods of other scripts may fail. Every character has a horizontal header line or the 'shirorekha' as shown in Fig. 2. This line serves as a reference to divide the character into two distinct portions: Head and Body if the top modifier is present. As shown in Fig. 2, a Devnagari word may be divided in three zones. Zone 1 gives top-modifier; Zone 2 the body of the word and Zone 3 is the lower modifier region. The basic character width ranges from very small र to large क्ष going through many medium sizes for typed characters but it is difficult to specify for handwritten data as the size is dependent on individual's stroke, pen width and style of writing. Many characters have a vertibar, which can be present in the middle region or in the end of the character as shown in Fig.3 (a). Some characters do not have vertibar as shown in Fig. 4. In Fig.5, characters from the basic set with top-modifier are shown.

अ आ इ ई	क ख ग घ
उ ऊ ए ऐ	च छ ज झ
ओ औ अं अः	ट ठ ड ढ ण
	त थ द ध न
	प फ ब भ म
	य र ल व श
	ष स ह क्ष ज्ञ

Figure 1(a). Basic Character Set (Typed)

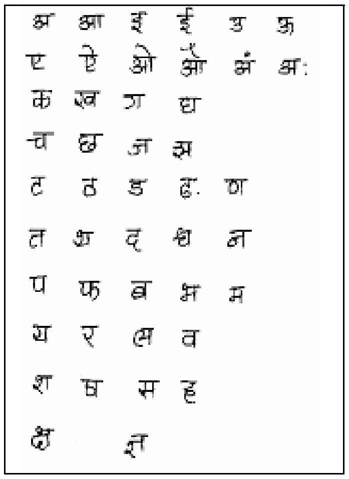


Figure 1(b). Basic Character Set (handwritten)

Another special feature of Devnagari script is conjuncts or structural compositions (*Yuktakshar)* of two or three consonants and/or vowels. Their formation is by simple rules and restrictions of the language of application. Each of the consonant (*Vyanjan)* and the conjuncts (*Yuktakshar)* can be further modified by vowel (*Matra*) modifier. In a manner similar to that of the formation of conjuncts, combinations of vowels, with the nasal sounds, gives rise to combines in vowels also. There are as many characters in Devnagari script as there are syllables in the spoken language. The proposed work is limited to the recognition of 45 characters with top modifiers only. This research work though, is a step towards the recognition of unconstrained Devnagari script recognition.

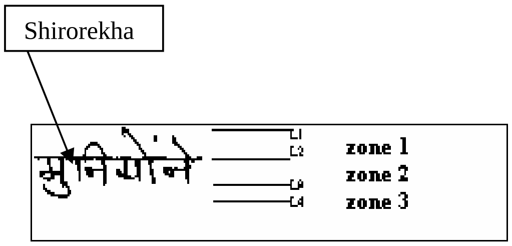


Figure 2. Devnagari word with modifiers and Zones

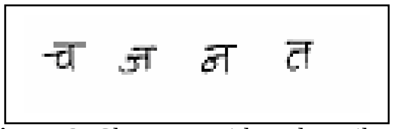


Figure 3. Character with end vertibar

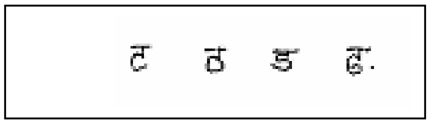


Figure 4. Characters without vertibar

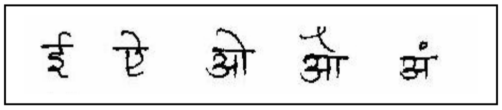


Figure 5. Characters with top modifier

3 Preprocessing

Each character is preprocessed before it can be recognized. Preprocessing includes filtering by Gaussian filter, binarization, and thinning.

Each handwritten document is scanned and converted into a digitized image using a desktop scanner. Noise removal and image smoothing is important to smoothen the strokes. Gaussian low pass filters can be used efficiently for this purpose [14]. The form of this filter in two dimensions is given by (1).

$$H(u,v) = e^{-D^2(u,v)/2D_0^2} \tag{1}$$

where D(u,v) is the distance from the origin of the transform and D_0 is the cut-off frequency.

Binarization is achieved by using Otsu's algorithm [15]. Next step, skeletonization [14] is an important aspect of feature extraction in our scheme. The spurs in the image are removed by the *spur* removal algorithm [15]. The process of cleaning up (or removing) these spurs is called pruning. Slant of a character is detected using a method of slices and applied successfully to Devnagari words in [13]. Keeping the topline as reference horizontal shear transform [14] is applied only if the slant is more than the set threshold of 10°.

4 Feature Extraction

A feature is a point of human interest in an image, a place where something happens. It could be an intersection between two lines, or it could be a corner, or it could be just a dot surrounded by space. These relationships are used for character identification, and hence feature points are exploited for the task of character recognition.

4.1 Top Modifier Features

As shown, in Fig. 2, the distinct feature of Devnagari script as well as individual characters is the topline (Shirorekha) that is detected using Hough Transform [15] and Horizontal Projection [6]. If top modifier is present, it is recognized on the basis of transitions [6] and shadows [16].

4.2 Segmentation of Character in Strokes

An adaptive thinning algorithm has been developed for separating an image in its constituent strokes. In the first step, the complete neighborhood pattern is mapped by finding and summing the contribution of all eight neighbors of a black pixel, Sum_of_Neighbours, $s(i, j)$ as given in (2).

$$\text{Sum\_of\_Neighbours} = s(i, j) = \sum_{i-1}^{i+1}\sum_{j-1}^{j+1} p(i, j) - 1 \tag{2}$$

Since the character image has already passed through skeletonization once, expectedly the maximum neighborhood can be of four pixels signifying a crosspoint as shown in Fig. 6(a). This point is given value zero and then the map is checked for values of $s(i, j)$ equal to three. Blindly thresholding these pixels to zero gives rise to over segmentation.

3	3	3
3	4	3
3	3	3

2	0	2
0	0	0
2	0	2

(a)Cross-point s(i,j) (b)Two Neighbor Cross-point

0	3	0
0	2	3
0	0	0

0	2	0
0	0	2
0	0	0

c) Lower Left Corner d) Two Neighbor corner

2	2	2
2	3	2
3	2	2

1	1	1
1	0	1
0	1	1

(e) Hole s(i,j) (f) Two-Neighbor Hole

Figure 6. Steps in Adaptive Thinning Algorithm.

Instead, the combinations shown in Fig. 6(c) and Fig. 6(d)-(f) are used to detect corners and smoothen them. The pruned map is then thresholded, as in (3).

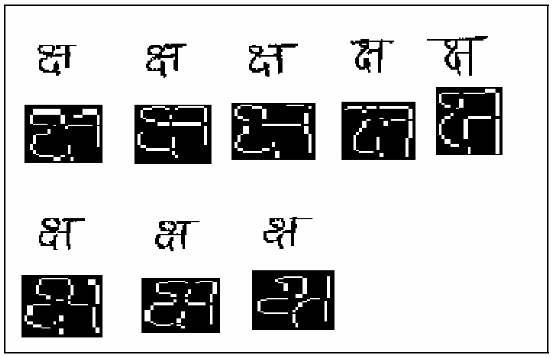


Figure 7. Character 'KSHA' and its strokes.

```
If  $\sum_{i-1}^{i+1}\sum_{j-1}^{j+1} s(i, j) - 1 \ge 3$
(3)
    $s(i, j) = 0;$
else
    $s(i, j) = 1;$
end
```

Fig. 7 gives various samples of character KSHA and their segmentation in strokes. As can be observed, various characters have some basic strokes in common. This knowledge is further exploited in extracting and matching of various strokes

4.3 Average compressed Direction Codes

After segmentation of the character in strokes, all segments are labeled from left to right in the image frame. As each stroke (segment) is labeled [15], its row and column indices are also stored. The next step is coding of these strokes using Freeman's chain codes with slight modification. Normally, Freeman chain codes define a closed boundary and directions are according to the Fig. 8. We use them here to indicate the shape of the strokes only one pixel thick. Each pixel has only two neighbors except the two endpoints. We code from the higher endpoint i.e. the endpoint with lower row index. If row indices are same, coding begins from the left endpoint, i.e. the endpoint with lesser column index. This is done to follow the intuitive left to right and top to bottom construction of Devnagari script.

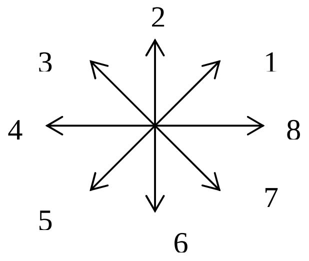


Figure 8. Freeman chain code direction

The obtained code is converted back into angles and averaged depending upon the length of the stroke. Each direction indicates an angle of 45deg. Averaging reduces minute variations in the strokes and reduces the vector space. Care has been taken when combinations of (8,1,2) and (7,1,2) are obtained, as direct averaging gives wrong result.

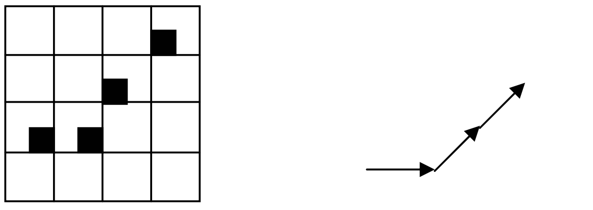


Figure 9. (a) Stroke **(b)** Quantized Angle

The code per section of the stroke is averaged as a combination of 1, 2, or 3 codes. For example consider a section of stroke shown in Fig. 9(a). the direction code is (8,1,1). This code is averaged by converting code directions in angles using anticlockwise angle representation. The angularrepresentation value is 33. 9°. This angle is then re-quantized as per the relation between code directions 1-8 and the angle given in Table 1 to a value of 2.

If after averaging using angles, average code obtained of a vertical line is 666666. This code is run length coded to obtain a compressed code of (6,6). This Average Compressed Direction Code (ACDC) is then stored as a part of the feature vector. Code 9 is used to represent a change from (8,1) in the averaged code. Code 10 indicates isolated single pixels. Some basic ACDC codes are shown in Fig.9.

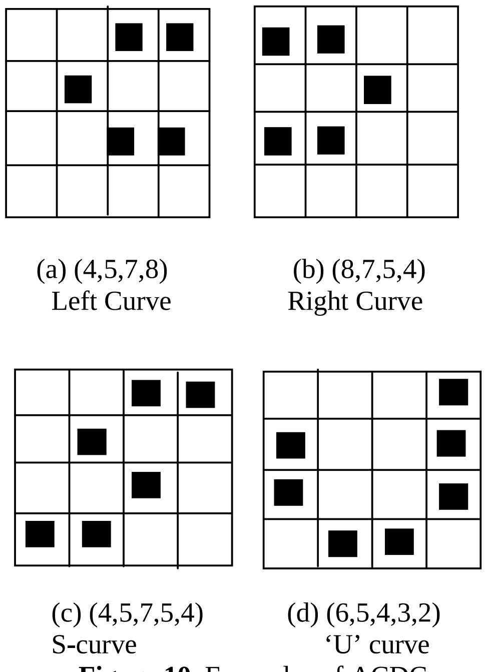


Figure 10. Examples of ACDC.

Apart from these directly obtained codes, insertions and deletions are done so as to accommodate the variations in handwritten strokes. This is from the perceptual study of how people read Devnagari script. For example the shape in Fig. 10(a) is equivalent to (4,5,6,7,8) and the basic shape defining code is (5,7) or (5,6,7) which indicates a left curve. In this way similar curve shapes and curves with distortions have similar ACDC codes and are classified as one type of stroke.

4.4 Stroke Features

Seven features extracted on all k strokes are listed below where k = 1 to Num, Num is the total number of strokes obtained in a character.

1. $L_k = \sum_{i,j} S(i,j)$, total no of black pixels in the stroke k.

2. $R_k = (\sum_i i)/M$, mean of row indices of the stroke k.

3. $C_k = (\sum_j j)/N$, mean of column indices of the stroke k.

4. $REL_k = L_k / Max(L_k)$, Length of stroke L_k, divided by the maximum length stroke.

5. $CIR_k = sqrt\ [(R1_k - R2_k)^2 + (C1_k - C2_k)^2] / L_k$, indicating a straight line or a curve, where $R1_k$ and $R2_k$, $C1_k$ and $C2_k$ are the row and column indices of endpoints of the stroke k.

6. $AREA_k = |(RE2_k - RE1_k)| * |(CE2_k - CE1_k)|$, area occupied in pixels by the stroke k where $RE2_k$ and $RE1_k$ are row indices of top most row and lower most row, $CE2_k$ and $CE1_k$ are leftmost and rightmost column indices of the stroke.

7. $REL\_AREA_k = AREA_k / max(AREA_k)$
Here (M,N) is the size of the image

All seven features for character 'Ksha' in Fig.10 (a) and (b) are given in Table 2 (last page). The algorithm accurately codes all strokes and assigns the codes as per their perceived directions

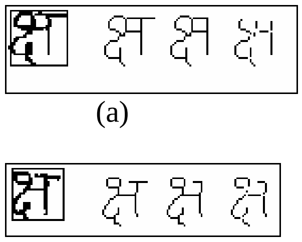


Figure 11. Results of Stroke Extraction of character 'Ksha'

From the values given in the Table 2, we can observe that the features extracted as well as stroke codes show similarity in their values, which proves the representational power of our ACDC algorithm.

4.5 Zoning Segment Frequency

The character is divided in 9 zones as shown in Fig.12. All character image co-ordinates are normalized to 0-1 range on both X and Y-axis. The zones are marked as Z1 – Z9. The zone boundaries are 0-0.33, 0.33-0.67 and 0.67 -1 on both X and Y-axis. The occurrence of stroke center of gravity is found from R_k and C_k. The number of stokes in the zone divided by the total number of strokes constitutes this ZSF feature. Zones are combined into regions in four ways and regional stroke frequencies (RSF) are calculated as given in (4)-(7).

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Figure 12. Zones of Character

$$\text{REGION 1} = Z1 + Z2 \tag{4}$$

$$\text{REGION 2} = Z1+Z2+Z4+Z5 \tag{5}$$

$$\text{REGION 3} = Z4+Z5+Z7 +Z8 \tag{6}$$

$$\text{REGION 4} = Z3+ Z6+Z9 \tag{7}$$

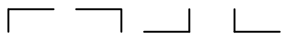
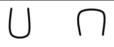
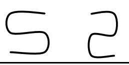
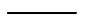
5 Character Classification

After character segmentation in strokes and feature extraction, classification is the main step for recognition. Classification is performed in three steps. First, strokes are classified according to their ACDC codes and fuzzy weights are assigned. In the second step ZSF and RSF features partition the vector space. In the third step, character is classified by unordered stroke matching based on Euclidean distance of stroke features.

5.1 Stroke Classification

The direction codes classify the strokes as one of the 15 primitives enlisted in Table 3 as left curve, right curve, u-curve, s-curve and straight lines. Column three of Table 3 gives equivalent ACDC codes of respective strokes. As is evident from the codes, this algorithm brings any stroke of arbitrary size in its basic shape defining form. This can be thus said to emulate human-vision. For example a deep left curve may have a code as 45678 whereas left curve can also be defined by codes 567 or 57. Similarly ACDC code 6 and 67, 76 are equivalent depending on the circularity feature $CIR_{k.}$

Table 3. Stroke Type and ACDC Code

Stroke Type	Segment	ACDC Codes
Straight Line		6,67,76,56, 65
Slant Line 1		7,76,67
Slant Line 2		5,65,56
Left curve		468,567, 45678,478
Right curve		864,765,
Circular		12345678, 2468
Corners		46,86,68,64
'U' curve		67812, 12876
'S' curve		4576854 87654658
Horizontal Line		8,4

Strokes are also accorded a weight according to the fuzzy membership function given in (8). The fixed weight in relation to other strokes is given in Table 4. This helps in case of over-segmentation of character. The fuzzy stroke feature enhances the recognition capability. The membership function is weighed by the circularity feature CIR $_k$ of the stroke.

$$\mu = (1/\mathrm{CIR}_k)^* \mu_{fix} \tag{8}$$

where μ_{fix} is a parameter based upon partial similarity of two segments and is given in the Table 3 . The matching parameter values range from 1-0. Lower value indicates a higher degree of matching. These values are developed intuitively for over-segmented segments.

In Table 4, relationship values for straight lines, curves and corners are given. Only one corner and its relationship value is enlisted as similar values are assigned to the remaining three corners. This feature is not useful for circular; u -curve and s-curve shape strokes, as their shape is already well defined and is fully recognizable.

Table 4. Stroke and μ_{fix}

Stroke	∣	╲	╱	⊏	⊐	_	└
∣	1	.5	.5	.33	.33	0	.5
╲	.5	1	.5	.33	.33	0	0
╱	.5	.5	1	.33	.33	0	0
⊏	.33	.33	.33	1	0	.33	.67
⊐	.33	.33	.33	0	1	.33	0
_	0	0	0	.33	.33	1	.5
└	.5	0	0	.67	0	.5	1

5.2 Coarse Classification

Feature vector is formed using number of strokes and the following :

1) No of strokes (NUM)

2) Nine ZSF (Z1-Z9)

3) Four RSF

The length of the vector is 14 for partitioning the vector space into five classes. These classes basically separate the characters as end vertibar, central vertibar, no vertibar, high activity in region R1 and low activity in region R1 characters.

5.3 Fine Classification

Pattern classification using distance measures is one of the earliest concepts in shape and pattern recognition. Dissimilarity, and conversely similarity, can be calculated by using a distance metric between two feature vectors. The nearest neighbour rule (NNR) [17], [18] classifies an unknown sample into the class of its nearest neighbour, according to some similarity measure (a distance). Given a distance, it is very simple to build up a classifier based on this rule, which often becomes unbeatable (or at least hard to beat) by other types of classifiers. Euclidean distance or L_2-Norm is the most common measure of dissimilarity and is used by our system. It is defined in (9).

$$\|d\|_2 = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2} \tag{9}$$

Where d is the distance between a_i and b_i, the feature vector elements. The feature vectors are of length n. The NNR is formulated as given in (10).

$$d\,(S_i, x) = \mathrm{MIN}_i(d(S_i, x)) \tag{10}$$

Where d is the distance measure, the set of samples $\{S_1, S_2, S_3, \ldots, S_i, \ldots, S_N\}$ is called the learning set, and x is the unknown sample to be classified. Two characters are similar if the Euclidean distance of their feature vectors is relatively small.

$$D = [d_{ij}] = \begin{bmatrix} d_{11} & d_{12} & d_{13} \cdots d_{1j} \\ d_{21} & d_{22} & d_{23} \cdots d_{2j} \\ d_{31} & d_{32} & d_{33} \cdots d_{3j} \\ \vdots & & \\ d_{i1} & d_{i2} & d_{i3} \cdots d_{ij} \end{bmatrix} = \begin{bmatrix} 0 & d_{12} & d_{13} \cdots d_{1j} \\ d_{21} & 0 & d_{23} \cdots d_{2j} \\ d_{31} & d_{32} & 0 \cdots d_{3j} \\ \vdots & & \\ d_{i1} & d_{i2} & d_{i3} \cdots 0 \end{bmatrix} \tag{11}$$

Thus, computing all the distances as given in (11) between sample and each prototype in the training set in a brute force manner, the nearest neighbour of a character can be selected by simply looking at row-min, i.e., the element in a row closest to zero in the distance metric. The diagonal zeros are the case when comparison is made between models characters. But as in this work hand drawn characters have been considered, perfect zero on the diagonal is not possible. However, smallest value on the diagonal will be a proof of system ability to recall and finding a closest match.

Total seven features are extracted on each stroke but all features are not used directly. The feature vector has stroke type as the main feature. Other supportive features are REL_k and REL_AREA_k. As stroke order is not considered this can be termed as unordered stroke matching as depicted in Fig.13.

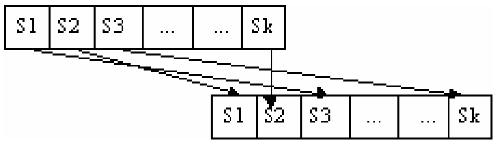


Figure 13. Unordered stroke matching.

The length of the vector (maximum no of segmented strokes is 15) is 15 * 3 i.e. 45. (Stroke type, REL_k, REL_AREA_k). S1, S2, S3 in Fig. 13 indicate the strokes and their associated features.

6 Experimentation and Results

Character recognition experiments were performed using three databases collected by us. Database I is of 270 trained writers each writing the 45 ‘Devnagari’ characters 3 times, in a sheet provided with boxes. A reference sheet is provided with standard characters to avoid too many variations in characters. Database II was collected from 20 persons, each one writing 45 characters 3 times. The writers were from different age groups and social backgrounds. They were given a sheet with boxes and no reference character sheet. Database III is totally unconstrained where 100 writers were given the instruction orally to write the characters 3-6 times. This data was collected from school children, college students, middle-aged working professionals and the older generation. 60 % of the database is used for feature extraction. Feature vectors of three prototypes of each character are stored. The remaining database is used for validation.

The database for character ‘Kha’ is shown in Fig. 14 and for character ‘Sha’ is shown in Fig. 15. As can be observed there is variation in shapes and also in the writing style of character ‘Kha’.

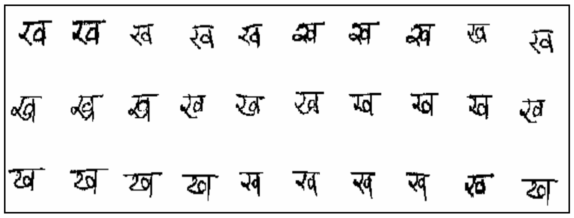


Figure 14. Samples of character ‘KHA’

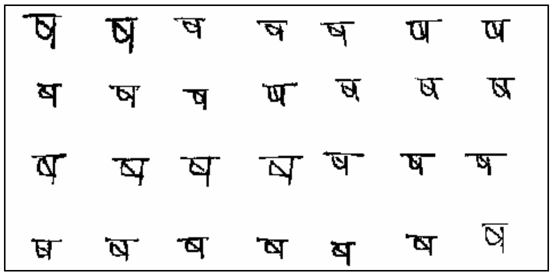


Figure 15. Samples of character ‘SHA’

The strokes of segmented character ‘KHA’ of is shown in Fig. 16. It indicates the fact that the algorithm is independent of character size and the algorithm developed is free of normalization..

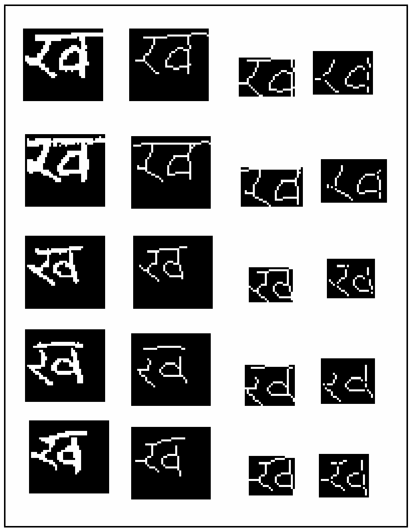


Figure 16. Strokes of one model of Character ‘KHA’

Similarly Fig. 17 indicates the strokes of one model of character ‘Sha’. Maximum variation in character ‘Sha’ is observed in the short tilted line in between the primary stroke.

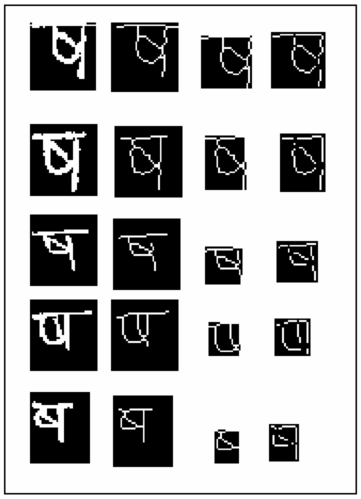


Figure 17. Strokes of one model of Character 'SHA'

The number of strokes extracted may slightly differ but it is distributed around the mean. The mean for 'Kha' is 7.3 and for' Sha' is 4.2. Fig.18 indicates the number of strokes on 'Y-axis' and number of samples 1-25 are marked on 'X-axis'for 'Kha' and 'Sha'.

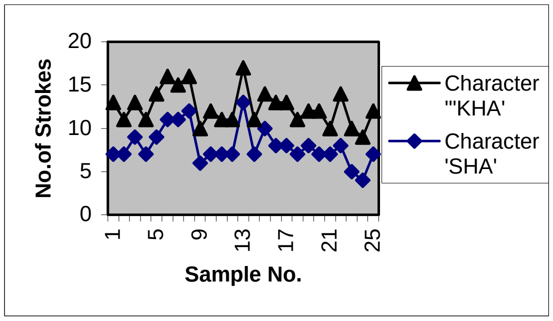


Figure 18. Number of Strokes Vs. Sample No

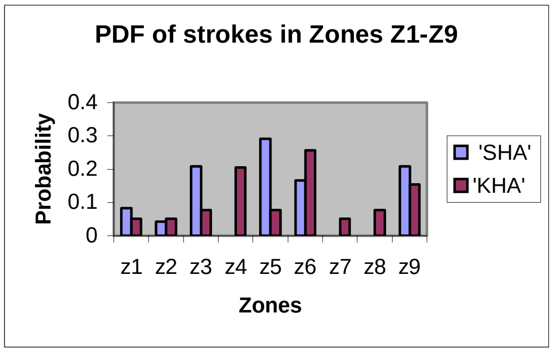


Figure 19. Zonal Stroke Frequency

Fig. 19 shows the plot of ZSF against the zones for character 'Kha' and 'Sha'. From the graph it can be inferred that every character or a group of similar looking characters have similar ZSF and RSF. It is noted for character 'SHA' there is no ZSF for zones Z4, Z7 and Z8. This clearly distinguishes it from character 'Kha' as can be done by human eye. Similarly other characters have been segmented in strokes and two models are stored for each character.

Both the characters shown in Fig. 15 and 16 have a vertibar at the end of the character and thus come in the same class of characters with end vertibar. The plot in Fig. 20 indicates that they have similar RSF.

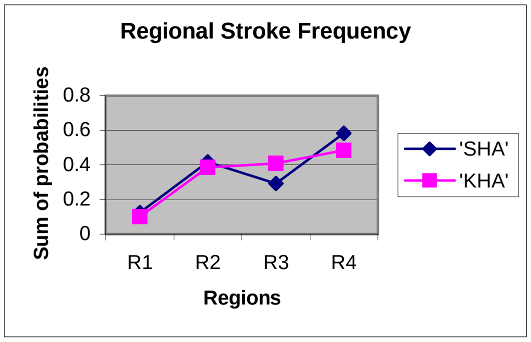


Figure. 20 Regional Stroke Frequency

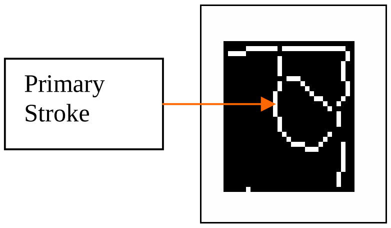


Figure 21. Character 'Sha' and its Primary Stroke

The primary stroke of character 'Sha' is the long left curve stroke as shown in Fig. 21.The distribution of this stroke over 11 samples is plotted in Fig. 22. This localized information of the primary stroke indicates that although the 'y co-ordinate of the center of gravity of the stroke is confined to a small range of values; its 'X' co-ordinate shows more variation. This is due to the difference in the length of topline, which changes the starting point of the image frame.

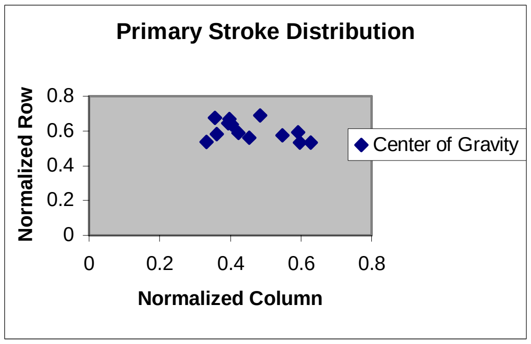
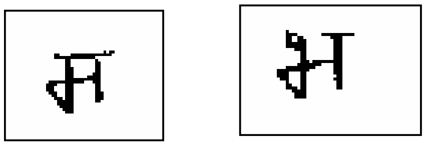


Figure 22. Primary Stroke Distribution.

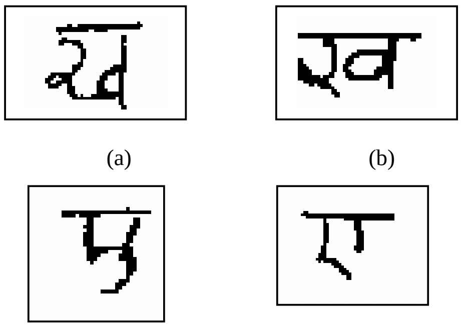
The result of recognition is dependent on database, varying from 89 - 97 % as given in Table 5. It indicates a need for developing a standard benchmark for Devnagari character recognition. The overall result for training samples is 93.67% and for unknown data is 92%. Feature vector size and accuracy results of existing schemes and our algorithm are compared in Table. 6. From the values it is evident that our algorithm is efficient and works with minimum vector size. The time complexity of the algorithm is very less compared to the one mentioned in [11],[12] as no size or line density normalization is carried out. In [12] the character is normalized to a size of 72 pixels and then pixel based Sobel filter output value is extracted. Our ACDC algorithm itself removes variations is very fast as it works on thinned and segmented strokes and stores very few features. Thus it is independent of character size and pen width. The total length of the feature vector is 45 only. The time to recognize one character is of the order of .01 ms on a Pentium V, 500 Mb Ram using Matlab 7.

The poor accuracy can be attributed to confusing characters and also due to different writing patterns. As India is a multilingual country and Devnagari script is used for different languages, it is observed that there are variations in characters when the language is changed. In Fig. 23(a) 'Ma' and (b) 'Bha' represent the group of structurally similar characters. Fig. 24 (a) - (b) show the character 'Kha' written differently. Fig. 24(c) and (d) show character 'Ae' written by Hindi language speaker and Marathi language speaker respectively.



(a) (b)

Figure 23. Structurally similar character



(a) (b) (c) (d)

Figure 24. Same characters written differently.

Table 5. Recognition Result

Database type	Recognition for Training	Recognition for Unknown Data
I (Trained writers)	97.4%	94.6%
II (Semi-trained)	93.5 %	92.3%
III (Unconstrained)	90.14%	89.2%
Overall Result	93.67%	92.0%

Table 6. Comparison with existing Techniques

Technique	**Feature vector Size**	**Accuracy**
Directional code histogram	64	80%
Sobel Filter	400	94%
Our ACDC algorithm	45	92.8%

7 Conclusion

The proposed work presents recognition of Devnagari characters free from normalization thereby giving flexibility and allowing size variation. Database was collected from writers of varied background. Modified thinning algorithm needed for separating a character in its constituent strokes was developed and tested successfully. Average Compressed Direction Codes (ACDC) algorithm, though simple, work efficiently to detect curves and turning points and emulate human vision. The overall recognition accuracy is 92.8%.

Rejection also plays an important role in system evaluation. Around 1.1% of samples are rejected as unrecognized. This work is a step towards unconstrained Devnagari script recognition, as in unconstrained text; words have to be segmented into its basic characters and modifiers in the order of top modifier, body (characters) and lower modifier. Also as the system is based on strokes it can be a step towards the conjunct separation and recognition. This is a maiden attempt at analyzing Devnagari characters on stroke-based model. Further preprocessing may improve the results and also more models may be stored. The Devnagari script is used extensively at the grass root levels in many states of India. This Devnagari character recognition system can be used in developing Indian postal automation system, bank check processing system, as a reading aid to blind and also in forensic science.

References:

[1] S. Kompalli, S. Nayak, S. Setlur and V. Govindaraju, Challenges in OCR of Devnagari Documents, *Proceedings of Eighth International Conference on Document Analysis and Recognition*, Seoul, South Korea, 2005, pp. 327- 331.

[2] N. Joshi, G.Sita, A.G. Ramakrishnan, Deepu V., S. Madhavnath, Machine Recognition of Online Handwritten Devnagari Characters, *Proceedings of Eighth International Conference on Document Analysis and Recognition*, Seoul, South Korea, 2005, pp. 1156- 1160.

[3] Nucharee Premchaiswadi, Wichian Premchaiswadi, Surakarn Duangphasuk, Seinosuke Narita, “SMILE: Printed Thai Character Recognition System”, *WSEAS Transactions on Computers*, Vol.2, No.2, 2003, pp. 424-429.

[4] Nucharee Premchaiswadi, Wichian Premchaiswadi, Surakarn Duangphasuk, Seinosuke Narita, “Broken Characters Identification forThai character Recognition Systems”, *WSEAS Transactions on Computers*, Vol.2, No.2, 2003, pp. 430-434.

[5] I.K. Sethi, *Machine Recognition of Constrained Handprinted Devnagari,* Pattern Recognition, Vol. 9, 1977, pp. 69-75.

[6] R. M. K. Sinha and H. N. Mahabala, *Machine Recognition of Devnagari Script,* IEEE Transactions on Systems, Man and Cybernetics, Vol. 9 (8), 1979, pp. 435- 441.

[7] R. M. K. Sinha and Veena Bansal, A complete OCR for Printed Hindi Text in Devnagari Script, *Proceedings of Fifth International Conference on Document Analysis and Recognition*, Seattle, US, 2001, pp. 800-804.

[8] B.B. Chaudhuri and U. Pal, An OCR system to read two Indian Language Scripts: Bangla and Devnagari (Hindi), *Proceedings of Fourth International Conference on Document Analysis and Recognition*, Ulm, Germany, 1997, pp. 1011-1015, 1997.

[9] Prachi Mukherji and Priti. P. Rege, A Survey of Techniques for Optical Character Recognition of Handwritten Documents with reference to Devnagari Script, *Proceedings of First International Conference on Signal and Image Processing,* Hubli, India, 2006, pp. 178 – 184.

[10] Latesh Malik, P.S. Deshpande & Sandhya Bhagat, Character recognition using relationship between connected segments and neural network., *WSEAS Transactions on Computers* Vol.5, No.1, January 2006 , pp. 229-235.

[11] N. Sharma , U. Pal, F. Kimura and S. Pal, Recognition of Off-Line Handwritten Devnagari Character Script, *ICVGIP,* Madurai, 2006, pp. 805- 816.

[12] U. Pal, N. Sharma, T. Wakabayashi and F. Kimura, Off- Line Handwritten Character Recognition of Devnagari Script, Proceeding of Ninth ICDAR, Brazil, 2007, pp. 496-500.

[13] Timothy J. Ross, *Fuzzy Logic with Engineering Applications,* MCGraw Hill International Editions, New York, 1997.

[14]Rafel C Gonzalez, Richard E.Woods, Digital Image Processing, Second Edition, Pearson Education, 2003.

[15] N.Otsu, A threshold selection method from gray level histograms, *IEEE Transactions on Systems*, *Man and Cybernetics*, 9(1), 1979, pp. 62-66.

[16] Prachi Mukherji, Priti P Rege and Leena K. Pradhan, Analytical Verification System for Handwritten Devnagari Script, Proceedings of the Sixth IASTED VIIP, Palma De Mallorca, Spain, 2006 pp. 237-242.

[17]B.B. Chaudhari and D. Datta Majumder, Two Tone Image Processing and Recognition, Wiley Eastern Limited, 1993.

[18] K. Fukunaga, *Introduction to Statistical Pattern Recognition (Second Edition),* Academic Press, New York, 1990.

Table 1. Angle quantization and code.

Theta (In Deg)	337.5 -22.5	22.5-67.5	67.5-112.5	112.5 -157.5	157.5 -202.5	202.5 -247.5	247.5-292.5	292.5-337.5
Code	1	2	3	4	5	6	7	8

Table 2. Results of different features for two character images in Fig.7

k	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
L_k *	18	10	4	3	1	1	4	4	11
L_2 *	15	17	4	4	1	6	9		10
R_k	0.6588	0.2759	0.7241	0.9138	0.3966	0.2241	0.3448	0.2414	0.5690
R_2	0.6250	0.1508	0.7143	0.9143	0.3214	0.3929	0.2179	-----	0.6071
C_k	0.2249	0.2727	0.4545	0.4886	0.5455	0.5909	0.7727	0.9545	0.9545
C_2	0.2202	0.3466	0.4476	0.4381	0.4762	0.6667	0.9000	-----	0.9134
REL_k	1.0000	0.5556	0.2222	0.1667	0.0556	0.1667	0.2222	0.2222	0.6111
REL_2	0.8824	1.0000	0.2353	0.2353	0.0588	0.3529	0.5294	-----	0.5882
$CIRk_k$	0.6136	0.8602	1.0000	1.2019	1.0000	1.0000	1.0000	1.0000	1.0000
$CIRk_2$	0.8246	0.1860	1.1180	1.0607	0	1.0000	0.7454		1.0050
$AREA_1$	96	56	5	12	2	4	5	5	12
REL_A_1	1	.5833	.0520	.1250	.020	.040	.0520	.0520	.125
$AREA_2$	104	48	15	16	1	7	35	-----	22
REL_A_2	1	.465	.156	.1666	.0001	.0673	.336		.2115
$ACDC_1$	545678	6787	6	7	6	6	8	6	6
$ACDC_2$	568	24768	65	7	10	8	86	-----	6