



Shape Feature and Fuzzy Logic Based Offline Devnagari Handwritten Optical Character Recognition

Prachi Mukherji

prachimukherji@rediffmail.com

Smt. Kashibai Navale College of Engineering (SKNCOE), Pune University, Pune, 411041, India

Priti P. Rege

ppr@coep.extc.com

College of Engineering (COEP), Pune University, Pune, 411005, India

Abstract

Devnagari script is a major script of India and is widely used for various languages. In this work, we propose a new shape based technique for recognition of isolated handwritten Devnagari characters. The thinned character is segmented into segments (strokes), using basic structural features like endpoint, crosspoint, junction points and adaptive thinning algorithm. The segments of characters are coded using our Average Compressed Direction Code (ACDC) algorithm. The segment shapes are classified as left curve, right curve, horizontal stroke, vertical stroke, slanted lines etc. based on these ACDC codes. The knowledge of script grammar is applied to identify the character using shapes of strokes, mean row and column co-ordinates, relative strength, straightness and circularity. Their location in the image frame is based on fuzzy classification. Characters are pre-classified using a tree classifier. Subsequently unordered stroke classification based on mean stroke features is used for final classification and recognition of characters. The system tolerates slant of about 10 deg left and right and a skew of 5 deg up and down. The average accuracy of recognition of the proposed system is 86.4%.

Keywords: Devnagari Script, Average Compressed Direction Codes, Fuzzy Logic, Unordered Stroke Matching.

1. Introduction

Handwritten character recognition is used most often to describe the ability of computer to translate human writing into text. The technical challenges in character recognition arise from three sources. First the symbols: the set of idealized shapes that can occur, often in a hierarchy where simple symbols are assembled into more complex ones, at several levels of organization. Second is deformation: the range of shape variations that each symbol is allowed to undergo, including geometric transformation (translation, rotation, scaling, stretching, etc.) and more complex or time dependent distortion. Third is image defects: the imperfections in image due to printing, optics, scanning, quantization, binarization, etc. Handwriting and machine print demand a different approach. Handwriting consists of elongated strokes, whereas the machine print consists of regularly spaced blobs.

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]. A marked distinction in Devnagari script from the 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 English, Latin, Chinese, Persian, Tamil, and Bangla scripts on both handwritten and machine printed texts. 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 [3] and for typed Devnagari script

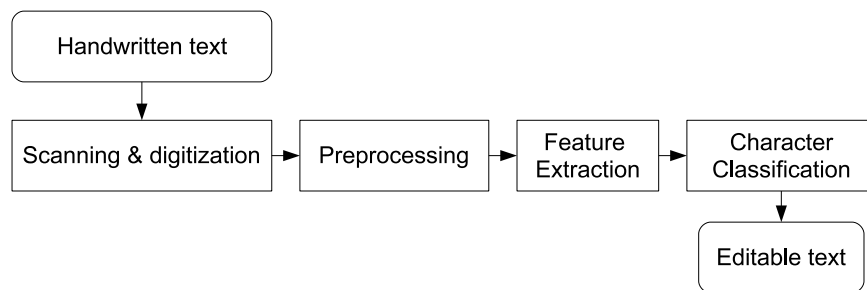


Fig. 1: Character recognition system.

by Sinha and Mahabala [4]. Sinha and Bansal [5] 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. Recognition of Devnagari text in Sanskrit manuscript 'Saddharma-pundarika' [6] is achieved with an accuracy of 98.09% using structural features and neural networks for classification. Pal and Chaudhuri have attempted OCR for two scripts, Bangla and Devnagari in [7]. Database evaluation methods are given in [8] and database for Devnagari numerals has been collected from mail addresses and job application forms in [9]. Machine recognition of online handwritten Devnagari characters has been reported in [2] with 82-85% accuracy. In [10] online Devnagari script recognition is attempted with 86.5% accuracy on a database of 20 writers. A combination of on-line and offline features has been used. In [11] Binary Wavelet transform is used for feature extraction of handwritten Devnagari characters. In [12], a survey of different structural techniques used for feature extraction in OCR of different scripts is given. Recently in [13], Quadratic classifier based method is proposed with 81% accuracy. Fuzzy features based approach for Bangla script is presented in [14]. Fuzzy logic based approach for Roman script has been used to account for variability in handwritten script in [15].

In this proposed work, attempt has been made to recognize 45 isolated handwritten characters in Devnagari script. This system allows variations that occur in individual handwriting, within specified limits. The handwriting should be legible and adhering to structural syntax of Devnagari script. The processing steps of our OCR system are as designated in Fig. 1. They can be summarized concisely. The documents are scanned and the digitized images are subjected to preprocessing techniques like filtering, binarization, skeletonization and pruning. The feature extraction module segments the character in segment strokes. Various features of these segmented strokes like mean row and column coordinate, circularity, area and Average Compressed Direction Codes (ACDC) are extracted. Cognitive scientists report that humans base their thinking on conceptual patterns and mental images rather than on any numerical quantities [16]. Keeping this view in mind and the construction of Devnagari script, the relative positions of segment strokes in image frame are classified using fuzzy logic. The character is first pre-classified using a tree classifier and final classification is achieved using unordered stroke matching. The applications of this system are many as in postal automation, reading aid for the blind and unconstrained Devnagari script recognition.

2. The Devnagari Character Set

The basic character set is of 48 characters in which there are 12 vowels (swar) and 36 consonants (Vyanjan). A unique property of Devnagari script is the formation of conjuncts

(Yuktakshar) that are combines of two (bi-consonantal) or three (tri-consonantal) consonants. About 176 bi-consonantal and 24 tri-consonantal conjuncts can be formed. Their formation is by simple rules and restrictions of the language of application. Each of the consonant and the conjuncts can be further modified by vowel modifier (Matra). 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 the Devnagari script as there are syllables in the spoken language. The 45 characters of the basic handwritten character set for experimentation is based on their present-day usage. The handwritten character set is shown in Fig. 2.



Fig. 2: Basic character set (handwritten).

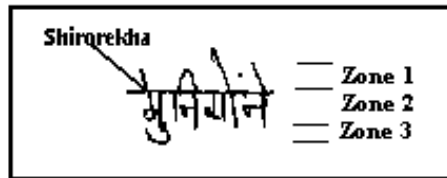


Fig. 3: 'Devnagari' word with modifiers and zones.

Every character has a horizontal header line on the top, called the 'Shirorekha' shown in Fig. 3. This line serves as a reference to divide the character into two distinct portions: Head and Body or zone 1 and zone 2, if the top modifier (matra) is present. The lower modifier occupies zone 3. The basic character width ranges from very small to large going through many medium sizes. Many characters have a vertibar, which can be present in the end of the character as shown for a few representative characters 'ch', 'ja', 'na' and 't' in Fig. 4(a). The vertibar can be present in the middle region as shown in Fig. 4(b)for

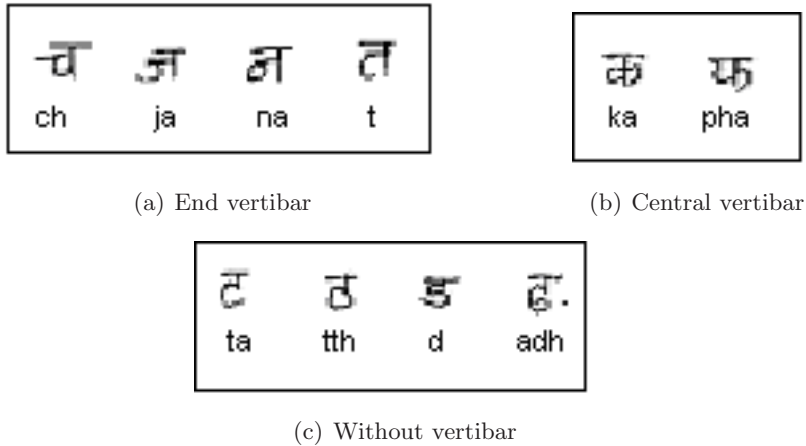


Fig. 4: Vertibar and non-vertibar 'Devnagari' characters.

characters 'ka' and 'pha'. Twelve characters do not have a vertibar. Some representative characters 'ta', 'tth', 'd' and 'adh' are shown in Fig. 4(c).

Fig. 5(a) shows four characters that have gaps ('aa', 'ga', 'ana' and 'sh') and Fig. 5(b) shows five characters with top modifiers (namely 'ee', 'ai', 'o', 'au' and 'am'). These five characters can be segmented in two parts: the top modifier and the body. The body of these characters resembles the characters 'e', 'ae', 'aa', 'aa' and 'a' respectively as shown in Fig. 5(c). This reduces the total number of characters to be recognized to 40. The body of both third and fourth character ('o' and 'au') in Fig. 5(b) are identified as character 'aa'. The segmented shape of top modifiers above the shirorekha of second and third character ('ae' and 'o') shown in Fig. 5(b) is also similar as shown in Fig. 5(d). Thus a total of four top modifiers have to be recognized.

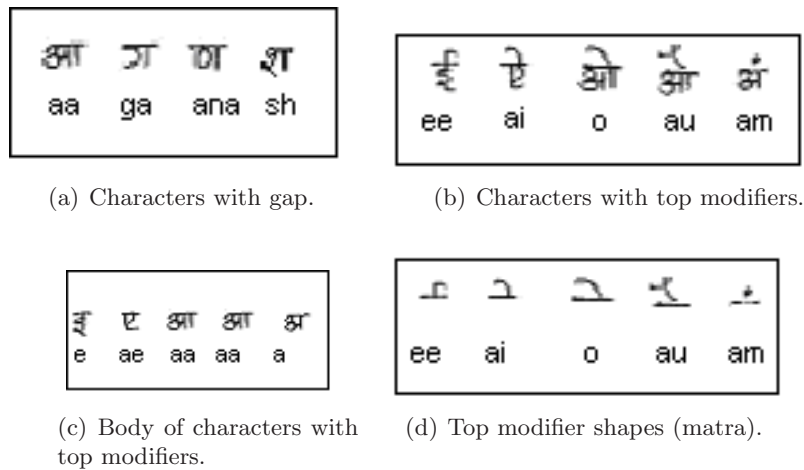


Fig. 5: Features of Devnagari characters.

3. Preprocessing

Each character has to undergo preprocessing before it can be recognized. This is done to increase the probability of recognition by correcting detectable abnormalities, if any, in the

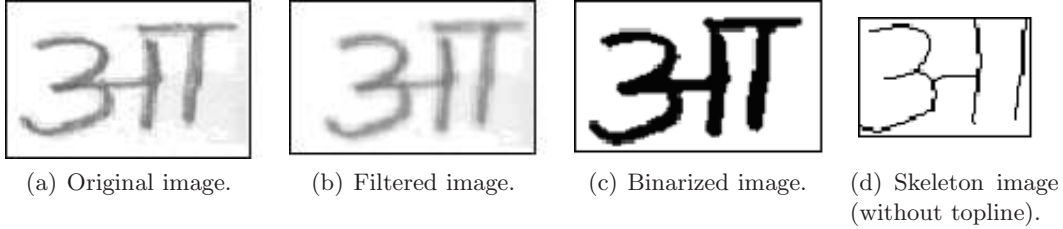


Fig. 6: Preprocessing steps for character 'aa'.

character. 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 done using the Gaussian filter of size 5X5. Binarization is achieved using Otsu's algorithm [17]. Next step, skeletonization [18] is an important aspect of feature extraction in our scheme. Skeletonization always produces short extraneous spurs. The spurs in the image are removed by the spur removal algorithm [18]. 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 [19]. In this method successive projections are taken and peak projection obtained gives the direction of slant of the near vertical strokes. Keeping the topline as reference, horizontal shear transform [20] is applied only if the slant is more than the set threshold of 10 deg. Different steps of the preprocessing module on character 'aa' are shown in Fig. 6(a) - Fig. 6(d). The character skeleton image is enclosed in a component box devoid of all extra pixels as shown in Fig. 6(d).

4. Feature Extraction

Feature extraction is the main module where isolated characters are analyzed and feature vector is created. The character under consideration is first checked for the presence of top modifier. The top modifier and the body of the character are recognized separately. Characters are segmented into strokes using our segmentation algorithm. The segmented strokes are coded using our ACDC algorithm. Fuzzy and crisp features are extracted on the segmented strokes of characters.

4.1 Top Modifier Features

As shown, in Fig. 3, the distinct feature of Devnagari script as well as individual characters is the topline (Shirorekha), which is detected using Hough Transform [18]. The top modifier is present above this topline and is recognized on the basis of transitions [5] and shadows [19]. For extracting top modifier, region above the topline is considered and segmented from the main character. The average number of transitions is counted in the segmented part of the character. The left to right shadow is plotted and coded according to increase (a jump of +1) or decrease in the shadow (a value of -1). Decreasing shadow from the base of the modifier to the top is for modifier of characters 'ai', 'o' or 'au' and increasing shadow from the base of modifier to the top is for modifier of character ee. If the segmented shape does not fall in any of the five top modifier shape categories ('ee', 'ai', 'o', 'au' and 'am') shown in Fig. 5(d), then the character is combined again and sent for feature extraction through the main module.

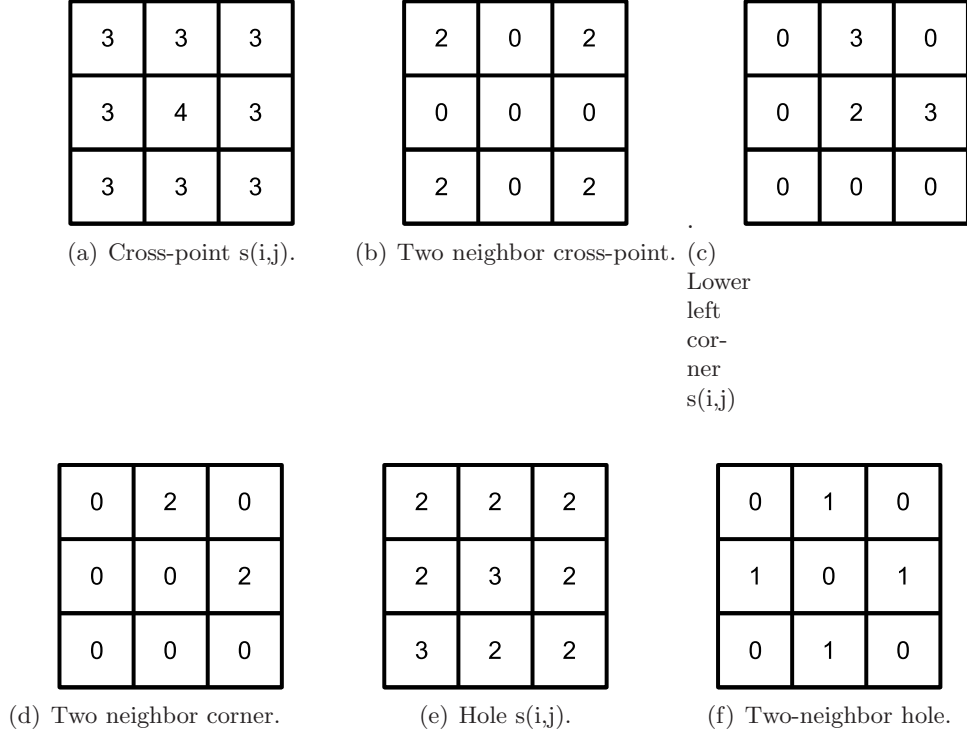


Fig. 7: Steps in Adaptive Thinning Algorithm.

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, termed as SumofNeighbours, as given in (1). Since the characters have already passed through skeletonization once, expectedly the maximum neighborhood can be of four pixels signifying a crosspoint as shown in Fig. 7(a). This point is given value zero and then the map is checked for values of three.

$$SumofNeighbours = s(i, j) = \sum_{i-1}^{i+1} \sum_{j-1}^{j+1} p(i, j) - 1 \quad (1)$$

Blindly changing the value of these points to zero gives rise to over segmentation, as shown in Fig. 7(b). Instead, the combinations shown in Fig. 4.2 and Fig. 7(d) detect corners and smoothen them in a process termed as pruning. The pruned map is then thresholded, as given in pseudocode. This ensures that a stroke has only two neighbors as required for its coding by our ACDC algorithm. A hole in Fig. 7(e) is modified to two-neighbor hole in Fig. 7(f).

```

    % Pseudocode for thresholding %
    start
    {
    if s(i,j) >= 3
        s(i,j) =0;
    else
        s(i,j) =1;
    end

```

```
}
stop
```

4.3 Average Compressed Direction Coding Algorithm

Shape description of strokes is obtained by applying our Average Compressed Direction Coding algorithm. This algorithm has four main steps. The first one is labeling of strokes, second one is coding of strokes, third step is averaging and fourth step is compression.

In the first step after segmentation of the character in strokes, all segment strokes are labeled [18] from left to right in the image frame. This scheme preserves temporal and shape information of the character. As each stroke (segment) is labeled, its row and column indices are also stored. Then for each stroke, mean row and mean column co-ordinate is calculated. These values are normalized by dividing them with the number of rows and number of columns respectively.

The second step is coding of these strokes using Freeman's chain codes [18] with slight modification. Normally, Freeman chain codes define a closed boundary and their codes and directions are shown in Fig. 8(b).

In this algorithm, they are used to indicate the shape of the strokes only one pixel thick. Each pixel has only two neighbors except the two endpoints. Coding begins from the higher endpoint i.e. the endpoint with lowest row index. If row indices are same, coding begins from the leftmost 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. The coding algorithm pseudocode is:

```
% pseudocode for character coding
start
{
    for i = 1: NUM
        for all pixels of value i
            Stpoint_x = min endpoint_row_index ;
            Stpoint_y = min endpoint_column_index;
            find neighbor
            dir code(i,j) = code according to Freeman Chain code(1-8)
            reset Stpoint_x to row index of neighbor
            % once the pixel direction is coded, that pixel should be discarded
            reset Stpoint_y to column index of neighbor
            set the previous black pixel of stroke to zero
        end
    end
}
stop
```

As shown in Fig. 8(a), the segmented stroke is a curve of twelve pixels. Coding begins from Endpoint1, which is also the starting point Stpoint_x, Stpoint_y. The coding directions and codes are shown in Fig. 8(b). The first direction code (dircode) obtained is 4. This pixel is then set to zero and coding continues by resetting the starting point. The dircode is 44 as shown in Fig. 8(c). The dircode after taking all pixels in account is of length eleven and assigned the value 44455678887.

The third step in the ACDC algorithm is averaging. This reduces the vector space while maintaining the direction information in the code. Depending upon the length of the stroke, code is averaged. Code lengths of 1 and 2 are treated separately. Code length obtained is divided by 3 and the remainder is stored. Every section of code length 3 of the stroke is

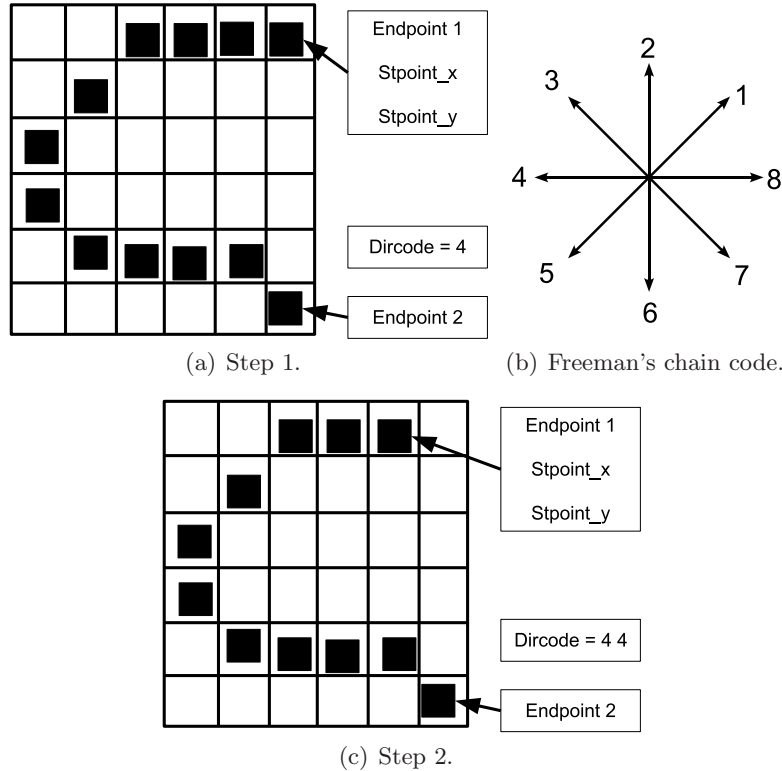


Fig. 8: A stroke and its coding.

converted into its angular contribution and then averaged. Each code is multiplied by 45 deg as the difference between two directions is 45 deg, with anticlockwise angle representation. These angles are then quantized according to values in Table 1.

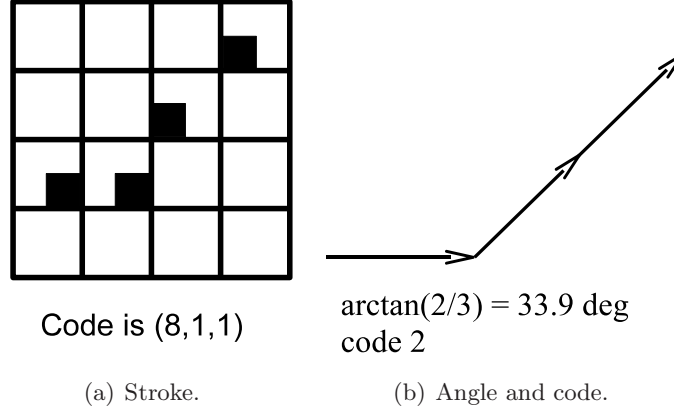
Here care has been taken when combinations of (8,1,2) and (7,1,2) are obtained, as direct averaging will give wrong value. Such combinations are treated separately as shown in Fig. 9(a) and Fig. 9(b). Here dircode (8,1,1) is coded as dircode 2. The algorithm is given as follows:

```
% Algorithm for averaging %
start

Multiply each code by 45
Sum them
Divide by 3
Store the angle in $ang_array$
Find the angle contribution of remainder codes (1 or 2) separately
Add in array $ang_array$
Convert the $ang_array$ obtained to a code as per table
Store $Av_dir_code$

stop
```

In the fourth step, the average direction code thus obtained is further compressed so as to extract the basic primitive from the stroke. Standard runlength coding is used for compression, where the code is used, not the run number. The changes in the average direction code and their values are noted and stored. Various examples are given in Fig. 10(a) -

**Fig. 9:** Stroke and its code.**Table 1:** Angle quantization and code.

<i>Theta (InDeg)</i>	Code
337.5 – 22.5	1
22.5 – 67.5	2
67.5 – 112.5	3
112.5 – 157.5	4
157.5 – 202.5	5
202.5 – 247.5	6
247.5 – 292.5	7
292.5 – 337.5	8

Fig. 10(d). This Average Compressed Direction Code 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.

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 perceptual study two experiments have been performed. Ten children of age six to ten and ten graduate students were shown Devnagari characters and asked to describe in words the shape of Devnagari characters. It was found that everyone described the characters in terms of curves, straight lines and slanted lines. Also there was a relationship like a small notch, or a long right curve followed by a small horizontal line. These descriptions have been used to set the vectors for the characters in this work. In the second experiment they were shown only curves and lines in different distorted shapes and asked to categorize as one of the standard shapes like a left curve, right curve, straight line, line tilted to left, line tilted to right, horizontal line etc. These standard shapes have been decided on the basis of descriptions provided in the first experiment. This led to the development of ACDC algorithm in which a shape is coded by averaging and compressing the direction code obtained so as to approximate the different distorted shapes to simple standard shapes. The conclusions of both the experiments are in line with Gestalts principles on Perception [20] that information content in a two-dimensional shape is contained in its contour i.e. its vertices, corners and curves.

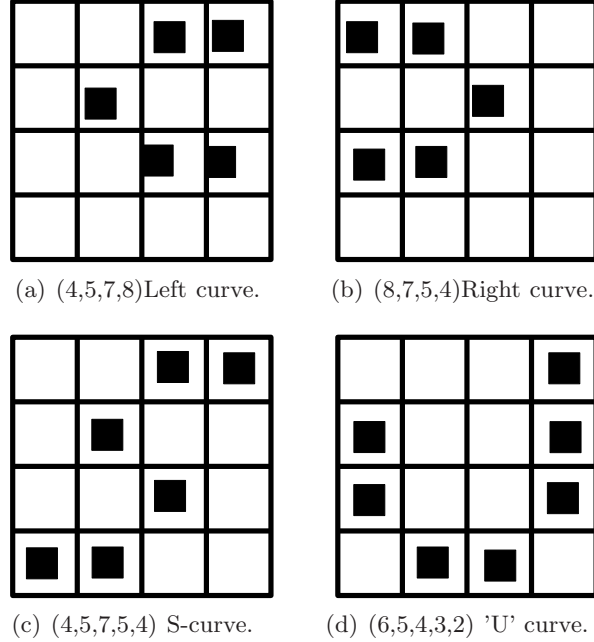


Fig. 10: A stroke and its coding.

For example, the ACDC code of the shape in Fig. 11(a) is (4, 5, 7, 8) and the basic shape defining code is (5, 7) shown in Fig. 11(b) or (5, 6, 7) shown in Fig. 11(c) and all are classified as left curve.

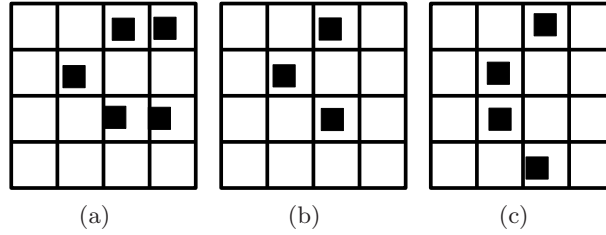


Fig. 11: Different curves classified as left curves.

4.4 Stroke Features

Stroke features are found on each stroke of the character. Stroke features are divided in two categories: Crisp features and Fuzzy features. The term crisp [21] is used to differentiate from fuzzy features. The term crisp is used in fuzzy set theory where crisp numbers or situations are definite values and used to build the model directly compared to a fuzzy set where fuzzy set is calculated with the help of a membership function.

4.4.1 Crisp Features

Ten crisp features are extracted on all k strokes, where $k = 1$ to Num. Num is the total number of strokes obtained in a character. They are

1. $L_k = \sum_{i,j} S(k)$, total number of black pixels in the stroke k .
2. $R_k = \frac{\sum_i i}{M}$, mean of the row indices of the stroke k .

3. $C_k = \frac{\sum_j j}{N}$, mean of the column indices of the stroke k.
4. $REL_k = \frac{L_k}{Max(L_k)}$, Length of stroke L_k , divided by the maximum length stroke.
5. $CIR_k = \sqrt{[(R1_k - R2_k)^2 + (C1_k^2 - C2_k^2)]/L_k}$, 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. $Dleft_k = \sum_j j$, distance of stroke k from the left frame of the character image.
8. $Dright_k = \sum_j N - j$, distance of stroke k from the right frame of the character.
9. $Dupper_k = \sum_i i$, distance of stroke k from the top of the character image.
10. $Dlower_k = \sum_i M - i$, distance of stroke k from the bottom of the character.

To extract the aforementioned features, strokes are labeled. The row and column co-ordinates of each labeled stroke are also stored along with the label number. The index (M,N) is the size of the character where (1,1) is the left top most point of the image. The sum of row co-ordinates of a segment stroke is divided by the total number of rows (M) for normalization of features 2 and 3. This is a decimal value between 0 and 1, where the lower limit is not inclusive of 0. Similarly the column co-ordinates are also added for each stroke and then divided by the number of columns, N. This gives the average column index between 0-1. Therefore both column and row indices are between 0 and 1, without actual resizing. This reduces computation and avoids distortion when the aspect ratio cannot be maintained.

Feature 4 and 5 are both calculated from stroke within the same character as the longest stroke is found to be generally same in different samples of the same character and this feature is used to find the circularity feature that has a value between 0 and 1.

4.4.2 Fuzzy Features

Image frame is divided in four parts as shown in Fig. 12. Stroke location features are extracted according to modified fuzzy logic membership [20] functions as given in (2), (3), (4), (5).

$$\mu_{left}(S_k) = [1 - (4 * D_{leftk}/N)]^\beta, \text{ if } D_{leftk} \leq N/2, \text{ Else, } \mu_{left}(S_k) = 0. \quad (2)$$

$$\mu_{right}(S_k) = [1 - (4 * D_{rightk}/N)]^\beta, \text{ if } D_{rightk} \geq N/2, \text{ Else, } \mu_{right}(S_k) = 0. \quad (3)$$

$$\mu_{upper}(S_k) = [1 - (4 * D_{upperk}/M)]^\beta, \text{ if } D_{upperk} \leq M/2, \text{ Else, } \mu_{upper}(S_k) = 0. \quad (4)$$

$$\mu_{lower}(S_k) = [1 - (4 * D_{lowerk}/M)]^\beta, \text{ if } D_{lowerk} \geq M/2, \text{ Else, } \mu_{lower}(S_k) = 0. \quad (5)$$

D_{leftk} is the sum of normal distance of pixels in stroke from left side of the image frame divided by the area of the stroke. Similarly, D_{rightk} is the sum of normal distance of pixels in stroke k from right side of the image frame divided by the area of the stroke, D_{upperk} is

the sum of normal distance of pixels in stroke k from upper side of the image frame divided by the area of the stroke and D_{lowerk} is the sum of normal distance of pixels in stroke k from lower side of the image frame divided by the area of the stroke. Parameter β has a value greater than or equal to one [20]. A stroke is classified as one of the four types as left upper, left lower, right upper and right lower region segment/stroke depending on threshold set from experimentation.

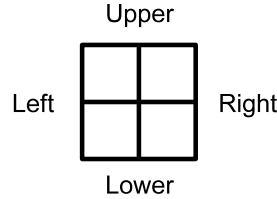


Fig. 12: Location in image frame.

5. Character Classifier

The recognition is based on a multi-class tree pre-classifier. The decision tree has root node at the top connected by successive branches to other nodes, till leaf nodes are reached. CART [22] approach is followed. This requires attention to number of splits, the property to be tested at each node, the number of nodes, pruning, impurity of leaf node and handling of missing data.

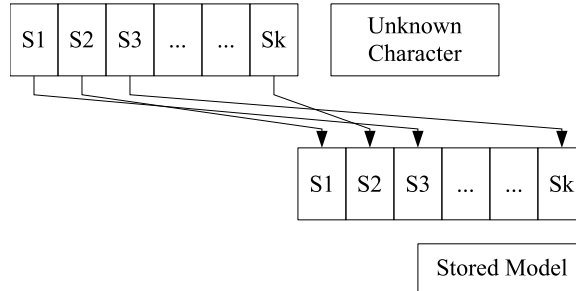


Fig. 13: Unordered stroke matching.

The character model is made up of number and type of strokes in left, right, upper and lower region and their mean crisp features. The stroke type is left curve, right curve, 'U'curves, 'S' curves, horizontal line, vertical line and slant strokes. This is further supported with the help of fuzzy features that categorize the stroke as right, left, upper or lower stroke. The strokes of the unknown character are first identified as stroke type and then matched using Euclidean [18] distance classifier with the stored average values of six crisp features. As the number of strokes is not fixed due to variations in the handwritten characters of the same class, multi-modeling is used. The feature vector length is variable depending upon the number of strokes. The number of strokes varies from three to twelve and therefore the feature vector varies from 18 to 72 depending upon the node traversed in the tree of the character.

Presence of certain strokes like vertical lines preclassifies the character in different classes. The first property that is checked is if the character has a gap in its vertical histograms.

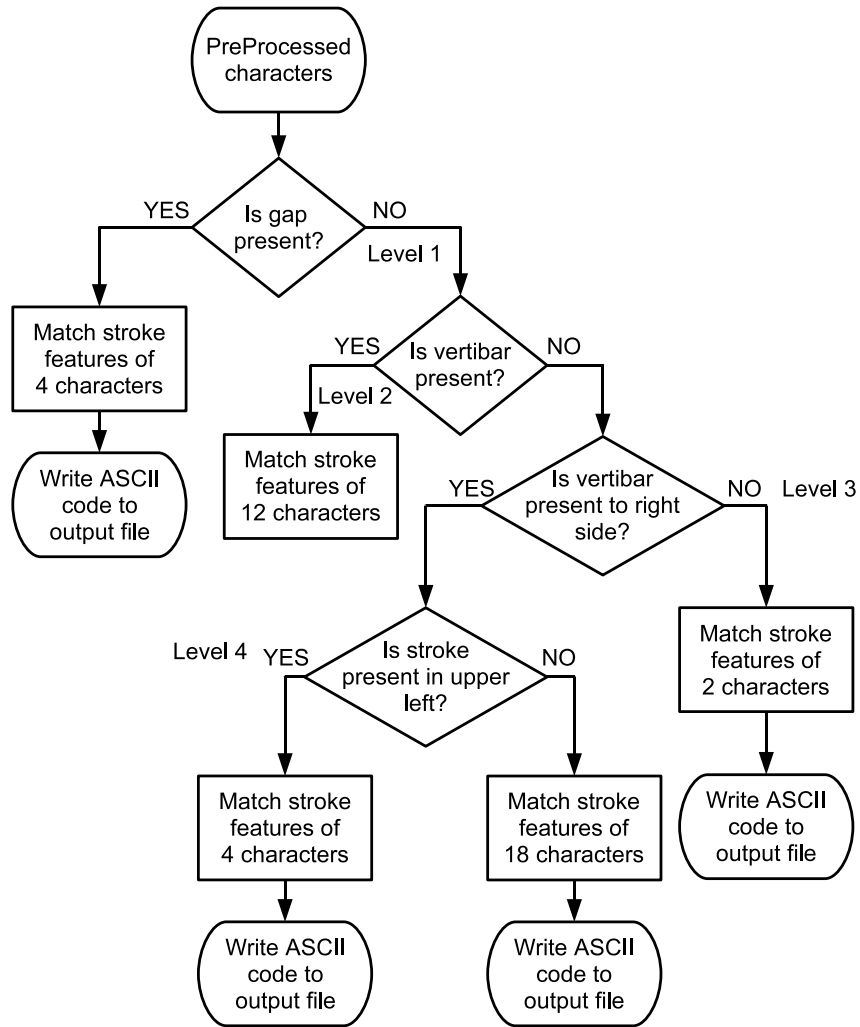


Fig. 14: Flow chart for the classifier.

Characters identified with gaps are matched with the stored model of four characters shown in Fig. 5(a). The non-gap characters are tested for the presence of vertibar. At this node vertibar and non-vertibar characters are separated thereby reducing the number of characters to be matched at the non-vertibar category to 12. The characters identified with vertibar are again split in two classes: vertibar in center and vertibar in the end. This splits the characters again in groups of 2 and 22. After this, 4 characters with high activity in the upper left region are separated and recognized. At this node, tree is split into two branches. After reaching the end nodes unordered stroke matching of the unknown character as described earlier is performed with the stored model as shown in Fig. 13. The flow chart of the classifier is shown in Fig. 14.

6. Experimentation and Results

As no standard database is available for handwritten Devnagari characters, 3 samples of 48 characters were collected from 250 persons belonging to different age groups and social backgrounds. 60% of the database is used for feature extraction. Feature vectors of three



Fig. 15: Stroke Segments (Inverted Image).

prototypes of each character are stored. The remaining database is used for validation. The results for features extracted for character 'A' shown in Fig. 15 are given in Table 2.

Table 2: Results of different features for image in Fig. 15.

k	1	2	3	4	5	6	7
L_k^*	38	33	11	3	13	17	19
R_k	0.8659	0.2298	0.5417	0.5365	0.5446	0.7396	0.3229
C_k	0.3732	0.4167	0.3611	0.5463	0.7500	0.8920	0.9111
REL_k	1.0000	0.8684	0.2895	0.0789	0.3421	0.4474	0.5000
CIR_k	0.8503	0.6976	1.0365	1.2019	1.0030	1.0017	1.0014
$AREA_k^*$	576	384	48	12	28	54	60
$Dleft_k^*$	786	765	234	118	567	867	984
Dl_k	2.7292	3.9844	9.7500	19.6667	40.5000	32.1111	32.8000
$\mu left(S_k), = 2$	0.8989	0.8524	0.6389	0.2716	0	0	0
$\mu left(S_k), = 3$	0.7910	0.7038	0.3732	0.0568	0	0	0
$\mu left(S_k), = 4$	0.5870	0.4501	0.1064	0.0015	0	0	0
$ACDC$	6543	898765	4	7	4	67	6

* values in pixels

In Fig. 16 (a), (c), (e) thinned characters of three different samples of 'YA' are shown and their segments are shown in Fig. 16(b),(d) and (f) respectively. The segments are shown inclusive of the topline. Results for fuzzy membership to left side of the frame for the segments obtained are given in Table 3 for four different values of parameter β . As can be observed value 2 gives good judgment of leftness. The top value 1-6 indicates the segment number.

Accuracy of segmentation is given in Table 4. The poorest segmentation accuracy is for character 'AU'. Overall accuracy of segmentation is 71.19% including characters without top modifier identified as characters with top modifier. Classification results of our database at different nodes in the classifier tree are given in Table 5. The system rejects 1.1% of the characters that are broken or do not match with any of the stored features of strokes. Recognition accuracy for characters with top modifier (matra) is 71.68 % and without top modifier (matra) is 88.33 %. Overall recognition accuracy is 86.4%.

Errors are observed when the character is not segmented accurately especially when there is overwriting. Due to similarity in some characters shown in Fig. 17(a) , high rate of error is observed in these characters. Some characters are written in structurally different ways depending on the educational and regional background of the writer as shown in Fig. 17(b) Fig. 17(c). Additional models have to be considered to recognize them.

7. Conclusions

The proposed work presents recognition of handwritten Devnagari characters free from normalization thereby giving flexibility and allowing size variation. Database has been col-

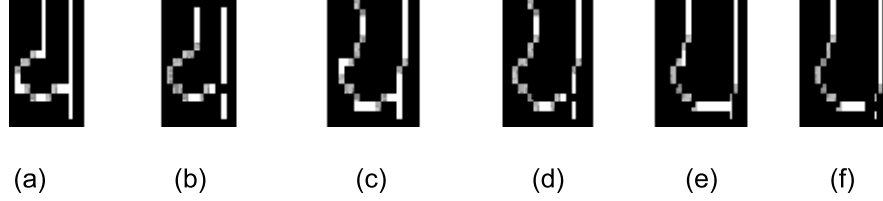


Fig. 16: Different samples of Character YA and their segments.

Table 3: Results of Fuzzy Membership to left side of image of segments of characters in Fig. 16.

Character	$\mu_{left}(S_1)$	1	2	3	4	5	6
'YA1'	$\beta = 1$	0	0.6854	0	0	0	0
	$\beta = 2$	0	0.4698	0	0	0	0
	$\beta = 3$	0	0.3220	0	0	0	0
	$\beta = 4$	0	0.2207	0	0	0	0
'YA2'	$\beta = 1$	0.1765	0.7664	0	0	0	0
	$\beta = 2$	0.0311	0.5873	0	0	0	0
	$\beta = 3$	0.0055	0.4501	0	0	0	0
	$\beta = 4$	0.0010	0.3450	0	0	0	0
'YA3'	$\beta = 1$	0.2222	0.7590	0	0	0	0
	$\beta = 2$	0.0494	0.5760	0	0	0	0
	$\beta = 3$	0.0110	0.4372	0	0	0	0
	$\beta = 4$	0.0024	0.3318	0	0	0	0

lected from writers of varied background. Modified thinning algorithm needed for separating characters in its constituent strokes has been developed and tested successfully. Modified direction codes (ACDC) algorithm, though simple, works efficiently to detect curves and turning points. The overall accuracy obtained is 86.4%. It can be compared with the result of 81% in [13], although database is different. Also this algorithm avoids resizing, a necessity when statistical techniques as in [13] are used. 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. Contextual recognition will enhance accuracy as will 'learning' from mistakes. Rejection also plays an important role in system evaluation. 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 it also has application in forensic science.

References

- [1] S. Kompalli, S. Kayak, S. Setlur and V. Govindaraj, *Challenges in OCR of Devnagari Documents*. Proceedings of Eighth International Conference on Document Analysis and Recognition (ICDAR'05), pp. 327- 331, Seoul, South Korea, September,2005.
- [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 (ICDAR'05), pp. 1156- 1160, Seoul, South Korea, September 2005

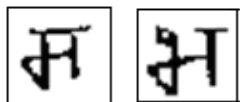
Table 4: Results of Top Modifier Segmentation.

Set Type	Number of Characters	Segmentation Accuracy of 'Ai' in %	Number of Characters	Segmentation Accuracy of 'Au', %	Total No. of Characters	Segmentation Accuracy, Average, %
Training	125	87.1	125	72.25	625	84.13
Testing	125	86.2	125	69.15	625	82.79

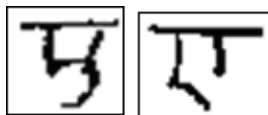
Table 5: Results of recognition at different levels in tree classifier.

Type of set	Level 1 (in %)	Level 2 (in %)	Level 3 (in %)	Level 4 (in %)
Accuracy of Training Set	98.15	95.24	95.35	90.4
Accuracy of Testing Set	90.23	87.56	88.50	80.4

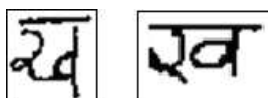
- [3] I.K. Sethi, *Machine Recognition of Online Handwritten Devnagari Characters*. Pattern Recognition, Vol. 9, pp. 69 - 75, 1977.
- [4] R. M. K. Sinha and H. N. Mahabala, *Machine Recognition of Devnagari Script*. IEEE Transactions on Systems, Man and Cybernetics, Pattern Recognition, Vol. 9(8), pp. 435 - 441, 1979.
- [5] 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, pp. 800 - 804, Seattle, USA, September, 2001.
- [6] K. Keeni, T-Nishino, H. Shimodaria and Y. Tan, *Recognition of Devnagari characters using Neural Networks*. IEICE Transactions on Information and Systems, Vol. E79-D, No.5, pp. 523 - 528, May, 1996.
- [7] 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, pp. 1011 - 1015, Ulm, Germany, August, 1997.
- [8] S. Kompalli, S. Setlur, V. Govindaraju and R. Vemulapati, *Creation of data resources and design of an evaluation test bed for Devnagari script recognition*. Proceedings of the thirteenth International Workshop Research Issues on Data Engineering: Multi-lingual Information Management, pp. 55 - 61, Hyderabad, India, March, 2003.
- [9] U. Bhattacharya and B. B. Chaudhuri, *Databases for Research on Recognition of Handwritten Characters of Indian Scripts*. Proceedings of Eighth International Conference on Document Analysis and Recognition, pp. 789 - 793, Seoul, South Korea, September, 2005.
- [10] S. D. Connel, R.M.K. Sinha, A. K. Jain, *Recognition of Un-constrained On-Line Devnagari Characters*. Proceedings of ICPR, pp. 2368 - 2371, Barcelona, Spain, September, 2000.
- [11] Prachi Mukherji, Vaishali B Gapchup and Priti P. Rege, *Feature Extraction of Devnagari Characters using sub bands of Binary Wavelet Transform*. Proceedings of RETIS-06, pp. 176 - 179, Kolkata, India, July, 2006.
- [12] 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, pp. 178 - 184, Hubli, India, December, 2006.
- [13] N. Sharma, U. Pal, F. Kimura and S. Pal, *Recognition of Off-Line Handwritten Devnagari Characters Using Quadratic Classifier*. Proc. of Indian Conference on Computer Vision Graphics and Image Processing (ICVGIP), India, pp. 805 - 816, Madurai, India, December, 2006.
- [14] Shamik Sural and P.K. Das, *Recognition of an Indian Script using Multilayer Perceptrons and Fuzzy Features* Proceedings of Sixth International Conference on Document Analysis and Recognition, pp. 1120 - 1124, Seattle, USA, September, 2001.
- [15] M. Hanmandlu, K.R. Murali Mohan and S. Chakraborty, *Fuzzy Logic based Handwritten Character Recognition*. Proceedings of International Conference on Image Processing, Vol 3, pp. 42 - 45, Thessaloniki, Greece, October, 2001.
- [16] Timothy J. Ross, *Fuzzy Logic with Engineering Applications*. MCGraw Hill International Editions, New York, 1997.



(a) Structurally similar characters.



(b) Same character written differently.



(c) Same character written differently.

Fig. 17: Characters with special properties.

- [17] N.Otsu, *A threshold selection method from gray level histograms*. IEEE Transactions on Systems, Man and Cybernetics, 9(1), pp. 62-66,1979.
- [18] Rafael C Gonzalez and Richard E.Woods, *Digital Image Processing* Second Edition, Pearson Education,2003.
- [19] Prachi Mukherji, Priti P Rege and Leena K. Pradhan, *Analytical Verification System for Handwritten Devnagari Script*. Proceedings of the Sixth IASTED VIIP, pp. 237-242, Palma De Mallorca, Spain, August,2006.
- [20] B.B. Chaudhari and D. Datta Majumder, *Two Tone Image Processing and Recognition* Wiley Eastern Limited,1993.
- [21] S. Rajasekaran and G.A. Vijaylakshmi Pai, *Neural Networks, Fuzzy Logic, and Genetic Algorithms* Prentice Hall of India Pvt. Ltd,2006.
- [22] R. O. Duda, P. E. Hart and David G. Stork, *Pattern Classification*. Second Edition, Wiley Student Edition,2001.