# Hierarchical Recognition System for Machine Printed Kannada Characters

*Dinesh Achaya U, N V Subba Reddy and Krishnamoorthi*

*Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal University, Manipal-576104 India*

*Abstract*— Extensive research has been done on optical character recognition in the last few decades. Most of the efforts were made to develop OCR systems for foreign languages like English, Japanese, Roman and Arabic characters. Many commercial OCR systems for these foreign languages are available in the market. In the context of Indian languages, majority of work is reported on Hindi and Bangla. And very few reports are available on South Indian languages. This paper describes a character recognition system that can handle machine printed text documents in Kannada, which is the official language of the South Indian state of Karnataka. Initially, the scanned image is preprocessed to remove noise. Lines, words and character components are segmented using two-stage segmentation technique. Classification of the character components is done in two stages. In the first stage, the character components are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized using a nearest neighbor classifier. We adopted this hybrid approach instead of using only a tree classifier because it is nearly impossible to find a set of stroke features that are simple to compute, robust and reliable to detect, and are sufficient to classify a large number of basic and complex shaped compound characters. The system is tested with the data set containing 8400 characters of different font and size. On average, the system recognizes characters with an accuracy of about 92.68%.

*Index Terms*— Character recognition, Structural features, Direction code, Binary decision tree, k-Nearest Neighbor, Multi-stage classifier.

## I. INTRODUCTION

Character recognition is one of the benchmark problems of Artificial Intelligence research. Optical character recognition (OCR) is a process of automatic computer recognition of characters in optically scanned and digitized pages of text. OCR is one of the most fascinating and challenging areas of pattern recognition with various practical application potentials. It can contribute immensely to the advancement of an automation process and can improve the interface between man and machine in many applications[3],[4],[5].

Some practical application potentials of OCR system are: Reading aid for the blind, Automatic text entry into the computer for desktop publication, library cataloging, ledgering, etc., Automatic reading for sorting of postal mail, bank cheques and other documents, Document data compression, Language processing, Multi-media system design, etc.

OCR for Indian languages in general is more difficult than for European languages[1] because of the large number of vowels, consonants, and conjuncts(combination of vowels and consonants). Further, most scripts spread over several zones. Segmentation has to deal with the positioning of the conjuncts and half syllables. These factors coupled with the inflectional and agglutinative nature of Indian languages make the OCR task quite challenging. Language models and computational linguistics as pertains to Indian languages is an area of recent research.

Traditionally, pattern recognition techniques are classification as template- and feature-based approach [6],[8],[9]. Early OCR systems employed only template-based approach, but modern systems combine this with feature-based approaches to obtain better results. For example, the initial Bangla OCR system [7] employed feature-based approach for basic characters, while template matching for compound character recognition.

Feature-based approaches derive important properties (features) from the test patterns and employ them in a more sophisticated classification model. The feature-based approaches can be of two types, namely spatial domain and transform domain approaches [10],[11]. In the context of Indian script OCR, spatial domain features are mostly used for various scripts like Bangla, Devnagari, Tamil, Telugu, etc. However, singular value decomposition and Cosine transform have also been used.

Some studies on Hindi, Gurumukhi, Oriya, Punjabi, Telugu, Gujrathi, Tamil, Kannada [1] etc are reported. To the best of our knowledge OCR development work

Manuscript received November 5, 2008
Manuscript revised November 20, 2008

on Kannada scripts are at its infancy and a very few standard work has been reported for Kannada documents [12],[13]. A font and size independent OCR system for printed Kannada documents is reported by Ashwin and Sastry [12]. The system first extracts words from the document image and then segments these into sub-character level pieces. The segmentation algorithm is motivated by the structures of the script. A set of zoning features is extracted after normalization of the characters for recognition. The final recognition is achieved by employing a number of 2-class classifiers based on the support vector machines (SVM). An on-line system for Kannada characters is described by Rao and Samuel [13]. The described system extracts Wavelet features from the contour of the characters. The convolutional feed-forward multi-layer neural network is used as the classifier.

In this paper, a character recognition system for machine printed Kannada characters is presented. Initially, the scanned image is preprocessed to remove noise. Lines, words and character components are segmented using two-stage segmentation technique. Classification of the character components is done in two stages. In the first stage, the character components are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized using a nearest neighbor classifier. A lexicon will combine the proper sequence of character components into a complete character.

## II. KANNADA SCRIPT

Kannada script[2] is the visual form of Kannada language. It is a south Indian language spoken in Karnataka state of India. It originated from southern Bramhi lipi of Ashoka period. It underwent modifications periodically in the reign of Sathavahanas, Kadambas, Gangas, Rastrakutas, and Hoysalas. Even before seventh-Century, the Telugu-Kannada script was used in the inscriptions of the Kadambas of Banavasi and the early Chalukya of Badami in the west. From the middle of the seventh century the archaic variety of the Telugu-Kannada script developed a middle variety. The modern Kannada and Telugu scripts emerged in the thirteenth Century. Kannada script is also used to write Tulu, Konkani and Kodava languages.

Kannada along with other Indian language scripts shares a large number of structural features. The writing system of Kannada script encompasses the principles governing the phonetics and a syllabic writing systems, and phonemic writing systems (alphabets).

The Kannada block of Unicode Standard (0C80 to 0CFF) is based on ISCII-1988 (Indian Standard Code for Information Interchange). The Unicode Standard (Version 3) encodes Kannada characters in the same relative positions as those coded in the ISCII-1988 standard.

We describe here some of the properties of Kannada Script that are useful for building the recognition system[12].

The Kannada alphabet is classified into two main categories: vowels and consonants. There are 16 vowels and 35 consonants as shown in Fig. 1 and Fig. 2.

Words in Kannada are composed of *aksharas* which are analogous to characters in an English word. While vowels and consonants are *aksharas*, the vast majority of *aksharas* are composed of combinations of these in a manner similar to most other Indian scripts. An *akshara* can be one of the following, (1) A stand alone vowel or a consonant (i.e. symbols appearing in Fig. 1 and Fig. 2). (2) A consonant modified by a vowel. (3) A consonant modified by one or more consonants and a vowel.

When a vowel constitutes the whole *akshara*, the vowel normally appears at the beginning of a word.

A consonant can also form the whole *akshara* and can come anywhere in the word. These *aksharas* appear in the middle region of the line and are represented by the same glyph as shown in Fig. 1 and Fig. 2.

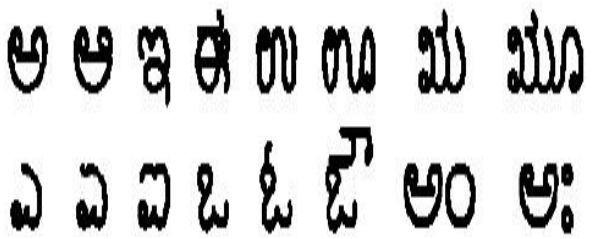


**Fig. 1** Vowels in Kannada

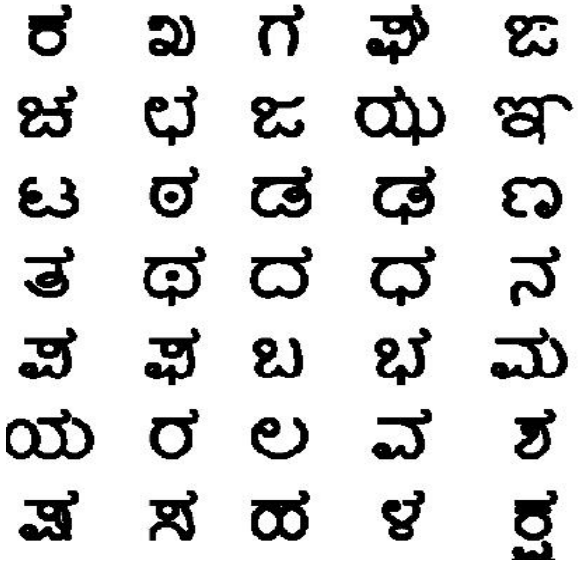


**Fig. 2** Consonants in Kannada

A consonant C and a vowel V can combine to form an *akshara*. Here the *akshara* is composed by retaining most of the consonant glyph (Fig. 2) and by attaching to it the glyph corresponding to the vowel modifier. The vowel modifier glyphs are different from those of the vowels and are shown in Fig. 3. The glyph of the vowel modifier for a particular vowel is attached to all consonants mostly in the same way, though, in a few cases the glyphs of the vowel modifier may change depending on the consonant. Fig. 3 shows two of the consonants modified by all the 16 vowels. In this figure, the second row shows the vowels, the third row shows the glyphs of the vowel modifiers, the fourth and fifth rows show the consonant–vowel(C–V ) combinations for two consonants which phonetically correspond to c as in cat and y as in yacht. As can be seen from the figure, the vowel modifier glyphs attach to the consonant glyphs at up to three places corresponding to the top, right and bottom positions of the consonant. It can be observed that the widths of the C–V combinations vary widely and also that the image of a single *akshara* may be composed of two or more disconnected components.

In the third form of *akshara* composition, consonants C1;C2 : : :Cj and a vowel V can combine to form an *akshara*. In practice j is limited to 3. The consonant C1 forms the *base consonant* and the modifier for the vowel V attaches to it.

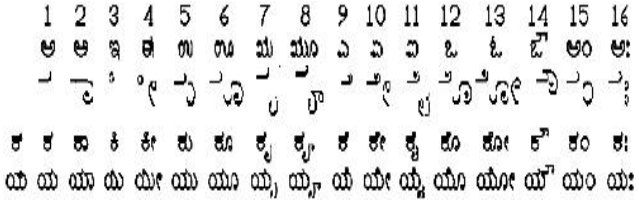



**Fig. 3** Example of consonant–vowel combinations

**Fig. 4** Consonant conjuncts in Kannada

The rules for this consonant vowel combination are the same as described above. The consonants C2 : : :Cj are called the consonant conjuncts. The glyphs of many consonant conjuncts resemble those of the consonants though there are a sizeable number of exceptions. Some of the consonant conjunct glyphs are shown in fig. 4. The consonant conjunct glyphs always appear below the C–V combination formed by C1 and V . A few examples showing *aksharas* formed by a consonant, a consonant conjunct and a vowel are shown in Fig. 5.

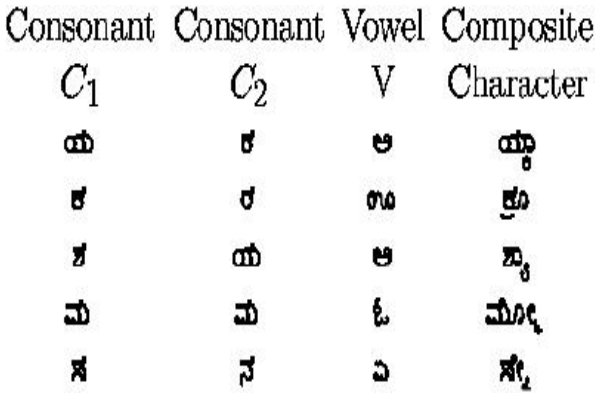



**Fig. 5** Some examples of consonant-consonant-vowel combinations.

## III. PREPROCESSING

The developed system comprises two main processes: (i) Preprocessing and (ii) Classification. The preprocessor is developed to address complexity issue of the problem domain by incorporating the segmentation mechanism. The samples to be classified are segmented to generate a set of components for each sample. Further a rule generator is used to process the information of each component of the sample mapped to obtain input in the form of decision rules. The classification of the system has been devised based on decision tree model in the first phase and k-nearest neighbor classifier in the second phase.

The preprocessing[3]-[5] process of the developed system further comprises (i) Text digitization and noise removal, (ii) Segmentation and (iii) Feature Extraction.

### *A. Text Digitization and Noise Removal*

The input image is digitized using a flatbed scanner with 300 dpi. The digitized image is binarized using histogram-based thresholding approach. The threshold value is chosen as the midpoint between two histogram peaks.

Median filtering is used to remove the noise in the binarized image. In median filtering an output pixel is determined by the median of the neighborhood pixels. The median is much less sensitive to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. It is a smoothing technique that causes minimal edge blurring.

### *B. Segmentation*

*Aksharas* in Kannada are formed by graphically combining symbols corresponding to consonants, consonant conjuncts and vowel modifiers using well-defined rules of combination. This general structure of

forming characters of a word is a feature common to many other Indian scripts. It also necessitates some special ways of segmenting a word into its constituent symbols while designing OCRs for Kannada[12].

In a language like English (written in the standard Roman script), each word consists of a linear sequence of characters written next to each other in a line. There are only fifty two possible character symbols. Since there is always some space between characters of a word, a general strategy for handling such scripts would be to segment a word into individual characters and then recognize each character separately. However, such a strategy is not feasible for Kannada script.

As in English script, in Kannada also the glyphs of *aksharas* are placed next to each other; but the *aksharas* themselves are quite complicated with considerable variation in widths and heights (Fig. 5). The number of possible consonant–vowel combinations is 35 X 16=560. The number of possible consonant–consonant–vowel combinations is 35 X 35 X 16=19600. Thus, if we consider each *akshara* as a separate category to be recognized, building a classifier to handle these many classes is very difficult. Also, such an approach does not exploit the fact that the *aksharas* are formed through well-defined geometric combination of individual symbols. Many of the letters or *aksharas* are very similar and differ only in having an additional stroke or an attachment. Structure of the script is such that it is feasible to break the *aksharas* into their constituents and recognize these components independently. This has been the chosen approach for most Indian scripts and we have considered this approach in developing the recognition system.

As can be seen from Fig. 5, the image of an *akshara* may not be a single connected component. Hence correctly segmenting the image of a word into those corresponding to individual *aksharas*, prior to recognition, is very difficult. Hence, in our system we segment the word into components each of which can be only part of an *akshara*. Ideally, we may want to split the word so that each segment corresponds to the base consonant or vowel modifier or consonant conjunct. Even this is not generally feasible because some of the vowel modifiers themselves do not correspond to a single connected component. Further, due to the structure of some of the consonant and vowel symbols, it was observed that, for getting consistent segmentation, it is easier to allow even some of the consonant symbols to be split into two or more parts while segmenting a word. Once a word is split using our segmentation algorithm, we label each piece (using a pattern recognition technique) and then combine the labels on neighboring segments to effect final recognition of *aksharas*. Due to the well defined graphical combination rules of the script, this step of combining labels of individual segments into *aksharas* is fairly simple[12].

*Line and Word Segmentation:* The segmentation[17],[18] Technique uses projection profile technique and zoning algorithm along with connected component analysis to segment the characters. The digitized image is processed to lines and words using appropriate horizontal and vertical projection profiles [1],[12],[15]. The projection profile gives valleys of zero height for these OFF pixels between the text lines. In the Fig. 6, a text document along with its horizontal histogram is shown. Segmentation of the image into separate lines is done at these valley points. A vertical projection is applied to the image for word segmentation, as shown in the Fig. 7. The space between each word, which is sufficiently large compared to inter character space, is used to determine the word boundaries. With the vertical projection profile approach OFF pixels between each word give valleys with zero value, which indicate the word boundary. This information is used to separate the words from the text lines.

*Character Segmentation*: The words are then segmented into smaller parts and given to the classifier to recognize the characters For English and some other foreign languages the normal projection profile method is enough to segment the machine printed characters. For Kannada characters the projection profile approach alone will not give the desired output, as the characters in Kannada are composed by attaching the glyph of a consonant, the glyphs of vowel modifiers and the glyphs of the consonant conjuncts.

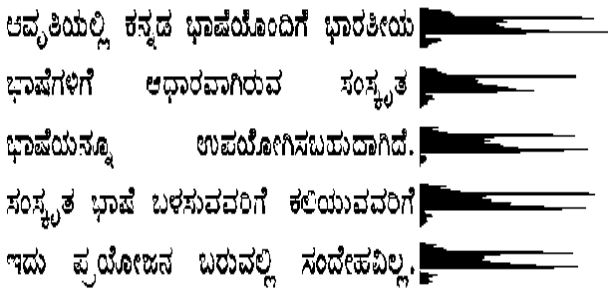



**Fig. 6** Line Segmentation by horizontal projection profile


**Fig. 7** Word Segmentation by vertical projection profile

From the recognition point of view, the characters have to be segmented into its constituents, i.e., the base consonant, the vowel modifier and the consonant conjunct. This is addressed by a two-stage approach. First, the zone level features (reference points) are extracted, in the coarse level and then in second stage, these reference points are used for connected component analysis in segmenting the characters.

**Fig. 8** Character Segmentation by vertical projection profile

Zoning is used to find the reference points, which is used to separate the top and bottom modifiers from the character. Two peak drops(Fig. 9) in the histogram values are used for dividing the text line into three zones, top zone, mid zone and bottom zone, which separates the modifier from the main character component. The top zone(Fig. 9) contains the vowel modifiers(VM) and sometimes part of the base consonant. The mid zone contains the consonant glyphs(C) and vowel modifiers. The vowel modifier glyphs may appear as either connected or disconnected components to the right of the base component. The bottom zone consists of glyphs for the consonant conjuncts(CC) and glyphs for some vowel modifiers. These glyphs generally appear disconnected from the base consonant and the vowel modifiers present in the mid zone. Zonal reference points are calculated for each line, once the line segmentation is done, and are used along with connected component analysis for the separation of components in the characters(Fig. 10). Consonant conjuncts, which penetrate into the mid zone, multiple components in the consonant conjuncts and vowel modifier on the right side of the base consonant are separated by using the connected component analysis(Fig. 10).

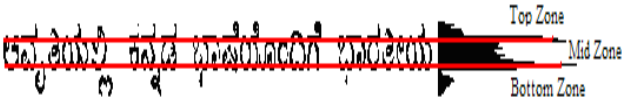



**Fig. 9** Zones in a Kannada text line

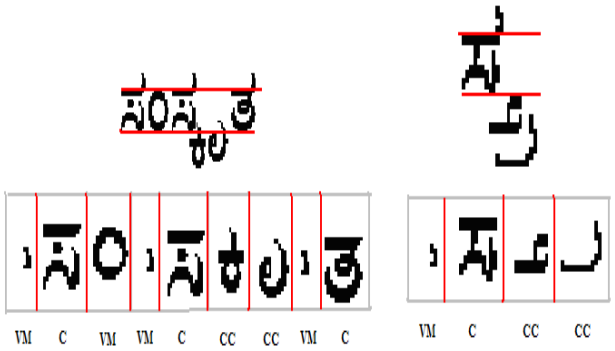



**Fig. 10** Separation of character components using zoning and connected component analysis

The character separation poses some problems due to the fact that the consonant conjuncts, which appear below the base consonant, appear frequently overlapped with the base consonant of next character in the word(Fig. 11). Such false cases can be handled, heuristically, by using statistics of the character width, separation between character components in the mid zone and number of connected components.

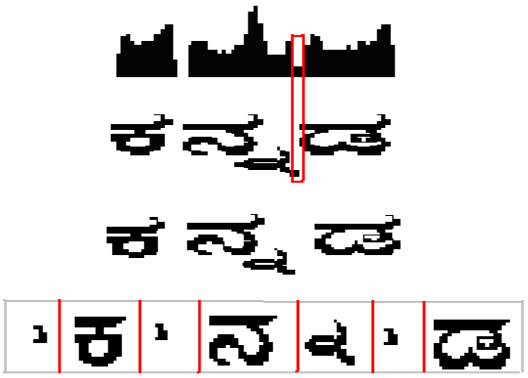


**Fig. 11** Consonant conjunct overlapping with the base consonant of next character in the word

*C. Feature Extraction*

We consider topological features and structural features for character recognition. We term these the principal features. The features are chosen with the following considerations: (a) Robustness, accuracy and simplicity of detection, (b) speed of computation, (c) independence of size and fonts, and (d) tree classifier design need[14].

We considered a few structural and topological features for the initial classification of characters. These features are used to design a tree classifier where the decision at each node of the tree is taken on the basis of

the presence/absence of a particular feature. Structural features include the presence of shirorekha, vertical and horizontal strokes and their position.. The topological features used include existence of holes, their number and position of holes with respect to character bounding box, ratio of hole height to character height etc. In addition, presence of isolated points, no. of zero crossings, aspect ratio and number of connected components are used as features in the recognition scheme.

The features selected here are simple, linear in structure and hence quick and easy to detect. They are fairly robust to noise and quite stable with respect to font variation. The methods for detecting the stroke features are described below.

'Shirorekha' corresponds to the presence of a horizontal line at the top of the image in consideration. Its presence is found by applying minimal bounding box on the character component and counting the number of character pixels in the first row of the character component. If it is more than $3/4^{th}$ of width of the character component, then it is having a shirorekha. Region descriptors function, 'regionprops' of MATLAB IPT, provide the necessary descriptors of the character component image. This includes Bounding box, Euler number, Image and Filled Image. These descriptors, along with the the connected component analysis, facilitates the calculation of number of holes, their position and size with respect to the size of character component.

Whether a side(bottom, left, right) of the character is convex or not is found by comparing the character pixels of n(=$1/4^{th}$ of character component height/width) rows from the side with that 'conveximage' descriptor of regionprops. A perfect matching character component is considered as a component having a convex shape on that side. 'Zero crossings' of a line with a character component is calculated by counting the number of zero to one change in the sequence of pixels in the specified line crossing the character component. Existence of isolated point in a particular hole is found by first extracting the required 'hole' region and then finding for holes in the inverted 'hole' image. A small vertical stroke below the 'shirorekha' is verified by the connected component analysis of character component, after the removal of 'shirorekha'.

In the second stage of classification, the direction code frequency of the character component is measured and is given to k-nearest neighbor[20] for final classification. Each character image is first thinned and then the minimal bounding box containing the numeral is divided into three equal horizontal and vertical blocks(Fig. 13). In each of these blocks, the direction chain code[19] for each contour point is noted and the frequency of the direction codes is computed. Here we use chain code of eight directions(Fig. 12). Thus, in each block we get an array of eight integer values representing the frequencies and these frequency values together form a feature set. Thus, for 3+3 blocks, we get (3+3) X 8= 48 features. To normalize features, we divide each of the frequency values by the maximum possible frequency.

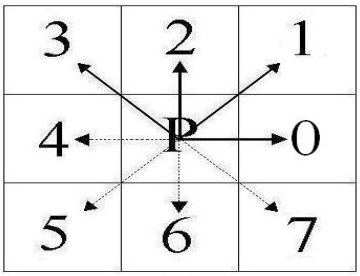



**Fig. 12.** For a point P the direction codes for its eight neighboring points are shown.

**Fig. 13** Horizontal and Vertical blocks for calculating direction code frequency.

## IV. CLASSIFICATION

Classification of the characters is done in two stages[14]. In the first stage, the characters are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized using a nearest neighbor classifier. We adopted this hybrid approach instead of using only a tree classifier because it is nearly impossible to find a set of stroke features that are simple to compute, robust and reliable to detect, and are sufficient to classify a large number of basic and complex shaped compound characters.

The design of a tree classifier has three components: (1) a tree skeleton or hierarchical ordering of the class labels, (2) choice of features at each non- terminal node, and (3) the decision rule at each non-terminal node[14]. The developed tree is a binary tree where the number of descendants from a non-terminal node is two. While traversing the tree, only one feature is tested at each non-terminal node. The selection of the feature at a particular non-terminal node is done by considering its impact on the length of the tree. If the set of patterns at a non-terminal node can be sub-divided into two sub-groups by examining a feature so that the number of elements of one group is roughly equal to that of the other group, the resulting binary tree will have zones. Thus the possible classes for the segments in the top two zones would be

the vowels, base consonants or vowel modifiers. Hence the total number of classes here should be 16 +35 + 16 = 67. The bottom zone contains mostly consonant conjuncts and hence we should have another 35 classes.

The binary tree classifiers for the character components in mid and bottom zones are given in table I to table III. At each non-leaf node, the presence of some stroke based feature is checked. The leaf nodes represent one to four unique character components. In the case of single character leaf node, the specific character is the classification result. However, in the case of multiple characters leaf node, the subsequent character recognition will be done by a nearest neighbor based on the direction code frequency feature. The mid zone character components are first checked for the presence of shirorekha. If shirorekha is present, then tree classifier of Table I is followed. Otherwise, the tree classifier represented in Table II is used. In case of bottom zone components, the tree classifier of Table III is used. However, we may not always get a set of features to design such an optimal tree. A semi-optimal tree is generated out of the available features. For a given non-terminal node, we select a feature that best separates the group of patterns in the above sense.

Each character segment is divided into top, mid and bottom components based on the zoning reference points and they are used for recognition. Each of the components contains a vowel, a vowel modifier, a base consonant, consonant conjunct or a part of them. We use three different classifiers for classifying the segments in each of the three vertical zones. Thus the possible classes for the segments in the top two zones would be the vowels, base consonants or vowel modifiers. Hence the total number of classes here should be 16 +35 + 16 = 67. The bottom zone contains mostly consonant conjuncts and hence we should have another 35 classes.

The binary tree classifiers for the character components in mid and bottom zones are given in Table I to Table III. At each non-leaf node, the presence of some stroke based feature is checked. The leaf nodes represent one to four unique character components. In the case of single character leaf node, the specific character is the classification result. However, in the case of multiple characters leaf node, the subsequent character recognition will be done by a nearest neighbor based on the direction code frequency feature. The mid zone character components are first checked for the presence of shirorekha. If shirorekha is present, then tree classifier of Table I is followed. Otherwise, the tree classifier represented in Table II is used. In case of bottom zone components, the tree classifier of Table III is used.

While traversing the tree, only one feature is tested at each non-terminal node. For example, with ಕ k as the input, first we check for shirorekha in the mid zone of ಕ k. Next, at successive non-terminal nodes(Table I), we check for the presence of holes, number of holes, whether the hole size is of character size or not, whether the hole touches the bottom line of character and finally whether the hole size is less than half the character size or not. When we reach the leaf node, since it contains only ಕ k, ಕ k becomes the result of tree classifier. However, if the input is ಜ j, the tree traversal(Table II) ends up with a leaf node containing four characters ಜಣಬಚ j, N, b, ch. In such cases, we use nearest neighbor classifier in the second stage. Seventy percent of the data set, which cannot classified by binary tree classifier, is used for training and the remaining thirty percent for the testing.

**TABLE I**
DECISION TREE FOR VOWELS/CONSONANTS WITH SHIROREKHA

<table>
<tr>
<td rowspan="11">Shirorekha: Yes<br>ಕಗ ಘಙಚ ಛಜಝಞ ಟಠಡಢ ಣತಥದ ಧನಪಫಮರ ವಶಷಸಹ<br>k, g, gh, Th, D, Dh, t, th, d, n, p, ph, m, r, v, S, Sh, s, h<br>Right Modifier: ಿ r</td>
<td rowspan="8">Holes:Yes<br>ಕಖಡಢತಥದ ಧಮರವಶಹ<br>k,Th, D, Dh, t, th, d, dh, r, v, S, h</td>
<td>No, of holes>1<br>ಡಢಹ<br>D. Dh, h</td>
<td colspan="3">Two holes of ch. size: ಹ h</td>
</tr>
<tr>
<td colspan="3">ಡಢ D, Dh</td>
</tr>
<tr>
<td rowspan="6">=1<br>ಘಙಚಛ ಜಝಟ ಠಣ<br>ಮರವಶಳಱ<br>k,Th, t, th, d, dh, r, v, S,</td>
<td rowspan="6">Size of hole: ch. size<br>ದಧ ರಶ<br>Th, th, d, dh, r, S , h</td>
<td>Isolated point in hole: Yes ಙಛ Th, th</td>
<td>Vertical line bottom modifier: Yes ಥ th</td>
</tr>
<tr>
<td>No ಶ Th</td>
</tr>
<tr>
<td rowspan="3">No<br>ದಧ ರಶ<br>d, dh, r, S</td>
<td>Bottom Curve: convex ರಶ r, S</td>
<td>Vertical line @ 1/3rd width: zero crossings = 3 ಷ S</td>
</tr>
<tr>
<td>R ರ</td>
</tr>
<tr>
<td colspan="2">No ದಧ d, dh</td>
</tr>
<tr>
<td rowspan="2">No<br>ಕತ ಮಪ<br>k, t, v</td>
<td>Hole touch bottom ಕ ಮಪ k, m, v</td>
<td>Hole size <1/2 glyph size ಮಪ V, m</td>
</tr>
<tr>
<td colspan="2">No ಕ K</td>
</tr>
<tr>
<td colspan="3">ತ T</td>
</tr>
<tr>
<td rowspan="5">No<br>ನಪಫನ ಪಫ ಷಸ<br>g, gh, n, p, ph, m, Sh, s<br>ಿ r</td>
<td colspan="4">Short Vertical line @ lower mid pt of shirorekha: yes<br>ಫಪಫ gh, p, ph → (NN Classifier)</td>
</tr>
<tr>
<td rowspan="4">No<br>ನನಷಸ<br>g, n, Sh, s</td>
<td rowspan="2">Horizontal line below shirorekha : no. of zero crossings :1 yes ನ n, ಿ r</td>
<td colspan="2">No ch. pixels below shirorekha on right edge of Bounding Box: Yes ಿ r</td>
</tr>
<tr>
<td colspan="2">No ನ n</td>
</tr>
<tr>
<td rowspan="2">No<br>ಗಷಸ<br>g, Sh, s</td>
<td colspan="2">V line at mid : no. of zero crossings: 1 ಗ g</td>
</tr>
<tr>
<td>No ಷಸ Sh, s</td>
<td>Convex Bottom: ಸ s<br>No ಷ Sh</td>
</tr>
</table>

**TABLE II**
DECISION TREE FOR VOWELS/CONSONANTS WITHOUT SHIROREKHA

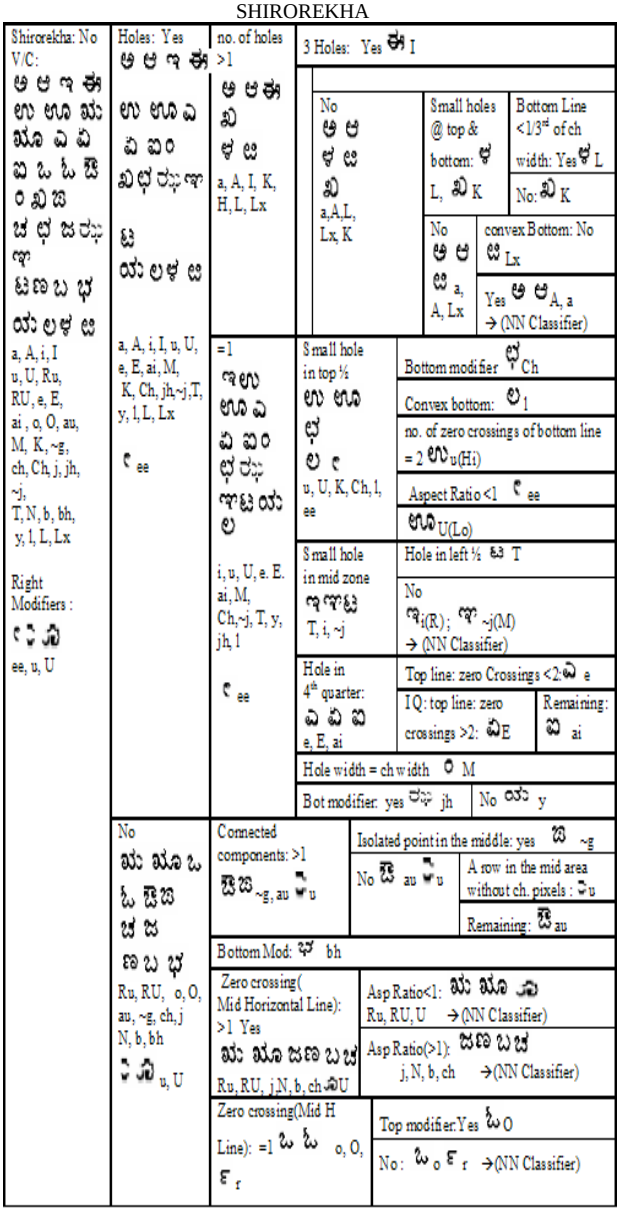
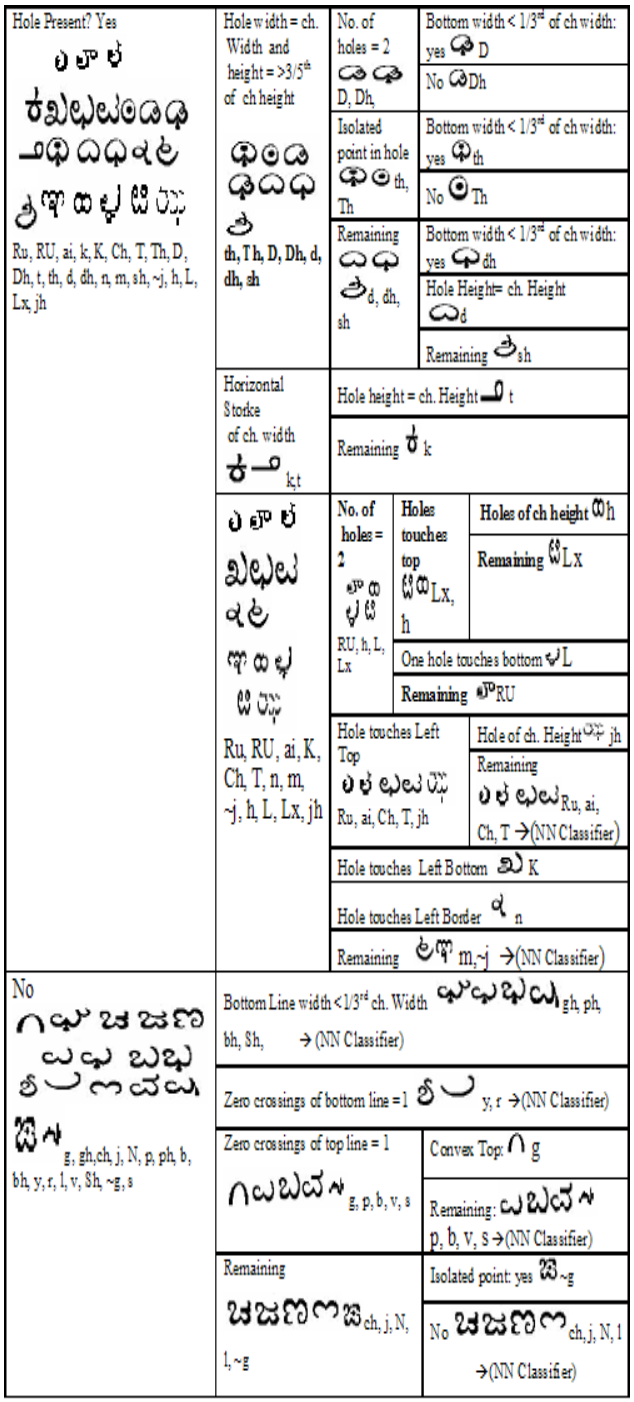



**TABLE III**
DECISION TREE FOR BOTTOM MODIFIER




The possible top modifiers are listed in Fig. 14. As the list includes only four symbols, a nearest neighbor alone is used for the identification. Once again, we use direction code frequency as the feature set. Since the symbols are less complex, while extracting the features,

we divide the modifier image into only two equal horizontal and vertical blocks. This reduces the dimension of the feature set to 4X8=32.

In most of the cases, the modifiers appear as a separate symbol(for instance, ೆ, ೦), either at the bottom or to the right of the base consonant. However, all the top modifiers(for instance, ಕ, ಿ) and some of the right modifiers(for instance, ಾ) appear connected to the base consonant. Because of zoning, the top modifiers will automatically get separated from the base consonant and facilitates the recognition. However, some extra effort is required to separate the touching right modifiers. One possible solution for this is based upon the statistics of the mid zone components in a text. We first calculate the average width of the mid zone character components. During recognition, if a mid zone component's width is 50% more than the average width and if the component has a shirorekha at least on the 50% of the component width starting from the right, then it is very likely that ಆ aa modifier is attached to the base consonant(ಕಾ, ಭಾ).

Special symbols(Fig. 16) are recognized by their size with respect to average character size, at the first level, and then they are classified by nearest neighbor classifier, at the second level. This group also include ಃH vowel modifier. Before entering vowel/consonant decision tree, the character component of mid zone should be verified for special symbols.

## V. RESULTS AND DISCUSSION

Majority of the Kannada characters are composed of a base consonant and a vowel modifier or/and consonant conjunct. Hence the recognition of characters is done in two stages. First the classification of character components is done, followed by combining the proper sequence of character components into a complete character using a lexicon[16]. The result generated by the classifier is used by the lexicon to take the decision to combine the character components. A simple lexicon based on dynamic finite state automata is used to support the recognition process. All possible modes of splitting of the characters is known and this information is encoded in a lexicon. The lexicon uses a look-up table to generate the ASCII text for each character. The ASCII string generated by the Lexicon is supplied to the Kannada editor. For a given series of character components the Lexicon is searched which is able to retrieve a list of N-best combinations that match the correct character.

The main difficulty faced with Kannada character recognition is the non-availability of a standard data set. The data set used in this work has been prepared from the available fonts Baraha multilingual editor. The fonts used

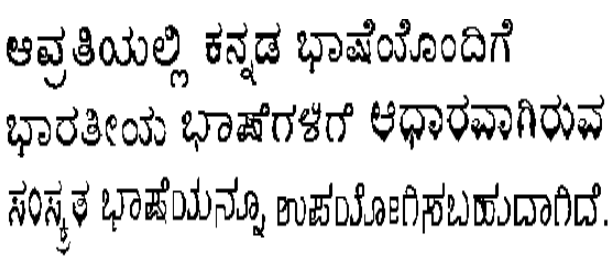



**Fig. 17** A sample Kannada Data Set with multiple fonts

include BRH Kannada, BRH Kannada RN, BRH Kannada Extra, BRH Vijaya, BRH Kailasam, and BRH Sirigannada.

A sample set with multiple fonts is given Fig.3.17. Multiple pages of Kannada text with a combination of multiple fonts and size are prepared and scanned through a flat bed scanner at 300 DPI and binarized. It includes a total of 8400 characters. Effort is made to include every possible character component in the sample data set. The results of various stages of the system are summarized below.

*Line segmentation:* Our system identifies individual text lines with an accuracy of 100%. Occasionally, when two adjacent text lines are close to each other, the lower zone(e.g., ಸ್ಥೈ of the upper line has some overlap with the upper zone (e.g., ಕೌ)of the lower line. In such situations, there is no clear valley between the two lines in the projection profile, and our system fails to detect the boundary between the two lines. Such cases are handled by using the statistics of the text. Whenever the gap between successive valley points of zero height, in the projection profile, is an abnormal value, we use the average height of different lines in the text as a reference value for segmentation.

*Word segmentation:* The overall word segmentation accuracy of the system is 98.8%. The threshold value, chosen for the inter-word gap based on the statistics of the text, works well in most cases. However, because of non-uniform printing, some words are printed closer together and lead to word segmentation error.

*Character segmentation:* The character segmentation accuracy of the system is 97.6%. The proposed method for separating overlapping characters(Fig. 10) based on the text statistics is generally successful.

*Character recognition:* On average, the system recognizes characters with an accuracy of about 92.68%., i.e., the overall error rate is 7.32%. With respect to different category of character components, recognition rate of the system is 92.32% for vowels, 91.41% for

consonants 94.67% for vowel modifiers and 89.64% for consonant conjuncts. A comparison of the proposed method with the contemporary work is given in Table IV.

## VI CONCLUSION

An automatic recognition of printed Kannada script employing two stage classifier based on structural and topological features is implemented. The result generated by the classifier is used by the lexicon to take the decision to combine the character components. Binary tree classifier is used in the first stage. If a leaf node of the tree contains only one character component, then that is the classification result. However, if it represents a group of character components, then the final classification is done by a nearest neighbor classifier in the second stage. The system is tested with the data set containing 8400 characters of different font and size. On average, the system recognizes characters with an accuracy of about 92.68%., i.e., the overall error rate is 7.32%. With respect to different category of character components, recognition rate of the system is 92.32% for vowels, 91.41% for consonants 94.67% for vowel modifiers and 89.64% for consonant conjuncts.

**TABLE IV**
COMPARISON OF PROPOSED METHOD WITH THE CONTEMPORARY WORK

| Method | Segmentation | Feature Extraction | Classifier | Data Size | Recognition Results(in %) | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | Vowels | Consonants | Vowel Modifiers | Consonant Conjuncts | Overall |
| Shastri [12] | Heuristic Merging; Recognition Based Approach | Pixel density in circular grid | SVM | 3000 | 93.78 | 93.78 | 87.21 | 94.9 | NA |
| Proposed Method | Connected Component Analysis | Stroke and Topological | Tree and k-NN | 8400 | 92.32 | 90.41 | 94.67 | 87.64 | 92.68 |

## REFERENCES


[1] U Pal, B B Choudhuri: Indian Script Character Recognition: A Survey. Pattern Recognition, Vol. 37,pp. 1887-1899, 2004

[2] Technology Development for Indian Languages: Kannada Script. http://tdil.mit.gov.in/

[3] R. Plamondon and S. N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 2000, pp. 63- 84.

[4] S. Mori, C. Y. Suen and K. Yamamoto, "Historical Overview of OCR Research and Development", *Proceedings of the IEEE*, 80, 1992, pp. 1029-1058.

[5] S. N. Srihari and S. W. Lam, "Character Recognition", *Technical Report*, 1995, CEDAR-TR-95-1.

[6] L. O' Gorman, R. Kasturi, "Document Image Analysis", *IEEE Computer Society Press*, Los Alamitos, CA, 1995.

[7] B. B. Chaudhuri, U. Pal, "A complete printed Bangla OCR system', *Pattern Recognition* 31 (1998) 531– 549.

[8] V.K. Govindan, A.P. Shivaprasad, "Character recognition— a survey", *Pattern Recognition* 23 (1990) 671–683.

[9] S. Wu, P. Shi, "Unconstrained hand written numeral recognition using Hausdor8 distance and multi- layer neural network classifier", *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, 1999, pp. 249–252.

[10] S.A. Mahmoud, 'Arabic character recognition using Fourier descriptors and character contour encoding', *Pattern Recognition* 27 (1994) 815–824.

[11] Wang, J.M. Mendel, "A fuzzy approach to handwritten rotation invariant character recognition'. *Proceeding of International Conference on ASSP*, 1992, pp. 145–148.

[12] T V Ashwin, P S Sastry: A font and size-independent OCR system for printed Kannada documents using support vector machine. *Sadhana*, vol. 27, pp. 35-38, Feb. 2002

[13] R.S. Rao, R.D. Sudhaker Samuel, "On-line character recognition for handwritten Kannada characters using Wavelet features and Neural classifier", *IETE J. Res*. 46 (2000) 387–392.

[14] B.B. Chaudhuri, U. Pal, M. Mitra, "Automatic Recognition of Printed Oriya Script", *Sadhana 27*, 2002, 23–34.

[15] U Pal and Anirban Sarkar: Recognition of Printed Urdu Script. In the proceedings of Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)

[16] E Kavallieratou, K Sgarbas, N Fakotakis and G Kokkinakis: Handwritten Word Recognition based on Structural Characteristics and Lexical support. In the proceedings of Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)

[17] U Pal and Sagarika Datta: Segmentation of Bangla Unconstrained Handwritten Text. In Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, Scotland

[18] Shuyan Zhao, Zheru Chi, Penfei Shi and Hong Yan: Two-Stage Segmentation of Unconstrained Handwritten Chinese Characters, Pattern Recognition. Vol. 36, pp. 145- 156, November 2001

[19] N. Sharma, U. Pal, and F. Kimura, "Recognition of Handwritten Kannada Numerals", *Proc, of IEEE-ICIT* 2006.

[20] Cover, T.M., Hart, P.E. "Nearest neighbor pattern classification", *IEEE Trans. Inform. Theory*, IT-13(1):21–27, 1967.