Character Recognition using Discrete Curve with the Use of Approximate String Matching

Samit Kumar Pradhan
M. Tech. (Computer Science)
Department of Computer and Information Science
University of Hyderabad

Sujoy Sarkar
M. Tech. (Computer Science)
Department of Computer and Information Science
University of Hyderabad

ABSTRACT

This paper deals with the recognition of printed basic Telugu characters using the discrete curves and approximation string matching. The features are extracted from smoothed images, obtained after the thinning operation. As by only thinning, spines may arise which will affect the recognition of the character. The features are the discrete curves, specified using the 3×3 regions of connected component representation. We represent the discrete curves in the form of a string, so the set of discrete curves result a set of strings. Then using the string matching operation we compare the string obtained from the stored character with the string obtained from the extracted character. As we are dealing with the characters so there may be the presence of noise which will affect the performance of the method so we are considering the approximation string matching instead of the exact string matching. The extracted features of the character are represented as a string and the string is stored in a trie data structure so that a uniform time will take to compare the strings. For the efficient approximate string matching we are using the Look ahead branch and bound scheme with the trie. We apply our method on 42 printed basic Telugu characters for demonstration and it gives promising results. However more extensive study on realistic data is required for betterment of the approach.

General Terms

Basic Telugu character, Optical character Recognition

Key words

Discrete curve, Approximate string matching, Trie, Look Ahead Branch and Bound

1. INTRODUCTION

Pattern Recognition in image processing [1, 2] encompasses several areas of research, viz. face recognition, signature recognition, text recognition and fingerprint recognition. High accuracy text recognition or optical character recognition (OCR) is a challenging task for Indic scripts. The OCR research in English character is at matured position. For the Indic scripts like Telugu we have proposed a new strategy which involves discrete curves, trie and look ahead branch bound pruning.

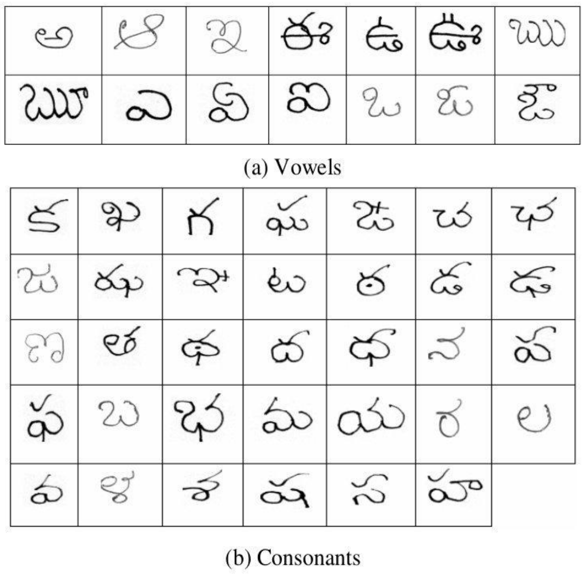


Fig. 1. Examples of Telugu basic characters

This paper primarily focuses on the recognition of basic Telugu characters shown in Fig. 1. There are 52 Telugu characters with vowels and consonants modifiers. These 52 characters can be modified to result in more than a thousand. Here we only consider about the 52 basic characters of Telugu scripts.

Various character recognition schemes have been proposed for the OCR [3]. This paper introduces new character recognition technique based on the connected component representation of the discrete curves. The image of a character can be represented as a set of discrete curves each of size 3×3. We use the connected components for the representation of the discrete curves in a string format. After that the string matching algorithm can be applied for the matching of the characters.

Let *X[1 ... N]* be the string representation of a training character of length *N* stored in the database which is implemented in a trie and *Y[1 ... M]* be the string obtained from the connected components of an unclassified character. *Y* may contain insertion, deletion and substitution errors. There are many algorithms for the approximate matching of the two strings. We include a brief review later section. We use one of the approximation scheme for the matching of the *X* and *Y* .

Trie is a data structure that offers search cost that is independent of the document size as it only depends on the maximum string length present in the document. Trie also combines prefixes together, and so by using the trie in the

approximate string matching [4], we can utilize the information obtained by evaluating the $D(X_i, Y)$ to compute other $D(X_j, Y)$. Here X_i and X_j share common prefix and $D(X, Y)$ be the string edit distance between two string X and Y. As opposed to this in artificial intelligence Branch and Bound (BB) techniques are there [5] and have been used to prune paths that have costs above a certain threshold.
The organization of this paper as follows, Section 2 presents a idea of the related work in this field. Section 3 presents some background that are used in this paper, discrete curve, connected component, string matching. Section 4 presents the details of the technique used in this paper, the character recognition procedure using this proposed method. Section 5 presents the results and discussion related to this scheme and section 6 concludes the paper.

2. BACKGROUND

In the present work we propose a new method of recognizing Telugu characters. Before there are many approaches for the character recognition like using fringe distance, template matching[6] and wavelet analysis [7]. Here we only focus on the character recognition not on the extraction of the characters from a document image [8]. We make use of connected component analysis to capture the invariant features of the Telugu scripts. The basic characters of this script are inherently circular in nature. Unlike Latin script, Chinese script, Devnagari or Bengali script, Telugu characters very rarely contain horizontal, vertical or diagonal line. We observe that the Telugu characters are obtained by joining circular shapes (full or partial) of different sizes with some modifiers. The modifiers are either of circular shape or oblique linear strokes.

2.1 Scanning and Size Normalization

We first scan a text as grey-scale image at appropriate resolution (at present we scan with 300 dpi resolution). The text may consist of running text material of multiple pages. In this present study we assume that the text is free of mathematical symbols, figures, or tables. The digitized image is segmented to extract individual characters. We follow the projection (horizontal, for line segmentation and vertical for the word segmentation) techniques. Each individual character can be of different size and hence, we normalize them to images of fixed size. We observe that though sizes vary for different characters, a frame of 32×32 is appropriate as the normalized size to contain distinct features. The Fig. 2. shows the normalization process.

2.2 Thinning and Smoothing

Thinning [9] is a morphological operation that is used to remove selected foreground pixels from binary images, but is particularly useful for skeletonization. In this mode it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output. After the scanning of the gray image

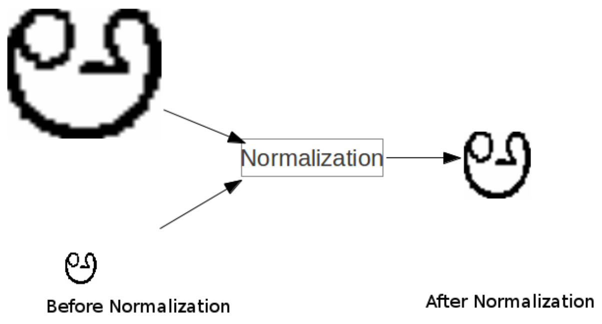


Fig. 2. The normalization process which normalized to a standard size of 32×32

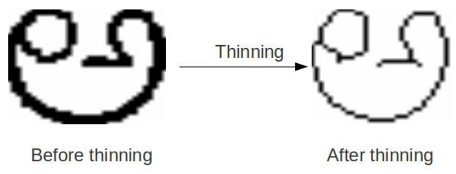


Fig. 3. A character before and after the thinning [9]

the image font may be different from the stored font of the character. We have to thin the image so that the character will be like a curve representation. The thinning operation is shown in the Fig. 3. After the thinning the character contains various spines. So a technique called smoothing is used to remove the spines from the thinned character. The purpose of the smoothing is to recovering an image from the noisy or blurred version of an image.

2.3 Discrete curve

A discrete arc [10] is a sequence of different adjacent pixels which don't touch on sides other pixels than neighbors in the sequence and don't touch on corners other pixels than neighbors in the sequence. A discrete arc is a set of elements of a one-to-one pixel sequence that satisfies,
For $I \subset Z$ and Type equation here. $I \subset Z^2$ discrete continuity of functions $f : I \rightarrow U$ and $g : U \rightarrow I$ will be described by the following conditions [10]:

(1) $\forall\, m, n \in I,\ |m - n| = 1 \Longrightarrow ((f(m) = f(n)) \vee (f(m) \odot f(n)))$,
(2) $\forall\, P, Q \in U,\ (P \oplus Q) \Longrightarrow |g(P) - g(Q)| \leq 1) \,\&\, (P \otimes Q \Longrightarrow |g(P) - g(Q)| \leq 2)$,

Where the symbols "$\oplus$", "$\otimes$", "$\odot$" denotes the direct neighborhood (pixels share a side), the indirect neighborhood (pixels touch only at a corner) and the neighborhood (direct and indirect) respectively.
A discrete curve is a sum of discrete arcs such that any two arcs can have only common ends. The more important is internal pixels of one arc (except direct neighbors of arc ends) cannot touch pixels of another arc. In other words when we represent a part of the curve in 3×3 region the possible combination of white and black pixels represents a discrete curve. The possible set of discrete curves is given in Fig. 4.

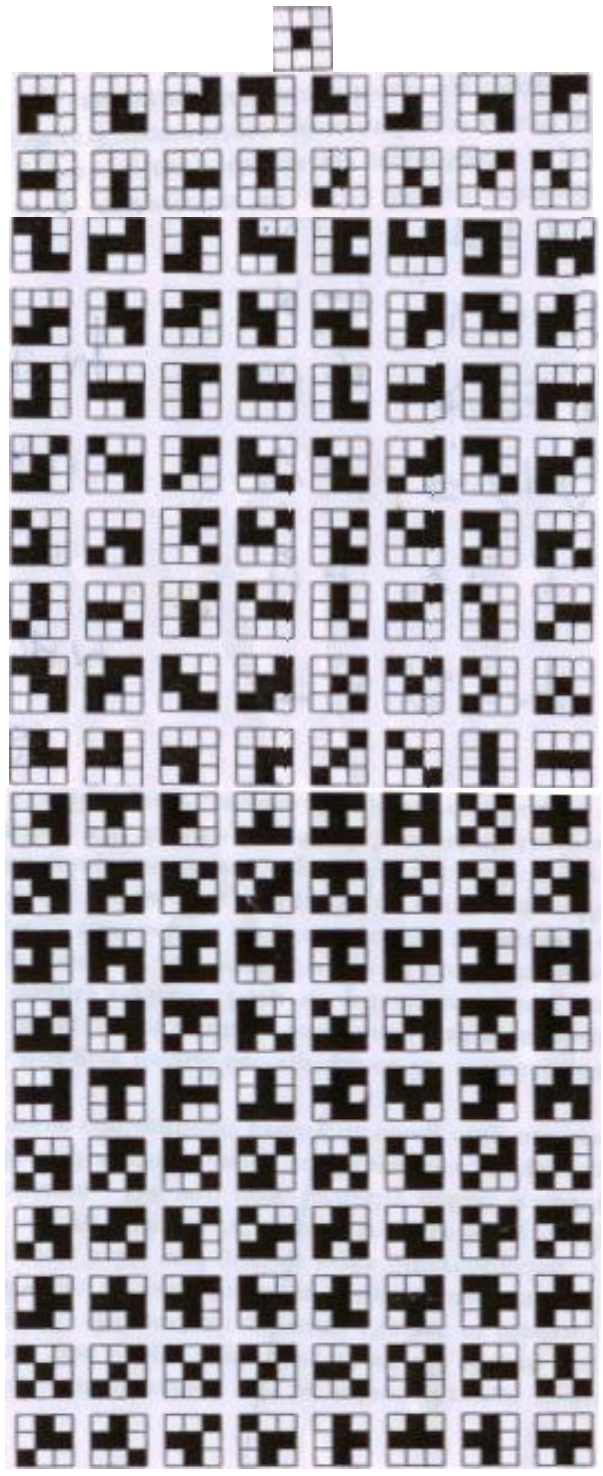


Fig. 4. The possible set of discrete curves[10]

2.4 Trie Data Structure

Trie is a data structure [11] which offers text searches independent of the document size. The searching cost depends on the maximum length word present in the trie. Its searching complexity is *O(m)* in worst case, where m is the length of the longest string. The data are represented not in the nodes but in the path from the root to the accepting node. Thus, all strings sharing a prefix will be represented by paths branching from a common initial path. Fig. 5. shows an example of a trie for a simple dictionary with the following number strings 135, 1356, 1357, 46, 2347, 24, 2469, 4678, 4689. In the trie of Fig. 5. the accepting node is labeled as black and a key value is stored. When search for a word is done if it detects the node key value is a non zero then it assumes it as a accepting node. Shang and Merrettal [4] used the trie data structure for exact and approximate string searching. They presented a trie based method whose cost is independent of the document size.

The main use of the trie is to compare the stored feature strings with the extracted feature string which is represented as a set of numbers.

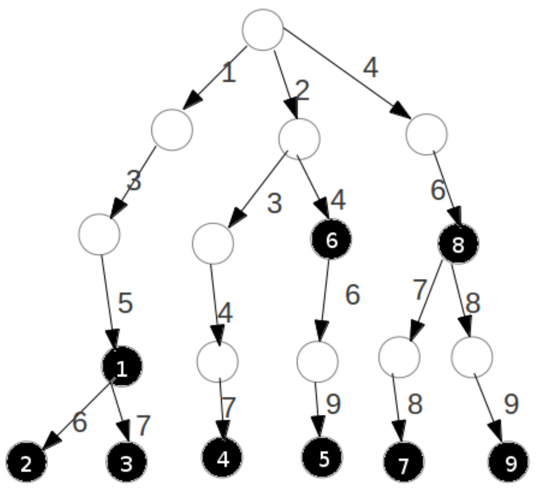


Fig. 5. A trie data structure that store the set of numbers, 135, 1356, 1357, 46, 2347, 24, 2469, 4678, 4689

2.5 Approximate String Matching

The problem of string matching [12] is that, given a body of text *T[1 … n]* we try to find a pattern *P[1 … m]* where $m \leq n$. But in approximate string matching [13, 14] instead of searching for the string exactly, it searches for patterns that are close to P. In other words approximate string matching allows for a certain amount of error between the two strings being compared. Mathematically the problem can be defined as, consider two strings of text *T[1 … n]* and *P[1 … m]*, and a distance function *D(x[i … j], y[a … b])* where *x[i … j]* and *y[a … b]* denotes sub strings of *x* and *y*. The distance function computes the minimal cost of converting *x[i … j]* in to *y[a … b]*. The final input to the approximate string matching problem is *k*, the maximum allowable error. Then the problem is to calculate the set of *P[i … j]* such that $D(T[x \ldots y], P[i \ldots j]) \leq k$.

3. OUR APPROACH

This paper attempted a technique that is usually not affected by the size of the training set and not greatly affected by the noise. As we are using a trie, it saves comparing time and as approximation matching is used, so error can be avoided.

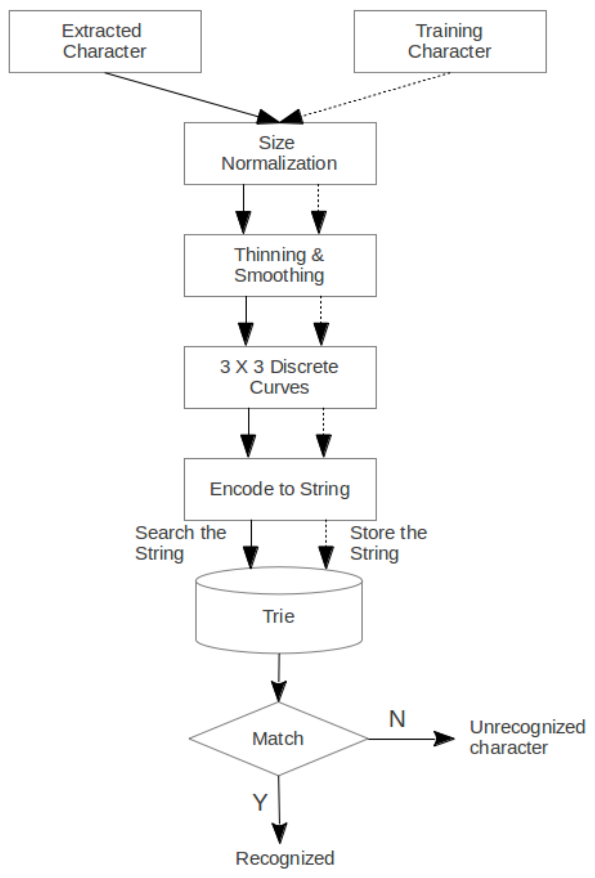


Fig. 6. The Block diagram of the character recognition approach

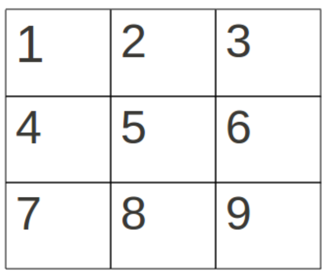


Fig. 7. Our numbering scheme of the pixels in a 3×3 connected component representation

The block diagram of our approach is given in Fig. 6. In the training dataset we first make thinning[9] of the character to make the character in skelitonised form and represents the boundary in single pixel width so the curve can be distinguished correctly. There may be spines present in the thinned character so the smoothing operation is required to remove the spines present on the boundary of the character. Then the character is processed to calculate the 3×3 non overlapping connectedness of the entire region and convert these connected component regions into sequence of digits on the basis of position of the black pixels as shown in Fig. 7. As per the curve properties, it will not contain any 2×2 continuous block of the black pixel.

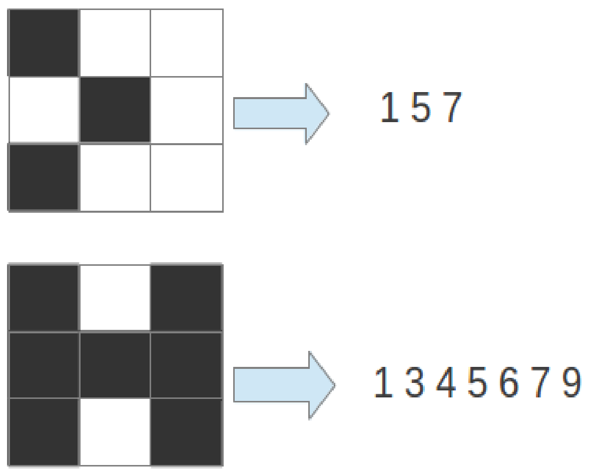


Fig. 8. Our numbering scheme of the pixels in a 3×3 connected component representation

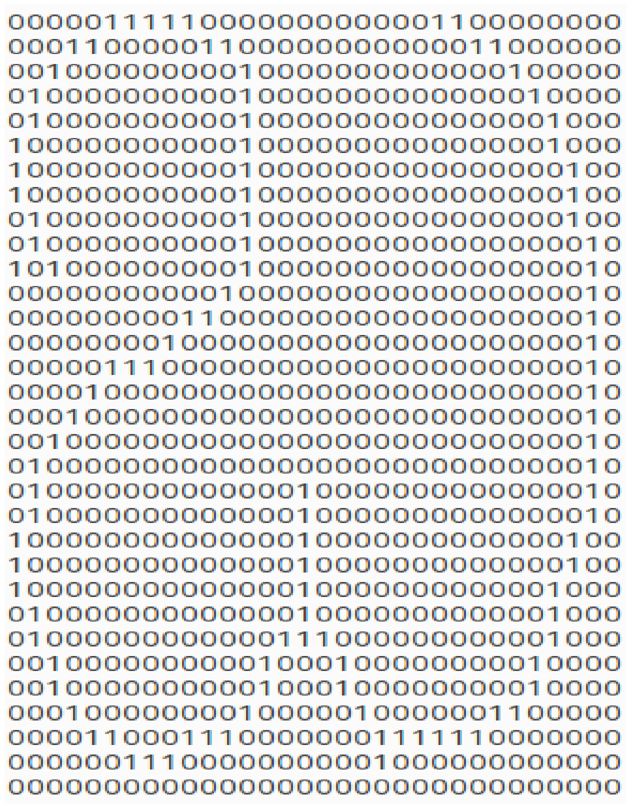
The maximum length of the numbers string will be 7 digits and the digits of the numbers are in sorted order. All the numbers obtained from the character image are concatenated into a single string. In the Fig.8 we show an example of encoding a 3×3 connected component region to a sequence of digits using our numbering rule.

For every discrete curve we will get one sequence of digits shown in Fig. 8. For a character we get a number of discrete curves that we can convert into a sequence of numbers. These sequence numbers form a unique string for each training character. The string is to stored in the trie. In the trie the numbers are stored in the path between the root and the accepting node. The accepting node stores a character label.

When an unclassified character is to be recognized it is processed through a set a operations like size normalization to a standard size, thinning, smoothing and finding set of numbers by the connected component representation of discrete curve method like the process of training data. In the process of generating string from the unclassified character some noise may be introduced. The noise can be due to the improper preprocessing that will not properly generate the string. For encountering this problem in the matching we allow some string error while matching by using the approximate string matching with error bound *'k'*. We use the Branch and Bound look ahead method of approximate string matching [13]. In branch and bound approach we enumerate all the combinations of the solution and discard the fruitless candidate from the search space. In the look ahead branch and bound approximate string matching approach we reduce the search space by pruning some paths from the search space. The pruning is done based on some conditions described below.

Let the training data for a character that is stored in the trie is denoted as *X[1 … n]* and the extracted set of number is *Y[1 … m]*. In the reference to the paper [13], in the trie it will not traverse the *subtrie(c)* unless there is a hope of determining a suitable string in it, where the latter is defined as the string that could be garbled into Y with less than K errors. For the calculation some condition is checked before proceeding to check the hope of string.

Let N' : be the length of the prefix calculated so far.
K : be the maximum permissible error in the string.
M : length of the searched string.



```
00000111110000000000001100000000
00011000001100000000000011000000
00100000000010000000000000100000
01000000000010000000000000010000
01000000000010000000000000001000
10000000000010000000000000001000
10000000000010000000000000000100
10000000000010000000000000000100
01000000000010000000000000000100
01000000000010000000000000000010
10100000000010000000000000000010
00000000000100000000000000000010
00000000011000000000000000000010
00000000100000000000000000000010
00000111000000000000000000000010
00001000000000000000000000000010
00010000000000000000000000000010
00100000000000000000000000000010
01000000000000000000000000000010
01000000000000010000000000000010
01000000000000010000000000000010
10000000000000010000000000000100
10000000000000010000000000000100
10000000000000010000000000001000
01000000000000010000000000001000
01000000000000111000000000001000
00100000000001000100000000010000
00100000000001000100000000010000
00010000000010000010000001100000
00001100011100000001111110000000
00000011100000000001000000000000
00000000000000000000000000000000
```

Fig. 9. A 0/1 representation of a Telugu character where the 1 represents the black pixels and the 0 represents the background pixels.

Maxlen : A value stored at a node which indicates the length of the path between this node and the most distant node representing an element of the trie.

Minlen : A value stored at a node which indicates the length of the path between this node and the least distant node representing an element of the trie.

At each node of the trie, before we do any further computations, we test the following conditions refer to as Look Ahead Branch and Bound conditions:

(1) $Minlen > M - N' + K$

(2) $Maxlen < M - N' - K$

If any of the above two equations satisfy then there is no hope of finding a solution within the present subtrie then prune the subtrie from the searched space.

4. ANALYSIS AND RESULTS

We performed the experiment to get the 3×3 discrete curves. We scanned the training dataset of all basic characters at 300 dpi. Then using the thinning algorithm[9] thinned the characters. Using Open CV library function we calculated the 0/1 representation of the characters. We consider the standard size of the dataset as 32×32 pixels. The output of the 0/1 representation of ‘ba’ character is shown in Fig. 9.

When we calculated the 3×3 discrete curves from the Fig. 9 then the string of numbers that resulted is,

{357, 1, 1269, 3, 159, 1, 369, 148, 47, 369, 259, 2, 9, 357, 369, 567, 24, 369, 357, 369, 14, 369, 368, 7, 369, 247, 158, 2347, 158, 69, 1, 3, 459, 678, 345, 3, 4567, 456, 12}

By concatenating all the above resulted numbers we get a string of numbers, where each number will be stored in the path of the trie from root to accepting node. If a set of numbers is searched against the numbers in the trie and it follows a path in the trie with at most K errors then the searched character will be found and the label of the accepting node is assigned to the searched string. Here we tested the approach with the ‘K’ value of 10. When we tested some basic characters of the Telugu script it showed efficient result and it correctly recognized the characters.

5. DISCUSSION AND FURTHER WORK

It is a technique which basically deals with the discrete curves, approximate string matching. In experimentation it is providing promising results. Some local analysis for a specific scripts can leads to hand written character recognition and can also leads to the complete set of character recognition not only the basic sets. It is not a language method for a set of dataset. It can be easily used for the other Indian scripts with some language specific analysis. As it uses the pruning strategy to reduce the search space the disadvantage of excess training character can be avoided. Using trie for the matching of the features is a very efficient method because

6. CONCLUSIONS

In this paper a novel approach for the character recognition for the Telugu basic characters is proposed. The two new ideas that we introduce is the use of string comparison in the character recognition and the other is the use of the trie in the computation which results matching in constant time, also we use the curve analysis to get the features of a character. As it is giving promising results the best result can be obtained by further analysis.

7. ACKNOWLEDGEMENT

We would like to express our gratitude to all those who gave us the possibility to complete this paper. We are deeply indebted to our project guide Prof. Atul Negi whose help, stimulating suggestions, knowledge, experience and encouragement helped us in all the times of study and analysis of the idea in the pre and post research period.

8. REFERENCES

[1] R. E.W. Rafael C. Gonzalez, Digital Image Processing. New Delhi, India: Pearson/Prentice Hall, 2008.

[2] A. Fred, T. Caelli, R. Duin, A. Campilho, and D. de Ridder, Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004 Proceedings, ser. Lecture Notes in Computer Science. Springer, 2004.

[3] A. Negi, K. N. Shanker, and C. K. Chereddi, “Localization, extraction and recognition of text in telugu document images,” in ICDAR, 2003, pp. 1193–1197.

[4] H. Shang and T. Merrettal, “Tries for approximate string matching,” Knowledge and Data Engineering, IEEE Trans-actions on, vol. 8, no. 4, pp. 540 –547, aug 1996.

[5] M. Firebaugh, Artificial intelligence: a knowledge-based ap-proach, ser. PWS-Kent series in computer science. Boyd & Fraser, 1988.

[6] A. Negi, C. Bhagvati, and B. Krishna, “An ocr system for telugu,” in ICDAR, 2001, pp. 1110–1114.

[7] A. K. Pujari, C. D. Naidu, M. S. Rao, and B. C. Jinaga, “An intelligent character recognizer for telugu scripts

using mul-tiresolution analysis and associative memory," Image Vision Comput., vol. 22, no. 14, pp. 1221–1227, 2004.

[8] V. K. Koppula, A. Negi, and U. Garain, "Robust text line, word and character extraction from telugu document image," in ICETET, 2009, pp. 269–272.

[9] P. Wang and Y. Zhang, "A fast and flexible thinning algorithm," Computers, IEEE Transactions on, vol. 38, no. 5, pp. 741 –745, may 1989.

[10] B. Oommen and G. Badr, "Dictionary-based syntactic pattern recognition using tries," in Structural, Syntactic, and Statistical Pattern Recognition, ser. Lecture Notes in Computer Science, A. Fred, T. Caelli, R. Duin, A. Campilho, and D. de Ridder, Eds. Springer Berlin Heidelberg, 2004, vol. 3138, pp. 251–259.

[11] R. A. Wagner and M. J. Fischer, "The string-to-string correction problem," J. Assoc. Comput. Mach., vol. 21, pp. 168–173, 1974

[12] G. Badr and B. Oommen, "A novel look-ahead optimization strategy for trie-based approximate string matching," Pattern Analysis & Applications, vol. 9, pp. 177–187, 2006, 10.1007/s10044-006-0036-8.

[13] G. Navarro, "A guided tour to approximate string matching," ACM Computing Surveys, vol. 33, p. 2001, 1999.