A Character Recognition Approach using Freeman Chain Code and Approximate String Matching

Samit Kumar Pradhan
Lecturer in Computer Science & Engineering
RGUKT, Basar
Andhra Pradesh, India

Sujoy Sarkar
Lecturer in Computer Science & Engineering
RGUKT, Basar
Andhra Pradesh, India

Suresh Kumar Das
M. Tech in Computer Science
University of Hyderabad
Hyderabad, India

ABSTRACT
This paper deals with a syntactic approach for character recognition using approximate string matching and chain coding of characters. Here we deal only with the classification of characters and not on other phase of the character recognition process in a Optical character Recognition. The character image is first normalized to a specified size then by boundary detection process we detect the boundary of the character image. The character now converted to boundary curve representation of the characters. Then the curve is encoded to a sequence of numbers using Freeman chain coding. The coding scheme gives a sequence of numbers ranges from 0 to 7. Now the characters are in form of strings. For training set we will get a set of strings which is stored in the trie. The extracted unclassified character is also converted to string and searched in the trie. As we are dealing with the character which can be of different orientation so the searching is done with approximate string matching to support noisy character that of different orientation. For approximate string matching we use Look Ahead Branch and Bound scheme to prune path and make the approximation accurate and efficient. As we are using trie data structure, so it take uniform time and don't dependent on the size of the input. When we performed our experimentation for noiseless character that is printed character it successfully recognize all characters. But when we tested with the different variation of the character then it detect most of the character except some noisy character.

General Terms
Pattern Recognition, Character Recognition, OCR.

Keywords
Syntactic Pattern Recognition, Freeman Chain Coding, trie, Character Recognition, Approximate string matching, Boundary detection.

1. INTRODUCTION
Character recognition is an area of research in image processing of Pattern Recognition domain. There are Optical Character Recognition (OCR) for recognition of characters of different scripts. The OCR research of English character is in matured state where the OCR for Indian scripts still in evolution stage for efficient recognition. Generally the character recognition contain various stages like, line segmentation, word segmentation, character segmentation, binarization, feature extraction, classification. From all the stages classification of characters plays an important role in the accuracy of character recognition. Generally for classification statistical pattern recognition are used. Here we proposed one syntactic PR method for classification[1][2] of characters. Our method involves string encoding, approximate string matching[3] and trie data structure.

Each character is converted to a string for the classification. String matching is used to compare training characters and the extracted unclassified character. Let X[1 ... n] be a string of length n from the training set to be compare with Y[1 … m] of length m is a string encoded from a unclassified character. The string Y may contain string error with respect to training characters. The string errors are deletion, insertion, substitution. The approximation matching is there to compare the string with certain number of above error in the string. Here we use one approximation string matching scheme for matching of training characters with extracted characters.

For storing of the training characters string, we use trie data structure. The search complexity in a trie data structure depends on the longest string present in the trie and independent of the number of input to the trie. The benefit of using trie is to reduce the complexity in matching.

The organization of this paper as follows. Section 2 presents some background concepts that are used in this paper, such as Freeman chain encoding, Trie data structure, and Approximate string matching. Section 3 presents the details of the proposed approach, the character recognition procedure using this proposed method. Section 4 presents the results and discussion related to this approach and section 5 concludes the paper.

2. BACKGROUND
All material on each page should fit within a rectangle of 18 x

Here we are only dealing with the character recognition not with the extraction of the character from document image. Here we first use the boundary of the image to encode that to a string. The boundary detection[4] process gives the boundary of the character image. That means the character image now represented with boundary line of the image. Going in the direction of the curve starting from a point until we reached again to that point. We will get a chain of code as defined in freeman chain encoding process. We are generally developing this classification method for the Indian script and we are operating on the basic Telugu scripts. The encoding and string representation of the character image which we describe in later section. There are many approaches in previous work[5][6] for character recognition, with different strategies such as using fringe distance, template matching[7] and wavelet analysis [8]. Now we build up some essential background for our approach.

2.1 Freeman Chain Encoding
Freeman chain code[4][9] is a code obtained by following the boundary in an assigned direction (clockwise or anti-clockwise). We start following the boundary from a fixed point and move along the boundary of the image until we reached to the starting point again.

While following the path depending upon the location of the next pixel from the current pixel as shown in fig. 1,

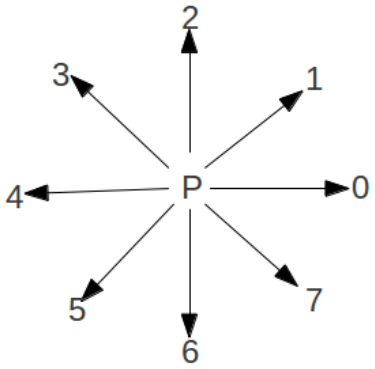

Fig 1. The Freeman chain coding direction.

The *P* is the current black pixel and there are 8 possible neighbor of *P*. Depending on the eight direction we will get one code number *1* to *7*. While moving in total path we get number of code and that form one string which represent the image. From the Fig. 2, using freeman coding, starting from the point '*S*' and move in anticlockwise direction gives '5555011077033334'.

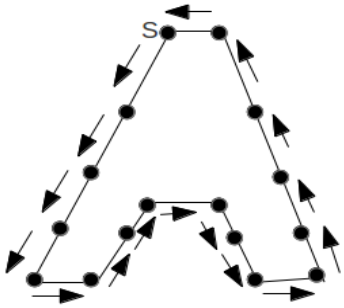

Fig 2. An example showing the coding of an image.

2.2 Trie Data Structure

Trie data structure generally used in dictionary build up as its search time complexity not depends on the size of the input rather on the longest string present in the trie. It starts search from the start node to depth reading single input symbol at a time and rejecting all path except the symbol. Here the string is stored in the path between start node to an accepting node. Each accepting node represents one String. Shang and Merrettal[10] used the trie data structure for exact and approximate string searching. They presented a trie based method whose cost is independent of the number of the words in the document. Trie indexes combine suffixes and so are compact in storage.

2.3 Approximate String Matching

Consider two strings of text T [1 ... n] and P [1 ... m], and a distance function *ed( )*. The problem is to match the two strings with an error bound of '*K*'. The edit distance *ed( )* between two strings is defined as the minimum number of character insertions, deletions and substitutions needed to convert one to another. The problem of approximate string matching[11] is typically divided into two sub-problems first one is, finding approximate sub string matches inside a given string and second one is finding dictionary strings that match the pattern approximately[12]. We are dealing with the second type of the approximate string matching problem.

Here we are storing all the string in the trie data structure and finding a approximate string with respect to searched string.

The main approach of the approximate string matching is to prune some path while searching approximate match in the trie. There are some heuristic for pruning the path in the trie so that the search path will reduce and the approximate match will be faster. In our approach we are using Look Ahead Branch and Bound method to prune path while doing approximate matching in the trie.

2.4 Look Ahead Branch and Bound

Look Ahead Branch and Bound is an Artificial Intelligence concept in which we will go ahead only when there is a hope of finding a solution. Here in trie we travel only those path where there is a hope to get the matching string. Here in trie we store one number in each node and along with this we need some other value for the heuristic. We store one field '*key*' to differentiate between accepting node and the non accepting node. We store the *key* value as non zero to mark as accepting node and *key* is zero for non accepting node. Also at every node we store some value defined below,

Let the training data for a character set that is stored in the trie is denoted as *X[1 ... n]* and the extracted character string representation is *Y[1 ... m]*. In the reference to the paper[3], In trie we will not traverse the subtrie(c) unless there is a hope of determining a suitable string in it, the expected suitable string is the string that contains less than *K* edit distance with the searched string. For pruning some conditions are needed to checked before proceeding to check whether there is a hope of finding a string or not.

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.

MAXL - 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.

MINL - 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. MINL > m - N + K
2. MAXL < m - N – K

If any of the above two equations satisfy then there is no hope of finding a solution within the present subtrie, so prune the subtrie from the searched space. While searching if we reached at a accepting node that is with non zero key value and the distance of the string represented by accepting node and searched string is in error bound *K*, accept that string. There is a chance that for a given error bound *K* two or more string may result, here the string with less distance with the searched string is the resulted string.

3. OUR APPROACH

We have proposed an effective syntactic character recognition method. The flow of our approach is given in fig. 3. The solid lines define the flow of the training character and the dotted lines define the extracted character, which is unclassified character.

For training data set we first normalize the binarize character image into a specified size, let it be 32 X 32. The normalization step is required for getting a valid encoded string. The extracted image can be of any size, so the normalization stage convert the image into a fixed size as the one taken in the training character. After the size normalization we detect the boundary of the character image as shown in fig. 4 and fig. 5. So for a character now we have only the boundary of the image and ready to encode the boundary curve to a string.

The boundary of the character is now encoded to a string using freeman chain coding as shown in fig. 1 and fig. 2. For each character we will get one sequence of numbers between 0 to 7. The strings obtained from a training character set is stored in a trie. When an extracted unclassified string is matched against the trie, we use approximate string matching. The main reason of the using approximate string matching not exact string matching is the noise, the unclassified character string may contain some noise. For slightly changed shape character also give different string than the original string. So to overcome this problem we use approximate string matching with some error bound instead of exact string matching. The error bound should enough large to uniquely classify the characters. While matching we are using levenshtein edit distance to compare string. We build a confusion matrix by finding the distance among the strings obtained from characters. Here we have given an example of 10 Telugu characters and the confusion matrix among them is shown in fig. 6.

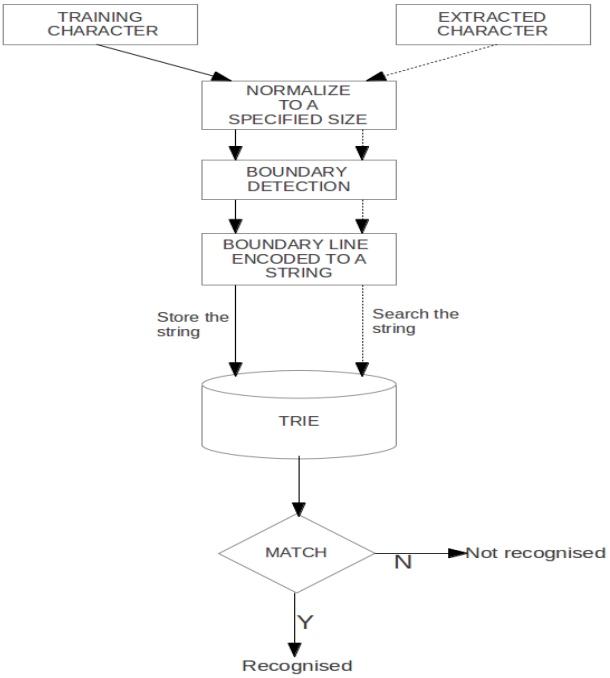

Fig. 3. The flow chart of our approach

Fig. 4. Character 'ba' of Telugu

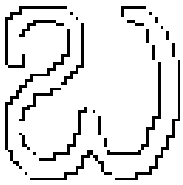

Fig 5. Character 'ba' (Telugu) after boundary detection

From the confusion matrix it is clearly stated that the distance among the string obtained from each character is very high, so by string matching we can classify the characters. Once the character string are stored in the trie the approximate matching is needed to match the character.

There are many approaches for the approximate string matching. Here we are considering Look Ahead Branch and Bound method of approximate string matching. The main aim of the approximate matching is to prune path by a heuristic, so that the search space will reduce and search will be faster. Here we approximately match the extracted string against the trie containing strings from training data set. We fix an error bound for approximate match depending on the error bound we get a set of approximate match strings. There is a chance of getting more than one approximate matched string for a given searched string. In this case we calculate the distance between searched string with all resulted string and the string with least distance with searched string is the result.

4. ANALYSIS AND RESULTS

We performed the experiment to get the encoded string. We performed the test on basic Telugu character which is scanned at 300 dpi. All the characters are normalized to a specified size of 32X32. We have generated all the characters boundary and performed freeman chain encoding to represent as a string using Open CV library. We have tested for the 44 Telugu basic characters and the distance between characters are calculated. Some part of the distance table that is confusion matrix shown in fig. 6, which shows the distance between the string representation of the characters and has large value, it says the method can be used to classify the characters. The bold letter in the fig. 6 shows some character that are similar to each other having less distance. We have tested all 44 basic character and implemented with *K* value 5. This method uniquely classify all the character.

To check the efficiency of the method for different orientation of characters, we checked the method for 15 variation with *K* value 50 of all character. We have shown the example of character 'a' of Telugu script in fig. 7 and table 1.

	అ	ఆ	ం	బ	భ	చ	ద	డ	న	ట
అ	**0**	103	110	126	178	143	148	151	128	128
ఆ	103	**0**	124	153	170	147	171	183	138	138
ం	110	124	**0**	122	174	110	140	153	112	118
బ	126	153	122	**0**	178	126	133	150	**87**	126
భ	178	170	174	178	**0**	133	154	163	169	165
చ	143	147	110	126	133	**0**	127	125	108	117
ద	148	171	140	133	154	127	**0**	112	109	143
డ	151	183	153	150	163	125	112	**0**	150	146
న	128	138	112	**86**	169	108	109	150	**0**	131
ట	128	138	118	125	165	117	143	146	131	**0**

Fig. 6. Confusion matrix.

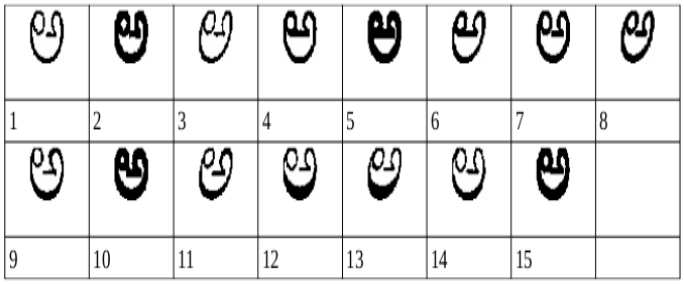

Fig 7 . Different orientation of character 'a' of Telugu.

Table 1. Distance among the different variation of character 'a'.

Image	2	3	4	5	6	7	8	9
1	95	89	83	104	101	74	91	83

Image	10	11	12	13	14	15
1	94	89	53	85	81	100

From the confusion matrix shown in fig.6 and from the table 1, the distance among the character is very large and among its own variation it is less. Although some of the entry in the confusion matrix in fiq. 6 are more than some of the entry in the fig. 8. It stated that some variation of the character is accurately classified but if the orientation is more that certain level for some character it is unclassified but if the error is not there then it can accurately classify all. So this method can be used to classify the printed character and character with certain level of variation. But extensive study in this area may be useful for accurate classification of all character.

5. DISCUSSION AND FURTHER WORK

As this method is based on the approximate string matching and string encoding and it is showing promising result for the printed character and some small variation of the character it can be used for the hand written character recognition. It is not a method for a specified language dataset. It can be easily used for the other Indian scripts basically efficiently those whose size are circular in shape like Telugu and Oriya. Here we only considered the basic character of the Telugu script but we can also consider all the characters of the Telugu script. As it uses the pruning strategy to reduce the search space the disadvantage of excess training character can be avoided where we have to compare with all the set of characters.

6. CONCLUSION

We have presented one syntactic approach for the character recognition. The new approach that we introduced here are approximate string matching and the string comparison for the character recognition. We also used trie for the efficient search and also of less complexity. As this result giving promising result for without noise and also some variation of the character image so further study in this field may result more efficient result.

7. REFERENCES

[1] S. K. Pradhan and S. Sarkar, "Article: Character recognition using discrete curve with the use of approximate string matching," IJCA Proceedings on International Conference in Distributed Computing and Internet Technology 2013, vol. ICDCIT, pp. 17–22, January 2013,.

[2] S. K. Pradhan and A. Negi, "A syntactic pr approach to telugu handwritten character recognition," in Proceeding

of the workshop on Document Analysis and Recognition, ser. DAR '12. New York, NY, USA: ACM, 2012, pp. 147–153.

[3] 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.

[4] H. Bunke and P. Wang, Handbook of Character Recognition and Do.World Scientific, 1997.

[5] N. Kato, M. Suzuki, S. Omachi, H. Aso, and Y. Nemoto, "A handwritten character recognition system using directional element feature and asymmetric mahalanobis distance," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 21, no. 3, pp. 258 –262, mar1999.

[6] H. Liu and X. Ding, "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes," in Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on, aug.-1 sept. 2005, pp. 19 – 23 Vol. 1.

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

[8] A. K. Pujari, C. D. Naidu, M. S. Rao, and B. C. Jinaga, "An intelligent character recognizer for telugu scripts using multiresolution analysis and associative memory," Image Vision Comput., vol. 22, no. 14, pp. 1221–1227, 2004.

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

[10] H. Shang and T. Merrettal, "Tries for approximate string matching," Knowledge and Data Engineering, IEEE Transactions on, vol. 8, no. 4, pp. 540 –547, aug 1996.

[11] B. J. Oommen and R. K. S. Loke, "Pattern recognition of strings with substitutions, insertions, deletions and generalized transpositions," Pattern Recognition, vol. 30, pp. 30–5, 1995.

[12] R. Baeza-yates and G. Navarro, "Fast approximate string matching in a dictionary," in In Proc. SPIRE'98. IEEE Computer Press**,** 1998, pp. 14–22.