Applications of Hidden Markov Model to Recognize Handwritten Tamil Characters

R. Jagadeesh Kannan, R.M. Suresh, and A. Selvakumar

Department of Computer Science and Engineering,
R.M.K. Engineering College,
Kavaraipettai - 601206, Tamil Nadu, India
{dr_rjk,rmsuresh}@hotmail.com

Abstract. Optical Character Recognition (OCR) refers to the process of converting printed, hand printed and handwritten Tamil text documents into software translated Tamil Text. As part of the preprocessing phase the image file is checked for skewing. If the image is skewed, it is corrected by a simple rotation technique. Then the image is passed through a noise elimination phase and is binarized. The preprocessed image is segmented. Thus a database of character image glyphs is created out of the segmentation phase. Then all the image glyphs are considered for recognition. Each image glyph is passed through various routines which extract the features of the glyph. The glyphs are now set ready for classification and recognition based on the above said features. The extracted features are considered for recognition using Hidden Markov Model (HMM). The recognition rate achieved is 89%.

Keywords: Optical Character Recognition, Hidden Markov Model, Preprocessing, Binarization, Segmentation, Feature Extraction, Preliminary Classification, Training and Recognition.

1 Introduction

Optical Character Recognition (OCR), involves a system designed to translate images of typewritten or handwritten text into machine-editable text. The process of handwriting recognition involves extraction of some defined characteristics called features to classify an unknown handwritten character into one of the known classes. A typical handwriting recognition system consists of several steps, namely: preprocessing, segmentation, feature extraction, and classification. Several types of decision methods, including statistical methods, neural networks, Hidden Markov Model (HMM), structural matching (on trees, chains, etc.) and stochastic processing (Markov chains, etc.) have been used along with different types of features.

1.1 Tamil Characters

Tamil alphabet consists of 12 vowels, 18 consonants, 216 composite letters, one special character (AK) and 14 other characters. Vowels and consonants are combined to form composite letters, making a total of 247 different characters. In addition to the standard characters, six characters taken from the Grantha script, which was used in the Tamil region to write Sanskrit, are sometimes used to represent sounds not native to Tamil, that is, words borrowed from Sanskrit, Prakrit and other languages. The complete Tamil alphabet and composite character formations are given in Table 1.

From Tamil language, some 67 Tamil characters as the basic characters (Vowels, Consonants and composite letters) are identified and if one recognizes these 67 characters then all the 247 characters can be recognized. The lists of 247 characters are represented in Table 1.

Table 1. Tamil Characters

Tamil Characters
க ங ச ட ண ப ம ய ல வ ள ன அ ஈ உ ஊ எ ை ா
இ ஃ ி ு ெ ே ் ு
ஞ த ந ர ழ ற ஆ ஏ ஐ ஒ ஓ ஔ ூ
றி றீ ரி ரீ தி தீ ழி ழீ நி நீ ஞி ஞீ

2 Literature Survey

In [1] the author described a method for recognition of machine printed Tamil characters using an encoded character string dictionary. The scheme employs string features extracted by row- and column-wise scanning of character matrix. The features in each row (column) are encoded suitably depending upon the complexity of the script to be recognized.

In [2] the author has proposed an approach for hand-printed Tamil character recognition. Here, the characters are assumed to be composed of line-like elements, called primitives, satisfying certain relational constraints. Labeled graphs are used to describe the structural composition of characters in terms of the primitives and the relational constraints satisfied by them. The recognition procedure consists of converting the input image into a labeled graph representing the input character and computing correlation coefficients with the labeled graphs stored for a set of basic symbols.

In [3] & [4] the authors attempts to use the fuzzy concept on handwritten Tamil characters to classify them as one among the prototype characters using a feature called distance from the frame and a suitable membership function. The prototype characters are categorized into two classes: one was considered as line characters/patterns and the other was arc patterns. The unknown input character was classified into one of these two classes first and then recognized to be one of the characters in that class.

In [5] a system was described to recognize handwritten Tamil characters using a two stage classification approach, for a subset of the Tamil alphabet. In the first stage, an unknown character was pre-classified into one of the three groups: core, ascending and descending characters. Then, in the second stage, members of the pre-classified group are further analyzed using a statistical classifier for final recognition.

In [6] a system was proposed to recognize printed characters, numerals and handwritten Tamil characters using Fuzzy approach. In [7] the author proposed an approach to use the fuzzy concept to recognize handwritten Tamil characters and numerals. The handwritten characters are preprocessed and segmented into primitives. These primitives are measured and labeled using fuzzy logic. Strings of a character are formed from these labeled primitives. To recognize the handwritten characters, conventional string matching was performed. However, the problem in this string matching had been avoided using the membership value of the string.

In [8] the authors proposed a two stage approach. In the first stage, an unsupervised clustering method was applied to create a smaller number of groups of handwritten Tamil character classes. In the second stage, a supervised classification technique was considered in each of these smaller groups for final recognition. The features considered in the two stages are different.

In [9] an approach was proposed to recognize handwritten Tamil characters using Neural Network. Fourier Descriptor was used as the feature to recognize the characters. The system was trained using several different forms of handwriting provided by both male and female participants of different age groups.

3 System Architecture

The scanned input document image file is loaded into the preprocessing steps, After the preprocessing steps is over, Preliminary Classification is done for each character and features are extracted from each of the characters and then they are sent into the recognition stage, which is done by Hidden Markov Model (HMM) to get the recognized output.

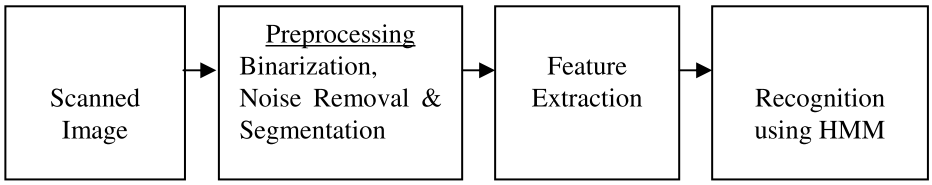


Fig. 1. Architecture of the Proposed System

The scanned image is preprocessed, i.e., the image is checked for skew correction, then the image is binarized, then unwanted noise is removed and finally the characters are segmented.

3.1 Preprocessing Steps

3.1.1 Binarization

Image binarization converts an image (up to 256 gray levels) to a black and white image (0 or 1). Binarization is done using Modified Otsu Global Algorithm. This algorithm is the combination of Otsu Global algorithm and Sauvola algorithm. This method is both simple and effective. The algorithm assumes that the image to be threshold contains two classes of pixels (e.g. foreground and background) and calculates the optimum threshold separating those two classes so that their combined spread (intra-class variation) is minimal. As in Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes:

$$\sigma_{\omega}^{2}(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \tag{1}$$

Weights ωi are the probabilities of the two classes separated by a threshold *t* and variances of these classes. Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance

$$\sigma_b^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2 \tag{2}$$

Which is expressed in terms of class probabilities ωi and class means μi which in turn can be updated iteratively.

Algorithm 1

```
Input: Scanned Image Output: Binarized Image
Step 1: Compute histogram and probabilities of each intensity level
Step 2: Set up initial $\omega_i(0)$ and $\mu_i(0)$
Step 3: Step through all possible thresholds t=1...... maximum intensity
        1. Update $\omega_i$ and $\mu_i(t)$
        2. Compute $\sigma_b^2(t)$
Step 4: Desired threshold corresponds to the maximum $\sigma_b^2$(t)
```

In the case of the bad quality image global thresholding cannot work well. For this, we would like to apply a technique. Sauvola binarization technique (window-based), which calculates a local threshold for each image pixel at (x, y) by using the intensity of pixels within a small window W (x, y). The Threshold T (x, y) is computed using the following formula

$$T(x,y) = \mathrm{Int}[X.\ (1 + k / (R-1))] \tag{3}$$

Where X is the mean of gray values in the considered window W (x, y), is the standard deviation of the gray levels and **R** is the dynamic range of the variance, **k** is a constant (usually 0.5 but may be in the range 0 to 1).

3.1.2 Noise Removal

Morphological Image Cleaning Algorithm(MIC) is used to remove noise from the image

Algorithm 2

```
Input: Noise Added Image    Output: Noise Removed Image
Step 1: Consider a noisy grayscale image I.
Step 2: Let S be the image with smoothing I which has openings and closings.
Step 3: Assume S is noise free.
Step 4: Then the difference image, D contains all the noise in I.    $D=I-S$    (4)
Step 5: Again the residual image (D) is added to the S smoothed image, so that the
        resulting image will be as sharp as the original image with smoothened
        regions between them.
```

3.1.3 Segmentation

The skewed images have been made available for segmentation process.

Algorithm 3

```
    Input: Noise Removed Image
    Output: Segmented Image
Step 1: The gray level image of a Tamil word is median filtered and then converted
        into a binary image using Otsu threshold technique.
Step 2: Apply the skew detection and correction algorithm and then detects the
        headline and Baseline as well as the bounding box.
Step 3: Connected components of a word to be segmented are detected.
Step 4: Lower contour of each connected component is traced anticlockwise. During
        this tracing Process the relevant features are extracted.
Step 5: The feature vectors are normalized. Also the MLP is trained with the
        normalized feature set.
```

After pre-processing the image, the image is cropped

3.1.4 Universe Of Discourse (or) Cropping

Universe of discourse is defined as the shortest matrix that fits the entire character skeleton. The following figure 2 shows the

Universe Of Discourse

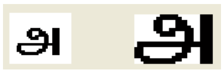


Fig. 2. Universe of Discourse

3.2 Preliminary Classification

The Tamil scripts can be categorized into four groups according to their structure. Basically three (3) zones can be identified in a typical Tamil character. The three zones are named Upper, Middle and Lower zones. The heights of the Upper and Lower zones are normally 25% of the line height and the Middle zone is of height 50% of the line height. The following figure 3 illustrates the Tamil character classification.

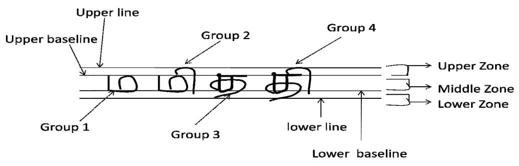


Fig. 3. Tamil Character Classification

The aim of the preliminary classification process is to classify an unknown character into one of the above four groups so that the number of characters is restricted to the members of that group. For each character, its upper and lower boundaries are compared with the four reference lines, and the pre-classification is performed according to it.

Following table lists all the characters in four groups.

Table 2. Tamil Characters categorized into Different Groups

Group	Characters of the Group
Group 1	க ங ச ட ண ப ம ய ல வ ள ன அ ஈ உ ஊ எ ை ா
Group 2	இ ஃ ி ் ெ ே ˙ ீ
Group 3	ஞ த ந ர ழ ற ஆ ஏ ஐ ஒ ஓ ஔ ு
Group 4	றி றீ ரி ரீ தி தீ ழி ழீ நி நீ ஞி ஞீ

The images of segmented characters are then rescaled into 32x32 pixel size, using a bilinear interpolation technique. Each image is divided into N X M zones and thus the image enters into Feature Extraction Stage

3.3 Feature Extraction

This follows the segmentation phase of character recognition system where the individual image glyph is considered and extracted for features. A character glyph is defined by the following attributes

Height of the character, Width of the character, Numbers of horizontal lines present - short and long, Numbers of vertical lines present - short and long, Numbers of circles present, Numbers of horizontally oriented arcs, Numbers of vertically oriented arcs, Centroid of the image, Position of the various features, Pixels in the various regions.

3.4 Hidden Markov Model

Hidden Markov Model (HMM) is a "doubly stochastic process with an underlying Markov process that is not directly observable, but can only be observed through another set of stochastic processes that produce the sequence of observed symbols

3.4.1 HTK Toolkit Operations for Training and Recognition

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models (HMM). HTK is in use at hundreds of sites worldwide. HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis.

Parts of HMM modeling are divided into three phases:

1. Data Preparation
2. Training
3. Recognition

3.*4.1.1 Data Preparation.* In the Data Preparation stage, first we will define a word network using a low level notation called HTK Standard Lattice Format (SLF) in which each word instance and each word-to-word transition is listed explicitly. This word network can be created automatically from the grammar definition. At this same stage, we have to build a dictionary in order to create a sorted list of the required character or words to be trained. Then we will use the tool HSGen to generate the prompts for test sentences. Note that the data preparation stage is required only for recognition purpose. It has absolutely no usage for the training purpose.

3.*4.1.2 Training.* The first task is to define a prototype for the HMM model to be trained. This task will depend on the number of states and the extracted feature of each character or word. The definition of a HMM must specify the model topology, the transition parameters and the output distribution parameters. HTK supports both continuous mixture densities and discrete distributions. In our application we will use discrete distributions as we see that the observation state is finite for each frame (8 x 90 = 720 states). Then we have to initialize the estimation of the HMM model parameters. The model parameters contain the probability distribution or estimation of each model. By far the most prevalent probability density function is the Gaussian probability function that consists of means and variances and this density function is used to define model parameters by the HTK. For this we have to invoke the tool HInit. After the initialization process is completed the HMM model is written into .mmf file that contains all the trained models and is used in recognition.

3.4.1.3 Recognition. Comparative to the training, recognition is much simpler. To complete this task we have to create a HMM model of the character or word image. Then this model will match with all the HMM models and the most likely model will be given as output. To perform this task using HTK we have to invoke the recognition tool HVite that uses the word network describing the allowable word sequence build up from task grammar, the dictionary that define each character or word, the entire list of HMMs and the description of each HMM model. HVite is a general-purpose Viterbi word recognizer. It will match a HMM model against a network of HMMs and output a transcription for the recognized model into a Master Label File .mlf. After the recognition process is completed, the model name is read from the Master Label File (.mmf) and the associated Unicode character for the recognized model is written to the output file.

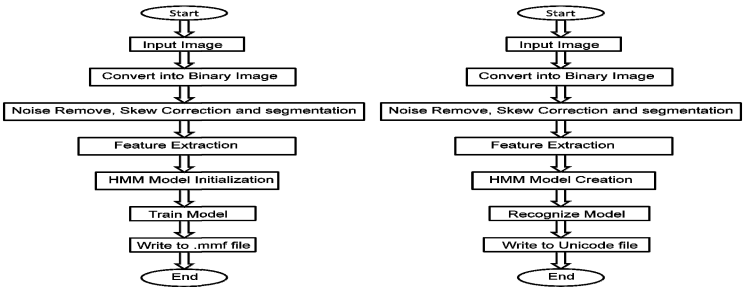


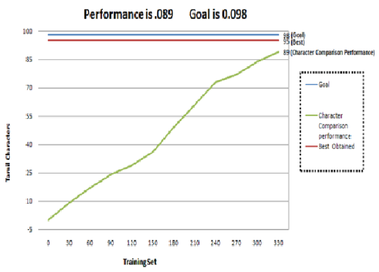
Fig. 4. Flowchart of Training and Recognition Procedure

4 Conclusion

The Preprocessing Steps, Preliminary Classification, Feature Extraction, Training and Recognition mechanism of the Hidden Markov Model (HMM) based Optical Character Recognizer (OCR) for Tamil character is being presented. The system is mainly divided into two parts; one is preprocessing stages and the other part consists of Preliminary classification, Feature Extraction and Recognition to recognize a particular sample. The performance of the system is acceptable. Performance of the recognizer depends on the number of trained sample. The following table 3 illustrates the recognition rate of Tamil characters for various groups considered in this paper.

Table 3. Recognition Rate of the Tamil character

Group	Total No of Characters	No of characters Recognized	Recognition Rate (%)
Group 1	70	62	88.57
Group 2	60	53	88.37
Group 3	65	58	89.23
Group 4	70	62	88.57



The following graph illustrates the Comparison performance of Tamil Characters 89%.The recognition rate obtained from this methodology is 89%. The performance of the system can be reached up to 95% and still the system can be trained to give the higher Recognition rate (98%).

References

1. Siromoney, G., Chandrasekaran, R., Chandrasekaran, M.: Computer Recognition of Printed Tamil Character. Pattern Recognition 10, 243–247 (1978)
2. Chinnuswamy, P., Krishnamoorthy, S.G.: Recognition of Hand printed Tamil Characters. Pattern Recognition 12, 141–152 (1980)
3. Suresh, R.M., Ganesan, L.: Recognition of Hand printed Tamil Characters Using Classification Approach. In: ICAPRDT, pp. 63–84 (1999)
4. Suresh, R.M., Arumugam, S., Ganesan, L.: Fuzzy Approach to Recognize Handwritten Tamil Characters. In: International Conference on Computational Intelligence and Multimedia Applications, pp. 459–463 (2000)
5. Hewavitharana, S., Fernando, H.C.: A Two Stage Classification Approach to Tamil Handwriting Recognition. Tamil Internet 2002, pp. 118–124 (2002)
6. Suresh, R.M., Ganesan, L.: Recognition of Printed and Handwritten Tamil Characters Using Fuzzy Approach. In: International Conference on Computational Intelligence and Multimedia Applications, pp. 291–286 (2002)
7. Patil, P.M., Sontakke, T.R.: Rotation, Scale and Translation Invariant Handwritten Devanagari Numeral Character Recognition Using General Fuzzy Neural Network. Pattern Recognition 40, 2110–2117 (2007)
8. Bhattacharya, U., Ghosh, S.K., Parui, S.K.: A Two Stage Recognition Scheme for Handwritten Tamil Characters. In: International Conference on Document Analysis and Recognition, pp. 511–515 (2007)
9. Sutha, J., Ramaraj, N.: Neural Network Based Offline Tamil Handwritten Character Recognition System. In: International Conference on Computational Intelligence and Multimedia Applications, vol. 2, pp. 446–450 (2007)