

Tamil Handwritten City Name Database Development and Recognition for Postal Automation

S. Thadchanamoorthy
Trincomalee Campus, Eastern University,
Sri Lanka
Email: stmoorrrthy@gmail.com

Umapada Pal
Computer Vision and Pattern Recognition Unit; Indian
Statistical Institute, Kolkata-108, India
Email: umapada@isical.ac.in

N. D. Kodikara; H. L. Premaretne
University of Colombo School of Computing, Sri Lanka
Email: ndk@ucsc.cmb.ac.lk;
hlp@ucsc.cmb.ac.lk

Fumitaka Kimura
Graduate School of Engg., Mie University; 1577
Kurimamachiya-cho, TSU, 514-8507, Japan
Email: kimura@hi.info.mie-u.ac.jp

Abstract- Although there are some reports on offline Tamil isolated handwritten character recognition, to our knowledge there is only two reports on Tamil off-line handwritten word recognition. Also no city name dataset is available for Tamil script. In this paper we present a Tamil offline city name dataset, we developed, and propose a scheme for recognition. Because of the different writing style of various individuals, some of the characters in a Tamil city name may touch and accurate segmentation of such touching into individual characters is a difficult task. Avoiding proper segmentation here, we consider a city name string as a word and the recognition problem is treated as lexicon driven word recognition. In the proposed method, binarized city names are pre-segmented into primitives (individual character or its parts). Primitive components of each city name are then merged into possible characters to get the best city name using dynamic programming. For merging, total likelihood of characters is used as the objective function and character likelihood is computed based on Modified Quadratic Discriminant Function (MQDF), where direction features are applied. A dataset of 265 Tamil city names is developed, and the database will be available freely to the researchers. From the experiment of the proposed scheme 96.89% city name accuracy is obtained from this dataset.

Keywords- Handwriting recognition, City name recognition, Tamil script, MQDF, Postal automation.

I. INTRODUCTION

Recognition of handwritten characters has been a popular research area for many years because of its various application potentials. Some of its potential application areas are Postal Automation, Bank cheque processing, automatic data entry, etc. There are many pieces of work towards handwritten recognition of Roman, Japanese, Chinese and Arabic scripts [1] and a few reports are also available towards handwritten word recognition of some of the Indian scripts [2]. Tamil is an important language (one of the official languages of Sri Lanka and one of the State official languages in India) but so far only two reports are available for offline handwritten Tamil word recognition. In [3], only 40 words are considered with the concept of HMM and obtained 80.75% recognition accuracy. In [4], a holistic approach is proposed for recognition of offline unconstrained handwritten Tamil words with 217 classes and 86.36% accuracy was obtained. For postal automation

there are many pieces of work towards city name recognition on non-Indian languages [5] and also some work have been done for some Indian scripts [6]. To the best of our knowledge there is no work on city name recognition in Tamil. Also there is no standard dataset for Tamil handwritten word recognition and hence this paper discusses, at first, about a Tamil offline handwritten city name dataset and then a scheme is proposed for Tamil city name recognition.

Tamil is an ancient, classical Dravidian language in existence for over two thousand years. The Tamil script traces its roots to the Brahmin script and continues to undergo a lot of changes to transgress itself as a portable medium. There are 30 basic shapes (12 vowels and 18 consonants) in Tamil. In addition to this there are six Grantham characters and one Ayutham. These shapes are shown in Fig.1. With the combination of these 30 basic shapes and one ayutham letter, in total two hundred and forty seven characters can be obtained that are used in Tamil language.

Vowels	அ ஆ இ ஈ உ ஊ எ ஏ ஐ ஒ ஓ ஔ
Ayutham	ஃ
Consonants	கங் ச்ஞ் ட்ண் த்ந் ப்ம் ய் ர் ல் வழ் ள் ற் ன்
Grantha letters in use	ஜ ஸ ஷ ஹ க்ஷ ழ்

Fig. 1. Basic shapes of Tamil characters

Touching with nearby letters is a serious problem in segmentation of handwritten character recognition. To avoid such issues in segmentation, here we consider a city name string as a word and the city name recognition problem is treated as lexicon driven word recognition. Some of the city names may contain two or more words and for their proper handling we have concatenated these words to have a single string (word). In the proposed city name recognition scheme, at first, the images of city names are binarized and pre-segmented into possible primitive components (individual characters or its parts). Each primitive ideally consists of a single character or a sub-image of a single character. In order to merge these primitive components into characters and to find optimum character segmentation of a city name, dynamic programming (DP) is applied using the total likelihood of

characters as the objective function. To compute the likelihood of a character, an MQDF based on the directional features of the contour points of the components is used. Block diagram of the proposed system is shown in Fig.2

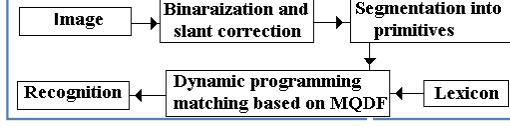


Fig. 2. Block diagram of the proposed system.

Rest of the paper is organized as follows. Details about dataset developed for Tamil city names are discussed in Section II. Different preprocessing steps utilized in our scheme are discussed in Section III. Section IV deals with feature extraction and Section V deals with segmentation recognition using dynamic programming. Detailed experimental results are presented in Section VI. Finally, the paper is concluded in Section VII.

II. TAMIL CITY NAME DATASET

Since Tamil script is used both in Sri Lanka and India (In Tamil Nadu State of India), in the dataset development, we consider city names both from Sri Lanka and India. In the proposed handwritten Tamil city name dataset, we considered 265 city names (109 Tamil Nadu city names and 156 Sri Lankan city and town names). There are 100 different samples collected for each city name. Samples of the dataset were collected from a different group of population of 500, which includes Officers (government and private), Students (University and School), Housewives, Unemployed and Retired persons. The samples were collected using pre-printed sheets of rectangular boxes. A part of the data collection sheet is shown in Fig. 3. While filling these sheets, a gel pen with 0.5 mm tip was used. Using suitable algorithms, the names were extracted from the images automatically and saved in the respective city name folder, in TIF formats. The dataset contains various types of data and to have the idea about the data some samples with our observations are shown in Fig.4.

Out of the 265 city names, there are 5 city names that contain two words, 2 city names that contain three words and the rest 258 city names contain only one word. The average length (number of characters in a city name) of the city names is 7. There are two modes obtained from the city name lengths of this data set. The first mode is at the length of 6 and the second is at the city name length of 8 (See Fig. 5). The inter-quartile range of the dataset lies between 5 character city names and 8 character city names. The maximum and the minimum lengths of the city names of the dataset are three (03) and eighteen (18). A graph for percentage of city names versus city name length is shown in Fig.5.

To the best of our knowledge, there is no dataset available for Tamil city names and hence we have developed this dataset and the database will be available freely to the researchers.

தென்மேல்	திண்டுக்கல்	மணப்பாளை	மணப்பாளை
தென்மேல்	சூதமங்கு	கொடைக்கானல்	பிள்ளைக்கானல்

Fig. 3. Part of a data collection sheet

சிம்பாளை	The upper dot (small circle) is touching with the main letter and some of the other letters also touching
சிம்பாளை	All the letters are touching side by side
சிம்பாளை	ன is written instead of ண – a careless mistake
சிம்பாளை	Negatively slanted style
சிம்பாளை	Positively slanted style

Fig.4. Some samples data with our observation.

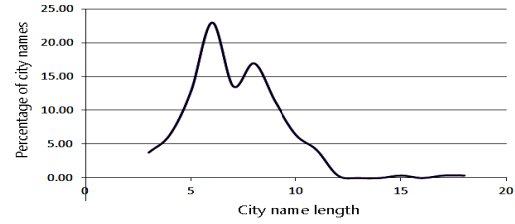


Fig.5. Graph of city names versus their length.

III. SLANT CORRECTION AND CHARACTER PRE-SEGMENTATION

A. Slant correction

Slant correction is an important preprocessing step for handwriting recognition. In this paper the gray scale image of an input city name is binarized by Otsu's algorithm [7] and the binary image is then slant estimated and corrected using the method proposed by Kimura et al. [8]. The slant estimation is done based on the chain code information of the character contours of an input city name image.

B. Character pre-segmentation

To find the optimal character segmentation by the segmentation-recognition scheme using dynamic programming, we pre-segmented the city names into primitives. When two or more characters sit side by side in a city name the neighboring characters may touch and generate a big cavity region. We find this cavity region based on the profile information and the deepest point of each cavity region is considered for pre-segmentation. Please note that all the cavities are not considered for pre-segmentation. We compute the heights of different cavities obtained in a city name and those cavities having height greater than $4 \cdot R_L$ (This value is decided based on the experiment. Here R_L is the stoke width and its

computation is discussed later) are considered for segmentation. Because of the structural shape of some characters, some small cavities may be obtained where segmentation should not be done and we use this threshold to ignore the segmentations of such small cavities. The value of this threshold is decided from the experiments and it depends on the stroke width of an input city name. In character pre-segmentation, our aim was to segment a city name into individual characters as much as possible avoiding much over-segmentation. Hence, the optimal threshold value has some impact on both the segmentation and the running time of the dynamic programming. The upper (lower) cavity portions of a Tamil city name shown in Fig.6(a) are marked by light (deep) grey in Fig.6 (b). After detection of pre-segmentation columns from the cavities of an input city name image, the image is split vertically at each pre-segmentation column and is separated into non-overlapping zones. A connected component analysis is applied to the split image to detect the boxes enclosing each connected component. These boxes are usually disjoint and do not include parts of other connected components. Connected components in the split city name image and their enclosing boxes are shown in Fig.6(c). These boxes are numbered (from left to right) and these numbers are shown at the upper/lower side of each box (see Fig.6(c)). These connected components are regarded as primitive segments and each of which corresponds to a full character or a part of a character.

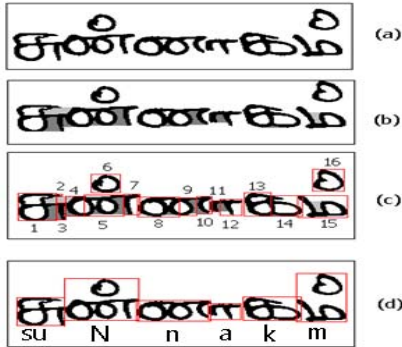


Fig.6. Examples of city name primitive segmentation. (a) An input city name (b) upper (lower) cavity portions of the city name are marked by light (deep) grey. (c). Bounding boxes of pre-segmented individual components are numbered from left to right (d) Optimum character segmentation. Segmented characters and their code are shown.

The stroke width (R_L) is calculated as follows. The image is scanned in row and column-wise and different run-lengths (of foreground pixels) with their frequencies are computed. If a component has n different run-lengths r_1, r_2, \dots, r_n with frequencies f_1, f_2, \dots, f_n respectively, then $R_L = r_i$ where $f_i = \max(f_j), j = 1, \dots, n$.

IV. FEATURE EXTRACTION

Histograms of direction chain code of the contour points of the components are used as features for recognition [8]. Extraction procedure of the 64 dimensional features is described below.

At first, the bounding box is divided into 7×7 blocks. In each of these blocks the direction chain code for each contour point is noted and frequency of direction codes is

computed. Here we use chain code of four directions only as shown in Fig.7. [direction n_0 (horizontal), n_1 (45 degree slanted), n_2 (vertical) and n_3 (135 degree slanted)]. Thus, in each block, we get an array of four integer values representing the frequencies of chain code in these four directions. These frequencies are used as feature. Thus, for 7×7 blocks we get $7 \times 7 \times 4 = 196$ features. To reduce the feature dimension, after the histogram calculation in 7×7 blocks, the blocks are down sampled into 4×4 blocks using a Gaussian filter. As a result we have $64 (4 \times 4 \times 4)$ dimensional features for recognition. The feature vector is divided by the height of the bounding box to make it size independent.

One critical point in segmentation-recognition techniques using dynamic programming is the speed of feature extraction, because the correct segmentation points have to be determined in optimization process with respect to the total likelihood of the resultant characters. The use of the cumulative orientation histogram enables one to realize high-speed feature extraction. Border following for feature extraction and orientation labeling are performed only once to an input city name image and the orientation feature vector of a rectangular region including one or more boxes is extracted by a small number of arithmetic operations for high-speed feature extraction [8].

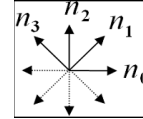


Fig.7. Different chain code directions used are shown.

V. SEGMENTATION RECOGNITION USING DYNAMIC PROGRAMMING

The core of a dynamic programming algorithm is the module that takes a city name image, a string to incorporate contextual information, and a list of the primitives from the city name image and returns a value that indicates the recognition confidence that the city name image will represent the string. Given a lexicon city name, the primitive segments of the city name image are merged and matched against the characters in the lexicon city name so that the average character likelihood is maximized using dynamic programming.

A. Markov chain representation

The number of the primitives of an input city name image is usually 1.2 to 2.5 times the number of characters in the city name image. In order to merge these primitive components into characters and find the optimum character segmentation, dynamic programming (DP) is applied using the total likelihood of characters as the objective function [8]. The ASCII (or Tamil code) lexicon of the considered city names of Tamil is utilized in the process of dynamic programming to incorporate contextual information. The likelihood of each character is calculated using the modified quadratic discriminant function. To apply the DP, the boxes are sorted from left to right according to the location of their centroids. If two or more boxes have the same x coordinates of their centroids, they are sorted from top to bottom. Numbers at

the top/bottom of the boxes in Fig.6(c) show the order of the sorted boxes. It is worth observing that the disjoint box segmentation and the box sorting process reduce the segmentation problem to a simple Markov process, in most cases. For example, the boxes 1 to 3 correspond to character “su” of the Tamil city name ‘suNnakm’ (shown in Fig.6(a)), boxes 4 to 7 correspond to character “N”, boxes 8 to 10 correspond to character “n”, boxes 11 to 12 correspond to character “a”, boxes 13 to 14 corresponds to character “k”, boxes 15 to 16 corresponds to character “m”. See Fig.6(d) where characters’ codes of this city name image are given. These assignments of boxes to character are represented by:

	su	N	n	a	k	m
$i \rightarrow$	1	2	3	4	5	6
$j(i) \rightarrow$	3	7	10	12	14	16

where i denotes the letter number, $j(i)$ denotes the number of the last box corresponding to the i -th letter. Note that the number of the first box corresponding to the i -th letter is $j(i-1)+1$. Given $[j(i), i=1,2,...,n]$ the total likelihood of characters is represented by

$$L = \prod_{i=1}^n l(i, j(i-1)+1, j(i)) \quad (1)$$

where $l(i, j(i-1)+1, j(i))$ is the likelihood for i -th letter and n denotes the total number of the letters. The optimal assignment (the optimal segmentation) that maximizes the total likelihood is found in terms of the dynamic programming as follows. The optimal assignment $j(n)^*$ for n -th letter is the one such that:

$$L^* = L(n, j(n)^*) = \text{Max}_k L(n, j(k)) \quad (2)$$

where $L(k, j(k))$ is the maximum likelihood of partial solutions given $j(k)$ for the k -th letter, which is defined and calculated recursively by

$$L(k, j(k)) = \text{Max}_{j(1), j(2), \dots, j(k-1)} \{ l(i, j(i-1)+1, j(i)) \} \\ = \text{Max}_{j(k-1)} [l(k, j(k-1)+1, j(k)) + L(k-1, j(k-1))] \quad (3)$$

and

$$L(0, j(0)) = 0 \quad \text{for } j(0) = 1, 2, \dots, m \quad (4)$$

where m denotes the total number of primitive segments.

Starting from (4), all $L(k, j(k))$'s are calculated for $k = 1, 2, \dots, n$ using (3) to find $j(n)^*$ using (2). The rest of $j(k)^*$ s ($k=n-1, n-2, \dots, 1$) are found by back tracking a pointer array representing the optimal $j(k-1)^*$ s which maximizes $L(k, j(k))$ in (3).

B. MQDF for Character likelihood

Character likelihood is calculated by the following modified quadratic discriminant function [9].

$$g(X) = \frac{1}{h^2} \|X - M\|^2 - \sum_{i=1}^k \frac{\lambda_i}{\lambda_i + h^2} [\phi_i^T (X - M)]^2 \\ + \ln[h^{2(n-k)} \prod_{i=1}^k (\lambda_i + h^2)] \quad (5)$$

where X denotes the input feature vector, M denotes the sample mean vector for each character class, and λ_i and ϕ_i denote the eigen values and eigenvectors of the sample covariance matrix. Values of constants h^2 and k are selected experimentally to achieve the best performance. In the following experiments, k is set to 20 and h^2 to $3/8 * \sigma^2$, where σ^2 is the mean of Eigen values λ_i 's over i and character classes.

Given a feature vector, $g(X)$ is calculated for a character class specified by a city name lexicon.

VI. RESULT AND DISCUSSIONS

Data details: As explained in Section II we have developed a Tamil city name dataset and we have used this set for the experiment here. Total number of city name classes is 265 and the number of samples for each country name is 100.. In total there were 26500 samples collected from all the classes. The learning samples of Tamil characters for the MQDF classifier were extracted from city name samples collected independently.

Measures of result computation: For recognition result computation we used different measures and they are defined as follows: Recognition rate = $(N_C * 100) / N_T$, Error rate = $(N_E * 100) / N_T$, Rejection rate = $(N_R * 100) / N_T$, Reliability = $(N_C * 100) / (N_E + N_C)$, Where N_C is the number of correctly classified city names, N_E is the number of misclassified city names, N_R is the number of rejected city names and N_T is the total number of city names tested by the classifier. Here $N_T = (N_C + N_E + N_R)$.

Global recognition results: From the experiment we noted that the city name recognition accuracy of the proposed scheme was 96.89%, when no rejection was considered. Also, from the experiment we noted that 98.13% (98.40%) accuracy was obtained when first two (three) top choices of the recognition results were considered without any rejection. Detailed results with different choices are shown in Table I. From the table it can be noted that recognition accuracy increases 1.24% when we consider two top choices instead of one top choice.

TABLE I. TAMIL CITY NAME RECOGNITION RESULTS BASED ON DIFFERENT TOP CHOICES (WITHOUT ANY REJECTION)

Number of top choices	1	2	3	4	5	6
Recognition rate (%)	96.89	98.13	98.40	98.65	98.77	98.86

Accuracy with respect to city name lexicon size: To give the idea of accuracy with respect to lexicon size, we computed some results with different number of lexicon sizes of the city names. We obtained 98.19% (without any rejection) accuracy when lexicon of only 53 city name classes are considered. Accuracy of 97.82% obtained when city name lexicon size was 106. Details results with respect to different lexicon sizes are given in Table II.

TABLE II. CITY NAME ACCURACY WITH RESPECT TO CITY NAME LEXICON SIZE

Number of city name classes	Recognition rate
53	98.19%
106	97.82%

159	97.40%
212	97.20%
265	96.89%

Accuracy with respect to length of city names: To get the idea about the city name recognition accuracy with respect to the length of city name, we computed city name length versus its recognition accuracy and the results are provided in Table III. From the table we can see that the proposed scheme gives higher results when city names are longer.

TABLE III. CITY NAME ACCURACY IN TERMS OF CITY NAME LENGTH

City name length	Recognition rate
01-05	95.77%
06-10	97.22%
11-15	97.56%
16-20	97.60%

Rejection versus reliability results: From the system we also computed rejection vs. reliability of our system. We noted that system provides 99.51% (99.90%) reliability when error and rejection rates are 0.45% (0.08%) and 7.00% (18.67%), respectively. City name recognition reliability with different rejection rates is given in Table IV. Rejection is done based on: (i) optimal likelihood value of the best recognized city name, and (ii) difference of the optimal likelihood values of the best and the second-best recognized city names.

TABLE IV. ERROR AND RELIABILITY RESULT OF THE PROPOSED SYSTEM WITH RESPECT TO DIFFERENT REJECTION RATES

Reliability	Error rate	Rejection rate
98.02%	1.94%	1.89%
99.01%	0.94%	4.44%
99.51%	0.45%	7.00%
99.90%	0.08%	18.67%

Comparison of results: To the best of our knowledge there are only two pieces of work on handwritten word recognition on Tamil scripts. To get an idea of comparative results of our system, we report the results here. Sigappi et al. [3] obtained 80.75% accuracy when they considered only 40 word classes. In our earlier work [4] we obtained 86.36% accuracy where 217 word classes are considered. From the present scheme we obtained 96.89% accuracy from the proposed system considering 265 Tamil city name classes.

Error analysis: From the experiment we noted that most of the errors occurred due to inaccurate segmentation i.e. when the set of character pre-segmentation points obtained from pre-segmentation module did not contain actual character segmentation points. Another reason of misrecognition was the shape similarity of some of the city names. Some examples of erroneous samples are shown in Table V.

From the experiment we have also noted pairs of mostly confusing characters and some of these pairs are shown in Table-VI. Here samples are shown in printed form to have the idea about their exact shapes.

TABLE V. SOME MISCLASSIFIED SAMPLES

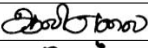
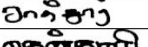
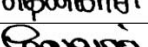
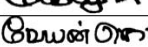

Printed version of the actual sample	Actual Sample	Recognized Sample
ஆலிஎலை		சூர்சூர்
சாத்தூர்		ஆத்தூர்
புதுக்காசி		சிவகாசி
வேலூர்		பேலூர்
வேலங்கோட்டை		மிதுவன்மலை

TABLE VI. CONFUSING TAMIL CHARACTER PAIRS

ஐ - ண	க - ச	த - ர	ரு - கு	ப - ப்
ஐ - ண	க - கு	த - ந	ரு - கு	ளி - ணி
ம - ம	க - ள	ற - த	ல - ள	ந - த
ஐ - ல	கி - சி	ற - ந	ல - வ	லி - வி

VII. CONCLUSION

In this paper we proposed a system for Tamil handwritten city name string recognition and the city name string recognition problem is treated as lexicon driven word recognition. We obtained 99.90% reliability from our proposed system when error and rejection rates are 0.08% and 18.67%, respectively. This research will be continued to address any further issues in the segmentation of touching and larger overlapping characters. The above proposed technique is general technique and can be used on any script. Before applying the technique, the coding information according to the script needs to be designed. Also the optimal threshold value adjustment based on stroke width for the segmentation need to be adjusted.

REFERENCES

- [1] R. Plamondon and S. N. Srihari. On-Line and off-line handwritten recognition: A comprehensive survey. IEEE Trans on PAMI, Vol.22, pp.62-84, 2000.
- [2] U. Pal, R. Jayadevan and N. Sharma, "Handwriting Recognition in Indian Regional Scripts: A Survey of Offline Techniques", ACM Trans. Asian Language Info. Processing, Vol-11, No-1, pp.1-35, 2012.
- [3] A. N. Sigappi, S. Palanivel and V. Ramalingam. Handwritten Document Retrieval System for Tamil Language: Int. J. of Computer Applications, Vol. 31, No.4, pp. 42-47, 2011.
- [4] S. Thadchanamoorthy, U. Pal, N. D. Kodikara and H. L. Premaretna "Holistic Recognition of Handwritten Tamil Words", In Proc. 3rd Intl Conf. on EAIT, pp. 165-169, 2012.
- [5] S. N. Srihari and E. J. Keubert, "Integration of handwritten address interpretation technology into the United states Postal service Remote Computer Reader System", In Proc. ICDAR, pp. 892-896, 1997.
- [6] U. Pal, K. Roy and F. Kimura, "A Lexicon Driven Handwritten City Name Recognition Scheme for Indian Postal Automation", IEICI Trans. on Information and Systems, Vol.E92-D, pp. 1146-1158, 2009.
- [7] N. Otsu, "A threshold selection method from gray-level histograms", IEEE Trans. SMC, vol.9, pp.62-66, 1979.
- [8] F. Kimura, M. Shridhar and Z. Chen, "Improvements of A Lexicon Directed Algorithm for Recognition of Unconstrained Handwritten Words", In Proc. ICDAR, pp.18-22, 1993.
- [9] F. Kimura, K. Takashina, S. Tsusuka and Y. Miyake, "Modified quadratic discriminant functions and the application to Chinese character recognition", IEEE Trans on PAMI, Vol.9, pp.149-153, 1987.