December 10, 2007

RE-TARGETABLE OCR WITH INTELLIGENT CHARACTER SEGMENTATION

MUDIT AGRAWAL

Re-targetable OCR with *intelligent character segmentation*

Mudit Agrawal
Language and Media Processing Laboratory
Institute of Advanced Computer Studies
University of Maryland
College Park, MD 20742
mudit@umiacs.umd.edu

Abstract

We have developed a font-based intelligent character segmentation and recognition system. Using font-characteristics of structurally similar looking font-files, this system aims at building a font-model to aid in the recognition of new scripts irrespective of glyph composition. Three feature extraction schemes have been used to demonstrate the importance of appropriate features for classification. The schemes have been tested on Latin as non-syllabic and Khmer as syllabic scripts and results are reported. Experiment results show the recognition accuracy can reach 92% for Khmer and 96% for Latin degraded image documents, both at character level. This work is a step towards recognition of scripts of low-density languages which typically do not warrant the commercial development of OCR.

Keywords: OCR, Feature extraction, Intelligent Character segmentation, Generalized character recognition, Font based recognition

TABLE OF CONTENTS

List of Figures

List of Tables

1 Introduction

There are many script-specific recognizers in the market. Such recognizers are generally based on some assumptions on the dataset and try to cover a wide variety of font sizes, script properties and request a very large collections of training samples (with ground-truth). Unfortunately, they are often too general and can not adapt well to outliers caused by noise or non-traditional fonts. Our challenge is to train and build a system which can tune itself to any new script with a minimal set of training samples and limited user interaction. With the base of recognition remaining same, accuracy should be enhanced by using some script specific features, if needed. Despite much ongoing research on non-Latin script recognition, most of the commercial OCRs still feature Latin script recognition as one of the primary objectives with high accuracies. Efforts on non-Latin scripts are quite segregated and continue to be tailored for specific scripts using their inherit features explicitly. The outcomes of such efforts are costly and do little to advance the field.

Traditionally, pattern recognition techniques for character recognition have been divided into two major categories [1] – template-based and feature-based [2, 3]. Template-based approaches aim to create a probabilistic template of each character model from the training data. During testing, the unknown pattern is superimposed directly on the ideal template pattern and degree of correlation is used to decide about the classification. This is generally the first step towards any new script analysis and classification. To enhance the accuracy, template based results are often combined with feature-based approaches.

Feature-based approaches extract feature vectors from training samples and aim to create a class-model out of all vectors of a given class. The challenge is to derive those features that aid in differentiating the class from other classes rather than solely representing that class. The feature-based approaches can be of two types, namely spatial domain and transform domain approaches [4, 5]. Spatial domain approaches derive features directly from the pixel representation of the pattern. In a transform domain technique, the pattern image is first transformed into another space using, for example, Fourier, Cosine, Slant or Wavelet transform and useful features are derived from the transformed images [4]. Support Vector Machines (SVMs) have also been successful at recognizing various non-Latin scripts. Though SVMs are suited for binary classification problems, DAG-SVMs can solve a *set of* 2-class classification problems, hence a multi-class problem – by choosing the maximum of the outputs of all SVMs.

Apart from these two techniques, there is plethora of other techniques which do not explicitly derive features from the patterns [6, 7]. During training, after normalization, the system adjusts its parameters to minimize the misclassifications. The system, thus trained is used for classifying unknown patterns. One of the most popular methods in this domain is artificial neural network – which adjusts its weights from the training samples – and uses these weights as features during classification. Though many papers have been published in this domain [8, 9], the weights in neural networks can not be analyzed like feature vectors – thus making the process, a black-box mechanism. Hidden Markov Model (HMM) [10] is a non-explicit feature-based method which works on large number of training samples to estimate the probability parameters. It has been quite successful in handwriting and speech recognition. Fuzzy rules [9], Mahalanobis and Hausdorff distance, Evolutionary algorithms [11] are other techniques used for the recognition purposes.

The paper has been divided into six sections. Section 2 deals with the complexity of various scripts, our recognition system with various modules. Section 3 elaborates on the challenges which we faced with the system and how font-files analysis, models, training and testing proposed a solution. This is followed by experiments in section 5. Some conclusions and directions of future work are presented in section 6.

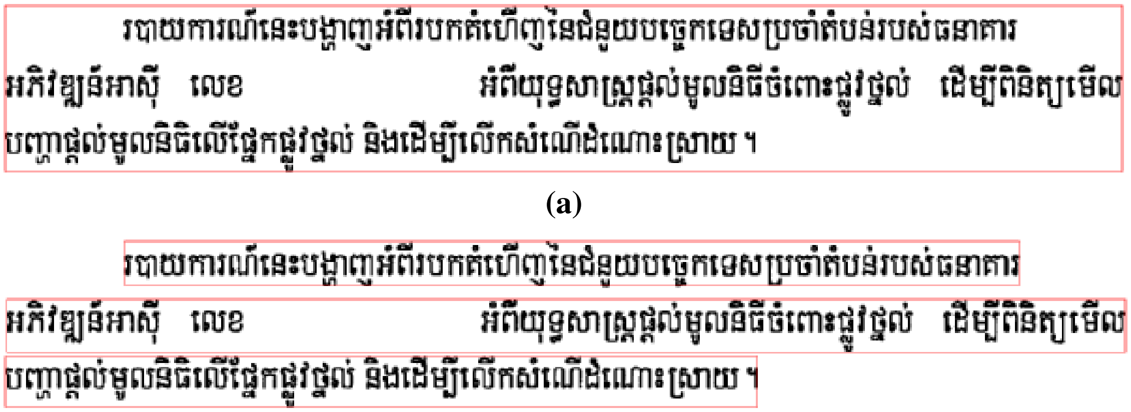
2 Approach

The objective is to create a generic script recognizer which can be trained using minimal number of document samples. Since our research is more targeted towards unexplored scripts like Khmer etc., easy availability of ground-truth data can not be assumed [12]. Due to this, many techniques like SVMs, which require a high amount of training data, can not be used for training and classification. Apart from that, limited user feedback is also the key to system's adaptiveness. The system contains three different functional components (1) Hierarchical segmentation (2) Feature Extraction (3) Classification. The modules are described in detail as under:

2.1 Segmentation

It is practically infeasible to segment out characters for complex non-Latin scripts for feature extraction. Hence, a limited user feedback mechanism was developed in which whole document ground-truth (text) is fed along with the document image. The text-alignment is done by aligning zone, line and words in image and document text. Since ground-truth can be erroneous, only the best-match parts are returned. Syllabic and non-syllabic scripts have different procedures for character alignment. Words in non-syllabic scripts (e.g. English) are composed of vertically separable characters whereas in syllabic scripts (like Khmer, Devanagari etc.) they are composed of vertically separable syllables. Each syllable can be further broken down into characters – a process which is very script specific. The system was developed for Latin as non-syllabic and Khmer as a syllabic script.

Figure 1 shows the segmentation of Khmer document in zones, lines, words, syllables and characters.

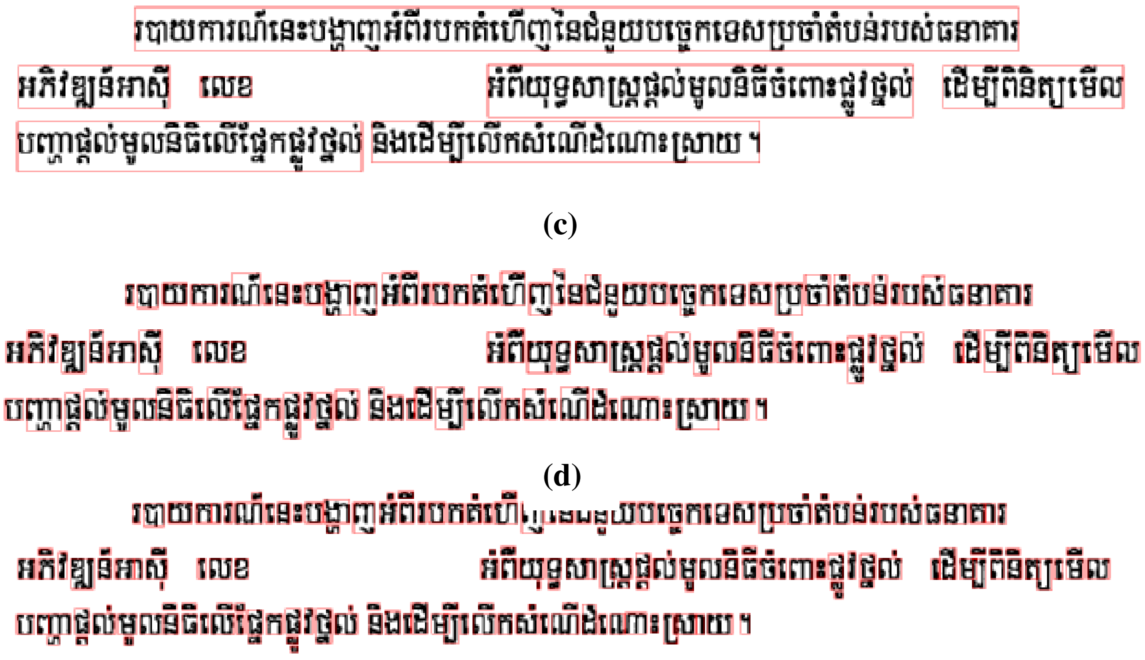


របាយការណ៍នេះបង្ហាញអំពីរបកគំហើញនៃជំនួយបច្ចេកទេសប្រចាំតំបន់របស់ធនាគារ
អភិវឌ្ឍន៍អាស៊ី លេខ អំពីយុទ្ធសាស្រ្តផ្តល់មូលនិធិចំពោះផ្លូវថ្នល់ ដើម្បីពិនិត្យមើល
បញ្ហាផ្តល់មូលនិធិលើផ្នែកផ្លូវថ្នល់ និងដើម្បីលើកសំណើដំណោះស្រាយ ។

(a)

របាយការណ៍នេះបង្ហាញអំពីរបកគំហើញនៃជំនួយបច្ចេកទេសប្រចាំតំបន់របស់ធនាគារ
អភិវឌ្ឍន៍អាស៊ី លេខ អំពីយុទ្ធសាស្រ្តផ្តល់មូលនិធិចំពោះផ្លូវថ្នល់ ដើម្បីពិនិត្យមើល
បញ្ហាផ្តល់មូលនិធិលើផ្នែកផ្លូវថ្នល់ និងដើម្បីលើកសំណើដំណោះស្រាយ ។

(b)


របាយការណ៍នេះបង្ហាញអំពីរបកគំហើញនៃជំនួយបច្ចេកទេសប្រចាំតំបន់របស់ធនាគារ
អភិវឌ្ឍន៍អាស៊ី លេខ អំពីយុទ្ធសាស្ត្រផ្តល់មូលនិធិចំពោះផ្លូវថ្នល់ ដើម្បីពិនិត្យមើល
បញ្ហាផ្តល់មូលនិធិលើផ្នែកផ្លូវថ្នល់ និងដើម្បីលើកសំណើដំណោះស្រាយ ។

(c)

(d)

(e)

Figure 1 (a) Zone Segmentation (b) Line Segmentation (c) Word Segmentation (d) Syllable Segmentation (e) Character Segmentation of Khmer script

For each connected component in a word, accents or separate dots are merged to form a character [13]. With the assumption that a character won't be too wide or too narrow, a connected component satisfying the following conditions is considered a template candidate:

(1) The aspect ratio falls in the range $[r_{low}, r_{high}]$;

(2) The area is larger than $A_{min;}$

where r_{low}, r_{high} are the predefined low and high aspect ratio thresholds respectively and A_{min} is the area threshold (derived from the data, the values were found to be 0.2, 1.0 and 5 respectively [13]). For each matched (aligned) character in the ground-truth text and document word-image, a class-template is created. However, for each word, the newly extracted components may leave another isolated component which can be added to the template set. The remaining part of the word is then rechecked and if doesn't match any template above a threshold, a new template map is generated. The new templates can be used to find more matches, with additional templates generated by the same procedure. This process is iterated until no new template forms.

2.2 Feature Extraction

After character segmentation, each character is processed through a featurization routine where the best-describing or differentiating features are extracted out in form of a feature vector. These feature vectors are then used in training and testing. Three featurization routines were developed and can be used interchangeably using configuration files:

Template initialization: Each character image is first resized to a 32-by-32 vector map. A probabilistic template is generated through all samples of sample class from the training data [12]

Zernike Moments: Moment descriptors have been studied for image recognition and computer vision since 1960s. Teague [14] first introduced the use of Zernike moments to overcome the shortcomings of information redundancy present in the popular geometric moments. Zernike moments are a class of orthogonal moments which are rotation invariant and can be easily constructed to an arbitrary order. And it was shown in [15] that Zernike moments are effective for the optical character recognition (OCR).

Directional Features: Template and Zernike moments do not utilize inherit 'directional' property of complex scripts. The relative placement of neighboring pixels is more important than the overall placement of pixels forming the character. On one hand where templates are too rigid about character's shape and for noisy documents they can result in poor models for classification, Zernike moments are transform-based feature analysis method – which are difficult to visualize in that dimension. Directional Feature [16] records the relative neighboring pixel positions for each contour pixel and generates a feature vector using that information.

The character image is normalized and contour extracted. It is then resized to a 64-by-64 mesh. This mesh is divided into 49 (7-by-7) sub-areas of 16-by-16 pixels where each sub-area overlaps eight pixels of adjacent sub-area (see figure 2). For each sub-area, a four-dimensional vector (x1, x2, x3, x4) is defined where x1, x2, x3, x4 record the relative direction (vertical, horizontal, forward inclined, backward inclined) of neighboring pixels for each pixel in the sub-area. Hence, a 49 x 4 = 196 unit long feature vector is produced. Figure 2 shows the directional feature extraction process step-by-step.

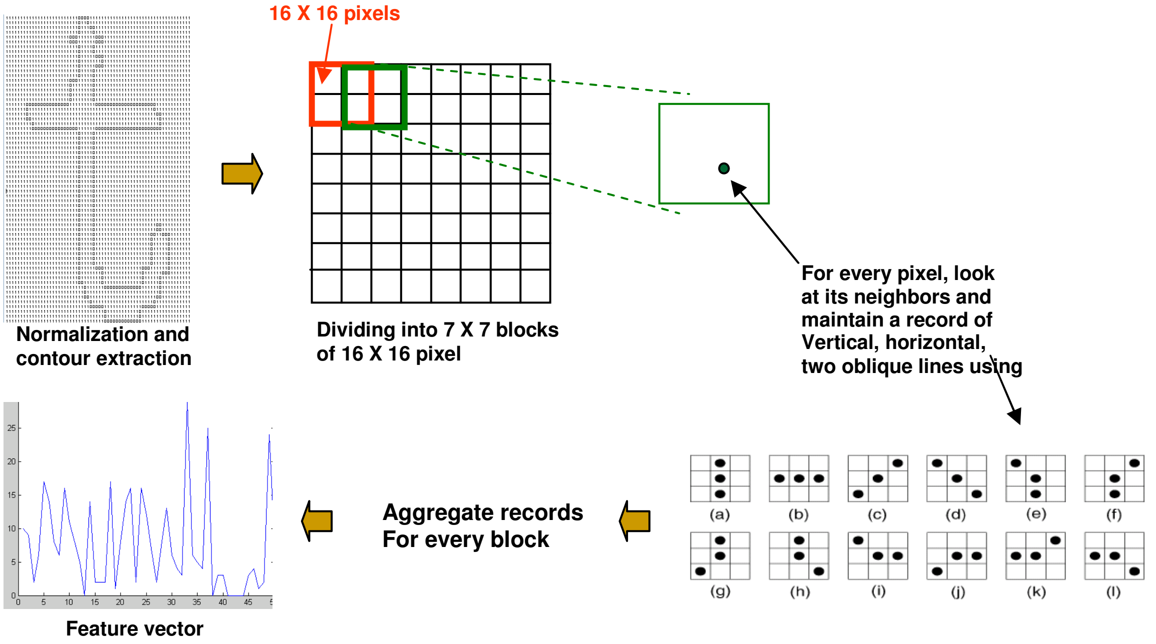

16 X 16 pixels
Normalization and contour extraction
Dividing into 7 X 7 blocks of 16 X 16 pixel
For every pixel, look at its neighbors and maintain a record of Vertical, horizontal, two oblique lines using
(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)
Aggregate records For every block
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40
45
Feature vector

Figure 2 Directional Element Feature Extraction

2.3 Classification

During the testing phase, after character segmentation and featurization, each feature vector is classified to one of the trained classes. The following methods were tested for classification.

2.3.1 Template Matching

Awarding probabilities when template pixel matches with the corresponding pixel in a candidate character image and penalizing otherwise, forms the core objective of template matching. The template which has the best match is considered to be the class of the character image. The candidate character image is binary, while the pixel values of the template map g(x, y) are in a range [0,N_{inst}], therefore g(x,y) is first normalized. The similarity of a character image f(x,y) and a template g_b(x,y) is defined as a weighted similarity as:

$$S_w(f,g) = 1.0 - \frac{1}{N^2}\sum_{x=1}^{N}\sum_{y=1}^{N} w(x,y)|f(x,y) - g_b(x,y)|$$

where the weight w(x,y) is defined as:

$$w(x,y) = \begin{cases} 1.0 & \text{if } g_b(x,y) \text{ is background} \\ g(x,y)/N_{inst} & \text{if } g_b(x,y) \text{ is foreground} \end{cases}$$

2.3.2 Hierarchical Classification

Kanji and South-East Asian scripts have a large set of alphabets. Hence, one-stage discrimination doesn’t generally suffice. In this approach, two-stage classification was used: rough and fine classification. The aim of rough classification is to cluster similar-looking characters into groups and then perform fine classification to extract the right class [16].

a. City Block Distance with Deviation (CBDD)

Let v = (v_1, v_2, … v_n) be an n-dimensional input vector and μ = (μ_1, μ_2, … μ_n) be the standard vector of a category. The CBDD is defined as:

$$d_{CBDD}(\mathbf{v}) = \sum_{j=1}^{n} \max\left\{0, \left|v_j - \mu_j\right| - \theta \cdot s_j\right\},$$

where s_j denotes the standard deviation of j^{th} element, and θ is a constant.

b. Asymmetric Mahalanobis Distance

For each cluster, the right class is obtained by finding the minimum asymmetric Mahalanobis distance from the templates in that cluster. The function is given by:

$$d_{AMD}(\mathbf{v}) = \sum_{j=1}^{n} \frac{1}{\left(\hat{\sigma}_j\right)^2 + b} \left(\mathbf{v} - \hat{\mu}, \phi_j\right)^2,$$

where b is the bias, - $\hat{\mu}$ is the quasi mean vector of the samples of the class m, φ_j is the eigenvector of covariance matrix of this category and $\hat{\sigma}_j$ is the quasi variance. In case of a tie, N-nearest neighbor is called, with N = 3.

3 Font-file based Intelligent Character Segmentation

3.1 Motivation

The paper has already discussed the differences in character segmentation approaches for Latin and non-Latin scripts. The structure of non-Latin scripts vary a lot from Latin scripts not only in form of character shapes but also in writing order, layout and word-compositions. Presence of language-specific constructs, in the domain of non-Latin scripts, such as shirorekha (Devanagari), modifiers (South-East Asian scripts), or non-regular word-spacing (Arabic and Chinese) require different approaches to layout analysis. Character segmentation is the major challenge in Chinese and South-East Asian scripts like Khmer, Devanagari etc. Asian scripts though share a lot of common properties yet pose different complications while dealing with segmentation. For example, figure 3 (c) shows the word العربية *al-arabiyyah*, "the Arabic [language]" in Arabic, in stages of rendering. The first line shows the letters as they are unprocessed, the result that would be given by an application without complex script rendering. In the second line the bidirectional display mechanism has come to play, and in the third the glyph shaping mechanism has rendered the letters according to context. 3 (b) shows a word in Devanagari where characters are combined together to form a word. A character's appearance is affected by its ordering with respect to other characters, the font used to render the character, and the application or system environment. Additionally, like Burmese (figure 3(a)), in Devanagari or other Indic scripts, few characters cause a change in the order of the displayed characters. These features propose greater challenges in segmentation of characters from word-images. While conventional vertical or horizontal profiling methods fail in segmenting out characters directly from words, character segmentation from syllables using only connected component analysis itself is a complex task which is highly correlated with the script characteristics.

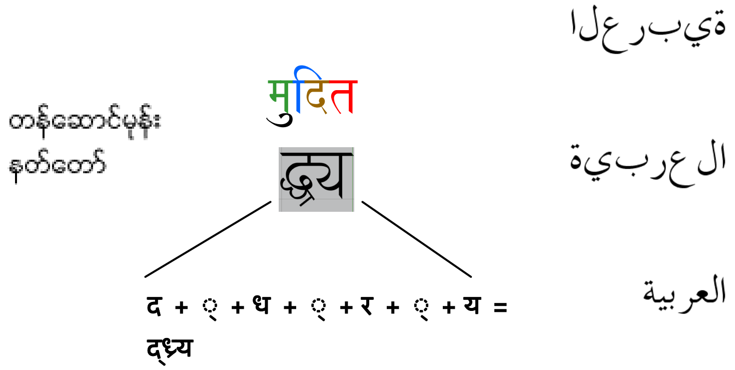


Figure 3 (a) Burmese script (b) Devanagari character composition (c) Arabic Word rendering

Apart from these, degraded text-documents have their own pool of recognition problems. Due to broken and touching characters, many times the character segmentation fails which in turn effects feature extraction and classification. One of the major components of a good recognition system is a feature-extraction module. [17] talks about various feature extraction methods for off-line recognition of segmented (isolated) characters. The choice of right feature extraction algorithm is considered to be the single most important factor in achieving high recognition performance. Various algorithms reported in the paper talks about invariance properties, reconstructability, expected distortions and variability of the characters. The assumption carried throughout is the availability of *segmented isolated characters!* Some of the complex problems in the field of syllabic character segmentation have been already enlisted. This implies that the benefits of good feature extraction modules (followed by classifiers or their combinations) can not be reaped until we have a robust *generic* solution to character segmentation problem.

[18] talks about four methods of character segmentation. First being *dissection,* where image is decomposed into classifiable units *before* featurization and classification. This solution faces the curse of recluse! Due to its disconnectivity from the later modules,

the process doesn't enjoy the benefits of feedback from those modules. Second set of methods try to classify subsets of spatial features collected from a word image as a whole. No complex 'dissection' algorithm has to be built and recognition errors are basically due to failures in classification. Segmentation hypotheses are generated and choice of best hypotheses along the word, gives best recognition result. The challenge is this approach is to come up with minimal number of possibly correct hypotheses. The third set of strategies is to *over-segment* the word-image at some heuristics. Though these techniques do fairly well in handwriting domain, their fruitfulness has not yet been established in printed-character recognition. The fourth method recognizes an entire word as a unit, and is termed as holistic strategy. A major drawback of this class of methods is that their use is usually restricted to a predefined lexicon.

3.2 Solution: Font-Models

With the motivation of building up an intelligent *generic* character segmentation and recognition system for any complex syllabic script, font files, being the source of script-renderings, were analyzed! Due to the unavailability of ground-truth data for less-researched scripts, glyphs from a bunch of font-files can be used as training samples. Font-files have a wealth of information and possess a *generative model* for every character. They contain the following set of information for a given script font:

- List of characters
- Glyphs of each character
- Font ascender
- Font descender

For each character at a particular font size, the file contains its

- Unicode value
- Height
- Width

- Horizontal Advance: the horizontal distance between the origins of present and next character in a word
- Vertical Advance: the vertical distance between the origins of present and next character in a word
- Bounding Box
- Left Bearing: the horizontal distance between the left-end of bounding box and its origin
- Right Bearing: the horizontal distance between the right-end of bounding box and origin
- Rules of combination of various characters forming complex new shapes

Many of such parameters are redundant, as they can be derived from other parameters. e.g.

Right bearing = |*left bearing*| + *horizontal_advance* – *width*

where width = bounding box right edge – bounding box left edge

A word's rendering takes place in the following manner. First character is placed with the given font face and size. Using the horizontal-advance and vertical-advance, origin of next character is determined. When the next character is printed, its left-bearing and bounding-box top-edge determines where to place it with respect to the previous character. Figure 4 shows the process steps.

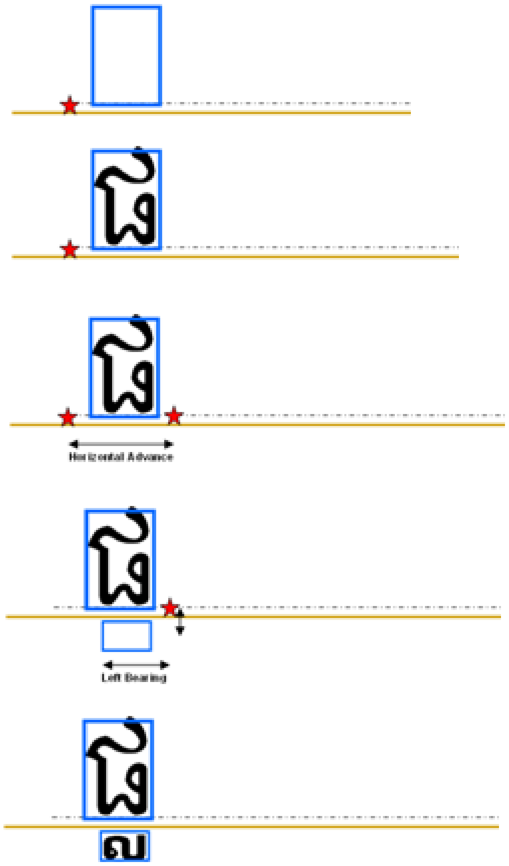


Figure 4 shows rendering of characters in a word in Khmer using font-models (a) Locating first char (b) Placing char into the location (c) Determination of Origin of next character (d) Determination of next char's position (e) Placing the second character

Hence, using a group of similar looking font files, glyphs can be extracted [19, 20] and used for training purposes. This eradicates the problem of unavailability of all characters in the huge alphabet of syllabic scripts with limited ground-truth. The glyphs extracted can substitute for missing or lesser available character classes from ground-truth. For a given font face and size, following information was extracted: (a) glyph of characters (to train them along with samples available from limited ground-truth) (b) horizontal-advance of each character (to determine the position of next character's origin) (c) bounding box location of present character (given its origin). This information is then used to segment characters from word-images during training using the process described above (figure 4).

Before venturing into training using font-files, a number of structurally similar looking font-files were processed to visually analyze the consistency in these model-parameters for a given font-size. The figure 5 summarizes the results for three fonts of Devanagari script. As shown, all the fonts place each character nearly at same position, hence validating the consistency in these parameters. Similar analysis was done for non-syllabic script (English) as well.

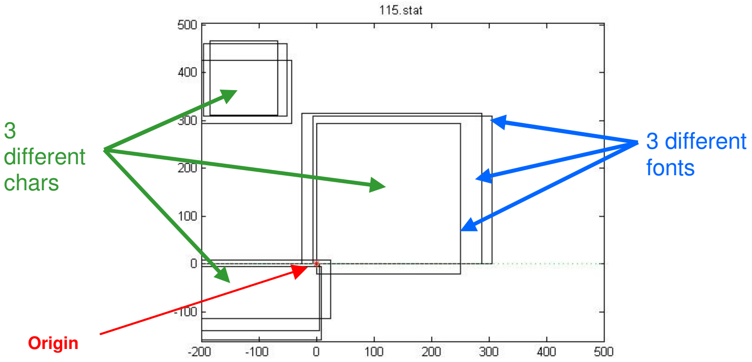


Figure 5 shows locations of 3 different characters of Devanagari script using 3 structurally similar Devanagari font files

3.3 Training using Font files

The flowchart below describes the step-by-step process of training document images with sparsely available ground-truth data.

Step 1, 2: A bunch of similar looking font files, resembling the text in documents under train, are entered along with ground-truth files.

Step 3: Bounding box and horizontal advance properties for each character are extracted from the font files and averaged out.

Step 4: The character glyphs from font files are passed through the feature extraction routines.

Step 5: Each document image along with its corresponding ground-truth file, is passed through zone segmentation module and a CNode structure (containing Page → Zone → Line → Word) is created containing word-alignments.

Step 6: Each word in this CNode structure is further segmented into characters using font-properties extracted out in step 2 and these characters are aligned with their corresponding ground-truth.

Step 7: For each character segmented out in the document image, feature extraction is performed and merged with that of its corresponding glyph from step 4.

Step 8: System is trained using the features

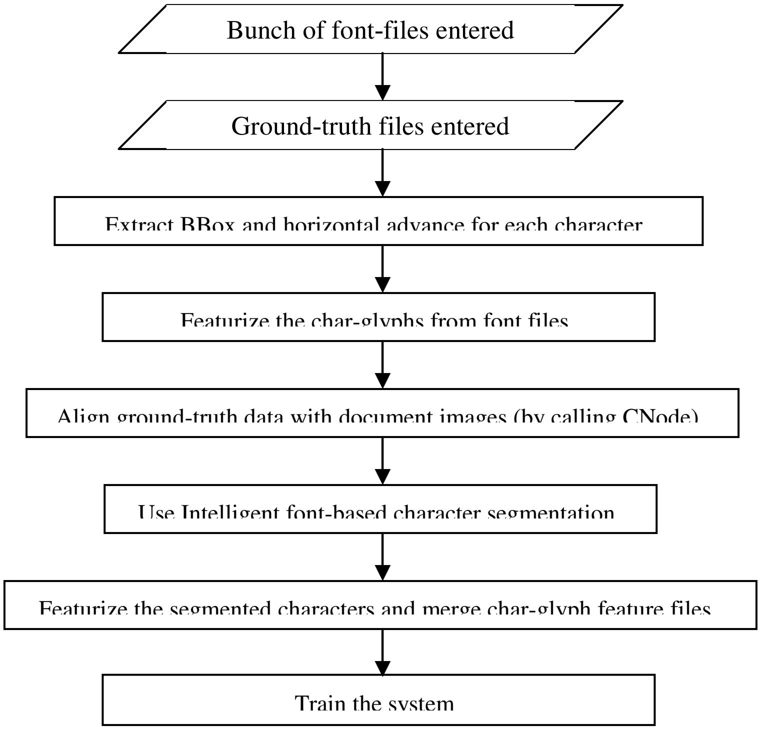

Bunch of font-files entered
Ground-truth files entered
Extract BBox and horizontal advance for each character
Featurize the char-glyphs from font files
Align ground-truth data with document images (by calling CNode)
Use Intelligent font-based character segmentation
Featurize the segmented characters and merge char-glyph feature files
Train the system

Figure 6 Training Chart using Font Models

3.4 Segmentation and Recognition

With the objective of grouping broken characters, segmenting conjuncts and touching characters, the technique of font-based intelligent character segmentation and recognition was developed. As discussed earlier, it falls in second category of character segmentation with an advantage of reducing the hypotheses by the knowledge of next character's position, given the present character. This is achieved using the font-file parameters.

Algorithm:

The document image file is classified into zones, lines and words. For each word, connected component analysis is performed. Assuming maximum N uncovered components can be combined together to form the next character, there can be ${}^{N}C_1$ + ${}^{N}C_2$ + … ${}^{N}C_N$ possible nodes (η_i) for next stage (typically N = 3). Given the present character, propositions (ρ_i) are made for next-character's locations (using font-model). Those η_i which do not overlap (with threshold τ) with any ρ_i, are discarded. η_i which overlap (with threshold τ) with any ρ_i are inserted into a set γ. η_i which enclose any ρ_i are inserted into a conjunct set δ. Nodes of set γ are ranked as per their confidences returned from the recognizer. Nodes of the conjunct set δ are given for *conjunct-test* (described later). If they pass the test, the conjunct is broken into possible characters using Dijkstra's algorithm [12, 21] and individual character confidences are returned. Only the first character (along with its confidence) from every conjunct is kept in the set δ and later characters are removed. The best confidence character is picked from set γ and δ combined. The process is repeated for the uncovered connected components in next stage. In case of dead-ends (when no possible character location coincides with the present connected-component nodes), back-tracking is performed. Figure 7 below shows the process.

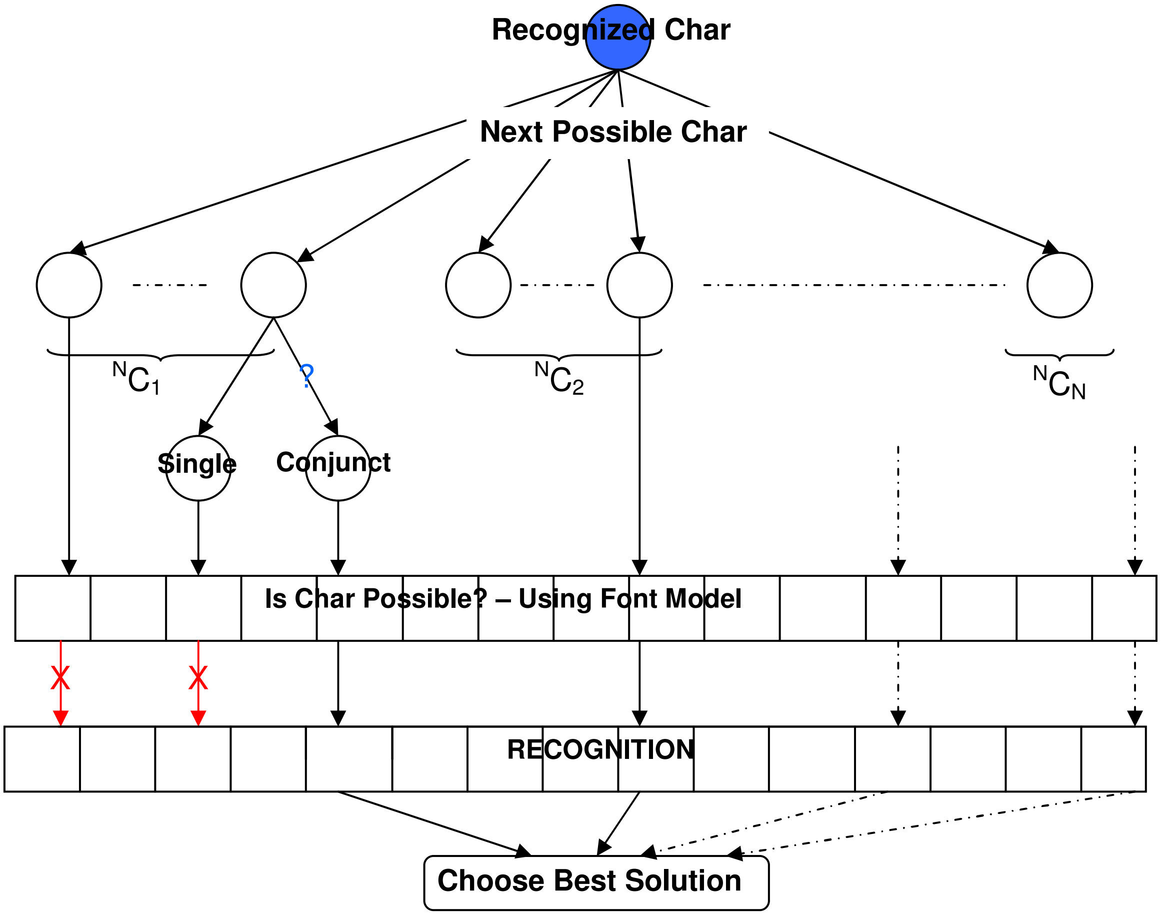


Figure 7 Dynamic Network created during best-path search of word-recognition (using Font-models)

Conjunct-test: The conjuncts form an integral part of any syllabic scripts. Many characters combine together to form a single shape. Without the prior knowledge of the script and using special modules, it has been nearly impossible to detect and split a conjunct into its character components. Techniques so far have relied on a crude method of aspect ratio threshold to determine if a character is a conjunct and needs to be broken down further. With font-models, an intelligent conjunct detection procedure has been developed. As shown in figure below, a character is passed for conjunct-analysis only if it encompasses the possibility of two or more characters of the given script under test. This position-analysis can be done only through font-models of the script, as illustrated in figure 8.

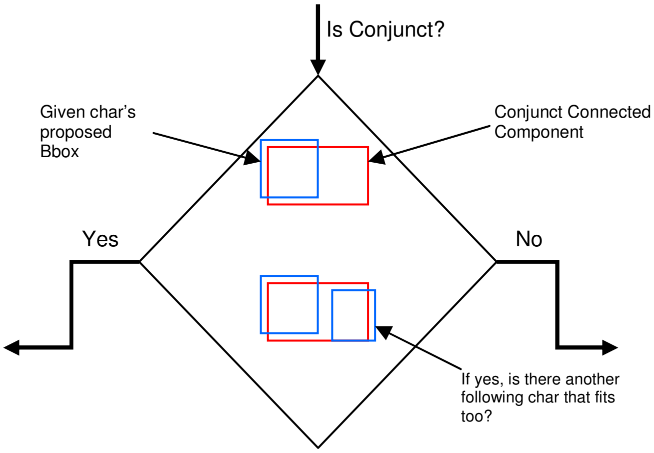

Is Conjunct?
Given char's proposed Bbox
Conjunct Connected Component
Yes
No
If yes, is there another following char that fits too?

Figure 8 Conjunct Detection. Red Box shows a possible conjunct under detection. If two characters (blue boxes) fit in (using font-model), it is a conjunct

4 Experiments

4.1 Datasets

Our experiments were conducted on two scripts – Latin (non-syllabic) and Khmer (syllabic). Two datasets of English – KTI and DOE and one Khmer dataset were used. KTI documents have varying amount of clarity across the pages which lead to quite good number of broken and touching characters (figure 9). Apart from this, the documents contain noise introduced during printing and scanning process. Also, the characters in words are skewed and not aligned perfectly with word's bottom reference line. This imposes challenges for character segmentation and prediction of next-character position using font-models. The most resembling font is *NSimSum.* DOE dataset, on other hand, is a much cleaner dataset, with font resembling more closely to *Courier New.* A single English document had approximately 2000 characters and 330 words. Khmer dataset contains some documents from Cambodian Gazetteer (7 in number) and few documents (8 in number) scanned from other resources. These documents have dark prints and hence suffer badly from touching-character problem. This combined with the presence of numerous conjuncts in Khmer script, becomes a great dataset for evaluation of our techniques. The closest font to the documents is of *Limon S1.* A single Khmer document had approximately 1500 characters and 100 words.

In dataset, randomly three to five documents were chosen for training. The idea was to evaluate our schemes under limited user feedback and training set – which is generally the case for any new script under study. The accuracy figures reported are the average figures of test documents (2 to 5 in number)

4.2 Protocols for evaluation

The text-documents returned by the OCR system are matched against the ground-truth data using an in-lab evaluation tool called *Lamp Lab Evaluation Tool.* The evaluation tool prints out an elaborate description of insertion, deletion and substitution errors in form of one-to-one, one-to-two, two-to-one, two-to-two confusions. It also summaries (in descending order) the most confused characters along with their confusions. Apart from character confusions, it also dumps word-confusion matrices in the similar fashion. These results were used to analyze the problems with our schemes and helped recovering many bugs and errors in the programs.

4.3 Feature Extraction

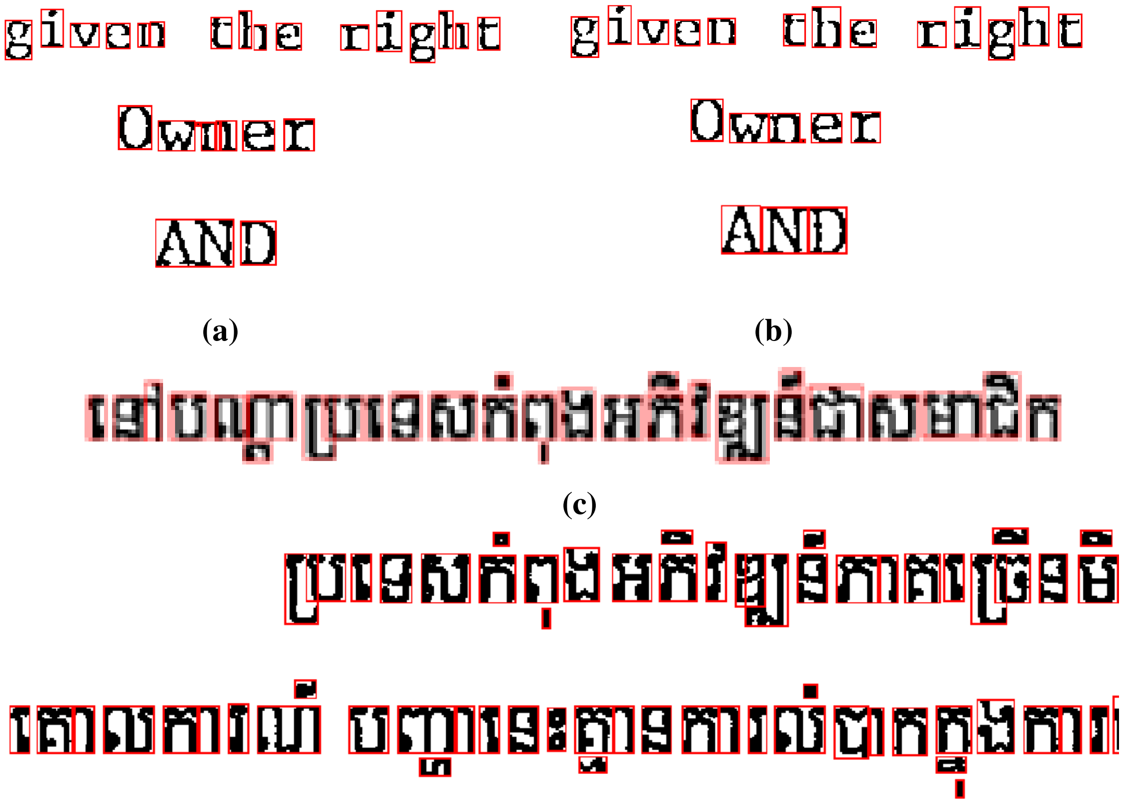
The template matching and directional feature extraction results are compared below, both for English and Khmer documents. Weighted similarity measure (section 2) was used to classify templates and CBDD was used to classify directional feature set.

Table 1 Compares character level accuracy results for Latin and Khmer script using Template and Directional features

	English		**Khmer**	
	Template Matching	**Directional Features**	**Template Matching**	**Directional Features**
Char Accuracy	86%	93%	84%	89%

4.4 Character Segmentation

The following figures show the improvements in character segmentation both for broken and touching characters. Figure 9 (a) shows the results of segmentation using *dissection-based* technique for English KTI document, whereas (b) shows intelligent font-based segmentation for the same document. (c) shows the results of segmentation using *dissection-based* technique for Khmer (d) using intelligent font-based segmentation



(a)

(b)

(c)

(d)

Figure 9 shows improvements of our technique over older dissection based techniques (a) shows Latin script character segmentation using dissection-based technique (b) shows the results using our technique (c) & (d) show results for Khmer script using dissection and our font-model based technique respectively

4.5 Recognition Results

The table below summaries the improvements gained using font-model based intelligent character segmentation and recognition – both for KTI English and Khmer dataset. Character accuracies as well as word accuracies have been reported. The accuracies reported are using *directional feature extraction scheme* and *CBDD classifier.* WOFM stands for WithOut Font Model (following *dissection-based* segmentation) and WFM stands for With Font Model

Table 2 Compares char and word level accuracies for Latin and Khmer scripts using dissection based and font-based techniques

	English		**Khmer**	
	WOFM	**WFM**	**WOFM**	**WFM**
Char Accuracy	93%	96%	89%	92%
Word Accuracy	83%	89%	38%	37%

Due to much higher word-length in Khmer, the word-accuracies plummet as compared to English.

5 Conclusion and Future Work

The paper presented a novel technique to intelligently segment and recognize characters in complex syllabic scripts, using font-models. It also emphasized the importance of a good feature extraction module (directional features over template or Zernike moments). These techniques not only enhanced degraded text-recognition results, but also obviated the need of a large number of training documents. An intelligent conjunct-detection scheme was also proposed which is more intuitive than aspect ratio. These techniques do not differentiate syllabic or non-syllabic approaches for segmentation and hence carry out direct character segmentation from words even for syllabic scripts. It aims for word-based recognition and hence is *ready* for language-models.

This technique however is slower than *dissection-based* segmentation and recognition, as it inculcates best possible recognition results analysis at every step of word-recognition. Also, it is susceptible to mis-recognitions if the font-size of a character changes abruptly within a single word. This is because the model works on a self-learning font-size model for each word and currently can not handle big aberrations in character font-sizes within a single word.

The next step towards improvement of recognition is a right combination of classifiers. Bootstrapping mechanism to re-learn or update training files during recognition will boost the recognition rate.

References

1. U. Pal, B.B. Chauduri, Indian script character recognition: a survey, Pattern Recognition, Vol. 37, 2004.

2. L. O’ Gorman and R. Kasturi. Document Image Analysis, IEEE Computer Society Press, Los Alamitos, CA (1995)

3. Y. Suen, M. Berthod and S. Mori, Automatic recognition of hand-printed character—the state of art. Proc. IEEE 68 (1980), pp. 469–487.

4. G.H. Granlund, Fourier pre-processing for hand-printed character recognition. IEEE Trans. Comput. C21 (1972), pp. 195–201.

5. S.A. Mahmoud, Arabic character recognition using Fourier descriptors and character contour encoding. Pattern Recognition 27 (1994), pp. 815–824.

6. C. Choisy and A. Belaid, Cross-learning in analytic word recognition without segmentation. Int. J. Document Anal. Recognition 4 (2002), pp. 281–289

7. R. Plamondon and S.N. Srihari, On-line and off-line handwritten recognition: a comprehensive survey. IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000), pp. 62–84.

8. U. Bhattacharya, T.K. Das, A. Datta, S.K. Parui and B.B. Chaudhuri, A hybrid scheme for handprinted numeral recognition based on a self-organizing network and MLP classifiers. Int. J. Pattern Recognition Artif. Intell. 16 (2002), pp. 845–864

9. Y. Chi and H. Yan, Handwritten numeral recognition using self-organizing maps and fuzzy rules. Pattern Recognition 28 (1995), pp. 56–66.

10. A.S. Britto, R. Sabourin, F. Bortolozzi and C.Y. Suen, The recognition of handwritten numerals strings using a two-stage HMM based method. Int. J. Document Anal. Recognition 4 (2003), pp. 102–117.

11. D. Stefano, A.D. Cioppa, A. Marcelli, Handwritten numeral recognition by means of Evolutionary Algorithms, Proceedings of the Fifth International Conference on Document Analysis and Recognition, 1999, pp. 804–807

12. H. Ma and D. Doermann, Adaptive OCR with Limited User Feedback, 8th Int'l Conf. Document Analysis and Recognition (ICDAR), 2005, pp 814-818

13. H. Ma and D. Doermann, Adaptive Hindi OCR Using Generalized Hausdorff Image Comparison, ACM Transactions on Asian Language Information Processing, Vol. 26, No. 2, 2003, pp198-213

14. M. Teague. Image analysis via the general theory of moments. Journal of the Optical Society of America, 70(8):920–930, 1979

15. A. Khotanzad and Y. H. Hong. Invariant image recognition by Zernike moments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(5):489–497, 1990.

16. Nei Kato, Masato Suzuki, Shin'ichiro Omachi, Hirotomo Aso, Yoshiaki Nemoto, A Handwritten Character Recognition System Using Directional Element Feature and Asymmetric Mahalanobis Distance, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 3, pp. 258-262, Mar., 1999

17. D. Trier, A. K. Jain, and T. Taxt, Feature extraction methods for character recognition - a survey, Pattern Recognition, vol. 29, no. 4, pp. 641-662, Apr. 1996.

18. G. Casey and E.Lecolinet, A Survey of Methods and Strategies in Character Segmentation, IEEE on pattern Analysis and machine intelligence, vol. 18, 1996.

19. G.E. Kopec and P.A. Chou, Document Image Decoding Using Markov Source Models, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, no. 6, pp. 602-617, June 1994.

20. A.C. Kam and G.E. Kopec, Document Image Decoding by Heuristic Search, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, no. 9, pp. 945-950, Sept. 1996.

21. http://en.wikipedia.org/wiki/Dijkstra's_algorithm

22. http://unicode.org/book/ch09.pdf