Composite Script Identification and Orientation Detection for Indian Text Images

Shamita Ghosh and Bidyut B. Chaudhuri
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute,
Kolkata – 700 108, India
e-mail: {shamita.ghosh, bbcisical}@gmail.com

Abstract— **A major preprocessing step in a multi-script OCR is to identify the script type of the test document image. The published papers on script identification usually assume that the test image is in correct i.e. 0° orientation. But by mistake a document may be fed to the system in wrong orientation, say at an angle of nearly 180° or ±90°. In this method we propose a script identification method that works for unknown orientation for all 11 official Indian scripts. Here, we first find the skew and counter-rotate the document by the skew angle. This will lead to correct (0°) or upside down (180°) orientation. Then script identification is done by a multi-stage tree classifier using features invariant to 0°/180° orientation. Next we go to find the orientation of the image by a two class classifier for each script. Performance of the proposed method has been tested on a variety of documents and promising results have been obtained.**

Keywords- Script type identification; Indian document processing; Reservoir principle; Orientation detection;

I. INTRODUCTION

The problem of determining the script in a document image has important applications in sorting and searching of images as well as multi-script optical character recognition (OCR). In India there are eleven official scripts and this paper deals with automatic identification of these scripts.

Among earlier studies on script identification, Spitz and co-workers [1, 2] used the spatial relationship of structural features for distinguishing between Han and Latin-based scripts. Asian scripts (Japanese, Korean and Chinese) were differentiated from the Roman ones by uniform vertical distribution of upward concavities. Then optical density is employed to distinguish among the Asian scripts. Hochberg *et al* [3] used cluster-based templates for thirteen scripts including Devanagari. Tan [4] suggested a method for six different scripts using Gabor filter output. For successful identification, this method requires image blocks containing text of single script in single font. Lee et al [5] attempted script identification of text in complex, un-oriented and degraded document images.

Among Indian scripts, Pal and Chaudhuri [6] proposed a method for Roman, Devanagari and Bangla based on a decision tree for recognizing the script of a line of text. This work was extended in [7] to other Indian scripts triplets. They used headline and structural properties like distribution of ascenders and descenders, position of vertical line in a text block, and the number of horizontal runs. Chaudhury and Sheth [8] used horizontal projection profile, Gabor transform and the aspect ratio of connected components for Roman, and three Indian scripts. Also, Pati and Ramakrishnan [9] have used Gabor transform and DCT features to separate bi-script, tri-scripts among eleven Indian scripts. A more elaborate review on script identification is given in [10].

However, all these studies presume that the document images are properly oriented and not skewed. But for quick and careless feeding in the scanner, 180° document orientation (ie. upside down) may result. For a square shaped page, orientation near ±90° is also likely. Lu and Tan [11] considered simultaneous orientation detection and script identification of Arabic, Chinese, Korean and Roman document images using Vertical Component Run (VCR) features through centroid of the component. Other studies on orientation detection are reported in [12, 16].

The approach in [11] appeared unsuitable for application in our problem, as discussed later. Our approach consists of three stages. The first stage contains preprocessing including skew correction as well as text line identification (for line-wise feature computation). The skew correction working on ±90° skew brings the image ideally into 0° or 180° orientation. Classification to one of the eleven scripts is done in the second stage using (0°/180°) orientation-invariant features. In the third phase, orientation detection and correction is done by one of the two class classifiers on the resultant script. This approach is more modular than [11].

Among rest of the paper, Section 2 briefly describes the characteristics of Indian scripts. The preprocessing steps of the document are proposed in Section 3. Section 4 considers the feature extraction while classification approaches for both stages are elaborated in Section 5. The experimental results and concluding remarks are presented in Section 6 and Section 7, respectively.

II. A GLIMPSE AT INDIAN SCRIPTS

Eleven scripts namely Bangla, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Oriya, Tamil, Telugu, Urdu and English are used for official works in Indian languages. Typical text lines in these scripts are shown in Fig. 1.

All these scripts excepting Urdu and English are descendants of ancient Brahmi which prevailed in India certainly around 500 BC. By 500 AD Brahmi branched into north and south Indian versions. The northern version

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.67

culminated in Bangla, Devanagari, Punjabi and Gujarati scripts while the southern version resulted in Telugu, Tamil, Malayalam and Kannada. Oriya script is a northern version with notable influence of the south. These scripts are Alpha-syllabary in nature, having vowel modifiers and compound characters (Gurmukhi and Tamil do not contain compounds).

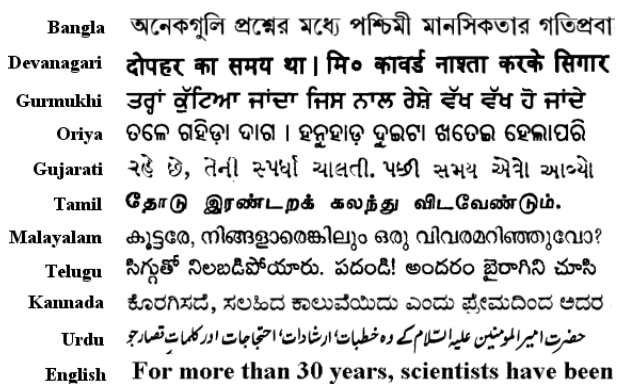

Fig 1. Typical Text lines of eleven scripts considered in the approach

Among the north Indian scripts, a fair amount of shape similarity is observed. As shown in Fig. 1, Devanagari and Gurmukhi (or Punjabi) having headline and vertical line strokes look quite similar. The Gurmukhi alphabet devised during 16th century has only 70 symbols. For scripts other than Gurmukhi, Tamil and Urdu, total number of shapes to be recognized in Indian scripts may vary from 500 to 2000.

Urdu alphabet, derived from Arabic, have distinct word composition rules. Unlike others, it is a highly cursive script. Urdu characters may take different shapes depending on word-initial, word-medial and word-final position.

English, belonging to Roman family too has different characteristics. For OCR the number of English symbols to be recognized is nearly as small as that of Gurmukhi.

III. Preprocessing Stage

The documents were initially digitized by a flatbed scanner in adequate (200/300/400 dpi) resolution. During scanning, various degrees of skew and orientation were randomly made to simulate the situation. Both fair and old document pages were scanned.

The digital images captured in gray tone were binarized by Otsu or Souvola algorithms depending on the image quality. Otsu method was used for freshly printed documents of good quality. For older document pages having local intensity variations, Sauvola approach has been used. More complex documents needed a mixture of global and local approach for proper binarization.

Sometimes binarization takes care of the noise to a good extent. The remaining noise is identified if there exist many more small-area connected components compared to elongated components having larger area (which are presumed to be text regions). Then a morphological noise removal approach is employed on the text. A guard zone around elongated components is used to save small textual symbols from removal.

A. Skew Detection and Correction

A new realization of Hough transform (HT), where instead of parametric space representation of straight line equation, a set of Digital Straight Lines (DSL) satisfying chord property due to Rosenfeld [17] are employed to simulate the HT. Advantage of DSL based HT will be described elsewhere. The angle corresponding to the highest accumulator value among the DSL Hough bins is chosen as the skew of the document. The Image is then rotated in the reverse direction to the nearest angle of multiple of 90°. Those around +90° or -90° are further rotated by 90° to make the text lines horizontal. The resulting image is now oriented either at 0° or at 180°. It is then passed to the text line detection module.

B. Individual Text Line Detection

The text line detection is essentially based on white to black transition count in horizontal scan, which will be high while scanning through a text line and low when scanning between two text lines. If two or more consecutive rows have zero transitions, we mark the top and bottom of the sequence of rows as primary text line separators. But they may also separate small disconnected components of characters e.g. diacritic marks but fail to find overlapping or touching lines. A few data driven heuristic thresholds are used to take care of both over-identification and under-identification, resulting in 99.7% accuracy.

IV. Features Extraction

Two sets of classifier are used for script identification and orientation detection, respectively. Features used for them are as follows.

A. Features for Script Identification

For script identification, features are chosen to have invariance to 0°/180° orientation and having good discriminative power.

1) Reservoir Area Difference Feature (f_{rd})

This class of features is obtained from the concept of water filling in a reservoir [18]. If there is concavity in some line drawing, when looking from say *top*, it can accumulate 2-D equivalent of water dropped from above. When the concavity is filled, there will be a spill-over. The water filled *area* is a measure of the concavity or capacity of the reservoir and can be used as a feature value. In size invariant pattern classification problem, normalization of this feature is needed. For the same stroke-drawing, we can get additional feature values by considering reservoirs viewed from the bottom, left and right side as well. More generalized reservoir morphology can be developed, but for our purpose the area-based featuring is sufficient.

To detect water reservoir area feature, first we find all connected components for each text line of the document

image. Then we calculate the filled area for each connected component for top-viewed reservoir as well as for bottom-viewed reservoir. The top-viewed and bottom-viewed reservoirs for a typical Urdu text line are shown in Fig. 2.

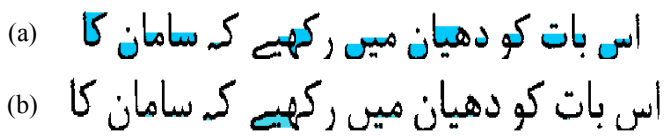

Fig.2(a)Top-viewed (b)Bottom-viewed reservoir for a typical Urdu text line

To make the reservoir feature 0°/180° orientation independent, we take absolute difference of top and bottom reservoir area. The feature is normalized by the sum of areas of these reservoirs and represented as.

$$f_{rd} = | W_T - W_B| / (W_T + W_B) \quad (1)$$

where W_T and W_B are the average per-line area in top viewed and bottom-viewed reservoirs, respectively. The normalization converts the feature value into pure number, insensitive to scaling of the image.

2) White Hole area feature (f_{wh})

Another feature for script identification which is also invariant to 0˚/180˚ orientation is white hole area. In Fig. 3 we show this hole area in gray shade. To make this feature size insensitive also, we normalized this total white hole area by dividing total area of text line boundary boxes of the image. According to our computation on training set, the normalized hole area is smallest in Gujarati followed by Urdu script. On the other hand, English script has the highest normalized hole area, closely followed by Telugu script.

Fig. 3. Hole area for eleven scripts considered in this experiment

3) Horizontal white-black transition per component (f_{thi})

In this case, at first the individual connected components of each text line are identified. Then horizontal scanning is conducted on these text lines and the numbers of white-black transition are counted on individual connected components. Now, we consider six bins for k transitions per component where k = 1, 2, 3, 4, 5 and ≥6. Initially, the counters of all bins are empty. When a scan line encounters a component in a text line, the number of white-black transitions is counted. Suppose it is 2. Then the counter in the bin corresponding to k = 2 will be incremented by one. In this way, the counters are incremented for all horizontal scans on the document. Let the counter values on training data for k =1, 2, 3, 4, 5 and ≥6 be n_1, n_2, n_3, n_4, n_5 and n_6, respectively. Then we get six normalized white-black transition feature set given by

$$f_{thi} = n_i / \sum n_i \; ; \; i = 1..6 \quad (2)$$

The members of the feature set show relative abundance of transitions per connected component in the image. For training samples of English, typical value of f_{th1} and f_{th2} are about 0.5 and 0.4, respectively but in case of Malayalam script f_{th1} and f_{th2} are 0.4 and 0.2, respectively.

4) Vertical white-black transition features (ftvi)

Here we consider scanning the text lines of the image in a vertical direction and count the number of white to black transitions. These vertical black-white transitions are accumulated in six bins corresponding to total number of columns having one, two, three, four, five and six or more transitions. The bins are filled by accumulating the number of transitions from text lines of the whole page. Note that unlike f_{thi} here we do not use individual connected components.

Let m_1, m_2, m_3, m_4, m_5 and m_6 be the entry in the six bins. The components are normalized by the total number of transitions to get the feature components.

$$f_{tvi} = m_i / \sum m_i \; ; \; i = 1, 2, ...6 \quad (3)$$

All these features are 0˚/180˚ orientation invariant.

B. Feature for Orientation Detection

For orientation detection in one earlier experiment we partitioned the text lines in upper and lower half to compute horizontal and vertical crossing counts, border following direction (horizontal, vertical, left and right) features etc in these two halves. The difference of such values between upper and lower half in 0˚ and 180˚ orientation were used to identify the orientation.

However, later on we have found that four water reservoir based features are enough to identify the orientation for all scripts more robustly. They are top, bottom, left and right reservoir area of connected components in the text, normalized with respect to component boundary box area and averaged over all components, called f_{rt}, f_{rb}, f_{rl} and f_{rr}, respectively.

Note that for poor documents there may be cuts in the text strokes where reservoir principle may not work. This is because water will leak through the hole and the reservoir can never be filled. In order to take care of this instead of water we consider using rectangular pebbles having say R pixel sidelength for plugging the hole.

V. Classification Strategy

A. Script identification

A tree classifier has been used for script identification. The tree classifier is constructed as follows. Initially we tried to classify all eleven scripts by a single classifier and analyzed the result. We noted that Bangla, Devanagari and Gurmkhi scripts confuse with one another. Similarly, English, Malayalam and Tamil were confused. Kannada and Telugu also misclassified into each other. So, we pooled them into three groups. These three groups as well as Gujarati, Oriya and Urdu created first layer of the

classification tree. Thus in Fig. 4 the classifier C_1 partitioned the document into six groups. By looking at the output of C_1 the second layers of the tree is formed, so that optimum classification accuracy is obtained. Thus, at the second level, the classifier C_2 separates Bangla from Devanagari and Gurmukhi, the classifier C_3 separates English, Malayalam and Tamil and another classifier C_4 distinguishes Kannada from Telugu. Finally, the third level classifier C_5 makes single-class decision between Devanagari and Gurmukhi.

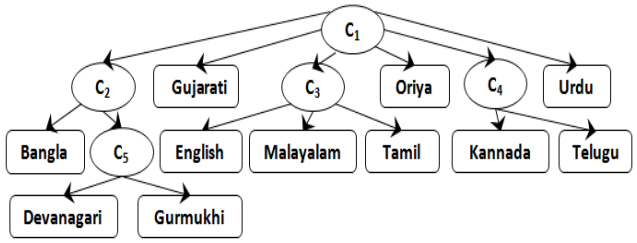

Fig. 4. Classification tree for script identification

Separate sets of features are employed for classifiers at different levels. They have been chosen on the basis of system performance on training data. At the first level, i.e. classifier C_1, the features are reservoir area difference (f_{rd}), hole area (f_{wh}) and six components of vertical white-black transitions f_{tvi}; i = 1,..6. Then in the second level, features for classifier C_2 are only f_{tvi}; i = 1,..6. For classifier C_3 the features are f_{wh} and f_{tvi}; i = 1,..6 and for classifier C_4 we use f_{tvi}; i = 1,..6. Finally, features for C_5 are only horizontal white-black transitions f_{thi}; i = 1,..6.

We compared the performance of three types of classifiers over the tree structure. They are minimum distance classifier, K-nearest neighbor (K-NN) classifier and Support Vector Machine (SVM) classifier [19]. The simplest is minimum distance classifier which is based on single representative feature vector per class. This vector is the average over feature vectors of all training samples for the class. For a given test pattern, its distance is computed from representative vector of all classes. The pattern is assigned to the class for which the distance is minimum.

The K-NN works on the basis of some good representative data in the feature space of each class. For a new test pattern, its K-nearest neighbors among these representatives are identified. The test pattern is assigned to the class from which majority of these representatives come.

More sophisticated approach is the Support Vector Machine (SVM), which is based on machine learning from support vectors derived from the training samples. It is a powerful technique that was originally proposed for two-class problem. The technique aims at generating a multi-dimensional hyperplane which maximizes the margin between two classes. The hyperplane is characterized by the normal vector expressed as linear combination of the neighboring examples of the two classes, called Support Vectors. For the generation of hyperplane, sometimes mapping to a higher dimensional space has to be done by using a Kernel function. Three types of kernel namely linear, polynomial and RBF kernels are commonly used.

These three types of classifiers are tested on the data and comparative results are presented in Section 6.

B. Orientation detection

For each script, a two-class classifiers is used for orientation detection. We tested with the three types of classifiers stated above. The four normalized water reservoir based features f_{rt}, f_{rb}, f_{rl} and f_{rr} are used in the classifiers for all scripts. Since SVM classifier performed uniformly better, results obtained by SVM is only presented in Section 6.

VI. EXPERIMENTAL RESULTS

We tested three different value of K for K-NN classifier. Also our SVM classifiers were run with Linear, Polynomial, and RBF kernels using OpenCV library. For each script, randomly chosen 30 document images are used for training and 250 images are considered for testing the accuracy of the approach proposed in this paper. For our experiment we considered only document containing single column text. More complex layout images is not considered in our experiments. Each document image contains 15-40 text lines with approximate 1-15 words per line and texts are printed in different fonts and style at 200/300/400 dpi. We observed that best result is obtained by SVM classifiers with RBF kernel for all but Malayalam script. For Malayalam, we obtained better result with SVM using Polynomial kernel. The result details are given in Table I.

TABLE I. SCRIPT IDENTIFICATION ACCURACY IN PERCENT

Script	Classifier						
	Min. Dist.	K-NN			SVM		
		K=7	*K=9*	*K=11*	*Linear*	*Poly*	*RBF*
Bangla	97.32	97.76	98.32	98.89	98.32	98.88	99.44
Devanagari	97.42	96.48	97.26	97.66	98.49	98.49	98.49
English	98.12	99.06	99.53	99.53	97.61	98.10	99.53
Gujarati	100.0	99.50	99.50	99.50	99.50	99.50	100.0
Gurmukhi	91.52	92.10	92.16	92.57	94.16	97.40	98.70
Kannada	90.20	90.55	91.35	92.51	98.99	98.99	99.33
Malayalam	97.83	98.33	98.50	98.50	99.01	100.0	99.50
Oriya	89.72	94.68	96.62	97.10	94.95	99.08	100.0
Tamil	91.20	91.04	92.59	93.21	97.99	98.04	98.37
Telugu	85.83	86.60	89.72	90.44	98.45	95.74	99.22
Urdu	100.00	100.00	100.00	100.00	100.00	100.00	100.00

For orientation detection, we obtained 100% accuracy on full page text for all considered scripts. So, we have studied the sensitivity and robustness of our method on regions smaller than a page. The minimum size considered was one line of text. The experiment was conducted on three hundred instances and the average result was taken for each script. The results are shown in Table II. Here once a script has attained 100% accuracy we did not go for more number of lines, since the results will obviously be 100%.

From Table II it is noted that the proposed approach works excellent for Bangla, Oriya, Urdu and reasonably well for Devanagari, Gurmukhi, Malayalam and Telugu. The others need more data (between 7-9 lines). This is with respect to the water reservoir based features. We are trying to find an approach where less data is needed for both script type and orientation detection.

TABLE II. ORIENTATION DETECTION ACCURACY IN PERCENT

Script	% of Accuracy for minimum document size (in no. of lines)								
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Bangla	100.00								
Oriya	100.00								
Urdu	100.00								
Devanagari	99.90	100.00							
Gurmukhi	99.31	100.00							
Malayalam	98.20	99.84	100.00						
Telugu	97.75	99.71	100.00						
Tamil	88.84	94.07	96.50	98.38	99.33	100.00			
Kannada	87.71	93.69	95.51	96.34	97.37	99.25	100.00		
English	85.90	89.63	95.83	96.90	97.79	99.53	100.00		
Gujarati	73.69	82.95	86.64	89.13	92.11	94.66	97.63	99.27	100.00

VII. CONCLUSION

A simple approach has been proposed for simultaneous script and orientation detection of all official Indian script document images. The basic idea here is to find a set of features which are invariant to 0°/180° orientation and yet powerful enough to discriminate the scripts. Once the script is identified, a proper feature set may be used to drive orientation detection classifiers for all individual scripts.

As stated, Lu and Tan [11] also considered simultaneous text categorization and orientation detection. However, this method did not work well for our purpose. One reason is that characters in Bangla, Devanagari, Gurmukhi and Urdu words are connected. So, we get less evidence through a centroid VCR of such connected components. Also, three-zone division cannot be robust for Urdu. The upper and lower zone in other scripts are also scaresly populated, often not giving discriminating evidence about orientation. Finally, Devanagari and Gurmukhi and Bangla have extremely similar overall shape vis a vis VCR features. These factors prompted us to devise different classification strategy described here. We believe that our approach will be effective for other scripts like Arabic, Farsi and Kashmiri (having identical structure as Urdu), as well as Tibetan, Sinhala and some south Asian scripts, that are identical to the Indian scripts.

The water reservoir feature has shown good potential in this problem. As stated before, small leakage in reservoir due to cut in stroke lines can be plugged by a 2D pebble. More involved incremental and embedded features can be developed using the reservoir concept. These and other mathematical expositions on reservoir morphology will be presented in a more theoretically oriented paper.

ACKNOWLEDGMENT

Partial support by DIT, Govt. of India in the form of a sponsored project is gratefully acknowledged.

REFERENCES

[1] A. L. Spitz and P. Sibun, “Natural language processing from scanned document images,” Proc. Applied Natural Language Processing, Stuttgart, 1994, pp. 115–121.

[2] A. L. Spitz, “Determination of The Script and Language Content of Document Images” IEEE Transactions on PAMI, vol 19, March 1997, pp. 235-245.

[3] J. Hochberg, P. Kelly, T. Thomas and L. Kerns, “Automatic script identification from images using clusterbased templates,” IEEE Trans. Pattern Anal. Machine Intell. vol. 19(2), February 1997, pp. 176–181.

[4] T. N. Tan, “Rotation Invariant Texture Features and Their Use in Automatic Script Identtification” IEEE Transactions on PAMI, vol. 20, 1998, pp. 751-756.

[5] D. Lee, C. R. Nohl. and H. S. Baird, "Language Identification In Complex, Unoriented, And Degraded Document Images," Proc. Second IAPR Workshop on Document Analysis Systems (DAS), 1996, pp. 17-39.

[6] U. Pal, and B. B. Chaudhuri, “Identification of different script lines from multi-script documents,” J. Image and Vision Computing. Vol. 20, 2002, pp. 945-954.

[7] U. Pal, and B. B. Chaudhuri, “Indian script character recognition: a survey,” J. Pattern Recognition, vol. 37, 2004, pp. 1887-1899.

[8] S. Chaudhury and R. Sheth, “Trainable script identification strategies for Indian languages,” Proc. Int. Conf. on Document Analysis and Recognition, (IEEE Comput. Soc. Press, 1999, pp. 657–660.

[9] P. B. Pati and A. G. Ramkrishnan, “Word Level multi-script identification,” Pattern Recognition Letters vol. 29, 2008, pp. 1218–1229.

[10] D. Ghosh, T. Dube and A. P. Shivaprasad, “Script Recognition-A Review,” IEEE Transactions on PAMI, vol 32(12), 2010, pp 2142-2161.

[11] S. Lu and C. L. Tan, “Automatic document orientation detection and categorization through document vectorization,” Proc. of the 14th annual ACM international conference on Multimedia, 2006, pp. 113-116.

[12] T. Akiyama, and N. Hagita, “Automated entry system for printed document,” J. Patteren Recognition, vol. 23(11), 1990, pp. 1141-1154.

[13] D. S Le, G. R. Thoma and H. Weschler, “Automated page orientation and skew angle detection for binary document images,” J. Pattern Recognition, vol. 27(10), Oct. 1994, pp. 1325-1344.

[14] D. Bloomberg, G. Kopec, and L. Dasari, “Measuring document image skew and orientation,” J. SPIE vol. 2422, 1995, pp. 302-316.

[15] R. S. Caprari, “Algorithm for text page up/down orientation determination,” J. Pattern Recognition Letters. vol. 21(4), 2000, pp. 311-317.

[16] B. T. Avila, and R. D. Lins, “A fast orientation and skew detection algorithm for monochromatic document images,” Proc. of the 2005 ACM symposium on Document engineering, 2005, pp. 118-126.

[17] A. Rosenfeld, “Digital straight line segment,” IEEE Trans. Computers, vol. 23(12), 1974, pp. 1264-1268.

[18] U. Pal, S. Sinha and B. B. Chaudhuri, “Multi-Script Line identification from Indian Documents,” Proc. Seventh International Conference on Document Analysis and Recognition (ICDAR) Vol 2, 2003, pp. 880-884.

[19] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” Data Mining and Knowledge Discovery, vol. 2(2), 1998, pp. 955-974.