Automatic Detection of Document Script and Orientation

Shijian Lu, Chew Lim Tan
Department of Computer Science, School of Computing
National University of Singapore, Kent Ridge, 117543, Singapore
{lusj, tancl@comp.nus.edu.sg}

Abstract

This paper presents an identification technique that automatically detects the underlying script and orientation of scanned document images. In the proposed technique, document script and orientation are identified by using the stroke density and distribution, which convert each document image into a document vector. For each script at each orientation, a number of reference document vectors are first constructed. Script and orientation of the query document are then determined according to the similarity between the query document vector and multiple preconstructed reference document vectors by using the K-nearest neighbor algorithm. Experiments show that the proposed technique is tolerant to the document skew and able to detect orientations of documents of different scripts.

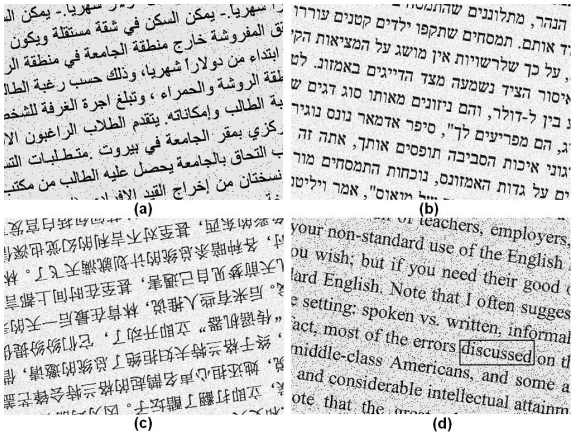

Figure 1. Document of different scripts and scanned at different orientations

1. Introduction

With the proliferation of digital libraries, an increasing number of documents of different scripts shown in Figure 1 are being produced. The knowledge of the underlying script will facilitate the management of these produced document images. Take optical character recognition (OCR) as an example. Though most OCR systems are equipped with multiple OCR engines that are capable of recognizing text of different scripts, manual routing is still required to switch incoming documents to the proper OCR engine. On the other hand, documents are sometimes captured upside down illustrated in Figure 1 due to the misalignment. The knowledge of the document orientation is also required before some document processing tasks such as layout analysis, OCR, and document retrieval.

Some works have been reported to detect the underlying script of scanned document images. The reported works can be classified into three categories including statistics based [5, 9, 11, 12], token based [7], and texture based [3, 13]. The statistics based works determine scripts through the analysis of the distribution of the upward concavity [9, 12] and horizontal projection profile [11]. Texture based works instead identify scripts based on the document texture measured by using the rotation invariant Gabor filter [3, 13]. In addition, specific character tokens have also been utilized in [7] for the script identification.

Some works have also been reported to detect the document orientation. Among the reported works, most [2, 4, 6] assume that document text is printed in Roman script. At the same time, document orientation is commonly detected based on the statistics of character ascenders and descenders where the number of character ascenders is statistically much larger than that of character descenders. To the best of our knowledge, the only exception is the work reported in [1] where character openness is utilized to detect the orientation of Roman document images and document images of some other scripts such as Hebrew and Pashto.

Though some document script and orientation detection works have been reported, two limitations still exist. Firstly, most reported script and orientation detection works [1, 2, 5, 9, 11, 12] commonly assume that document images

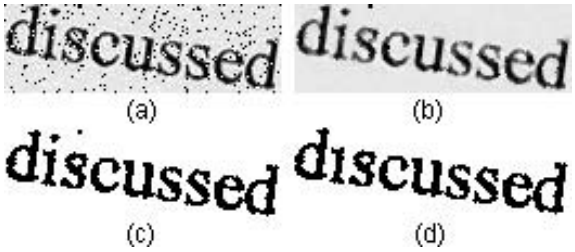

Figure 2. (a) Word image cropped from the document in Figure 1(d); (b) word image after the median filtering; (c) binarized word image; (d) word image after the image filtering.

are free of the document skew. Therefore, deskew is normally required because document skew is almost inevitable. Secondly, most reported orientation detection work [2, 4, 6] assumes that document text is printed in Roman and so cannot work on non-Roman scripts properly.

In this paper, we present a script and orientation identification technique that is tolerant to the document skew and able to detect orientation of documents of different scripts. We detect document script and orientation by using the stroke density and distribution, which convert each document image into a document vector that encodes the underlying script and orientation information. For document of each script at each orientation (upright or upsidedown), we create a number of reference document vectors and then determine the script and orientation of the query document by using the K-nearest neighbor algorithm.

2. Proposed Methods

This section presents the proposed identification technique. In particular, we will divide this section into four subsections, which deal with document image preprocessing, encoding of stroke density and distribution, script and orientation detection, and discussions, respectively.

2.1 Document Image Preprocessing

Document images need to be first preprocessed. In the proposed technique, noise of small size such as salt & pepper noise shown in Figure 1 is first suppressed by a median filter. Filtered document images are then binarized and labeled through the connected component analysis. Finally, noise of big size and text symbols of small size are removed through an image filtering process.

Noise of small size is first suppressed by using a median filter. As text images are rich in edges, the standard median filter may smear character strokes severely. We therefore utilize a center weighted median filter [8] given below, which "pulls" the median value towards the intensity of the center pixel and so keeps stroke shapes much better:

$$y(i,j) = M\{x(i-s, j-t), 2k\, x(i,j) \mid (s,t) \in W\} \quad (1)$$

where W refers to a 3×3 window and parameter k is set at 1 in case character strokes are blurred by the filter. $x(i,j)$ and $y(i,j)$ denote the intensity of the original and filtered image pixel at (i,j). For the word image in Figure 2(a) cropped from the document images in Figure 1(d), Figure 2(b) shows the filtered word image.

Binarization is then implemented to convert the filtered document image into a binary one. A large number of document binarization techniques have been reported and we directly adopt Otsu's global thresholding technique [10]. For the filtered word image in Figure 2(b), Figure 2(c) shows the resultant binary word image. Binary text images are then labeled through the connected component analysis and information including component size, component centroid, and component pixel list is determined and stored for the subsequent script and orientation detection.

Text symbols of small size such as the top part of characters "i" and "j" and noise of bigger size are then removed through an image filtering process. The filtering threshold is set as the size of the k^{th} labeled connected component such that the inter-class variance $\sigma_s(k)$ of component size histogram reaches the maximum:

$$\sigma_s(k) = \frac{\left[\varphi(k) \cdot \sum_{i=1}^{S} i \cdot p(i) - \sum_{i=1}^{k} i \cdot p(i)\right]^2}{\varphi(k) \cdot \big(1 - \varphi(k)\big)} \quad (2)$$

where S refers to the maximum size of the labeled connected components, $p(i)$ gives the normalized density of component size histogram and $\varphi(k)$ gives the zeroth-order cumulative moment of the histogram as follows:

$$p(i) = n(i)/N_t \qquad \varphi(k) = \sum_{i=1}^{k} p(i) \quad (3)$$

where N_t denotes the number of the labeled connected components and $n(i)$ gives the number of the connected components of size i. For the word images in Figure 2(c), Figure 2(d) shows the filtering results where text symbols of small size (top part of character "i") as well as noise of bigger size have been correctly removed.

2.2 Stroke Density and Distribution Encoding

We capture document script and orientation information by using the density and distribution of character stokes, which is characterized by the number and position of vertical component cuts illustrated in Figure 3. Scanning from

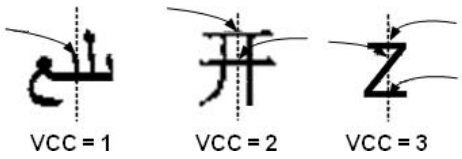

Figure 3. Vertical component cut illustration.

top to bottom, a vertical component cut is detected when a vertical scan line passing through the component centroid enters the component region from the background. We define the cut position as the position of the topmost stroke pixel that intersects the vertical scan line. Thus, in Figure 1, the three characters from left to right (i.e. Arabic, Chinese, English) have 1, 2, and 3 vertical component cuts, respectively, as indicated by the arrows.

In the proposed method, stroke density and distribution of a connected component are encoded by using a component vector of dimension ten as follows:

$$CV = [CV_1 \ \cdots CV_{10}] \tag{4}$$

where the first eight elements $CV_1 \cdots CV_8$ record the number of vertical component cuts. In the proposed method, we assume that the number of vertical cuts of each connected component is not more than 8 because most characters of scripts under study are found to have not more than 8 vertical cuts. $CV_1 \cdots CV_8$ are therefore defined as follows:

$$CV_i = \begin{cases} 1 & \text{if the character has i vertical cuts} \\ 0 & \text{Otherwise} \end{cases} \tag{5}$$

The last two component vector elements CV_9 and CV_{10} instead record and the number of vertical component cuts lying above and below the corresponding component centroid, respectively. Clearly, the ten elements of the component vector satisfy the following constraints:

$$CV_9 + CV_{10} = \sum_{i=1}^{8} CV_i \times i \tag{6}$$

Based on the encoding technique described above, a document image can be converted into a document vector of dimension ten through summing up of the corresponding component vectors as follows:

$$DV = \sum_{j=1}^{N_f} CV^j \tag{7}$$

where N_f refers to the number of connected component after the preprocessing. CV^j denotes the component vector of the j^{th} labeled connected component.

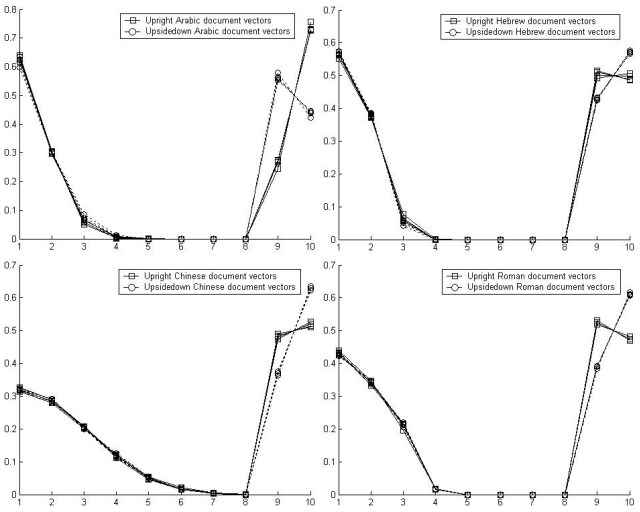

Figure 4. Four document vectors of each of the four scripts under study at each of two orientations are plotted.

Suppose a query document image contains just a single word "no" and the two characters "n" and "o" are correctly segmented. The component vectors of characters "n" and "o" can be determined as [1 0 0 0 0 0 0 0 1 0] and [0 1 0 0 0 0 0 0 1 1] according to the definition of the proposed component vector in Equations (4-5). The corresponding query document vector can therefore be determined as [1 1 0 0 0 0 0 0 2 1], which corresponds to the sum of the component vectors of characters "n" and "o" .

Before the ensuing document script and orientation detection, the converted document vector must be normalized as follows to remove the document length effect:

$$\overline{DV} = \frac{DV}{DV_9 + DV_{10}} \tag{8}$$

where DV_9 and DV_{10} refer to the last two elements of the converted document vector.

The converted document vector encodes the stoke density and distribution and can be used for script and orientation detection. For each of the four scripts under study (Arabic, Chinese, Roman, and Hebrew), Figure 4 shows eight normalized document vectors that are converted from four documents scanned at upright and upside down orientations, respectively. As Figure 4 shows, the first eight elements nearly coincide with each other for document vectors of the same script, but they are far different for document vectors of different scripts. Similarly, the last two elements are close for document vectors of the same script at the same orientation, but they are far different for document vectors of the same script but at opposite orientations.

2.3 Script and Orientation Detection

Script and orientation of document images can therefore be detected based on the document vector similarity illustrated in Figure 4. In particular, for each script under study, a number of reference documents are first created where text is printed in different fonts and styles. Each reference document then produces two reference document images, which are scanned at the upright and upsidedown orientations and the skew angle is controlled between -5 degrees and 5 degrees. Each produced reference document image is then converted into a reference document vector as described in the last subsection. Finally, the script and orientation of a query document can be determined according to the similarity between the query document vector and the pre-constructed reference document vectors by using the K-nearest neighbor algorithm.

In the proposed method, we prepare 80 reference document vectors for each script under study, which are converted from 40 reference documents when they are scanned at upright and upsidedown orientations, respectively. The similarity between the query document vector QDV and A reference document vector RDV is evaluated by using the cosine similarity as follows:

$$sim(QDV, RDV) = \frac{\sum_{i=1}^{N_v} DV_i \cdot RDV_i}{\sqrt{\sum_{i=1}^{N_v}(QDV_i)^2 + \sum_{i=1}^{N_v}(RDV_i)^2}} \quad (9)$$

where QDV_i and RDV_i denote the i^{th} element of DV and RDV. The script and orientation of the query document can therefore be detected based on the majority of K (10 in our system) nearest neighbor categories. In case of a tie, the script and orientation of the query document is assigned to be the same as that of the closest reference document vector.

2.4 Discussions

One distinctive characteristic of the proposed script and orientation detection technique is that it is tolerant to the document skew. Take the Roman script as an example. The vertical cut number of most Roman letters such as "e", "o", and "v" (with 3, 2, and 1 vertical cuts) will remain unchanged when the skew angle lies within a reasonable range. Besides, the relative position between vertical cuts and the corresponding component centroids remains unchanged when the skew angle lies within a reasonable range. Figure 5 shows the cosine similarity between document vectors of a document image when it is skewed by multiple different angles. As Figure 5 shows, the variation in document vectors is nearly negligible (small than 0.03) when skew angle lies between -5 degrees and 5 degrees.

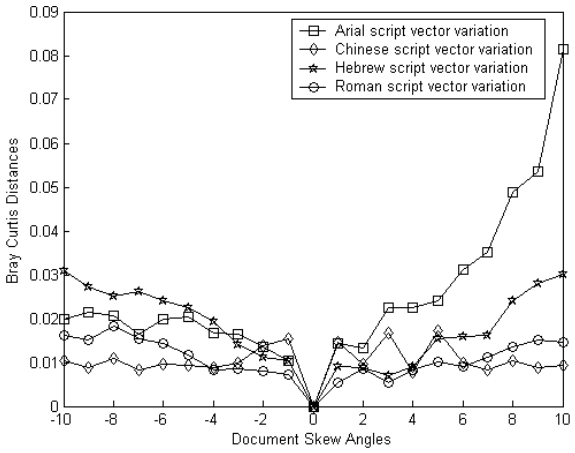

Figure 5. Cosine similarity between document vectors of different skew.

This ensures that script and orientation of most documents with mild skew can be correctly detected.

3. Experimental Results

This section presents the experimental results. Firstly, 80 test documents are collected where every 20 are printed in one of the four scripts under study. In particular, each test document contains at least 15 text lines and document text is typed in different fonts and styles. Each of the 80 test documents produces two document images, which are scanned at the two opposite orientations and skew angle is controlled within -5 degrees and 5 degrees as well. Script and orientation are then detected as described in Section 2.3. Experiments show that script identity of 160 test images is all correctly detected. Besides, the orientation identification rate reaches up to 95.63%

Table 1 shows the average cosine similarity between every 20 test document vectors of each script at one specific orientation (either upright or upsidedown) and the respective document vector templates estimated as the mean of the reference document vectors (described in Section 2.3) of the same script at the same orientation. As Table 1 shows, the distances between document vectors and the document vector template of the same script are normally smaller than those between document vectors and document vector templates of the different scripts. Furthermore, the distances between document vectors and document vector template of the same script at the same orientation are much smaller than those between document vectors and document vector templates of the same scripts but at the opposite orientation.

The proposed technique is sensitive to the number of characters within document images. As the character num-

Table 1. Average cosine similarity between every 20 document vectors and eight document vector templates (A, C, H, R: Arabic, Chinese, Hebrew, and Roman; U, D: upright and upsidedown orientation; T: template; DV: document vector).

	AU_DV	AD_DV	CU_DV	CD_DV	HU_DV	HD_DV	RU_DV	RD_DV
AU_T	0.0322	0.1716	0.2565	0.2021	0.1521	0.1166	0.2282	0.1677
AD_T	0.1734	0.0387	0.1964	0.2473	0.0565	0.1028	0.1158	0.1793
CU_T	0.2888	0.2190	0.0348	0.0859	0.1950	0.2055	0.1346	0.1375
CD_T	0.2314	0.2719	0.0846	0.0324	0.2484	0.2057	0.1899	0.1047
HU_T	0.1647	0.0751	0.1680	0.2213	0.0276	0.0529	0.0767	0.1353
HD_T	0.1307	0.1158	0.1871	0.1908	0.0614	0.0121	0.1171	0.1042
RU_T	0.2324	0.1168	0.1015	0.1546	0.1078	0.1435	0.0330	0.0733
RD_T	0.1681	0.1840	0.1211	0.0873	0.1589	0.1150	0.0962	0.0092

ber becomes too small, the converted document vectors may not reflect the stroke density and distribution of the underlying script. We test a set of text line images cropped from the 160 document images described above. Experiments show that scripts and orientations can be properly detected in most cases when document images contain six or more text lines. script and orientation detection of short documents need to be further studied. Besides, the extension to other scripts will be investigated in our future work as well.

4. Conclusion

This paper presents an identification technique that detects script and orientation of scanned document images. Based on the observation that documents of the same script at the same orientation have similar stroke density and distribution, the proposed technique converts each document image into a document vector and detects script and orientation according to the document vector similarity. Experiments show that the proposed technique is tolerant to the document skew and able to detect the orientation of document images of different scripts.

5 Acknowledgement

This research is supported by the Agency for Science, Technology and Research (A*STAR), Singapore, under grant no. 0421010085.

References

[1] H. B. Aradhye. A generic method for determining up/down orientation of text in roman and non-roman scripts. *Pattern Recognition Letters*, 38(11):2114 2131, 2005.

[2] D. Bloomberg, G. Kopec, and L. Dasari. Measuring document image skew and orientation. *SPIE 2422*, pages 302–316, 1995.

[3] A. Busch, W. W. Boles, and S. Sridharan. Texture for script identification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(11):1720–1732, 2005.

[4] R. S. Caprari. Algorithm for text page up/down orientation determination. *Pattern Recognition Letters*, 21(4):311–317, 2000.

[5] J. Ding, L. Lam, and C. Y. Suen. Classification of oriental and european scripts by using characteristic features. *International Conference on Document Analysis and Recognition*, pages 1023–1027, 1997.

[6] B. T. Ávila and R. D. Lins. A fast orientation and skew detection algorithm for monochromatic document images. *ACM symposium on Document engineering*, pages 118–126, 2005.

[7] J. Hochberg, L. Kerns, P. Kelly, and T. Thomas. Automatic script identification from images using cluster-based templates. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(2):176–181, 1997.

[8] S. J. Ko and Y. H. Lee. Center weighted median filters and their applications to image enhancement. *IEEE Transactions on Circuits and Systems*, 38(9):984993, 1991.

[9] D. S. Lee, C. R. Nohl, and H. S. Baird. Language identification in complex, unoriented, and degraded document images. *International Workshop on Document Analysis Systems*, pages 76–88, 1996.

[10] N. Otsu. A threshold selection method from graylevel histogram. *IEEE Transactions on System, Man, Cybernetics*, 19(1):62–66, 1978.

[11] U. Pal and B. B. Chaudhury. Identification of different script lines from multi-script documents. *Image and Vision Computing*, 20(13-14):945–954, January 2002.

[12] A. L. Spitz. Determination of script and language content of document images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 19(3):235–245, 1997.

[13] T. N. Tan. Rotation invariant texture features and their use in automatic script identification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(7):751–756, 1998.