Script and Language Identification in Degraded and Distorted Document Images

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

Abstract

This paper reports a statistical identification technique that differentiates scripts and languages in degraded and distorted document images. We identify scripts and languages through document vectorization, which transforms each document image into an electronic document vector that characterizes the shape and frequency of the contained character and word images. We first identify scripts based on the density and distribution of vertical runs between character strokes and a vertical scan line. Latin-based languages are then differentiated using a set of word shape codes constructed using horizontal word runs and character extremum points. Experimental results show that our method is tolerant to noise, document degradation, and slight document skew and attains an average identification rate over 95%.

Introduction

Script and language identification is normally the first step for multilingual optical character recognition (OCR) and multilingual information retrieval. Traditionally, script and language identification is frequently addressed in natural language processing areas where the identification is carried out based on the character-coded text (Cavnar & Trenkle 1994) or OCR results (Lee, Nohl, & Baird). N-grams, which represent *n* consecutive text symbols, are frequently exploited for script and language identification. With the promise of paper-less office and the proliferation of digital libraries, more and more degraded and distorted document images of different scripts and languages are archived. Script and language identification in these archived document images accordingly become a new challenge.

A few script identification techniques have been reported to determine scripts in scanned document images. The reported methods can be classified into three categories, namely, statistics based approaches, token based approaches, and texture based approaches. For statistics based approaches, the distribution of horizontal projection profile is frequently exploited (Ding, Lam, & Suen 1997; Pal & Chaudhury 2002). For document images printed in Roman, the horizontal projection profile of text normally holds two local peaks at the *x*-line and base line positions. But for oriental scripts, horizontal projection profile distributes more evenly from the top to the bottom of text lines. Besides, the distribution of upward concavity (Lee, Nohl, & Baird ; Spitz 1997) is also proposed for script identification.

Texture based approach instead utilizes the visual difference of document images printed in different scripts. In the work by Jain et al.(1996), Chinese and Roman are differentiated through convolving texture discrimination masks. Later, Tan (1998) captures textural features using the rotation invariant multi-channel Gabor filter. Similarly, Busch et al.(2005) determine scripts through gray-level co-occurrence matrix and wavelet analysis. Compared with the statistics based approach, texture based approach needs no text segmentation or connected component labeling. However, the identification process is normally much slower due to the time-consuming texture measurement and analysis. In addition to texture and statistical image features, text tokens (Hochberg *et al.* 1997) that are specific to different scripts have also been proposed for script identification.

A few language identification techniques have also been reported to determine languages in scanned document images. Currently, most language identification techniques focus on Latin based languages. Unlike various scripts that hold different alphabet structures (Spitz 1997; Hochberg *et al.* 1997) and texture features (Tan 1998; Busch, Boles, & Sridharan 2005; Jain & Zhong 1996), Latin based languages are all printed in the same set of Roman letters and so have similar texture features. Character sequences, which are generally organized in different patterns (words) in different languages, are widely exploited for Latin based language identification.

The reported language identification techniques normally start with a character categorization process, which classifies characters into several categories based on a number of character shape features. For example, the work in (Spitz 1997; Nobile *et al.* 1997; Suen *et al.* 1998) proposes to group characters and other text symbols into six, ten, and thirteen categories, respectively. Based on the character categorization results, word shape codes (*WSC*) are then constructed and languages are finally determined according to *WSC* frequency of a single word, word pair, and word trigram. As a common drawback, the performance of these methods depends heavily on character segmentation results. For de-

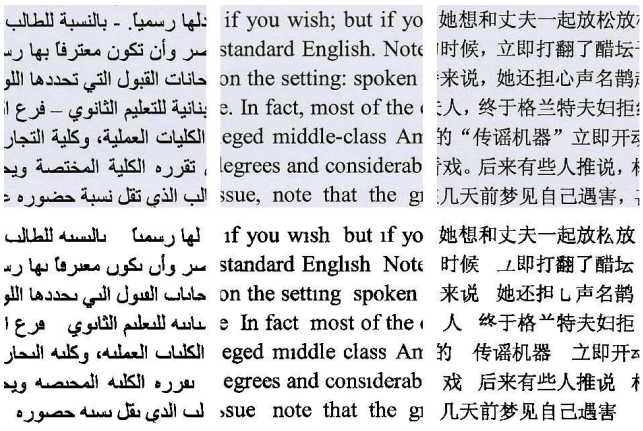

Figure 1: Three document patches in Arabic, Chinese, and English and preprocessing results.

graded documents that contain a large quantity of broken or touching characters, the constructed *WSC* are far from the real ones.

We propose to identify scripts and languages through document vectorization, which transform each document image into an electronic document vector that characterizes the shape and frequency of the contained character and word images. For each script and language studied, we first construct a vector template through a learning process. Script and language of the query image are then determined based on the distances between the query vector and multiple learned templates. In the proposed method, we vectorize document images using the number of character runs and character extremum points, which are both tolerant to noise, document degradation, and slight document skew. We first exploit the density and distribution of vertical runs to identify six frequently used scripts including Arabic, Chinese, Hebrew, Japanese, Korean, and Roman. For document images printed in Roman script, eight Latin based languages including English, German, French, Italian, Spanish, Portuguese, Swedish, and Norwegian are then differentiated based on the *WSC* that are constructed using character extremum points and numbers of horizontal runs.

Document Preprocessing

Some preprocessing operations are required before the ensuing script and language identification. Firstly, document text must be segmented from the background and we adopt Otsu's (Otsu 1979) global thresholding technique for document binarization. After document binarization, segmented document components are then labeled through connected component analysis. For each labeled document component, information including component size, centroid, and pixel list can accordingly be determined and stored for later image processing. Lastly, preprocessing is finished through two rounds of size filtering.

Noise of small size is first removed through the first round of size filtering. We set threshold at 10 because nearly all labeled character components contain much more than 10

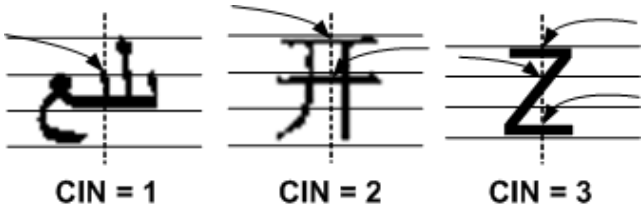

Figure 2: The number and position of vertical runs.

pixels. The second round filtering further removes noise of bigger size and small document components such as punctuations, the top parts of *i* and *j*, and character ascent and descent (such as *á* and *ü*) associated with extended Roman alphabets. Filtering threshold is determined as the size of the k^{th} document component so that the inter-class variance $\sigma_s(k)$ of document component size histogram reaches maximum:

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

where *S* refers to the maximum size of document components after the first round of size filtering. $p(i)$ gives the normalized density of component size histogram and $\varphi(k)$ gives the zeroth-order cumulative moment of the histogram. They are determined as:

$$p(i) = n(i)/N \qquad \varphi(k) = \sum_{i=1}^{k} p(i) \qquad (2)$$

where N denotes the number of document components after the first round of size filtering. $n(i)$ gives the number of document components with size equal to i.

For three document patches on the top of Figure 1, the three document patches on the bottom give filtering results. Currently, most language identification techniques depend heavily on small document components such as punctuation and character ascent and descent ($\hat{a}$ and $\ddot{u}$) for character shape coding. Unfortunately, degraded documents normally contain some quantity of noise with size similar to these small components, which results in coding inaccuracy. As our method does not require these small components, we just remove them together with noise and so produce a cleaner text image for ensuing feature detection.

Script Identification

We identify scripts based on the density and distribution of vertical character runs. For each script studied, a script template is first constructed through a learning process. The script of the query image is then determined according to the distances between the query document vector and multiple learned script templates.

Document Image Vectorization

We characterize the density and distribution of vertical character runs through document image vectorization, which

Num_1	……						Num_8	Upp_1	……						Upp_8	Mid_1	……						Mid_8	Low_1	……						Low_8
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
47	12	1	0	0	0	0	0	25	2	0	0	0	0	0	0	13	4	0	0	0	0	0	0	22	7	1	0	0	0	0	0
23	24	17	6	4	1	0	0	65	15	0	0	0	0	0	0	8	26	17	7	1	0	0	0	2	11	11	4	4	1	0	0
40	24	39	3	0	0	0	0	68	5	0	0	0	0	0	0	31	36	5	0	0	0	0	0	7	25	37	3	0	0	0	0

Figure 3: The format of the proposed *VRV* and *VRV* examples constructed using the three characters in Figure 2 and the three document patches in Figure 1.

Table 1: Standard deviations of the training *VRV* from the six constructed *VRT* (AR-RN: Arabic· · ·Roman; ART-RNT: Arabic· · ·Roman Template).

	AR	CN	HW	JN	KN	RN
ART	**.0137**	.1739	.1559	.1586	.1318	.1599
CNT	.1818	**.0083**	.1572	.0333	.0825	.0913
HBT	.1564	.1523	**.0049**	.1323	.1119	.1321
JPT	.1588	.0257	.1319	**.0110**	.0550	.0840
KRT	.1322	.0762	.1116	.0554	**.0091**	.0904
RNT	.1601	.0876	.1317	.0838	.0900	**.0123**

transforms each document image into an electronic document vector. Scanning from the top to the bottom, a vertical character run occurs as a vertical scan line passing through the character centroid enters the character region from the background. The run position is accordingly determined at the topmost pixel of a character stroke that meets the vertical scan line. For documents with skew distortion, the vertical scan line is defined as a straight line passing through character centroids with orientation orthogonal to that of the *x*-line or baseline of the related text line. The arrows in Figure 2 illustrate the numbers and positions of vertical character runs.

We propose to characterize the number and position of vertical character runs using a vertical run vector (*VRV*) of dimension 32. The first 8 elements of vertical character runs describe the density of vertical character runs. We set the upper limit of the number of vertical character runs in each scanning round at 8 as there is normally no more than 8 vertical runs for most scripts currently under study. To study the distribution of vertical character runs, we first divide text lines into three equidistant zones as illustrated in Figure 2, namely, the top zone around the top line, the middle zone around the middle line, and the base zone around the base line of text lines, respectively. The upper limit of vertical character runs in the three text zones are also set at 8 because vertical runs in each scanning round may all occur within a unique text zone. Each document can thus be transformed into a *VRV* of dimension 32 where the first 8 elements record the number of characters with vertical run number equal to their indices and the following 24 elements record the distribution of vertical runs within the top, middle, and base text zones, respectively.

The first row in Figure 3 illustrates the format of *VRV* where Num_i i = 1$\cdots\cdots$8 give the number of characters with the number of vertical runs equal to *i*. Upp_i, Mid_i, and Low_i, i = 1$\cdots\cdots$8 define the positions of vertical runs within three text zones. For the three characters given in Figure 2, the 2th-4th rows in Figure 3 give the corresponding *VRV*. For the Arabic character, there is only one vertical runs that occurs within the top zones. Therefore, only Num_1 and Upp_1 are set as 1. But for English character “Z”, there are three vertical character runs occurring within the top, middle, and base zones. As a result, Num_3, Upp_1, Mid_2, and Low_3 are set as 1 because the second and third vertical runs occur within the middle and base zones, respectively. For document images containing a large number of characters, the corresponding *VRV* can be determined as the sum of the *VRV* of all processed characters. The last three rows in Figure 3 give the *VRV* of the Arabic, English, and Chinese document patches given on the bottom of Figure 1.

Script Template Construction

For each script studied, we construct a vertical run template (*VRT*) based on the document vectorization technique described in the last subsection. We use 300 scanned document images (every 50 in one studied script) for *VRT* construction. In the proposed method, *VRT* are simply estimated as the mean of the training *VRV* of the same script::

$$VRT_i = \frac{\sum_{j=1}^{M} VRV_{i,j}}{\sum_{j=1}^{M}\sum_{k=1}^{N} VRV_{j,k}} \qquad (3)$$

where VRT_i refers to the i^{th} element of the estimated *VRT*. $VRV_{j,k}$ refers to the k^{th} element of the j^{th} *VRV* determined using the j^{th} training image. *M* and *N* are 50 and 8, referring to the 50 training images *VRV* and the first 8 elements of each training *VRV*, respectively. Therefore, the denominator item, known as a normalization factor, gives the total number of components within the 50 preprocessed training images.

The *VRV* of the training images of the same script are normally quite close to each other. We use standard deviation to study the distribution of the *VRV* of the training images around the constructed *VRT*:

$$\sigma^2 = \frac{1}{M}\frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N}(VRV_i^j - VRT^j)^2 \qquad (4)$$

where *M* and *N* are 50 and 32, referring to the 50 training *VRV* of the same script and the 32 elements of each training *VRV*, respectively. For the 300 training images studied, Table 1 gives the standard deviations of every 50 *VRV* of the same script around the six constructed *VRT*. As we can see, the standard deviation of *VRV* from the related *VRT* is much smaller than that of the *VRV* to other *VRT*.

Script Identification

As Table 1 shows, the *VRV* of documents of the same script are much closer to each other. We therefore determine scripts using the distance between the *VRV* of the query image and the six learned *VRT*.

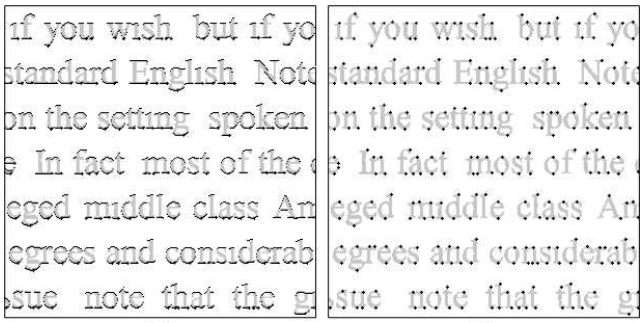

Figure 4: (a) Extracted upward and downward text boundaries; (b) detected character extremum points.

We evaluate the distances between the query *VRV* and the six constructed *VRT* using Bray Curtis distance, which has a nice property that its value always lies between 0 and 1. The distance between the query document vector and the i^{th} script template can be determined as:

$$BCD_i = \frac{\sum_{j=1}^{N}(|VRV_j - VRT_j^i|)}{\sum_{j=1}^{N}(VRV_j) + \sum_{j=1}^{N}(VRT_j^i)} \qquad (5)$$

where VRV_j represents the j^{th} element of the query *VRV*. VRT_j^i corresponds to the j^{th} element of the i^{th} learned *VRT*. As a result, query image is assigned to the script with the smallest Bray Curtis distance.

Latin based Language Identification

We identify Latin based languages using a word shape coding scheme, which transforms each document image into a document vector that characterizes the shape and frequency of the contained word images.

Word Shape Feature Extraction

We exploit two word shape features for word shape coding. The first refers to character extremum points detected from the upward and downward text boundary. The second is the number of horizontal word runs, which counts the intersections between character strokes within a word image and the middle line of the related text lines.

For each labeled document components, the upward and downward boundaries can be determined using a vertical scan line that traverses across the document component from the top to the bottom. The first and last character pixels of each scanning round corresponding to the highest and lowest pixels are determined as upward and downward boundary points. For the English document patch in Figure 1, Figure 4(a) shows the extracted upward and downward text boundary where texts are printed in light gray for highlighting.

For each labeled document component, its upward or downward boundaries actually form an arbitrary curve that can be characterized by a function *f(x)*. Character extremum points can accordingly be defined as the extrema of the function *f(x)*. For the text boundaries given in Figure 4(a), Figure 4(b) shows the detected character extremum points. It

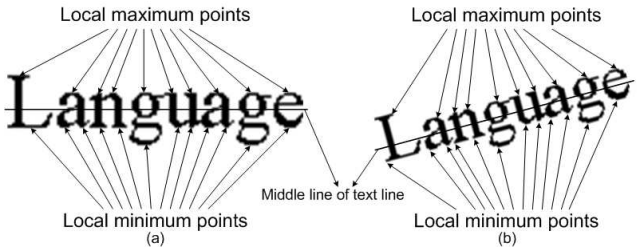

Figure 5: Character extremum points and horizontal word runs in degraded and distorted word image.

should be clarified that some single pixel concave and convex along the text boundary may affect the extremum point detection. These concave and convex with a single pixel can be removed using some morphological operators.

Character extremum point is tolerant to most segmentation errors and slight document skew. For word image "language" in Figure 5(a), characters "g", "u", and "a" are falsely connected after the binarization process. With traditional character shape coding technique (Spitz 1997; Nobile *et al.* 1997; Suen *et al.* 1998), these three characters will be treated as one and the resulting *WSC* will be totally different from the real one. But character extremum point is able to capture the word shape correctly while characters are connected as shown in Figure 5(a). Similarly, local extremum points can also be detected correctly in the presence of slight skew as illustrated in Figure 5(b).

Horizontal run number (*HRN*) refers to the number of intersections between character strokes within a word image and the related middle line of text. For example, the *HRN* of word image "the" is 4 because there are 4 runs between the character strokes and the related middle line. Similar to the character extremum point, *HRN* is also tolerant to segmentation errors and slight document skew. For the word image "language" in Figure 5, 14 horizontal runs can be correctly detected in the presence of segmentation errors and slight skew.

Word Shape Coding

We classify character extremum points into three categories based on their positions relative to the *x*-line and baseline of text. The first category pertains to the extremum points that lie far below the baseline. The second category pertains to the extremum points that lie within the region between the *x*-line and the baseline. The third category pertains to the extremum points that lie far above the *x*-line. We denote the above categories as 1, 2, and 3, respectively. For document images in Latin based languages, the *x*-lines and baselines roughly determine the position of character extremum points extracted from the text line under study.

Combined with the number of horizontal word runs, a word image can thus be transformed to a digit sequence. The first part on the left of the digit sequence is coded based on character extremum points. The following part on the right instead counts the number of intersections between character strokes and the middle line of text. Take the sample word

Table 2: *WSC* numbers learned from training images (EN-NO: English, German, French, Italian, Spanish, Portuguese, Swedish, Norwegian).

	EN	GE	FR	IT	SP	PO	SW	NO
EN	6344	1332	1586	1494	1526	1404	1064	1076
GE	1332	8280	1546	1448	1466	1400	1158	1230
FR	1586	1546	7472	1620	1646	1552	1232	1210
IT	1494	1448	1620	6381	1716	1708	1208	1194
SP	1526	1466	1646	1716	6811	1808	1278	1188
PO	1404	1400	1552	1708	1808	6144	1124	1102
SW	1064	1158	1232	1208	1278	1124	8773	1418
NO	1076	1230	1210	1194	1188	1102	1418	9147

"the" as an example. The corresponding word shape code is 3322/4 where 3322 is coded using character extremum points and the following digit 4 refers to the number of horizontal word runs, respectively.

Each document image can thus be transformed into a word shape vector (*WSV*) with each element recording a unique *WSC*. The *WSV* construction process can be summarized as follows. For each *WSC* translated from a word within the studied document image, the corresponding *WSV* is searched for the element with the same *WSC*. If such element exists, the occurrence number of the corresponding *WSC* is increased by one. Otherwise, a new *WSV* element is created where the *WSC* is set as that newly coded *WSC* and the word occurrence number is initialized as 1 instead.

A word frequency vector (*WFV*) can be constructed as well based on the occurrence numbers of the coded *WSC*. *WFV* must be normalized first to facilitate the ensuing language identification:

$$WFV_i = \frac{ON_i}{N_w} \tag{6}$$

where WFV_i gives the i^{th} element of the normalized *WFV*. N_w denotes the number of word images within the studied image and ON_i is the occurrence number of the i^{th} *WSC*.

Language Identification

We use the proposed word shape coding scheme for language identification. For each language studied, a word shape template (*WST*) and a word frequency template (*WFT*) are first constructed. Languages of query images are then determined based on the distance between the query vector and multiple learned language templates.

We construct *WST* and *WFT* through a learning process, which remember *WSC* and the related word frequency based on a set of training images. 400 document images (every 50 printed in one studied language) are prepared for *WST* and *WFT* construction. Table 2 gives the learning results where diagonal items give the numbers of *WSC* o f the eight studied languages and off-diagonal items give the numbers of *WSC* shared by two related languages. As Table 2 shows, the average *WSC* collision rate just reaches around 10%. Considering the facts that the learned *WSC* contain a large number of short frequently appeared words, the 10% collision rate is actually much higher than the real one. It may be reduced greatly after more sample images are trained and some longer *WSC* are remembered.

Table 3: The accuracy of script identification in relation to the number of lines of text processed.

No of Text Lines	No of Image Samples	Accuracy
12	67	98.51%
10	82	97.56%
8	97	95.88%
6	121	94.21%
4	168	91.07%
2	263	83.65%
1	324	76.54%

Languages can be determined based on *WSC* information alone. Treating *WSV* as a data set, languages of query images can be determined based on the number of *WSC* that are shared between the query *WSV* and the 8 learned *WST*::

$$HD_i = \frac{|WSV \cap WST_i|}{|WSV|} \tag{7}$$

where $|\cdot|$ give the size of a data set. The $|WSV|$, known as a normalization factor, removes the effect of document length difference.

Language can be alternatively determined based on similarity between the query *WFV* and multiple constructed *WFT*. We propose to evaluate such similarity using cosine measure:

$$SM_i = \frac{\sum_{j=1}^{N} WFV_j \cdot WFT_j^i}{\sqrt{\sum_{j=1}^{N}(WFV_j)^2 + \sum_{j=1}^{N}(WFT_j^i)^2}} \tag{8}$$

where WFV_j represents the j^{th} normalized word frequency within the query *WFV*. WFT_j^i is determined as follows. For each element within the query *WFV*, the i^{th} language template is searched for the element with the same *WSC*. If such element exist, the WFT_j^i is determined as the corresponding normalized word frequency. Otherwise, WFT_j^i is simply zero. As a result, query image is determined to be printed in the language with the biggest SM_i.

Experiments

Script Identification

276 testing documents are prepared to evaluate the performance of the proposed script identification method. The testing documents are collected from different sources and each contains around 30 text lines on average. All testing documents are captured using a generic document scanner where scanning resolution is set at 200 ppi to simulate the document degradation. Experimental results show that the scripts of all 276 testing images are correctly identified.

While the number of characters within query images becomes smaller, the script identification rate may deteriorate as the *CIV* of the query image cannot reflect the real density and distribution of vertical character runs. We create a set of

Table 4: Confusion Matrix for Language Identification Results (ER:Errors).

	Detected Languages								ER
	EN	GE	FR	IT	SP	PO	SW	NO	
EN	48	2							2
GE		46		1	3				4
FR			50						0
IT		1		49					1
SP		2			47	1			3
PO		2				48			2
SW							48	2	0
NO			1				1	48	2
ER		7	1	1	3	1	1	2	16

document samples as given in Table 3 to test the script identification performance in relation to character numbers. All sample images in Table 3 are cropped from the 276 scanned full page document images described above. As Table 3 shows, the identification rate is quite high as the number of text lines is bigger than 4, otherwise the identification accuracy deteriorates quickly.

Language Identification

To test our proposed language identification method, we prepare 347 testing documents printed in eight studied Latin based languages. Each document contains 1-40 text lines. Similarly, all testing images are scanned at resolution 200 ppi to simulate document degradation. Table 4 gives the identification results. As Table 4 shows, the language identification rate reaches over 95% on average.

Similar to script identification, the performance of the proposed language identification method is closely related to the number of words within the query images. We note that all testing images containing more than 50 word images are correctly identified. As the word number becomes smaller, the collided *WSC* may dominate the normalized the query *WSV* or *WFV* and this may result in identification errors. Experiments show 15 of the 16 falsely identified testing images in Table 4 contain 20 or less words.

Discussions

The proposed method is quite fast. Currently, it takes around 3 seconds on average to identify script or language from a document image of 1800×1200. The execution time can be further reduced through code optimization. As a direct application, the proposed method can be implemented for multilingual document categorization. In addition, it may be applied for multilingual OCR or multilingual retrieval where scripts or languages must be determined first. We will investigate these in our future work.

Conclusion

This paper reports a script and language identification technique that differentiates scripts and languages in degraded and distorted document images. In our proposed method, scripts and languages are differentiated through document vectorization, which transforms each document image into an electronic document vector. We vectorize document images using stroke run numbers and character extremum points, which are both tolerant to noise, document segmentation errors, and slight document skew. Experiments show the accuracy of the proposed identification method reaches over 95% on average.

Acknowledgement

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

References

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

Cavnar, W., and Trenkle, J. 1994. N-gram based text categorization. *3rd Annual Symposium on Document Analysis and Information Retrieval,* 161–175.

Ding, J.; Lam, L.; and Suen, C. Y. 1997. Classification of oriental and european scripts by using characteristic features. *International Conference on Document Analysis and Recognition,* 70(3):1023–1027.

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

Jain, A. K., and Zhong, Y. 1996. Page segmentation using texture analysis. pattern recognition. volume 29, 743–770.

Lee, D. S.; Nohl, C. R.; and Baird, H. S. Language identification in complex, un-oriented, and degraded document images. *International Workshop on Document Analysis Systems,* 76–88.

Nobile, N.; Bergler, S.; Suen, C. Y.; and Khoury, S. 1997. Language identification of online documents using word shapes. *4th International Conference on Document Analysis and Recognition,* 258–262.

Otsu, N. 1979. A threshold selection method from graylevel histogram. *IEEE Transactions on System, Man, Cybernetics* 19(1):62–66.

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

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

Suen, C. Y.; Bergler, S.; Nobile, N.; Waked, B.; and Nadal, C. P. 1998. Categorizing document images into script and language classes. *International Conference on Advances in Pattern Recognition,* 297–306.

Tan, T. N. 1998. Rotation invariant texture features and their use in automatic script identification. *IEEE Transaction on Pattern Analysis and Machine Intelligence* 751–756.