Script and Language Identification in Noisy and Degraded Document Images

Shijian Lu, Chew Lim Tan, Senior Member, IEEE

Abstract—This paper reports a knowledge-based identification technique that differentiates scripts and languages in noisy and degraded document images. In the proposed method, scripts and languages are identified through document vectorization, which transforms each document image into an electronic document vector that characterizes the shape and frequency of contained character or word images. Document images are vectorized using character extremum points and the number of character runs, which are both tolerant to noise, text fonts, and ranges of document quality. For each script or language studied, a script or language template is first constructed through learning from a set of training images. Scripts and languages of query images are then determined according to the distances between the document vectors of query images and multiple learned script and language templates. Experiments over documents of six scripts and eight Latin-based languages show that the proposed method is fast and accurate in the presence of various document degradations.

Index Terms—Multilingual script identification, language identification, document images.

I. INTRODUCTION

SCRIPT and language identification is normally the first step for multilingual text recognition and information retrieval. Traditionally, this issue is frequently addressed in natural language processing areas where script and language are determined based on the character-coded text [1], [2] or OCR results [3]. 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 being archived during document digitalization process. The identification of scripts and languages of these archived document images accordingly becomes a new challenge. One straightforward solution is to convert imaged text into electronic text through OCR and then determine scripts and languages based on the recognized text. However, OCR process is quite expensive, normally requiring arduous manual correction. More importantly, script or language are also a prerequisite to generic OCR systems in a multilingual environment. Under such circumstance, an alternative solution is to directly determine scripts and languages in scanned document images without OCR.

Some work has been reported to determine scripts from scanned document images. The reported work can be classified into three categories, namely, statistics based approaches [3], [4], [10], [11], token based approaches [5], [6], and texture based approaches [7], [8], [9]. For statistics based approaches, the distribution of horizontal projection profile [10], [11] and the distribution of upward concavity [3], [4], [10] are frequently exploited for script identification. Texture based methods instead utilize texture difference for script identification. For example, Jain et al. [7] attempt to differentiate Chinese from Roman through convolving trained masks. Later, Tan [8] and Busch et al. [9] propose to measure textural statistics, text tokens [5], [6] that are specific to different scripts are also exploited for script identification.

Some work has also been reported to differentiate languages from scanned document images. Currently, most reported language identification work focuses on Latin-based languages. Unlike various scripts that hold different alphabet structures [4], [5] and texture features [8], [9], Latin-based languages are all printed in the same set of Latin alphabets and so have similar texture features. Therefore, word shape coding, which transforms word images into a set of electronic codes, is widely exploited in the literature. The reported work normally starts with a character categorization process, which classifies character images into several categories based on a number of character shape features. For example, the work in [4], [12], [13] proposes to group characters and other text symbols into six, ten, and thirteen categories, respectively. With the character categorization results, word shape codes (WSC) are then constructed and languages are finally determined according to the frequency of a single word [4], [12], word pair, and word trigram [13].

The proposed script and language method has multiple advantages compared with the reported ones. Firstly, we integrate the script and language identification within the same framework instead of just focusing on the identification of scripts [5], [7], [8], [9], [10], [11] or Latin-based languages [12], [13]. Secondly, as the exploited word shape features are both tolerant to noise and document degradation, our language identification method can work well on noisy and degraded document images where traditional methods [4], [12], [13] may fail. Thirdly, though we focus on the study of six scripts and eight Latin-based languages in this paper, our method can be easily extended through learning from a training set to deal with document images of new scripts or languages.

We identify scripts and languages through document vectorization, which transform each document image into an electronic document vector that characterizes the shape and frequency of contained character and word images. For each script and language studied, a script and language knowledge template is first constructed through a learning process. Scripts and languages of query images are then determined according to the distances between query document vectors and multiple learned script or language templates. We perform document vectorization using vertical character runs, horizontal word
techniques have been reported and we directly adopt the background. A large number of document segmentation algorithms, respectively. Experimental results and discussions are then given in Section 5. Finally, we draw some concluding remarks in Section 6.

II. DOCUMENT IMAGE PREPROCESSING

Archived document images are often coupled with different types of degradation. The four document samples in Figure 1(a)-(d) illustrate three types of frequently encountered document degradation, namely, poor resolution, noise including Gaussian and pepper & salt noise, and physical paper degradation. Preprocessing is required to compensate document degradation and produce “clean” binary document text before the ensuing feature extraction and document vectorization. Firstly, captured document text must be segmented from the background. A large number of document segmentation techniques [20] have been reported and we directly adopt

runs, and character extremum points, which are all tolerant to noise, text font, and document degradation.

The framework of the proposed method is as follows. Given a query image, a query document vector characterizing the density and distribution of vertical character runs is first constructed. The script of the query document is then determined according to the distances between the query document vector and multiple learned language templates. We study documents of six scripts including Arabic, Chinese, Hebrew, Japanese, Korean, and Roman and eight Latin-based languages including English, French, German, Italian, Spanish, Portuguese, Swedish, and Norwegian.

The outline of the rest of this paper is as follows. In Section 2, we first describe a few document image preprocessing operations required before document vectorization. Section 3 and 4 then present the proposed script and language identification algorithms, respectively. Experimental results and discussions are then given in Section 5. Finally, we draw some concluding remarks in Section 6.

Fig. 1. Document image preprocessing including text segmentation, size filtering, and single pixel hole, concave, and convex removal.

Otsu’s algorithm [18]. After document segmentation, binary document components are then labeled through connected component analysis [19]. For each labeled component, information including component size, centroid, and pixel list is determined and stored for later image processing.

Noise must also be removed before the ensuing feature detection. In the proposed method, noise means not only traditional noise but also small document components such as character ascender & descender and character ascent & descent (such as ă and î), which are not required and may instead affect the ensuing document vectorization. Currently, most language identification techniques [4], [12], [13] depend heavily on these small components for character shape coding. Unfortunately, degraded documents normally contain some quantity of noise with size similar to those small components, which introduce coding errors. As our method does not require these small components, we just remove them together with noise of similar size and so produce a cleaner text image for ensuing processing. We remove noise 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 desired character components contain much more than 10 pixels. The second round filtering further removes noise of bigger size and small document components. Filtering threshold is determined as the size of the kth document component so that the inter-class variance σ(k) of document component size histogram reaches maximum:

\[
\sigma(k) = \frac{\left( \frac{\varphi(k) \cdot \sum_{i=1}^{S} i \cdot p(i) - \sum_{i=1}^{k} i \cdot p(i)}{\varphi(k) \cdot (1 - \varphi(k))} \right)^2}{\phi(1)}
\]  

where parameter 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:

\[
p(i) = n(i)/N, \quad \varphi(k) = \sum_{i=1}^{k} p(i)
\]  

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.

It should be clarified that some single pixel hole, concave, and convex may exist along character strokes due to document degradation such as pepper & salt noise. The single pixel hole,
concave, and convex must be removed beforehand because they may affect the detection of the three exploited text shape features, namely, vertical character runs, horizontal word runs, and character extremum points. The removal can be simply accomplished using some logical or morphological operators. Figure 2 illustrates the preprocessing process using a sample word “empirical” labeled with a rectangle in Figure 1(b). For the close-up view of the sample word in Figure 2(a), Figure 2(b) shows the segmentation result. Figure 2(c) then gives the size filtering result where pepper & salt noise and small document component (top part of “i”) have been filtered out. Figure 2(d) finally gives the preprocessing result where single pixel hole, concave, and convex have been removed.

III. SCRIPT IDENTIFICATION

We identify scripts through document vectorization, which transforms each document image into a document vector based on the density and distribution of vertical runs of the preprocessed document components. For each scripts studied, a script template is first constructed through clustering of a set of training document vectors. The scripts of query images are then determined according to the distance between the query document vectors and multiple learned script templates.

A. Document Vectorization I

We exploit the number and position of vertical component run (VCR) illustrated in Figure 3 for script identification. Scanning from the top to the bottom, a vertical run is detected when a vertical scan line passing through the component centroid enters the component regions from the background. The run position is located at the topmost pixel of the document component that meets the vertical scan line. The vertical scan line is defined to pass through the component centroid so that the VCR is more tolerant to text fonts and styles. To study VCR distribution along text lines, we divide text lines into three equidistant zones, namely, the x zone around the x line, the middle zone around the middle line, and the base zone around the base line, respectively. Detected VCR can be classified to three text zones accordingly as illustrated in Figure 2.

We characterize the number and position of VCR using a vertical run vector (VRV) of dimension 32. The first 8 VRV elements characterize the density of VCR along text lines in vertical direction. We set the upper limit of VCR number at 8 as the number of VCR in each scanning round is normally no bigger than 8 for most scripts currently under study. Considering the fact that all VCR may occur within a single text zone, the upper limit of VCR number in three text zones are set at 8 as well. Each document can thus be transformed into a VRV of dimension 32 where the first 8 elements record the number of document components with VCR number equal to their indices and the following 24 elements record the distribution of VCR within the top, middle, and base text zones, respectively.

\[
VRV = [N_1 \cdots N_8 \ U_1 \cdots U_8 \ M_1 \cdots M_8 \ L_1 \cdots L_8] \tag{3}
\]

where \(N_i, i = 1 \cdots 8\), gives the numbers of document components with VCR number equal to \(i\). \(U_i, M_i, \) and \(L_i, i = 1 \cdots 8\), define the positions of VCR within the three text zones, respectively.

Take the three characters in Figure 3 as an example. For the Arabic character, only \(N_3\) and \(U_1\) are set to 1 because there is only one vertical run that occurs within the top text zone. But for English character “Z”, there are three vertical runs occurring within the top, middle, and base text zones. Therefore, \(N_3, U_1, M_2, \) and \(L_3\) are set to 1 because the second and third VCR occur within the middle and base text zones, respectively. For document images containing multiple characters, the corresponding document VRV can be simply determined as the sum of the VRV of all preprocessed document components. As the length of documents may be different, document VRV must be normalized to cancel the effects of document length before script identification:

\[
VRV = \frac{VRV}{\sum_{i=1}^{N} VRV_i} \tag{4}
\]

where \(N\) is equal to 8. Therefore, the normalization factor (denominator) is set at the number of preprocessed document components within the studied document image.

The VRV of documents of the same script are normally quite similar. Figure 4 gives the VRV of four studied scripts where eight document vectors are shown for each script, demonstrating the density and distribution of the VCR of the underlying scripts. As we can see, document VRV of the same script are quite close to each other, while document VRV of
different scripts are quite different, either in VCR density or in VCR distribution. As Figure 4 shows, a large portion of Chinese characters hold four or more VCR, but most Hebrew characters hold just one or two VCR instead. In addition, the distribution of VCR of different scripts in the three text zones are far from each other as well.

B. Script Template Construction

As described in the last subsection, document VRV are closely related to the underlying scripts. Script templates can therefore be constructed through clustering of multiple training document vectors of different scripts. We adopt K-Mean clustering algorithm and script templates are estimated as the cluster centroids accordingly.

A training set is first constructed using 273 documents where each script hold at least 41 training documents. The 273 training documents are collected from different sources and each document is composed of 20-40 text lines. Document texts are printed in different fonts and styles and scanned to images at 600 ppi. Each training image is then converted to a VRV based on the Document Vectorization I described in the last subsection. Lastly, the training set is constructed using the VRV of the 273 training images.

$$VRV_{training} = \{VRV_i, i = 1, \ldots, N\}$$

where $VRV_i$ denotes the VRV of the $i^{th}$ training image and $N$ is equal to 273.

The main idea of K-Mean clustering is to classify a data set into $k$ clusters and determine $k$ centroids, one for each cluster. As we just study documents of six scripts, cluster number $k$ can be set at 6 as a priori. To speed up the clustering process and achieve better clustering results, we set initial cluster centroids (ICC) as $k$ (6) training VRV that are as far away as possible from each other. Therefore, the first ICC can be chosen randomly and the followings are determined to be the VRV that are farthest from the chosen ICC. With $k$ ICC determined, the first round clustering starts by associating each training VRV to its nearest centroid. After this, $k$ new cluster centroids are updated and the clustering loop proceeds until the object function below is minimized:

$$J = \sum_{i=1}^{K} \sum_{j=1}^{M} ||VRV_i^j - C_i||^2$$

where $M$ gives the number of training VRV associated to $C_i$. $||VRV_i^j - C_i||^2$ gives the distance between $VRV_i^j$ and $C_i$. We try several sets of ICC and every time the clustering process converges nicely. Script templates can be determined as the the six cluster centroids accordingly.

C. Script Identification

Based on the learned script templates, script of the query image can thus be determined according to the distance between the VRV of the query document and the six learned script templates. We evaluate the vector distances using Bray Curtis distance, which has a nice property that its value always lies between 0 and 1:

$$VD_i = \frac{\sum_{j=1}^{N}(VRV_j - ST_j)}{\sum_{j=1}^{N}(VRV_j) + \sum_{j=1}^{N}(ST_j)}$$

where parameter $N$ is equal to 32. $VRV_j$ represents the $j^{th}$ element of the query VRV. Parameter $ST_j$ corresponds to the $j^{th}$ element of the $i^{th}$ learned script template. As a result, query images are determined to be printed in the scripts with the smallest Bray Curtis distance $VD_i$.

IV. LATIN-BASED LANGUAGE IDENTIFICATION

For documents printed in Roman script, languages cannot be differentiated based on the VCR information because all Latin-based languages such as English and French share the same set of Roman alphabets. We therefore propose a new document vectorization scheme, which transforms each document image into a document vector through a word shape coding approach. For each Latin-based language studied, a language template is first constructed through a memory-based learning process. Languages of query images are then similarly determined according to the distances between query document vectors and multiple constructed language templates.

A. Word Shape Feature Extraction

We identify Latin-based language through a word shape coding approach. In our earlier work, a few word shape coding schemes have been proposed for topic categorization [14], keyword spotting [15], and language identification [16] within scanned document images. In this paper, we adopt our earlier work [16] and exploit two word shape features for document vectorization. The first word shape feature refers to character extremum points detected from the upward and downward text boundary. The second is the number of horizontal word runs, which is equal to the number of intersections between character strokes within a word image and the related middle line of text as illustrated in Figure 5(b).

For each filtered document components, its 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 component pixels...
of each scanning round corresponding to the highest and lowest pixels constitute upward and downward text boundary. Figure 5(a) shows the extracted upward and downward text boundaries where text is printed in light gray for highlighting. After removing the small boundary components, each upward or downward boundary normally forms an arbitrary curve that can be modeled using a function \( f(x) \). The extrema of the \( f(x) \), which correspond to the locally highest or lowest boundary pixels, can be mathematically defined as below:

**Definition 1:** Given an arbitrary curve \( f(x) \):

1. **We say that** \( f(x) \) **has a relative maximum at** \( x = c \) **if** \( f(x) \leq f(c) \) **for every** \( x \) **in some open interval around** \( x = c \).
2. **We say that** \( f(x) \) **has a relative minimum at** \( x = c \) **if** \( f(x) \geq f(c) \) **for every** \( x \) **in some open interval around** \( x = c \).

For Roman characters, the upward extremum points are normally detected with six boundary patterns as illustrated in Figure 6(a)-(f). The downward text boundary also takes six patterns, which actually correspond to the 180 degree rotation of the six upward patterns in Figure 6. For the sample word image “script” in Figure 5(a), the black dots in Figure 5(b) shows the detected character extremum points.

The two exploited word shape features are tolerant to font and text segmentation error caused by document degradation. Figure 5 illustrates these two properties. As Figure 5 shows, the number of horizontal word runs and character extremum points are correctly detected, though the sample word “script” is typed in two totally different fonts. In addition, the characters “r” and “i” in “script” on the top row are falsely connected due to document degradation. With traditional character shape coding approaches [4], [12], [13], these two characters will be treated as one and the resulting word shape code will be totally different from the real one. But the proposed features can capture word shapes correctly while characters are falsely connected. As Figure 5 show, both character extremum points and horizontal word runs are correctly detected in the presence of text segmentation errors.

**B. Document Vectorization II**

Similar to the classification of vertical component runs, character extremum points are also classified into three categories according to their positions relative to the x-line and base line 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 character extremum points within the above three categories as \( I \), \( 2 \), and \( 3 \), respectively.

Combined with the number of horizontal word runs, each word image can thus be transformed to a WSC where character extremum points are coded first and the number of horizontal runs follows behind. Take word image “the” as an example. The corresponding WSC is a digit sequence 3322 where the subsequence 322 is coded based on character extremum points and the following digit 4 refers to the number of horizontal word runs. Furthermore, within the subsequence 3322, the first digit 3, the following digits 32, and the last digit 3 are coded based on the extremum points of character “t”, “h”, and “e”, respectively. A document image can thus be converted into a word shape vector (WSV). In the proposed method, each vector element is composed of two components: the first is a unique WSC within the WSV and the second corresponds to a word occurrence number (WON), giving the frequency of the word within the studied document image.

\[
W SV = \left[ (WSC_1 : WON_1), \cdots, (WSC_n : WON_n) \right]
\]  

where \( n \) refers to the number of unique WSC within the studied document image.

Figure 7 gives the WSV construction algorithm. As Figure 7 shows, given a WSC \( WSC_j \) translated from a word image within the \( i^{th} \) training document, the corresponding WSV is searched for the element with the same WSC. If such element exists, the corresponding WON component \( WSV_{j,won} \) is increased by one. Otherwise, a new WSV element is created where the WSC component \( WSV_{j+1,wsc} \) is set as \( WSC_j \) and the WON component \( WSV_{j+1,won} \) is initialized as one instead. The conversion process terminates until all WSC within the \( i^{th} \) training document are examined.

**C. Language Template Construction**

For each Latin-based language studied, we construct a language template through a memory-based learning process.
Given a set of \( \text{WSV} \) of the training images of the same language and a blank word shape template \( \text{WST} \)
For \( i=1 \): size of \( \text{WSV} \)
For \( j=1 \): size of \( \text{WSV}_j \)
For \( k=1 \): size of \( \text{WST} \)
If \( \text{WSV}_{i,j,\text{wsc}} = \text{WST}_{k,\text{wsc}} \)
\[ \text{WST}_{k,\text{wsc}} = \text{WST}_{k,\text{wsc}} + \text{WSV}_{i,j,\text{wsc}} \]
Break;
End
End
Add a new \( \text{WST} \) element with
\[ \text{WST}_{k+1,\text{wsc}} = \text{WSV}_{i,j,\text{wsc}} \text{ and } \text{WST}_{k+1,\text{won}} = \text{WSV}_{i,j,\text{won}} \]
End
End

Fig. 8. The algorithm for word shape vector construction.

The target is to remember the \( \text{WSC} \) and the corresponding word frequency of languages.

We construct a training set using 417 documents where each studied language holds at least 46 training documents. Training documents are collected from different sources and each document contains 20-40 text lines. Document texts are printed in different fonts and styles and scanned to document images at 600 ppi. Each training document is then transformed into a \( \text{WSV} \) using the Document Vectorization II described in the last subsection. Lastly, the training set is constructed using the \( \text{WSV} \) converted from the 417 training images:

\[ LT\text{S} = \{ \text{WSV}_{\text{EN}}, \text{WSV}_{\text{FR}}, \ldots, \text{WSV}_{\text{NO}} \} \] (9)

where \( \text{WSV}_{\text{EN}}, \text{WSV}_{\text{FR}} \ldots \ldots \text{WSV}_{\text{NO}} \) denote multiple subsets of \( \text{WSV} \) transformed from document images of English, French, German, Italian, Spanish, Portuguese, Swedish, and Norwegian, respectively.

Different from script template construction, the language of each training document is known here. Figure 8 shows for each of the training \( \text{WSV} \) of the \( i \)th language currently under study, the \( \text{WST} \) of each of \( \text{WSV} \) element \( \text{WSV}_{i,j,\text{wsc}} \) is compared with that of language template elements. If the language template holds an element with the same \( \text{WST} \), the corresponding \( \text{WON} \) component \( \text{WST}_{k,\text{won}} \) is increased by the \( \text{WON} \) of the \( \text{WSV} \) element under study \( \text{WSV}_{i,j,\text{won}} \). Otherwise, the new \( \text{WST} \) element is created where the \( \text{WSC} \) element \( \text{WST}_{k+1,\text{wsc}} \) is set as the \( \text{WSV}_{i,j,\text{wsc}} \) and the \( \text{WON} \) component is set as \( \text{WSV}_{i,j,\text{won}} \). The learning process terminates until all training \( \text{WSV} \) of the \( i \)th language are examined.

For the 417 training \( \text{WSV} \) studied, Table I gives the numbers of the learned \( \text{WSC} \) where the diagonal gives the numbers of the \( \text{WSC} \) of each studied languages and the off-diagonal items gives the numbers of \( \text{WSC} \) shared by two related languages. As Table I shows, the average \( \text{WSC} \) collision rate just reaches around 10%. Considering the facts that the learned \( \text{WSC} \) 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 \( \text{WSC} \) are remembered.

### TABLE I

<table>
<thead>
<tr>
<th>Language</th>
<th>EN</th>
<th>DE</th>
<th>FR</th>
<th>IT</th>
<th>ES</th>
<th>PT</th>
<th>SV</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>6344</td>
<td>1332</td>
<td>1586</td>
<td>1494</td>
<td>1526</td>
<td>1404</td>
<td>1064</td>
<td>1076</td>
</tr>
<tr>
<td>German</td>
<td>6732</td>
<td>1280</td>
<td>1546</td>
<td>1448</td>
<td>1466</td>
<td>1400</td>
<td>1158</td>
<td>1230</td>
</tr>
<tr>
<td>French</td>
<td>1386</td>
<td>1546</td>
<td>7472</td>
<td>1620</td>
<td>1646</td>
<td>1532</td>
<td>1232</td>
<td>1210</td>
</tr>
<tr>
<td>Italian</td>
<td>1944</td>
<td>1448</td>
<td>1620</td>
<td>6381</td>
<td>1216</td>
<td>1708</td>
<td>1208</td>
<td>1194</td>
</tr>
<tr>
<td>Spanish</td>
<td>1526</td>
<td>1466</td>
<td>1646</td>
<td>1716</td>
<td>6811</td>
<td>1808</td>
<td>1278</td>
<td>1196</td>
</tr>
<tr>
<td>Portuguese</td>
<td>1404</td>
<td>1400</td>
<td>1552</td>
<td>1708</td>
<td>1808</td>
<td>6144</td>
<td>1124</td>
<td>1102</td>
</tr>
<tr>
<td>Swedish</td>
<td>1620</td>
<td>1646</td>
<td>1278</td>
<td>1400</td>
<td>1552</td>
<td>1404</td>
<td>1620</td>
<td>6381</td>
</tr>
<tr>
<td>Norwegian</td>
<td>1494</td>
<td>1230</td>
<td>1272</td>
<td>1208</td>
<td>1278</td>
<td>1102</td>
<td>1418</td>
<td>9147</td>
</tr>
</tbody>
</table>

### D. Language Identification

Latin-based languages are determined based on the similarities between the \( \text{WSV} \) of query images and multiple learned language templates. The similarities can be evaluated using solely \( \text{WSC} \) information because \( \text{WSC} \) collision rates are quite small. Disregarding \( \text{WON} \) components, each \( \text{WSV} \) or language template can be viewed as a data set that is composed of a number of \( \text{WSC} \) digit sequences. Similarities between query \( \text{WSV} \) and language templates can therefore be evaluated according to the number of \( \text{WSC} \) they share:

\[ SD_i = \frac{| \text{WST}_i \cap \text{WSV}_{\text{query}} |}{| \text{WST}_i |^{1/2} \times | \text{WSV}_{\text{query}} |^{1/2}} \] (10)

where function \(| \cdot |\) gives the cardinality (size) of the studied set. Therefore, The numerator term gives the number of \( \text{WSC} \) shared by the query \( \text{WSV} \) and the \( i \)th studied language template \( \text{WST}_i \). The denominator term, corresponding to a normalization factor, removes the effect of document length. Query image is accordingly assigned to the language with the smallest \( SD_i \).

The similarities between the \( \text{WSV} \) of query documents and language templates can be more elaborately evaluated using word frequency information. Before similarity measurement, word frequency must be normalized as follows:

\[ WF_i = \frac{\text{WON}_i}{\sum_{i=1}^{N} \text{WON}_i} \] (11)

where \( \text{WON}_i \) gives the occurrence number of \( i \)th \( \text{WST} \). The denominator term instead gives the sum of \( \text{WON} \) within the query \( \text{WSV} \) or the learned language templates. Based on the normalized word frequency information, the similarity can be simply evaluated using the cosine measure between query \( \text{WSV} \) and multiple learned language templates:

\[ SM_i = \frac{\sum_{j=1}^{N} WF_j \cdot LT_{ij}^{3}}{\sqrt{\sum_{j=1}^{N} (WF_j)^2 + \sum_{j=1}^{N} (LT_{ij}^{3})^2}} \] (12)

where \( N \) gives the size of the query \( \text{WSV} \). \( WF_j \) represents the \( j \)th normalized \( \text{WON} \) component within the query \( WFV \). \( LT_{ij}^{3} \) is determined as follows. For each element within the query \( WFV \), the \( j \)th language template is searched for the element with the same \( \text{WST} \). If such element exist, the \( LT_{ij}^{3} \) is determined as the corresponding normalized \( \text{WON} \) component. Otherwise, \( LT_{ij}^{3} \) is simply zero. As a result, query image is determined to be printed in the language with the biggest \( SM_i \).
TABLE II
ACCURACY OF SCRIPT IDENTIFICATION IN RELATION TO THE NUMBER OF LINES OF TEXT. (NTL: NUMBER OF TEXT LINES; ANC: AVERAGE NUMBER OF CHARACTERS; NTI: NUMBER OF TESTING IMAGE)

<table>
<thead>
<tr>
<th>NTL</th>
<th>NTI</th>
<th>ANC</th>
<th>Acc. of Our Method</th>
<th>Acc. of Larry’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>55</td>
<td>351</td>
<td>98.18%</td>
<td>98.18%</td>
</tr>
<tr>
<td>10</td>
<td>73</td>
<td>307</td>
<td>97.26%</td>
<td>98.63%</td>
</tr>
<tr>
<td>8</td>
<td>86</td>
<td>232</td>
<td>97.61%</td>
<td>96.51%</td>
</tr>
<tr>
<td>6</td>
<td>114</td>
<td>189</td>
<td>94.74%</td>
<td>92.11%</td>
</tr>
<tr>
<td>4</td>
<td>149</td>
<td>125</td>
<td>90.60%</td>
<td>87.25%</td>
</tr>
<tr>
<td>2</td>
<td>228</td>
<td>68</td>
<td>85.53%</td>
<td>84.21%</td>
</tr>
<tr>
<td>1</td>
<td>284</td>
<td>35</td>
<td>77.82%</td>
<td>74.30%</td>
</tr>
</tbody>
</table>

TABLE III
ACCURACY OF LANGUAGE IDENTIFICATION IN RELATION TO THE NUMBER OF LINES OF TEXT. (ANW: AVERAGE NUMBER OF WORDS)

<table>
<thead>
<tr>
<th>NTL</th>
<th>NTI</th>
<th>ANW</th>
<th>Acc. of Our Method</th>
<th>Acc. of Larry’s method</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>67</td>
<td>206</td>
<td>98.51%</td>
<td>98.51%</td>
</tr>
<tr>
<td>10</td>
<td>82</td>
<td>177</td>
<td>97.56%</td>
<td>97.56%</td>
</tr>
<tr>
<td>8</td>
<td>97</td>
<td>141</td>
<td>95.88%</td>
<td>94.85%</td>
</tr>
<tr>
<td>6</td>
<td>121</td>
<td>103</td>
<td>94.21%</td>
<td>92.56%</td>
</tr>
<tr>
<td>4</td>
<td>168</td>
<td>74</td>
<td>91.07%</td>
<td>88.10%</td>
</tr>
<tr>
<td>2</td>
<td>263</td>
<td>37</td>
<td>83.65%</td>
<td>79.85%</td>
</tr>
<tr>
<td>1</td>
<td>324</td>
<td>18</td>
<td>76.54%</td>
<td>66.37%</td>
</tr>
</tbody>
</table>

V. Experiments and Discussions

A series of experiments are conducted and described in this section. Firstly, a set of normal images with the same quality as that of training images used in Section III and IV are tested. After that, noise and document degradation effects are examined using several sets of noisy and degraded document images. Lastly, two sets of images of a new script and a new Latin-based language are experimented to study the extendibility of the proposed method. We compare our method with Larry’s throughout the experiments.

A. Script and Language Identification

276 and 347 testing documents of six scripts and eight Latin-based languages currently under study are prepared to evaluate the performance of the proposed identification method. Similar to the training image described in Section III and IV, each testing document contains 20-40 text lines and document texts are printed in different fonts and styles. All testing documents are scanned to document images at 600 ppi. 276 VRV and 347 WSV are then constructed and scripts and languages are determined based on the distances between the testing VRV or WSV and the learned script and language templates. Experiments show that the script and language of all testing documents are correctly determined.

Our method depends heavily on the number of characters and words within document images. As the number of characters or words becomes smaller, the VRV of the query image may not reflect the real density and distribution of VCR. At the same time, the effects of WSC ambiguity become relatively stronger while document images contain just a few words. To test the relations between the performance of our method and character and word number, we create two sets of testing images as shown in Table II and III, all of which are cropped from the 276 and 347 testing images described above.

Table II gives the script identification results. Similar to the work in [3], [10], Larry’s method can only differentiate Oriental from Latin script based on the distribution of upward concavity, though it can identify Chinese, Japanese, and Korean using the optical density information. As a result, Larry’s method misclassifies all testing images of Hebrew, one script currently under study, to Roman category. Therefore we gauge the accuracy of Larry’s method in Table II using just a subset of testing images, namely, the ones printed in Roman, Chinese, Japanese, and Korean, respectively. As Table II shows, the performances of Larry’s method and ours are both closely related to the number of the contained characters or words.

Table III gives the language identification results. As Table III shows, Larry’s method deteriorates more seriously while the number of words and text lines becomes small. This can be explained by higher WSC ambiguity of Larry’s word shape coding scheme where word images are coded based on character ascender & descender and character ascent & descent. Under such coding scheme, lots of different words such as “like” and “love” share the same WSC. The effects of higher WSC ambiguity becomes more apparent while the number of words within testing documents becomes smaller.

B. Noise and Degradation

Our method is more tolerant to noise and various document degradation. We experiment with 217 and 325 challenging images of six scripts and eight Latin-based languages currently under study. Among the testing images, some challenge images are captured from the physically degraded documents collected from Internet and digital libraries, some are synthetic through addition of noise including Gaussian noise () and pepper & salt noise (), and some are of poor quality through reduction of scanning resolution as shown in Figure 1. Text is printed in different fonts and styles and each document contains at least 15 text lines. We compare the performances of our methods with Larry’s and Table IV gives experimental results.

At Table IV shows, the proposed script identification method achieves around the same performance as Larry’s, which is also tested using a subset of testing images of Chinese, Japanese, Korean, and Roman. But for Latin-based languages, the performance of Larry’s method deteriorates much more seriously. This can be explained by two factors. Firstly, Larry’s methods including some other methods [12], [13] depends heavily on the small character components. But for documents with strong noise especially pepper and salt noise, it is hard to differentiate those small components from noise of similar size. Secondly, Larry’s methods assume that characters are correctly segmented, but degraded documents normally contain a large portion of broken and touching characters, which introduce coding errors.

Compared with Larry’s method, our language identification method is much better in the presence of noise and document degradation. We achieve noise tolerance through two rounds of size filtering discussed in Section II. As our method requires
C. Script and Language Extension

The proposed method can be easily extended to deal with documents of a new script or Latin-based language. To add a new script, the only thing required is to create a set of training documents of that script and do the K-Mean clustering again. If the density and distribution of VCR of the new script is far different from that of existing ones, the K-Mean clustering process always converges nicely and produces a new cluster centroid accordingly. The addition of a new Latin-based language can be similarly accomplished through creating a set of training documents of that new languages and remembering the converted WSC and related WON information.

We test the extendability of our method using documents of a new Indian script Bangla and a new Latin-based language Dutch. For each of the new script and language studied, we create 80 documents where 40 are for template construction and another 40 are for testing. The documents contain 2-40 text lines and are scanned as 600 ppi. Experimental results show that 39 of the 40 Bangla testing document and 37 of the 40 Dutch testing documents are correctly identified. Besides, all four falsely identified testing document contain 5 or less text lines, which can be expected as the proposed method works on document instead of character or word level.

VI. CONCLUSION

This paper reports a knowledge-based identification technique that differentiates scripts and languages in noisy and degraded document images. In the proposed method, scripts and languages are determined through document vectorization, which transform each document image into an electronic document vector. Documents of different scripts are vectorized using the density and distribution of vertical component runs, whereas documents of different languages are vectorized through a word shape coding approach. Experimental results show that the proposed methods is accurate and tolerant to text fonts, noise, and various document degradation.

Currently, the proposed method works well on most scanned document images. As resolution of sensor increases, more and more documents are being captured using a digital camera due to its advantages in speed, flexibility, and portability aspects. However, camera images of document generally contain perspective distortion. Besides, for documents lying over a smoothly curved surface, the document images captured by a digital camera are coupled with geometric distortion. Script and language identification in the presence of perspective and geometric distortion is worth further investigation. In addition, the proposed word shape coding scheme has the potential for the categorization of multilingual document images without OCR. We will study these in our future work.

ACKNOWLEDGMENT

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

REFERENCES


TABLE IV

<table>
<thead>
<tr>
<th>Script/Language</th>
<th>Acc. of Ours</th>
<th>Acc. of Larry’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low quality</td>
<td>93.88%</td>
<td>93.88%</td>
</tr>
<tr>
<td>Gaussian</td>
<td>96.88%</td>
<td>98.43%</td>
</tr>
<tr>
<td>Pepper-Salt</td>
<td>99.92%</td>
<td>96.34%</td>
</tr>
<tr>
<td>Degradation</td>
<td>95.83%</td>
<td>97.53%</td>
</tr>
</tbody>
</table>

The accuracy of script and language identification in relation to various document degradation.