

Texture for Script Identification

Andrew Busch, *Member, IEEE*, Wageeh W. Boles, *Member, IEEE*, and
Sridha Sridharan, *Senior Member, IEEE*

Abstract—The problem of determining the script and language of a document image has a number of important applications in the field of document analysis, such as indexing and sorting of large collections of such images, or as a precursor to optical character recognition (OCR). In this paper, we investigate the use of texture as a tool for determining the script of a document image, based on the observation that text has a distinct visual texture. An experimental evaluation of a number of commonly used texture features is conducted on a newly created script database, providing a qualitative measure of which features are most appropriate for this task. Strategies for improving classification results in situations with limited training data and multiple font types are also proposed.

Index Terms—Script identification, wavelets and fractals, texture, document analysis, clustering, classification and association rules.

1 INTRODUCTION

As the world moves ever closer to the concept of the “paperless office,” more and more communication and storage of documents is performed digitally. Documents and files that were once stored physically on paper are now being converted into electronic form in order to facilitate quicker additions, searches, and modifications, as well as to prolong the life of such records. A great proportion of business documentation and communication, however, still takes place in physical form and the fax machine remains a vital tool of communication worldwide. Because of this, there is a great demand for software which automatically extracts, analyzes, and stores information from physical documents for later retrieval. All of these tasks fall under the general heading of *document analysis*, which has been a fast growing area of research in recent years.

A very important area in the field of document analysis is that of optical character recognition (OCR), which is broadly defined as the process of recognizing either printed or handwritten text from document images and converting it into electronic form. To date, many algorithms have been presented in the literature to perform this task, with some of these having been shown to perform to a very high degree of accuracy in most situations, with extremely low character-recognition error rates [1]. However, such algorithms rely extensively on a priori knowledge of the script and language of the document in order to properly segment and interpret each individual character. While in the case of Latin-based languages such as English, German, and French, this problem can be overcome by simply extending the training database to include all character variations, such an approach will be unsuccessful when dealing with differing script types. At best, the accuracy of such a system will be necessarily reduced

by the increased number of possible characters. In addition, many script types do not lend themselves to traditional methods of character segmentation, an essential part of the OCR process and, thus, must be handled somewhat differently. For all of these reasons, the determination of the script of the document is an essential step in the overall goal of OCR.

Previous work has identified a number of approaches for determining the script of a printed document. A number uses character-based features or connected component analysis [2], [3]. The paradox inherent in such an approach is that it is sometimes necessary to know the script of the document in order to extract such components. In addition to this, the presence of noise or significant image degradation can also significantly affect the location and segmentation of these characters, making them difficult or impossible to extract. In such conditions, a method of script recognition which does not require such segmentation is required. Texture analysis techniques are a logical choice for solving such a problem as they give a global measure of the properties of a region, without requiring analysis of each individual component of the script. Printed text of different scripts is also highly preattentively distinguishable, a property which has long been considered a sign of textural differences [4].

In Section 2, we provide an overview of the problem of script recognition, highlighting its importance in a number of applications. The idea of using texture analysis techniques to determine the script of printed text is then investigated in Section 3, showing the rationale behind such an approach and the work done in this area to date. Due to the nature of most texture features, document images must be normalized to ensure accuracy and algorithms for such preprocessing stages, including binarization, skew detection and correction, and normalization of text are presented in Section 4. Details of the final texture features extracted from the normalized images are then given in Section 5, with experimental evaluation of each such feature set performed in Section 6, using a newly constructed document image database containing a total of 10 different scripts.

Section 7 outlines a technique for improving classifier performance by utilizing the common properties of printed text when training a Gaussian mixture model classifier. It is proposed that by taking advantage of this a priori information, less training data will be required to adequately describe

• A. Busch is with the School of Microelectronic Engineering, Griffith University Nathan Campus, Nathan, QLD. 4111 Australia. E-mail: a.busch@griffith.edu.au.

• W.W. Boles and S. Sridharan are with the School of Engineering Systems, Queensland University of Technology, GPO Box 2343, Brisbane, QLD. 4001 Australia. E-mail: {w.boles, s.sridharan}@qut.edu.au.

Manuscript received 1 Mar. 2004; revised 10 Dec. 2004; accepted 14 Dec. 2004; published online 14 Sept. 2005.

Recommended for acceptance by M. Pietikainen.

For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0112-0304.

each class density function, leading to increased over performance. The problem of multifont script recognition is addressed in Section 8. This is of much importance in practical applications since the various fonts of a single script type can differ significantly in appearance and are often not adequately characterized in feature space if modeling with a single class is attempted. A method for clustering the features and using multiple classes to describe the distribution is proposed here, with experimental results showing that this technique can achieve significant improvement when automatically dealing with many font types.

2 SCRIPT AND LANGUAGE RECOGNITION

Although a large number of OCR techniques have been developed in recent years, almost all existing work in this field assumes that the script and/or language of the document to be processed is known. Although it is certainly possible to train an OCR system with characters from many languages to obtain a form of language independence, the performance of such a classifier would naturally be lower than one trained solely with the script and/or language of interest. Using specialized classifiers for each language is also advantageous in that it allows for the introduction of language and script specific knowledge when performing other required tasks such as document segmentation character separation. Using such specialized classifiers in a multilingual environment requires an automated method of determining the script and language of a document.

A number of different techniques for determining the script of a document have been proposed in the literature. Spitz has proposed a system which relies on specific, well-defined pixel structures for script identification [2]. Such features include locations and numbers of upward concavities in the script image, optical density of text sections, and the frequency and combination of relative character heights. This approach has been shown to be successful at distinguishing between a small number of broad script types (Latin-based, Korean, Chinese, and Japanese) and moderately effective at determining the individual language in which the text is written. Results when using a wider variety of script types (Cyrillic, Greek, Hebrew, etc.) are not presented nor is any attempt made to define the conditions for an unknown script type.

Preprocessing of document images is required before this technique is applied. The purpose of this is to form a binary image, that is, an image composed entirely of white and black pixels only. By convention, white pixels are considered background and black pixels are considered text. Following this, connected components [5] are extracted from the binary representation. For each component extracted in this way, information such as the position, bounding box, and lists of pixels runs is stored. Representing the image in this way provides an efficient data structure on which to perform image processing operations, such as the calculation of upward concavities and optical density required in the latter stages of the process. For many script types, calculating connected components will also separate individual characters, although this is not always true in the general case. Further work has attempted to address this problem by segmenting individual connected components at this stage [6]. This is required primarily for languages that possess a high degree of connectivity between characters, such as types

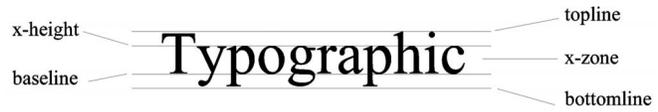


Fig. 1. Commonly labeled text regions for an English sample.

of Indian script, and certain fonts such as italics which enhance connectivity of consecutive characters.

Suen et al. also apply two extra segmentation algorithms at this point, in order to remove extremely large connected components and noise [3]. Large components are considered to be those with bounding boxes more than five times the average size. It is thought that these correspond to nontextual regions of the input image and can safely be discarded. Any bounding boxes with dimensions smaller than the text font stroke are considered noise and also removed. Practical experiments have shown some problems when using these techniques, as important image features such as the dots on "i" characters and accent marks are erroneously removed. If character segmentation is not accurate entire lines of text may also be removed, as they are considered to be a single large connected component.

Before feature extraction, it is necessary to define various regions within a line of text. Four horizontal lines define the boundaries of three significant zones on each text line. These lines are the top-line, x-height, baseline, and bottom-line. The top-line is the absolute highest point of the text region, and typically corresponds to the height of the largest characters in the script, for example "A." The x-height line is the height of the smaller characters of the script, such as "a" and "e," and is not defined for all scripts, for example, Chinese. The baseline is defined as lowest point of the majority of the characters within a script, excluding those which are classified as *descenders*. Finally, the bottom-line is the absolute lowest point of any character within the script, for example the "g" and "q" characters. The three zones encompassed by these lines are known as the descender zone, x-zone, and ascender zones. Fig. 1 shows the locations of each of these lines and regions for an English text sample.

Calculating the positions of these lines is performed using vertical projection profiles of the connected components. By projecting the lowest position, top position, and pixels for each connected component, the positions of each line can then be determined. The peak in the lowest position profile is taken as the baseline position, since the majority of characters in any known language will have their lowest point here. Searching upwards from this point, the peak in the top position profile is then labeled as the x-height, although this may lead to inaccurate x-line positioning when lines of text with a large number of capital letters and/or punctuation symbols are present [2]. The positions of the top and bottom lines are found simply by searching for the highest and lowest projected positions, respectively, and removing any values which are excessive. Having determined the positions of these lines, they are then used as a reference point for all location information for individual connected components.

The primary feature used in the script recognition algorithm of Spitz is the location and frequency of upward concavities in the text image. An upward concavity is present at a particular location if two runs of black pixels appear on a single scan line of the image and there exists a run on the line below which spans the distance between these two runs.

Once found, the position of each upward concavity in relation to the baseline of the character is noted and a histogram of these positions for the entire document constructed. Analysis of such histograms for Latin-based languages shows a distinctly bimodal distribution, with the majority of upward concavities occurring either slightly above the baseline or slightly below the x-height line. In contrast to this, Han-based scripts exhibit a much more uniform distribution, with the modal value typically evenly spaced between the baseline and x-height. Using this information, it is possible to accurately distinguish Latin-based and Han-based scripts using a simple measure of variance. No histograms were provided for other script types such as Greek, Hebrew, or Cyrillic, so it is unknown how such a method will perform for these script types. In constant use, this method has never been observed to incorrectly classify Latin or Han-based scripts [7].

As well as this technique, a number of other approaches to automatic script recognition have been proposed. Hochberg used textual symbols extracted from individual characters to classify text regions, with template matching used to classify each such symbol [8]. Pal and Chaudhuri use properties of individual text lines to distinguish between five scripts, including the challenging Devanagari and Bangla scripts [9]. This approach uses a rule-based approach relying on extensive knowledge of the distinctive properties of these scripts, making unsuitable for applications where unknown scripts are to be added and recognized.

3 TEXTURE ANALYSIS FOR SCRIPT RECOGNITION

The work presented in the previous section has shown excellent results in identifying a limited number of script types in ideal conditions. In practice, however, such techniques have a number of disadvantages which in many cases make identification of some scripts difficult. The detection of upward concavities in an image is highly susceptible to noise and image quality, with poor quality and noisy images having high variances in these attributes. Experiments conducted on noisy, low-resolution, or degraded document images have shown that classification performance drops to below 70 percent for only two script classes of Latin-based and Han. The second disadvantage of the technique proposed by Spitz is that it cannot effectively discriminate between scripts with similar character shapes, such as Greek, Cyrillic, and Latin-based scripts, even though such scripts are easily visually distinguished by untrained observers.

Determination of script type from individual characters is also possible using OCR technology. This approach, as well as others which rely on the extraction of connected components, requires accurate segmentation of characters before their application, a task which becomes difficult for noisy, low resolution, or degraded images. Additionally, certain script types, for example, Sanskrit, do not lend themselves well to character segmentation, and require special processing. This presents a paradox in that to extract the characters for script identification, the script, in some cases, must already be known. Using global image characteristics to recognize the script type of a document image overcomes many of these limitations. Because it is not necessary to extract individual characters, no script-dependent processing is required. The effects of noise, image quality, and resolution are also limited to the extent that it impairs the visual appearance of a sample. In most cases, the script of the document can still be readily

preattentively determined by a human observer regardless of such factors, indicating that the overall texture of the image is maintained. For these reasons, texture analysis appears to be a good choice for the problem of script identification from document images.

Previous work in the use of texture analysis for script identification has been limited to the use of Gabor filterbanks [10], [11]. While this work has shown that texture can provide a good indication of the script type of a document image, other texture features may give better results for this task, and we investigate this possibility in this work.

4 PREPROCESSING OF IMAGES

In general, blocks of text extracted from typical document images are not good candidates for the extraction of texture features. The varying degrees of contrast in gray-scale images and the presence of skew and noise could all potentially affect such features, leading to higher classification error rates. Additionally, the large areas of white space, unequal character word and line spacings, and line heights can also have a significant effect on these features. In order to reduce the impact of these factors, the text blocks from which texture features are to be extracted must undergo a significant amount of preprocessing. The individual steps which are performed in this stage are binarization, deskewing, and block normalization.

The segmentation and extraction of text regions from a document image is a difficult problem which has received significant attention in the literature [12], [13], [14], [15], [16]. This stage of processing, however, is beyond the scope of this paper, and manual extraction of text regions is performed for all experiments. Such an approach is consistent with previous work in this field, with manual extraction text regions used by a number of authors [2], [11].

Binarization can be described as the process of converting a gray-scale image into one which contains only two distinct tones, that is black and white. This is an essential stage in many of the algorithms used in document analysis, especially those that identify connected components, that is, groups of pixels which are connected to form a single entity. Although document images are typically produced with a high level of contrast for ease of reading, scanning artifacts, noise, paper defects, colored regions, and other image characteristics can sometimes make this a nontrivial task, with many possible solutions presented to date in the literature. In general, a decision threshold is used to determine the final binary value of each pixel, with the actual threshold value determined either globally for the entire image [17], [18], [19] or, locally, for different regions of the image [20], [21], [22]. Iterative approaches to binarization, which allow for better performance in the presence of textured and other irregular background, have also been proposed [23].

For the purposes of this evaluation, all of the images used are of high contrast with no background shading effects. Because of this, a global thresholding approach provides an adequate means of binarization, and the method proposed by Otsu in [17] is used.

4.1 Skew Detection and Correction

Knowing the skew of a document is necessary for many document analysis tasks. Calculating projection profiles, for example, requires knowledge of the skew angle of the

image to a high precision in order to obtain an accurate result. In practical situations, the exact skew angle of a document is rarely known, as scanning errors, different page layouts, or even deliberate skewing of text can result in misalignment. In order to correct this, it is necessary to accurately determine the skew angle of a document image or of a specific region of the image, and, for this purpose, a number of techniques have been presented in the literature.

Postl [24] found that the maximum valued position in the Fourier spectrum of a document image corresponds to the angle of skew. However, this finding was limited to those documents that contained only a single line spacing, thus the peak was strongly localized around a single point. When variant line spacings are introduced, a series of Fourier spectrum maxima are created in a line that extends from the origin. Also evident is a subdominant line that lies at 90 degrees to the dominant line. This is due to character and word spacings and the strength of such a line varies with changes in language and script type. Peake and Tan expand on this method, breaking the document image into a number of small blocks, and calculating the dominant direction of each such block by finding the Fourier spectrum maxima [25]. These maximum values are then combined over all such blocks and a histogram formed. After smoothing, the maximum value of this histogram is chosen as the approximate skew angle. The exact skew angle is then calculated by taking the average of all values within a specified range of this approximate. There is some evidence that this technique is invariant to document layout and will still function even in the presence of images and other noise [25]. A number of other techniques for the estimation of the skew angle have also been proposed [26], [27], [28].

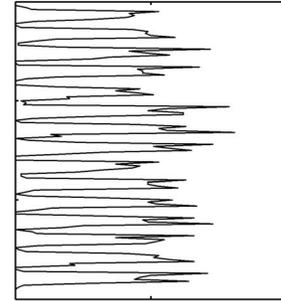
Expanding on the work of Peake and Tan, Lowther et al. use the Radon transform to accurately locate the peak of the Fourier spectrum of the document image [29]. In order to remove DC components and the higher weightings of the diagonals, a circular mask is first applied to the spectrum. Experimental results have shown that this technique provides superior accuracy in estimating the skew angle of a wide range of documents, with the correct skew angle of over 96 percent of test documents determined to within 0.25 degrees [29], and almost all within 1 degree. Because of these results, this technique has been used to detect and correct the skew of all text regions in each of the experiments outlined in Section 6.

4.2 Normalization of Text Blocks

Extraction of texture features from a document image requires that the input images exhibit particular properties. The images must be of the same size, resolution, orientation, and scale. Line and word spacing, character sizes and heights, and the amount of white space surrounding the text, if any, can also affect texture features. In order to minimize the effects of such variations to provide a robust texture estimate, our system attempts to normalize each text region before extracting texture features. This process will also remove text regions that are too small to be characterized adequately by texture features.

An effective algorithm for overcoming these problems and normalizing each region of text has been developed, based on the work done by Peake and Tan [10]. After binarization, deskewing, and segmentation of the document image, a number of operations are performed on each region in order to give it a uniform appearance.

A LONE Libyan bomber has been convicted of killing 270 people in the biggest mass-murder case in British history. Former Libyan intelligence agent and airline security chief Abdel Baset Ali Mohammed el-Megrahi, 48, was convicted of bombing Pan Am Flight 103 which crashed over Lockerbie in 1988. His co-accused Al Aminu Khalifa Fhimah, 44, a former Libyan Arab Airlines employee, was acquitted.



(a)

(b)

Fig. 2. Example of projection profile of text segment. (a) Original text and (b) projection profile.

First, horizontal projection profiles are taken for each segment. By detecting valleys in these profiles, the positions of line breaks, as well as the height of each line and line space is calculated, assuming that the text is correctly aligned following deskewing. An example of a typical projection profile obtained in this manner is shown in Fig. 2.

Having detected the lines of text, the average height of the lines in the region is then calculated and those that are either significantly larger or smaller than this average are discarded. Investigation of many regions has found that such lines often represent headings, captions, footnotes, or other nonstandard text, and as such may have an undesirable effect on the resulting texture features if they are retained. The remaining lines are then normalized by the following steps:

1. Each line is scaled to convert it to a standard height. Although 15 pixels has been found to provide good results in our experiments, larger or smaller values may be more appropriate in situations where the expected input resolution of the document images is significant higher or lower.
2. Normalization of character and word spacings. Often, modern word processing software expands spaces between words and even characters to completely fill a line of text on a page, leading to irregular and sometimes large areas of white space. Tabulation and other formatting techniques may also cause similar problems. By traversing the line and ensuring that each space does not exceed a specified distance (two-thirds of the standard height), this white space can be removed.
3. Removal and padding of short lines. After performing the above operations on each line, the length of the longest line is determined, and each of the others padded to extend them to this length to avoid large areas of white space at the ends of lines. To accomplish this, the line is repeated until the desired length is achieved. Clearly, for lines which are very short, such repetition may lead to peaks in the resulting spatial frequency spectrum of the final image, and hence lines which do not satisfy a minimum length, expressed as a percentage of the longest line, are simply removed.

Following normalization, lines must be recombined to construct the final block of text. When performing this stage of processing, it is important that the line spacings are constant to avoid significant white space between lines. Due to

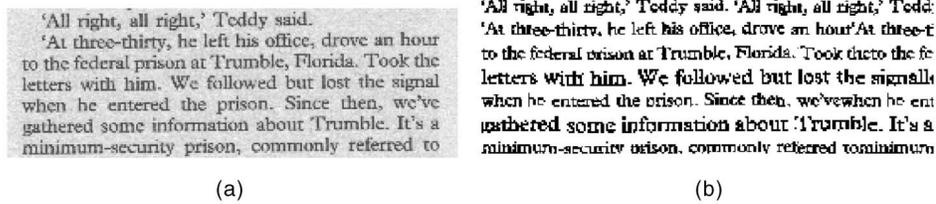


Fig. 3. Example of text normalization process on an English document image. (a) Original image and (b) normalized block.

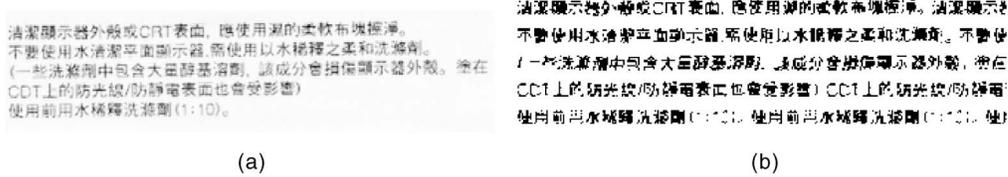


Fig. 4. Example of text normalization process on a Chinese document image. (a) Original image and (b) normalized block.

differences in the nature of various scripts, analysis of the line is required in order to determine the limits and relative frequencies of character heights. For example, Latin-based scripts comprise mostly of characters of a small height, such as “a,” “r,” and “s,” with a small but significant number of characters which protrude above and/or below these heights, for example, “A,” “h,” “p,” and “j.” In contrast to this, almost all characters in some other scripts, such as Chinese, have identical heights and, thus, require a somewhat larger line spacing in order to maintain a uniform appearance. Determination of which class a sample of text belongs to can be easily made by an examination of the projection profiles of each line. In order to obtain a uniform appearance over all script types, this information is taken into account when normalizing the line spacings. To allow for a more uniform appearance, samples of text with uniform or near-uniform character heights are combined using a larger line spacing.

An example showing the effect of the entire normalization process applied to a typical text segment is shown in Fig. 3. From this example, it can be seen that the original image, which is somewhat noisy and contains many large regions of whitespace, highly variable line spacing, and nonstandard text in form of equations, is transformed into a block of relatively uniform appearance. Closer inspection reveals the existence of repeated sections, however, preattentively this is not apparent. The algorithm described above works equally well on all tested scripts and languages, which is clearly an important property for this application. Fig. 4 shows the results obtained after processing a Chinese document image. Note in this example the increased line spacings due to the equal height of characters in the Chinese script.

5 TEXTURE FEATURE EXTRACTION

From each block of normalized text, the following texture features are evaluated for the purpose of script identification.

5.1 Gray-Level Co-Occurrence Matrix Features

Gray-level co-occurrence matrices (GLCMs) are used to represent the pairwise joint statistics of the pixels of an image and have been used for many years as a means of characterizing texture [30]. For a gray-scale image quantized to R discrete levels, such matrices contain $R \times R$ elements and can be defined for an image I as

$$P_d(i, j) = \frac{|\{(r, s), (t, v) : I(r, s) = i, I(t, v) = j\}|}{NM}, \quad (1)$$

where $|\cdot|$ represents the cardinality of a set. Due to the variable parameter d , the set of co-occurrence matrices is arbitrarily large. In practice, the most relevant correlations occur at short distances and, thus, the values of d are typically kept small, and expressed in the form (d, θ) , with d representing the lineal distance in pixels, and θ the angle between them. Typically, θ is restricted to the values $\{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$, and d limited to a small range of values. It is also possible to modify the GLCM somewhat to ensure diagonal symmetry of the matrix. This is achieved by the transformation

$$P_{(d, \theta)} = P_{(d, \theta)} + P_{(-d, \theta)}. \quad (2)$$

For a typical image with $R \geq 8$, the size of the resulting GLCMs makes their direct use unwieldy, and statistical features such as correlation, entropy, energy, and homogeneity are instead extracted and used to characterize the image [30]. Due to the binary nature of the document images from which the features are extracted, the extraction of such features is unnecessary and indeed counterproductive. Since there are only two gray levels, the matrices will be of size 2×2 , meaning that it is possible to fully describe each matrix with only three unique parameters due to the diagonal symmetry property. Using these values directly is feasible and has experimentally shown to give better results than attempting to extract the co-occurrence features of such a small matrix. Using values of $d = \{1, 2\}$ and $\theta = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$ leads to a total of 24 features.

5.2 Gabor Energy Features

The energy of the output of a bank of Gabor filters has been previously used as features for identifying the script of a document image, with good results shown for a small set of test images [10], [11]. In this work, both even and odd symmetric filters are used; they are described by

$$g_e(x, y) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)} \cos(2\pi\mu_0(x \cos \theta + y \sin \theta)) \quad (3)$$

$$g_o(x, y) = e^{-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)} \sin(2\pi\mu_0(x \cos \theta + y \sin \theta)), \quad (4)$$

where x and y are the spatial coordinates, μ_0 the frequency of the sinusoidal component of the Gabor filter, and σ_x and σ_y

the frequencies of the Gaussian envelope along the principal axes, typically with $\sigma_x = \sigma_y$. In the experimental results presented in [11], a single value of $\mu_0 = 16$ was used, with 16 orientation values spaced equidistantly between 0 and 2π , giving a total of 16 filters. By combining the energies of the outputs of the even and odd symmetric filters for each such orientation, a feature vector of same dimensionality is created. To obtain rotation invariance, this vector is transformed via the Fourier transform, and the first four resulting coefficients used for classification. Since skew detection and correction has been performed on the test images to be used in these experiments, such a transformation is not required, and the features will be used unmodified. By combining the energies of the odd and even symmetric filters for each resolution and orientation, a total of 16 features are obtained using this method. While these features have shown good performance on a small number of script types [11], using only a single frequency does not provide the necessary discrimination when a large set of scripts and fonts are used. To overcome this, an additional 16 filters with a frequency of $\mu_0 = 8$ are employed, giving a final dimensionality of 32.

5.3 Wavelet Energy Features

The wavelet transform has emerged over the last two decades as a formal, concise theory of signal decomposition and has been used to good effect in a wide range of disciplines and practical applications. A discrete, two-dimensional form of the transform can be defined as [31]

$$A_j = [H_x * [H_y * A_{j-1}]_{\downarrow 2,1}]_{\downarrow 1,2} \quad (5)$$

$$D_{j1} = [G_x * [H_y * A_{j-1}]_{\downarrow 2,1}]_{\downarrow 1,2} \quad (6)$$

$$D_{j2} = [H_x * [G_y * A_{j-1}]_{\downarrow 2,1}]_{\downarrow 1,2} \quad (7)$$

$$D_{j3} = [G_x * [G_y * A_{j-1}]_{\downarrow 2,1}]_{\downarrow 1,2}, \quad (8)$$

where A_j and D_{jk} are the approximation and detail coefficients at each resolution level j , H , and G are the low and high-pass filters, respectively, and $\downarrow_{x,y}$ represents downsampling along each axis by the given factors. The energies of each detail band of this transform, calculated by

$$E_{jk} = \frac{\sum_{m=1}^M \sum_{n=1}^N D_{jk}(m, n)}{MN}, \quad (9)$$

where M and N represent the size of each detail image, have been used by many authors as a texture feature vector [32], [33]. Although such features are relatively primitive in nature, their wide use and simple nature make them an ideal point of reference with which to compare more sophisticated approaches.

These features can be directly extracted from a region of normalized text, giving a total of $3J$ features, where J is the total number of decomposition levels used in the transform. In the evaluation conducted in this paper, a value of $J = 4$ is used, leading to a feature dimensionality of 12. The choice of analyzing is also of importance when extracting such features. Although the literature has presented results using a number of different analyzing wavelets, the family of biorthogonal spline wavelets [34] is a popular choice due to their symmetry, compact support and smoothness, and regularity properties. Previous work has shown that these wavelets perform well in texture characterization problems, and we have used a second-order wavelet of this type in all work presented in this paper [35].

5.4 Wavelet Log Mean Deviation Features

Previous work in the field of texture classification has shown that by applying a nonlinear function to the coefficients of the wavelet transform, a better representation of naturally textured images can be obtained [36]. By using a logarithmic transform and extracting the mean deviation of these values rather than the energy, significant improvements in overall classification accuracy were obtained when compared to the standard wavelet energy signatures, at negligible increase in computational cost. These features, named the wavelet log mean deviation features, are defined as [36]

$$LMD_{jk} = \frac{\sum_{m=1}^M \sum_{n=1}^N \log\left(\frac{|D_{jk}(n,m)|}{S_j \delta} + 1\right)}{MN}, \quad (10)$$

where δ is a constant specifying the degree of nonlinearity in the transform, and S_j represents the estimated maximum value of the coefficients at resolution level j . Although the optimal value of δ is high dependent upon the textures being used, previous work has found that a value of $\delta = 0.001$ performs well over a wide variety of natural textures. The total number of features obtained in this manner is equal to the wavelet energy features, thus when using four levels of decomposition a dimensionality of 12 is again obtained.

5.5 Wavelet Co-Occurrence Signatures

By extracting second-order features of the wavelet coefficients, represented by the co-occurrence features at small distances, it is possible to significantly improve the classification of natural textures [35]. Such features are extracted from each of the wavelet detail images in a similar manner to the GLCM features described previously, with linear quantization used to transform the near-continuous wavelet coefficients to discrete form. In order to avoid overly sparse matrices, an undecimated form of the wavelet transform is used in place of the standard two-dimensional FWT, providing greater spatial resolution, less sparse co-occurrence matrices at low resolutions, and translation invariance [37]. These features are known as the wavelet co-occurrence features.

The nonlinear transform described above can also be used when calculating the wavelet co-occurrence features by modifying the quantization function, with experimental results showing significantly reduced overall error rates when used to classify a variety of natural textures [36]. This is most easily achieved by modifying the quantization function $q(x)$ such that for a desired number of levels I ,

$$q_1(x) = \begin{cases} \text{round}\left[\kappa \log\left(\frac{x}{S_j \delta} + 1\right)\right], & x \geq 0 \\ -\text{round}\left[\kappa \log\left(\frac{|x|}{S_j \delta} + 1\right)\right], & x < 0, \end{cases} \quad (11)$$

where

$$\kappa = \frac{I - 1}{\log(1/\delta + 1)}. \quad (12)$$

Once the wavelet coefficients are quantized using (11), co-occurrence matrices are formed in an identical manner to the construction of the GLCMs described previously. From such matrices, the following co-occurrence features are extracted: energy, entropy, inertia, local homogeneity, contrast, cluster shade, cluster prominence, and information measure of correlation, as shown in Table 1 [30]. These features are known as the wavelet log co-occurrence features.

TABLE 1
Co-Occurrence Features Extracted from GLCMs

Co-occurrence features	Expression
Energy	$\sum_i \sum_j P^2(i, j)$
Entropy	$-\sum_i \sum_j P(i, j) \log P(i, j)$
Inertia	$\sum_i \sum_j (i - j)^2 P(i, j)$
Contrast	$\sum_i \sum_j P(i, j) i - j ^k, \quad k \in \mathbb{Z}$
Local Homogeneity	$\sum_i \sum_j \frac{1}{1 + (i - j)^2} P(i, j)$
Cluster Shade	$\sum_i \sum_j (i - M_x + j - M_y)^3 P(i, j)$
Cluster Prominence	$\sum_i \sum_j (i - M_x + j - M_y)^4 P(i, j)$
Inf. Measure of Correlation	$\frac{-\sum_i \sum_j P(i, j) \log P(i, j) - H_{xy}}{\max(H_x, H_y)}$

where $M_x = \sum_i \sum_j iP(i, j)$ and $M_y = \sum_i \sum_j jP(i, j)$

$$H_{xy} = -\sum_i \sum_j P(i, j) \log \left(\sum_j P(i, j) \cdot \sum_i P(i, j) \right)$$

$$H_x = -\sum_i \left\{ \sum_j P(i, j) \cdot \log \sum_j P(i, j) \right\}$$

$$H_y = -\sum_j \left\{ \sum_i P(i, j) \cdot \log \sum_i P(i, j) \right\}$$

Extracting these features for the first four resolution levels of the wavelet decomposition gives a total of 96 features for both the linear and logarithmic quantized cases.

5.6 Wavelet Scale Co-Occurrence Signatures

The wavelet scale co-occurrence signatures have been recently shown to provide unique texture information by describing the relationships between scales of the wavelet transform, allowing improved modeling of visual texture features which contain information on many scales and orientations [38]. A scale co-occurrence matrix is defined as [36]

$$S_{ji}(k, l) = \frac{|\{(u, v) : q_1(D_{ji}(u, v)) = k, q_2(A_j(u, v)) = l\}|}{NM}, \quad (13)$$

where $A_j(u, v)$ is the approximation image at resolution level j , $D_{ji}(u, v)$ are the three detail images, $q_1(x)$ and $q_2(x)$ are the quantization functions for the detail and approximation coefficients, respectively, and $(k, l) \in \{1 \dots I\}$, where I is the number of discrete quantization levels used. The logarithmic quantization function described in (11) is used for the detail coefficients, while linear quantization has been shown to be more suitable for the approximation data. From each of the scale co-occurrence matrices, a number of the features described in Table 1 are extracted. These features have been shown to perform well on a variety of naturally textured images, with lower overall classification errors when applied to some texture databases.

6 CLASSIFICATION RESULTS

The proposed algorithm for automatic script identification from document images was tested on a database containing eight different script types (Latin, Chinese, Japanese, Greek,

Cyrillic, Hebrew, Sanskrit, and Farsi). Examples of these images are shown in Fig. 5. Each such image was binarized, deskewed, and normalized using the algorithms described above, and 200 segments, each 64×64 pixels in size, extracted for each script class. This size sample corresponds to roughly four lines of printed text, typically, containing two or three words on each line. Although higher accuracies could be obtained by using larger areas or even complete regions, we have used these small regions to simulate situations where only a limited amount of text is available. The images obtained from this process were then divided into two equal groups to create the training and testing sets, ensuring that samples taken from the same image were placed into the same group.

In order to improve classification accuracy and reduce the dimensionality of the feature space, feature reduction is performed prior to classification by means of linear discriminate analysis [39]. This technique maps the feature space to one of lower dimensionality while maximizing the Fisher criterion, a measure of class separability. For a training set of N classes and a feature dimensionality of M , this analysis will return a $M \rightarrow (N - 1)$ mapping, representing the $N - 1$ hyperplanes necessary to segment the feature space linearly. To illustrate the effectiveness of this technique, the results obtained both with and without performing this step are shown in Table 2.

Classification of the samples is performed using a Gaussian mixture model (GMM) classifier, which attempts to model each class as a combination of Gaussian distributions in feature space [39], and is trained using a version of the expectation maximization (EM) algorithm. Due to the large range of possible scripts, a dynamic method of determining a classifier topology for each class is required. For this purpose, we have chosen to use the Bayes information criterion (BIC), which can be approximated for each candidate topology T_i by [40], [41], [42]

$$BIC(T_i) = \log p(\mathbf{X}_i | T_i, \hat{\lambda}_i) - \alpha \frac{K_i}{2} \log N_i, \quad (14)$$

where \mathbf{X}_i is the set of N training observations, $\hat{\lambda}_i$ the parametric form of the trained classifier, K the number of free parameters in the model, and α a scaling factor. From (14), it can be seen that this criterion is made up of the likelihood of the model given the training data minus a penalty factor which increases linearly with model complexity.

The overall classification error rates for each of the texture features are shown in Table 2. It can be seen that the wavelet log co-occurrence significantly outperform any of the other features for script classification, with an overall error rate of only 1 percent. This result is relatively consistent with those reported for natural textures [36], indicating that local relationships between wavelet coefficients are an excellent basis for representing texture. The relative increase in performance of these features compared to those extracted with linear quantization is again consistent with previously published results.

The scale co-occurrence features did not perform as well on the binary script images as has been previously reported for natural textures [38], with only a slightly reduced error rate when compared to the wavelet energy features. The GLCM features showed the worst overall performance, from which it can be concluded that pixel relationships at small distances are insufficient to characterize the script of a document image. The poor performance of the features proposed by Spitz can be attributed to the fact that they



Fig. 5. Examples of document images used for training and testing. (a) English, (b) Chinese, (c) Greek, (d) Cyrillic, (e) Hebrew, (f) Hindi, (g) Japanese, and (h) Persian.

were only designed to distinguish between Latin and Han-based scripts and cannot effectively discriminate script pairs such as Greek and Latin or Persian and Devangari.

Table 3 shows the distribution of the errors among the various script classes for the wavelet log co-occurrence features. In order to give a meaningful distribution, linear discriminate analysis was not used when generating these results. The Chinese script shows the lowest overall error rate for these features, with the largest errors arising from misclassifications between the Cyrillic and Greek scripts.

TABLE 2
Script Recognition Results for Each of the Feature Sets with and without Feature Reduction

Texture Features	Classification Error	
	no discriminate analysis	discriminate analysis
Spitz	48.7%	43.2%
GLCM Features	11.9%	9.1%
Gabor Energy	7.4%	4.9%
Wavelet Energy	7.9%	4.6%
Wavelet Log MD	8.3%	5.2%
Wavelet Coc.	5.0%	2.0%
Wavelet Log Coc.	4.9%	1.0%
Wavelet Scale Coc.	9.3%	3.2%

7 ADAPTIVE GMMs FOR IMPROVED CLASSIFIER PERFORMANCE

Printed text, regardless of the script, has a distinct visual texture and is easily recognized as such by a casual observer. With such a commonality between all script classes, it is possible to use this a priori knowledge to improve the modeling of each individual texture class. This is done by training a global model using all available training, then adapting this model for each individual class, rather than creating each model independently. By doing this, a more robust representation can be obtained, somewhat overcoming the blind nature of the learning algorithm. It is also possible to train a class using less training observations, since an initial starting point for the model is already available. This technique has been used with great success in applications where the general form of a model can be estimated using prior information, such as the modeling of speech and speakers, and is known as maximum a posterior (MAP) adaptation [43].

7.1 MAP Adaptation

When estimating the parametric form of a classifier, the maximum likelihood estimate is defined as the parameter set $\hat{\lambda}$ such that [44], [39]

$$\hat{\lambda} = \arg \max_{\lambda} l(\lambda), \tag{15}$$

TABLE 3
Confusion Matrix for the Wavelet Log Co-Occurrence Features

Classified	Actual Script							
	Latin	Chinese	Japanese	Greek	Cyrillic	Hebrew	Sanskrit	Farsi
Latin	96	0	0	2	2	0	0	0
Chinese	0	100	4	0	0	0	0	0
Japanese	0	0	95	1	0	0	0	1
Greek	3	0	0	92	5	0	0	0
Cyrillic	1	0	0	5	93	0	0	0
Hebrew	0	0	0	0	0	97	3	3
Sanskrit	0	0	1	0	0	3	94	3
Farsi	0	0	0	0	0	0	2	93
Error	4%	0%	5%	8%	7%	3%	5%	7%

where $l(\lambda)$ is the likelihood of the training observations for that parametric form λ defined as

$$l(\lambda) = p(\mathbf{o}|\lambda). \quad (16)$$

Given these definitions, the ML framework can be thought of as finding a *fixed* but *unknown* set of parameters $\hat{\lambda}$. In contrast to this, the maximum a posterior (MAP) approach assumes λ to be a *random* vector with a known distribution, with an assumed correlation between the training observations and the parameters λ [45]. From this assumption, it becomes possible to make a statistical inference of λ using only a small set of adaption data \mathbf{o} , and prior knowledge of the parameter density $g(\lambda)$. The MAP estimate therefore maximizes the posterior density such that

$$\lambda_{MAP} = \arg \max_{\lambda} g(\lambda|\mathbf{o}) \quad (17)$$

$$= \arg \max_{\lambda} l(\mathbf{o}|\lambda)g(\lambda). \quad (18)$$

Since the parameters of a prior density can also be estimated from an existing set of parameters λ_0 , the MAP framework also provides an optimal method of combining λ_0 with a new set of observations \mathbf{o} .

In the case of a Gaussian distribution, the MAP estimations of the mean \tilde{m} and variance $\tilde{\sigma}^2$ can be obtained using the framework presented above, given prior distributions of $g(m)$ and $g(\sigma^2)$, respectively. If the mean alone is to be estimated, this can be shown to be given by [46]

$$\tilde{m} = \frac{T\kappa^2}{\sigma^2 + T\kappa^2} \bar{x} + \frac{\sigma^2}{\sigma^2 + T\kappa^2} \mu, \quad (19)$$

where T is the total number of training observations, \bar{x} is the mean of those observations, and μ and κ^2 are the mean and variance, respectively, of the conjugate prior of m . From (19), it can be seen that the MAP estimate of the mean is a weighted average of the conjugate prior mean μ and the mean of the training observations. As $T \rightarrow 0$, this estimate will approach the prior μ , and as $T \rightarrow \infty$, it will approach \bar{x} , which is the ML estimate.

Using MAP to estimate the variance parameter, with a fixed mean, is accomplished in a somewhat simpler manner. Typically, a fixed prior density is used, such that

$$g(\sigma^2) = \begin{cases} \text{constant} & \sigma^2 \geq \sigma_{min}^2 \\ 0 & \text{otherwise,} \end{cases} \quad (20)$$

where σ_{min}^2 is estimated from a large number of observations, in the case of our application the entire database of script images. Given this simplified density function, the MAP estimate of the variance is then given by

$$\tilde{\sigma}^2 = \begin{cases} S_x & S_x \geq \sigma_{min}^2 \\ 0 & \text{otherwise,} \end{cases} \quad (21)$$

where S_x is the variance of the training observations. This procedure is often known as *variance clipping* and is effective in situations where limited training data does not allow for an adequate estimate of the variance parameter.

In the current application of script recognition, the prior parameter density $g(\lambda)$ can be estimated using a global model of script, trained using all available data. This choice is justified by the observation that printed text, in general, regardless of script type, has a somewhat unique appearance and as such the texture features obtained should be relatively well clustered in feature space. Training for each of the individual textures is then carried out by adapting this global model for each individual script class. In order to create more stable representations and limit computational expense, only the mean parameters and weights of these mixtures are adapted during this process, using (19).

7.2 Classification Results

Using the same training data as the previous experiment and the MAP approach outlined above, a global script model was created for each of the feature sets and adapted separately for each script class. The optimal number of mixtures for these models was again determined dynamically using the Bayes information criterion (BIC) described in Section 6. The overall classification results from this experiment are shown in Table 4. These results show a small improvement in overall classifier error when compared to those of Table 2, due to the more robust model obtained by utilizing prior information.

It is important to note that in these experiments a relatively large amount of training data (100 samples per class) is used, resulting in models which are stable and well-defined. In situations where less training data is available, it is expected that results will be somewhat poorer, and the benefit of using MAP adaptation to create a starting point

TABLE 4
Script Recognition Results for Various Feature Sets Using MAP Adaptation with Large Training Sets

Texture Features	Classification Error
GLCM Features	8.9%
Gabor Energy	4.5%
Wavelet Energy	4.2%
Wavelet Log MD	4.1%
Wavelet Cooc.	1.4%
Wavelet Log Cooc.	0.8%
Wavelet Scale Cooc.	3.0%

TABLE 5
Script Recognition Results with and without MAP Adaptation for Various Texture Features for Small Training Sets

Texture Features	Classification Error	
	no MAP adaptation	MAP adaptation
GLCM Features	12.1%	9.9%
Gabor Energy	7.8%	5.3%
Wavelet Energy	7.2%	5.8%
Wavelet Log MD	8.2%	6.1%
Wavelet Cooc.	3.3%	2.7%
Wavelet Log Cooc.	2.4%	1.3%
Wavelet Scale Cooc.	5.6%	4.0%

for each model more clearly illustrated. To test this hypothesis, the amount of training data was reduced to only 25 samples per class, and the experiment above repeated. The overall classification error rates obtained with and without using MAP are shown in Table 5. These results more clearly indicate the benefits of the MAP adaptation process, with error rates significantly reduced for each of the feature sets when compared to using models trained independently using the ML algorithm.

8 MULTIFONT SCRIPT RECOGNITION

Within a given script there typically exists a large number of fonts, often of widely varying appearance. Because of such variations, it is unlikely that a model trained on one set of fonts will consistently correctly identify an image of a previously unseen font of the same script. To overcome this limitation, it is necessary to ensure that an adequate amount of training observations from each font to be recognized are provided in order that a sufficiently complex model is developed.

In addition to requiring large amounts of training data, creating a model for each font type necessitates a high degree of user interaction, with a correspondingly higher chance of human error. In order to reduce this level of supervision, an ideal system would automatically identify the presence of multiple fonts in the training data and process this information as required.

8.1 Clustered LDA

The linear discriminate function described previously attempts to transform the feature space such that the interclass separation is maximized, while minimizing the intraclass separation, by finding the maximum of the cost function $tr(\mathbf{C}\mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{C}')$. While this function is optimal in this sense, it does make a number of strong assumptions regarding the nature of the distributions of each class in feature space. All classes are assumed to have equal covariance matrices, meaning that the resulting transform will be optimal only in the sense of separation of the class means. Additionally, since the function is linear, multimodal distributions cannot be adequately partitioned in some circumstances. Fig. 6 shows a simplistic synthetic example of this case, where two classes are clearly well separated in feature space, however have the same mean and therefore cannot be effectively separated by a linear discriminate function. When analyzing scripts containing multiple fonts, it is common to encounter such multimodal distributions in

feature space within a particular script, as the texture features extracted from different fonts can vary considerably due to the unique characteristics of each.

To overcome this limitation of LDA, we propose to perform automatic clustering on the data prior to determining the discriminate function, and assign a separate class label to each individual cluster. Training and classification is then performed on this extended set of classes, and the final decision mapped back to the original class set. Although this leads to less training data for each individual subclass, using the adaptation technique presented in the previous section can somewhat alleviate this problem.

The k-means clustering algorithm is a fast, unsupervised, nondeterministic, iterative method for generating a fixed number of disjoint clusters. Each data point is randomly assigned to one of k initial clusters, such that each cluster has approximately the same number of points. In each subsequent iteration of the algorithm, the distance from each point to each of the clusters is calculated using some metric, and moved into the cluster with the lowest such distance. Commonly used metrics are the Euclidian distance to the centroid of the clusters or a weighted distance which considers only the closest n points. The algorithm terminates when no points are moved in a single iteration. As the final result is highly dependent on the initialization of the clusters, the algorithm is often repeated a number of times, with each solution scored according to some evaluation function.

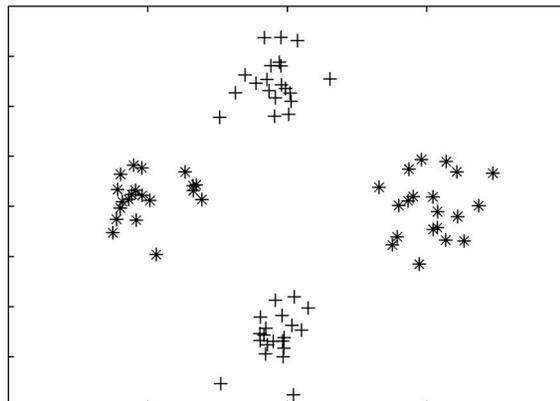


Fig. 6. Synthetic example of the limitations of LDA. The two multimodal distributions, although well separated in feature space, have identical means and, hence, an effective linear discriminate function cannot be determined.

TABLE 6

Script Recognition Error Rates for Scripts Containing Multiple Fonts When Trained with a Single Model

Texture Features	Classification Error
GLCM Features	15.9%
Gabor Energy	13.1%
Wavelet Energy	12.3%
Wavelet Log MD	12.6%
Wavelet Cooc.	14.9%
Wavelet Log Cooc.	13.2%
Wavelet Scale Cooc.	15.0%

Determining the optimal number of clusters is a problem which has been previously addressed in the literature [47], [48], [49]. However, for the purposes of multifont script recognition, using a fixed number of clusters has shown to provide adequate results at significantly reduced computational cost. In the experiments in the following section, 10 clusters are used in all cases, as this number was found to be generally sufficient to describe the font variations present within all of the tested scripts. Although the majority of classes can in fact be represented adequately using fewer than this number of clusters, using more clusters does not significantly degrade performance.

8.2 Classification Results

To illustrate the limitations of using a single model in a multifont environment, experiments using a number of fonts from each script class were conducted. A total of 30 fonts were present in the database, with 10 from Latin script, four each from Chinese, Japanese, and Persian, and three each from Sanskrit, Hebrew, Greek, and Cyrillic. 100 training and testing samples were extracted from each font type.

To illustrate the limitations of using a single model for multiple fonts, each of the scripts was trained as a single class using the MAP classification system proposed above. From the results shown in Table 6, it can be seen that large errors are introduced, with the most common misclassification occurring between fonts of the Latin and Greek scripts. Interestingly, these results show that the simpler texture features do not suffer the same performance degradation as the more complex features, with the wavelet energy signatures showing the lowest overall classification error of 12.3 percent.

The proposed clustering algorithm is implemented by using k-means clustering to partition each class into 10 regions. Each subclass is then assigned an individual label and LDA and classification performed as normal. The results of this experiment are shown in Table 7, with the wavelet log co-occurrence features again providing the lowest overall error rate of 2.1 percent. Although the error rates for each of the feature sets is slightly higher than the single font results of Table 5, a vast improvement is achieved when compared to the results obtained using a single model only.

9 CONCLUSIONS AND FUTURE WORK

This paper has shown the effectiveness of texture analysis techniques in the field of document processing and, more specifically, to the problem of automatic script identification.

TABLE 7

Script Recognition Error Rates for Scripts Containing Multiple Fonts When Clustering Is Used to Create Multiple Models

Texture Features	Classification Error
GLCM Features	12.5%
Gabor Energy	7.7%
Wavelet Energy	6.9%
Wavelet Log MD	7.0%
Wavelet Cooc.	3.2%
Wavelet Log Cooc.	2.1%
Wavelet Scale Cooc.	5.5%

A number of texture features were evaluated for the purpose of script recognition, including GLCM, Gabor filterbank energies, and a number of wavelet transform-based features. By using such features, it is not necessary to extract individual script components, making them ideal for degraded and noisy documents or situations where such segmentation is not possible. The amount of text required for accurate recognition is also quite small, with as little as five words sufficient in some cases. When classifying scripts containing a single font, experimental results have shown that texture features can outperform other script recognition techniques, with the wavelet log co-occurrence features giving the lowest overall classification error rate.

In order to provide more stable model of each script class, as well as reducing the need for excessive training data, a technique was proposed whereby MAP adaptation is used to create a global script model. Because of the strong interclass correlations which exist between the extracted features of script textures, this approach was found to be well-suited to the application of automatic script identification. Experimental results showed a small increase in overall classification performance when using large training sets, and significant improvement when limited training data is available.

Using a single model to characterize multiple fonts within a script class has been shown to be inadequate, as the fonts within a script class can vary considerably in appearance, often resulting in a multimodal distribution in feature space. To overcome this problem, a technique whereby each class is automatically segmented using the k-means clustering algorithm before performing LDA is presented. By doing this, a number of subclasses are automatically generated and trained without the need for any user intervention. Experiments performed on a multifont script database have shown that this technique can successfully identify scripts containing multiple fonts and styles.

REFERENCES

- [1] I. Bazzi, R. Schwartz, and J. Makhoul, "An Omnifont Open-Vocabulary OCR System for English and Arabic," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 495-504, June 1999.
- [2] A.L. Spitz, "Determination of the Script and Language Content of Document Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 235-245, Mar. 1997.
- [3] C. Suen, N.N. Bergler, B. Waked, C. Nadal, and A. Bloch, "Categorizing Document Images into Script and Language Classes," *Proc. Int'l Conf. Advances in Pattern Recognition*, pp. 297-306, 1998.

- [4] B. Julesz, "Visual Pattern Discrimination," *IRE Trans. Information Theory*, vol. 8, pp. 84-92, 1962.
- [5] C. Ronse and P. Devijver, *Connected Components in Binary Images: The Detection Problem*. Research Studies Press, 1984.
- [6] D.S. Lee, C.R. Nohl, and H.S. Baird, "Language Identification in Complex, Unoriented, and Degraded Document Images," *Proc. LAPR Workshop Document Analysis and Systems*, pp. 76-98, 1996.
- [7] A.L. Spitz and M. Ozaki, "Palace: A Multilingual Document Recognition System," *Proc. Int'l Assoc. for Pattern Recognition Workshop Document Analysis Systems*, pp. 16-37, 1995.
- [8] J. Hochberg, "Automatic Script Identification from Images Using Cluster-Based Templates," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 176-181, Feb. 1997.
- [9] U. Pal and B.B. Chaudhuri, "Automatic Identification of English, Chinese, Arabic Devnagari and Bangla Script Line," *Proc. Sixth Int'l Conf. Document Analysis and Recognition*, pp. 790-794, 2001.
- [10] G. Peake and T. Tan, "Script and Language Identification from Document Images," *Proc. Workshop Document Image Analysis*, vol. 1, pp. 10-17, 1997.
- [11] T. Tan, "Rotation Invariant Texture Features and Their Use in Automatic Script Identification," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 751-756, July 1998.
- [12] M. Acharyya and M.K. Kundu, "Document Image Segmentation Using Wavelet Scale-Space Features," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1117-1127, 2002.
- [13] P. Clark and M. Mirmedhi, "Combining Statistical Measures to Find Image Text Regions," *Proc. 15th Int'l Conf. Pattern Recognition*, pp. 450-453, 2000.
- [14] N. Jin and Y.Y. Tang, "Text Area Localization under Complex-Background Using Wavelet Decomposition," *Proc. Sixth Int'l Conf. Document Analysis and Recognition*, pp. 1126-1130, 2001.
- [15] H. Li, D. Doermann, and O. Kia, "Automatic Text Detection and Tracking in Digital Video," *IEEE Trans. Image Processing*, vol. 9, no. 1, pp. 147-156, 2000.
- [16] V. Wu, R. Manmatha, and E.M. Riseman, "Finding Text in Images," *Proc. Second ACM Int'l Conf. Digital Libraries*, 1997.
- [17] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62-66, 1979.
- [18] J. Kittler and J. Illingworth, "Minimum Error Thresholding," *Pattern Recognition*, vol. 19, pp. 41-47, 1986.
- [19] J.N. Kapur, P.K. Sahoo, and A.K. C. Wong, "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram," *Computer Vision, Graphics, and Image Processing*, vol. 29, pp. 273-285, 1985.
- [20] Y. Liu, R. Fenich, and S.N. Srihari, "An Object Attribute Thresholding Algorithm for Document Image Binarization," *Proc. Int'l Conf. Document Analysis and Recognition*, pp. 278-281, 1993.
- [21] J. Yang, Y. Chen, and W. Hsu, "Adaptive Thresholding Algorithm and Its Hardware Implementation," *Pattern Recognition Letters*, vol. 15, pp. 141-150, 1994.
- [22] J. Sauvola and M. Pietikainen, "Adaptive Document Image Binarization," *Pattern Recognition*, vol. 33, pp. 225-236, 2000.
- [23] Y. Liu and S.N. Srihari, "Document Image Binarization Based on Texture Features," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 540-544, May 1997.
- [24] W. Postl, "Detection of Linear Oblique Structures and Skew Scan in Digitized Documents," *Proc. Int'l Conf. Pattern Recognition*, pp. 687-689, 1986.
- [25] G. Peake and T. Tan, "A General Algorithm for Document Skew Angle Estimation," *Proc. Int'l Conf. Image Processing*, vol. 2, pp. 230-233, 1997.
- [26] B.B. Chaudhuri and U. Pal, "Skew Angle Detection of Digitized Indian Script Documents," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 703-712, Feb. 1997.
- [27] H.S. Baird, "The Skew Angle of Printed Documents," *Document Image Analysis*, L. O'Gorman and R. Kasturi, eds., IEEE CS Press, pp. 204-208, 1995.
- [28] A. Vailaya, H.J. Zhang, and A.K. Jain, "Automatic Image Orientation Detection," *Proc. Int'l Conf. Image Processing*, vol. 2, pp. 600-604, 1999.
- [29] S. Lowther, V. Chandran, and S. Sridharan, "An Accurate Method for Skew Determination in Document Images," *Digital Image Computing Techniques and Applications*, vol. 1, pp. 25-29, 2002.
- [30] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 3, pp. 610-621, 1973.
- [31] I. Daubechies, "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," *IEEE Trans. Information Theory*, vol. 36, pp. 961-1005, 1990.
- [32] T. Chang and C.C. Kuo, "Texture Segmentation with Tree-Structured Wavelet Transform," *Proc. IEEE Int'l Symp. Time-Frequency and Time-Scale Analysis*, vol. 2, p. 577 1992.
- [33] H. Greenspan, S. Belongie, R. Goodman, and P. Perona, "Rotation Invariant Texture Recognition Using a Steerable Pyramid," *Proc. 12th Int'l Conf. Pattern Recognition*, vol. 2, pp. 162-167, 1994.
- [34] M. Unser, A. Aldroubi, and M. Eden, "A Family of Polynomial Spline Wavelet Transforms," *Signal Processing*, vol. 30, pp. 141-162, 1993.
- [35] G. Van de Wouwer, P. Scheunders, and D. Van Dyck, "Statistical Texture Characterization from Discrete Wavelet Representations," *IEEE Trans. Image Processing*, vol. 8, no. 4, pp. 592-598, 1999.
- [36] A. Busch, W.W. Boles, and S. Sridharan, "Logarithmic Quantization of Wavelet Coefficients for Improved Texture Classification Performance," *IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, 2004.
- [37] S.G. Mallat, "Zero-Crossings of a Wavelet Transform," *IEEE Trans. Information Theory*, vol. 37, pp. 1019-1033, 1991.
- [38] A. Busch and W.W. Boles, "Texture Classification Using Wavelet Scale Relationships," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 4, pp. 3484-3487, 2002.
- [39] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*. New York: John Wiley & Sons, Inc., 2001.
- [40] R.E. Kass and A.E. Raftery, "Bayes Factors," *J. Am. Statistical Assoc.*, vol. 90, pp. 773-795, 1994.
- [41] J. Olivier and R. Baxter, "MML and Bayesianism: Similarities and Differences," Technical Report 206, Monash Univ., Australia, 1994.
- [42] G. Schwarz, "Estimating the Dimensionality of a Model," *Ann. Statistics*, vol. 6, no. 2, pp. 461-464, 1978.
- [43] D.A. Reynolds, "Comparison of Background Normalization Methods for Text-Independent Speaker Verification," *Proc. EUROSPEECH* vol. 2, pp. 963-970, 1997.
- [44] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, second ed. San Diego: Academic Press 1990.
- [45] C. Lee and J. Gauvain, "Bayesian Adaptive Learning and MAP Estimation of HMM," *Automatic Speech and Speaker Recognition: Advanced Topics*, Boston: Kluwer Academic, pp. 83-107, 1996.
- [46] C.-H. Lee, C.-H. Lin, and B.-H. Juang, "A Study on Speaker Adaptation of the Parameters of Continuous Density Hidden Markov Models," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 39, no. 4, pp. 806-814, 1991.
- [47] H.-S. Rhee and K.-W. Oh, "A Validity Measure for Fuzzy Clustering and Its Use in Selecting Optimal Number of Clusters," *Proc. Fifth IEEE Int'l Conf. Fuzzy Systems*, vol. 2, pp. 1020-1025, 1996.
- [48] K.S. Younis, M.P. DeSimio, and S.K. Rogers, "A New Algorithm for Detecting the Optimal Number of Substructures in the Data," *Proc. IEEE Aerospace and Electronis Conf.*, vol. 1, pp. 503-507, 1997.
- [49] I. Gath and A.B. Geva, "Unsupervised Optimal Fuzzy Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 773-780, July 1989.



Andrew Busch received a double degree in electronic engineering and information technology and the PhD degree in engineering from the Queensland University of Technology, Australia, in 1998 and 2004, respectively. He currently works as a lecturer within the school of Micro-electronic Engineering at Griffith University, Australia. His research interests include texture classification, multiresolution signal analysis, document analysis, and the use of imagery for biometric authentication. He is a member of IEEE.



Wageeh W. Boles is an associate professor at the School of Engineering Systems, Queensland University of Technology (QUT), Australia. Professor Boles obtained the BSc degree in electrical engineering from Assiut University, Egypt, and the MSc and the PhD degrees in electrical engineering from the University of Pittsburgh. He also obtained a graduate certificate in education (higher education) from QUT. He held the academic positions of assistant professor, Penn State University, then lecturer, senior lecturer, and associate professor at Queensland University of Technology. From 1999 to 2004, he held the position of assistant dean (Teaching and Learning), at the Faculty of Built Environment and Engineering, QUT. Professor Boles has been successful in obtaining numerous competitive research and teaching development grants and has more than 100 publications. He initiated and maintained a unique, active, and pioneering research effort in developing new image processing techniques and adopting them to various applications such as biometric human identification using iris and palm images, object recognition, and texture analysis. He is very passionate about his teaching and has published in the areas of work integrated learning and the study and utilization of learners' cognitive styles in the design and implementation of computer-based learning solutions. Professor Boles has been awarded two Outstanding Teaching Assistant Medals from the University of Pittsburgh, in 1987/1988. In 1999, he received the QUT Award for Outstanding Academic Contribution in Teaching and Leadership, the Faculty's Teaching Excellence Award, and was one of only three University nominees to the Australian Awards for University Teaching. He also won the national Engineers Australia—Australasian Association for Engineering Education (EA-AAEE) Award for Excellence in Teaching and Learning in Engineering Education, in 2004. Professor Boles is a member of the Executive of the Australasian Association for Engineering Education, AaeE, since 2001 and a member of the IEEE since 1984. He is also a member of the IEEE Computer Society.



Sridha Sridharan received the BSc (electrical engineering) degree and received the MSc (communication engineering) degree from the University of Manchester Institute of Science and Technology (UMIST), United Kingdom, and the PhD degree in the area of signal processing from University of New South Wales, Australia. He is a fellow of the Institution of Engineers, Australia, a senior member of the IEEE and the chairman of the IEEE Queensland Chapter in Signal Processing and Communication. He is currently with the Queensland University of Technology (QUT) where he is a professor in the School of Electrical and Electronic Systems Engineering. Professor Sridharan is the leader of the Research Program in Speech, Audio, Image, and Video Technologies (SAIVT) at QUT and is a deputy director of the Information Security Institute (ISI) at QUT. In 1997, he was the recipient of the award of Outstanding Academic of QUT in the area of research and scholarship.

► **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.**