SIViP (2015) 9 (Suppl 1):S225–S234
DOI 10.1007/s11760-015-0795-z

ORIGINAL PAPER

Classifying fonts and calligraphy styles using complex wavelet transform

Alican Bozkurt[1] · Pinar Duygulu[2] · A. Enis Cetin[3]

Received: 2 April 2015 / Revised: 15 June 2015 / Accepted: 20 June 2015 / Published online: 23 July 2015

Abstract Recognizing fonts has become an important task in document analysis, due to the increasing number of available digital documents in different fonts and emphases. A generic font recognition system independent of language, script and content is desirable for processing various types of documents. At the same time, categorizing calligraphy styles in handwritten manuscripts is important for paleographic analysis, but has not been studied sufficiently in the literature. We address the font recognition problem as analysis and categorization of textures. We extract features using complex wavelet transform and use support vector machines for classification. Extensive experimental evaluations on different datasets in four languages and comparisons with state-of-the-art studies show that our proposed method achieves higher recognition accuracy while being computationally simpler. Furthermore, on a new dataset generated from Ottoman manuscripts, we show that the proposed method can also be used for categorizing Ottoman calligraphy with high accuracy.

Keywords Font recognition · Ottoman calligraphy · Dual tree complex wavelet transform · SVM · Latin · Arabic · Chinese

Electronic supplementary material The online version of this article (doi:10.1007/s11760-015-0795-z) contains supplementary material, which is available to authorized users.

✉ Alican Bozkurt
alican@ece.neu.edu

[1] Department of Electrical and Computer Engineering, Northeastern University, 360 Huntington avenue, Boston, MA 02115, USA

[2] Department of Computer Science, Hacettepe University, Beytepe Campus, 06800 Ankara, Turkey

[3] Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

1 Introduction

The term font generally refers to a document's typeface, such as Arial. Each font can have variations such as bold or italic to emphasize the text, which are called emphases. Font recognition, which is the process of classifying different forms of letters, is an important issue in document analysis especially in multi-font documents [20,32]. In addition to its advantages in capturing document layout, font recognition may also help to increase the performance of optical character recognition (OCR) systems by reducing the variability of shape and size of the characters.

For printed fonts in languages that use the Latin alphabet, the main challenge is to recognize fonts in "noisy" documents, that is, those containing many artifacts. When we consider languages with cursive scripts, such as Arabic, the change in character shape with location (isolated, initial, medial or final) and dots (diacritics) above or below the letters cause further difficulties in character segmentation.

Handwritten documents add an extra component to the analysis because of writing style. Classifying handwriting styles continues to be a challenging yet important problem for paleographic analysis [3]. In recent studies of Hebrew, Chinese and Arabic calligraphy, researchers used characters as the basic elements to extract features [6,35,38]. However, these methods heavily rely on preprocessing steps and are prone to error.

Another example which we focus, the style of Ottoman calligraphy, is the artistic handwriting style of Ottoman Turkish. Different styles were used in different documents, such as books, letters, etc. [29]. Scholars around the world want to access efficiently and effectively to Ottoman archives, which contain millions of documents. Classifying Ottoman calligraphy styles would be an important step in categorizing large numbers of documents in archives, as it would assist further processing for retrieval, browsing and transliteration.

The Ottoman alphabet is similar to Farsi and Arabic. Hence, to recognize multi-font printed texts in Ottoman [25], existing methods of Arabic and Farsi font recognition can be utilized. However, due to the late adoption of printing technology in the Ottoman Empire, a high percentage of documents are handwritten. Documents in Ottoman calligraphy are very challenging compared to their printed counterparts, with intra-class variances much higher than what is found in printed documents. Some documents are hundreds of years old and the non-optimal storage conditions of historic Ottoman archives have resulted in highly degraded manuscripts. So as not to damage the binding, books are scanned with their pages only partially open, introducing non-uniform lighting in images.

We propose a simple but effective method for recognizing printed fonts independent of language or alphabet, and extend it to classifying handwritten calligraphy styles in Ottoman manuscripts. We present a new method based on analyzing textural features extracted from text blocks, which therefore does not require complicated preprocessing steps such as connected component analysis or segmentation. While Gabor filters are commonly used in the literature for texture analysis, they are computationally costly [20]. Alternatively, we propose using complex wavelet transform (CWT), which is not only more efficient but also achieves better recognition rates compared to other methods.

Unlike most existing studies focusing on a single language, we experiment on many previously studied printed fonts in four languages: English, Farsi, Arabic and Chinese. Our method also yields high accuracy in categorizing Ottoman calligraphy styles on a newly generated dataset consisting of various samples of different handwritten Ottoman calligraphy styles.

2 Related work

One of the first systems of optical font recognition was [39], in which global typographical features were extracted and classified by a multivariate Bayesian classifier. The authors extracted eight global features from connected components. They experimented on ten typefaces in seven sizes and four emphases in printed and scanned English documents. Ozturk et al. proposed a cluster based approach for printed fonts and exploited recognition performance for quality analysis [26].

Feature extraction methods for font recognition in the literature are divided into two basic approaches: local and global. Local features, usually refer to the typographical information gained from parts of individual letters, and are utilized in [15,33]. Local feature extraction relies on character segmentation, requiring the documents to be noise free and scanned in high resolution. Global features refer to information extracted from entire words, lines or pages, and are mostly texture based [2,4,10,20,32].

Zhu et al. addressed the font recognition problem as a texture identification issue and used multichannel Gabor filters to extract features. Prior to feature extraction, the authors normalized the documents to create uniform blocks of text. Experimental results were reported on computer-generated images with 24 Chinese (six typefaces and four emphases) and 32 English (eight typefaces and four emphases) fonts. Pepper and salt noise was added to generate artificial noise. Gabor filters were also used in [30] for feature extraction, and support vector machines (SVM) for classification. Experiments were carried out on six typefaces with four emphases on English documents. Similarly, Ma and Doermann used Gabor filters not for font but for script identification at word level. The authors used three different classifiers: SVM, k-nearest neighbor and the Gaussian mixture model (GMM), to identify the scripts in four different bilingual dictionaries (Arabic-English, Chinese-English, Hindi-English, Korean-English). The method was also used to classify Arial and Times New Roman fonts [23]. Aviles-Cruz et al. used high-order statistical moments to characterize the textures, and Bayes classifier for classification. Similar to [37], they experimented on Spanish documents digitally generated with eight fonts and four emphases. They also tested the effects of Gaussian random noise in [5].

Khosravi and Kabir approached the Farsi font recognition problem, and instead of using Gabor filter-based method, proposed a gradient-based approach to reduce the computational complexity. They combined Sobel and Roberts operators to extract the gradients and used AdaBoost for classification. In [31], Sobel-Robert operator-based features were combined with wavelet-based features. Another method, based on matching interest points is proposed in [36].

For Arabic font recognition, method based on fractal geometry is proposed in [8,9], where the authors generated a dataset consisting of printed documents in ten typefaces. Slimane et al. proposed a method for recognizing fonts and sizes in Arabic word images at an ultra-low resolution. They use GMM to model the likelihoods of large numbers of features extracted from gray level and binary images [32]. Bataineh et al. considered statistical analysis of edge pixel behavior in binary images for feature extraction from Arabic calligraphic scripts. They experimented on Kufi, Diwani, Persian, Roqaa, Thuluth and Naskh styles [7].

For classifying calligraphy styles, Yosef et al. presented a writer-identification method based on extracting geometric parameters from three letters in Hebrew calligraphy documents, followed by dimension reduction [35]. Azmi et al. proposed using triangle blocks for classifying Arabic calligraphy. In this method, triangles are formed using tangent values and gray-level occurrence matrices extracted from

Table 1 (L)ow and (H)igh pass (K)ingsbury and (F)arras coefficients

L_K^n	0	−0.0884	0.0884	0.6959	0.6959	0.0884	−0.0884	0.0112	0.0112	
H_K^n	0.0112	0.0112	−0.0884	0.0884	0.6959	0.6959	0.0884	−0.0884	0	0
L_F^n	0.0351	0	−0.0883	0.2339	0.7603	0.5875	0	−0.1143	0	0
H_F^n	0	0	−0.1143	0	0.5875	0.7603	0.2339	−0.0883	0	0.0351

individual characters [6]. Zhuang et al. introduced a generative probabilistic model for automatically extracting a presentation in calligraphic style for Chinese calligraphy works. The authors created a latent style model based on the latent Dirichlet allocation model [38].

3 Proposed method

We propose a new method to categorize writing styles, applicable to printed fonts and calligraphy styles. Given a text in a particular language, our goal is to classify the writing style as one of the known categories. We consider documents as textures, and use CWT, which has the ability to capture directional features at various angles and scales in a computationally efficient manner. We describe the details in the following.

3.1 Preprocessing

The input is a gray-level image of the text to be classified and that empty margins are cropped. Since the focus of this study is to classify the fonts, but not the layout extraction, text areas are cropped manually.

The proposed method has the ability to work on multi-font documents, and it can also be used for segmenting parts in different fonts. It is capable of detecting and discarding empty areas, which is achieved by block processing. A binarized document image is divided into blocks, and a block is marked as "empty" if the ratio of black pixels to white pixels is below a certain threshold, meaning that there is not sufficient text in that block. By properly choosing the block size, blocks can be arranged to include minimal heterogeneous text (in terms of font) as much as possible. Choosing block size is further discussed in Sect. 4.3. We use Otsu's method [24] for binarization, which is observed to be effective for the documents we experiment with. Non-empty blocks are fed to a feature extraction process.

It is common to do a normalization to fix the font sizes before performing feature extraction. This process usually requires line and character length prediction, with projection profiles commonly used for this purpose. After the normalization step, most studies use space filling to generate uniform text blocks [5]. The normalization step was not required for our study: we generated artificial datasets with a fixed font size, and Ottoman documents have relatively similar font sizes.

3.2 Complex wavelet transform for feature extraction

For feature extraction, we use CWT [13,17,18,28]. Ordinary discrete wavelet transform (DWT) is not as reliable as CWT when modeling textural features, because the former is shift-variant and only sensitive to horizontal (0°) and vertical (90°) directions. On the other hand, dual-tree complex wavelet transform (DT-CWT), proposed in [22], is almost shift-invariant, directionally selective at angles ±15°, ±45° and ±75°, and has perfect reconstruction capability. It introduces minimal redundancy (4:1 for images) and has a computational complexity of $\mathcal{O}(N)$ [22]. In this work, we use Farras filters [1] and the six-tap filter designed in [21] (see Table 1). Note that L_K^n, H_K^n, L_F^n and H_F^n only show the real parts of the filters. Imaginary parts of the filters can be calculated by reversing and taking the negative of each respective filter [1,21]. A dual tree is constructed from L_K^n, H_K^n, L_F^n, and H_F^n. Using this tree it is possible to decompose the input image into directional sub-bands with CWT (see Fig. 1). After a single stage of DT-CWT image decomposition, the image is decomposed into directional sub-bands with orientations of ±15°, 45° and 75°.

Since DT-CWT produces output images with different sizes at each tree level due to decimation, and these sizes depend on the input image size, it is not feasible to use output images of DT-CWT directly. Instead, we use statistical features of outputs of the complex wavelet tree, that is, the first and the second moments (i.e., mean and variance), because they are computationally more efficient and more robust to noise than higher-order moments. In experimenting with several levels of the complex wavelet tree, we find that recognition rate does not increase noticeably after three

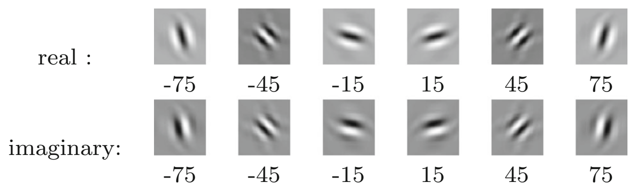

Fig. 1 Activations of CWT at level 4

level trees. Overall, our feature vector includes mean and variance values of 18 output images (six outputs per level of a three-level complex wavelet tree), resulting in a 36-element feature vector. We use the MATLAB implementation of DT-CWT given by [12].

3.3 Classification

We use SVMs [16] for classification. The radial basis function (RBF) is used as the kernel function. As discussed in [19], RBF nonlinearly maps samples into a higher-dimensional space, which makes it very efficient for problems with small numbers of features, as in our case. We used LIBSVM [14]. The parameters of SVM and the RBF kernel, C and γ, are optimized by a grid search and cross-validation on the training set. The search range is $[1 - 10^6]$ for C and $[10^{-6} - 1]$ for γ.

4 Experimental results

To compare our method with state-of-the-art studies, we use the available datasets provided by other studies and generate artificial datasets of our own. For English, we created lorem ipsum texts with fonts used in [5]. In a similar fashion, we create paragraphs with fonts used in [37] for Chinese, and with fonts used in [20] for Farsi. For Arabic, we perform our experiments on a publicly available dataset, used in [9]. In addition, we construct a new dataset of Ottoman calligraphy by scanning pages from Ottoman documents written in different calligraphy styles.

Although there are differences between the Arabic, Farsi and Ottoman languages, they use the same alphabet, with small variations. We use different datasets for Farsi and Arabic to compare our method with state-of-the-art methods for Farsi [20] and Arabic [8]. For instance, we use "Farsi texts" in this section to indicate the texts created with fonts used in [20], not as a general term for all documents in Farsi. The Ottoman dataset is fundamentally different from the Arabic and Farsi datasets because the former is in a handwritten form, whereas the latter two are created digitally. The procedure is called "calligraphy style recognition" for the Ottoman dataset, and "font recognition" for the others to emphasize the difference.

In the following, we first present the experimental results for recognize printed fonts in each dataset separately. Then, we show that our method is also capable of recognizing fonts in multi-font documents and that it can categorize fonts in multiple languages without a noticeable decrease in performance. We then present the results of the calligraphy style recognition. Finally, we provide a detailed analysis of our method.

4.1 Font recognition

This section presents the results for the English, Chinese, Farsi and Arabic documents with printed fonts. Descriptions of the datasets are followed by experimental evaluations of that dataset. The accuracy of the model for recognizing different fonts is computed over a 10-fold cross-validation. The optimization of SVM parameters is performed only on training set in order not to overfit and get unfairly high recognition rates.

4.1.1 Dataset 1—english texts

To test the noise performance of our method, we constructed three different English datasets. The first dataset, called "noise-free", consists of saved pages of English lorem ipsum texts typed in eight typefaces (Arial, Bookman, Courier, Century Gothic, Comic Sans MS, Impact, Modern, Times New Roman), and in four emphases (regular, italic, bold,

Table 2 Recognition rates (%) of the proposed method, and comparisons with the methods of Bozkurt et al., Aviles-Cruz et al. and Ramanathan et al. on English datasets

Font	Noise-free				Low noise				High-noise			
	Ours	[11]	[5]	[30]	Ours	[11]	[5]	[30]	Ours	[11]	[5]	[30]
Arial	**100**	**100**	**100**	**100**	96.90	96.30	81.80	**100**	**98.44**	94.45	–	91.70
Bookman	**100**	98.38	**100**	**100**	**100**	94.91	87.00	**100**	**98.40**	97.45	–	88.90
Century gothic	**100**	97.92	**100**	**100**	**98.50**	95.83	69.80	97.20	92.20	89.58	–	**94.40**
Comic sans	**100**	98.89	**100**	**100**	**100**	95.42	75.50	**100**	**100**	97.22	–	97.20
Courier new	**100**	**100**	**100**	**100**	**100**	97.78	96.30	**100**	**100**	95.56	–	94.40
Impact	**100**	**100**	**100**	**100**	**100**	**100**	99.00	**100**	**100**	**100**	–	94.40
Computer modern	**100**	**100**	**100**	**100**	**100**	89.82	97.00	**100**	**98.40**	93.48	–	88.90
Times new roman	**100**	98.15	**100**	**100**	**100**	97.22	91.00	**100**	98.44	89.82	–	**100**
Mean	**100**	99.17	**100**	**100**	99.40	95.91	87.20	**99.70**	**98.20**	94.69	-	93.80

The highest accuracies are shown in bold

bold-italic). The term "noise-free" means that no noise is introduced in generating the texts or saving them as images. This set is used for validating and comparing methods in an ideal case. We created noisy versions of the same texts by printing and scanning the pages in 200 dpi, as done in [5], using a Gestetner MP 7500 printer/scanner. This process introduced a small amount of noise to the images, hence we called the dataset created from these pages "low noise". We constructed the third dataset by photocopying and scanning the texts 10 times in succession, which resulted in a clear degradation of image quality and introduced a large number of artifacts. This is an approximation of a worn-out document, and called "high noise". The average signal-to-noise ratio (SNR) is 8.25 for the low-noise dataset and 7.41 for the high-noise dataset. These values are calculated by registering each image in the noisy dataset to the corresponding image in the noise-free dataset. The noise-free image is then subtracted from the registered image to extract the noise image.

We compared our proposed method with the methods in [5] and in [30] on the three datasets described above. We present the results in Table 2, with overall accuracy calculated over the four emphases for each font. In "noise-free", all methods classify all fonts perfectly. As noise is introduced, our method begins to outperform the other methods in almost all fonts. Gabor filters' performance [30] is close to CWT because both methods employ directional filters at various angles. In the "high-noise" dataset, because [5] uses horizontal and vertical projections in the preprocessing stage, the noise and artifacts in the images prevent the method from segmenting the words properly and the method cannot provide any result. Our method suffers only a 1.8 % decrease in average accuracy. It is also possible to use directional filters introduced in [11] instead of complex wavelets in our framework. Results for directional filters are also provided in Table 2.

4.1.2 Dataset 2—Chinese texts

Since the proposed method uses textural information, which is not dependent on a specific language, there is no limitation in the language or the alphabet selection regarding the method applied. We show this advantage of our method by testing it on Chinese texts. We used four different emphases (regular, italic, bold, bold-italic) for six different fonts used in [37]: SongTi, KaiTi, HeiTi, FangSong, LiShu and YouYuan. This set can be considered the Chinese equivalent of noise-free English set.

We compare the proposed method to the Chinese font recognition methods described in [37] and [34]. Gabor features are used in [37] and characters' stroke features are used in [34]. Recognition rates of the methods for each font and style are presented in Table 3. An average performance of

Table 3 Recognition accuracy of the proposed method compared with methods of Yang et al. and Zhu et al. on the Chinese dataset

	SongTi			KaiTi			HeiTi			FangSong			LiShu			YouYuan		
	Ours	[34]	[37]	Ours	[34]	[37]	Ours	[34]	[37]	Ours	[34]	[37]	Ours	[34]	[37]	Ours	[34]	[37]
Regular	**100**	98.20	**100**	**100**	**100**	98.40	92.86	93.20	**100**	**100**	**100**	92.80	**100**	**100**	**100**	**100**	**100**	99.60
Bold	**100**	93.20	**100**	95.24	91.70	**98.40**	**100**	**100**	**100**	95.24	92.90	**100**	**100**	**100**	**100**	**100**	96.50	99.60
Italic	**100**	**100**	97.60	95.24	**100**	89.60	**100**	96.50	**100**	**100**	93.30	94.00	92.86	**100**	**100**	**100**	**100**	**100**
Bold-italic	**100**	**100**	99.20	**100**	93.30	**100**	**100**	**100**	**100**	**100**	91.70	96.80	**100**	96.50	**100**	**100**	94.90	**100**
Mean	**100**	97.85	99.20	**97.62**	96.25	96.60	98.21	97.42	**100**	**98.81**	94.47	95.90	98.21	99.12	**100**	**100**	97.85	99.80

The highest accuracies for each font are shown in bold

Table 4 Recognition rates (%) of the proposed method and comparisons with [20] and [31] for Farsi texts

	Ours	[20]	[31]
Lotus	**92.2**	**92.2**	90.7
Mitra	**95.3**	93.4	93.7
Nazanin	90.6	85.2	**92.0**
Traffic	**98.4**	97.6	95.9
Yaghut	96.9	97.6	**98.5**
Zar	**92.2**	87.4	90.9
Homa	**100**	99.2	99.8
Titr	**100**	95.2	97.0
Tahoma	**100**	96.6	98.3
Times	98.4	97.2	**98.8**
Mean	**96.41**	94.16	95.56

Table 5 Recognition accuracy of the proposed method (96 × 160 sample size) and the method of [9] on Arabic text

	Ours	[9]
Ahsa	**99.63**	94.00
Andalus	**98.77**	94.00
Arabictransparant	**99.82**	92.00
Badr	99.44	**100**
Buryidah	98.30	**100**
Dammam	99.95	**100**
Hada	90.39	**100**
Kharj	**90.35**	88.00
Koufi	**99.35**	98.00
Naskh	**98.57**	98.00
Mean	**97.46**	96.40

97.16 % was reported in [34] for the six fonts, and a 98.58 % average performance was obtained in [37]. Our method has the highest overall recognition accuracy 98.81 %.

4.1.3 Dataset 3—farsi texts

The Ottoman alphabet is similar in nature to the Farsi alphabet. To compare the performance of the proposed method against [20], we replicated the dataset used in [20], which consists of scanned pages of Farsi lorem ipsum paragraphs written in four different emphases (regular, italic, bold, bold-italic) in ten different fonts: Homa, Lotus, Mitra, Nazanin, Tahoma, Times New Roman, Titr, Traffic, Yaghut, and Zar. Only regular and italic are used for Titr, because bold emphasis is not available for it.

Table 4 presents accuracies of the proposed method for Farsi texts and comparisons with [20] and [31]. Overall, our method performs better than the others.

4.1.4 Dataset 4—arabic texts

The ALPH-REGIM dataset for printed Arabic scripts, provided by [8], consists of text snippets of various fonts, sizes, and lengths. We use ten typefaces which are also used in [8]: Ahsa, Andalus, Arabictransparant, Badr, Buryidah, Dammam, Hada, Kharj, Koufi and Naskh.

We compare our method with the work of Ben Moussa et al. [9] for the ALPH-REGIM dataset. Since this dataset contains images with various sizes, block size must be chosen accordingly. The smallest image in the dataset is a 100×1322 image containing two lines. Thus, we choose a block size of 96×160, to efficiently sample every image in the dataset. For six out of ten fonts, our method results in much better performances than the other methods (see Table 5). Our method also performs better when mean accuracy is considered.

4.1.5 Font recognition in documents with multiple fonts

To show that our method can also handle documents with multiple fonts, samples from the “noise-free English” dataset with 96×96 sized blocks were used to classify a collage of texts written in different fonts (Fig. 2).

4.1.6 Font recognition in the combined font dataset

We considered all four datasets combined, resulting in a large dataset with 104 classes. We choose a block size of 96×160. The average recognition accuracy of our method for each dataset, for both the single dataset case and the combined dataset case, are presented in Table 6. The results show that the fonts of different alphabets do not overlap with each other in the proposed feature space and can be classified with a SVM without a noticeable decrease in performance.

4.2 Calligraphy style recognition

We show that the proposed method is capable of categorizing calligraphy styles as well as printed forms through our experiments on a dataset generated from handwritten Ottoman manuscripts. To the best of our knowledge, automatic classification of Ottoman calligraphy has not been studied. We created a new dataset from documents written in Ottoman calligraphy by scanning 30 pages in five different styles: divani, matbu, nesih, rika, and talik. Example documents from this dataset are presented in Fig. 3.

Ottoman documents have some characteristic page layouts that, together with calligraphy styles, is usually specific to a form of manuscript. These distinct layouts help with high accuracy discrimination. To normalize this layout difference, we created an additional set uniform in terms of layout. The areas that contain text are stitched together to eliminate empty spaces in the document.

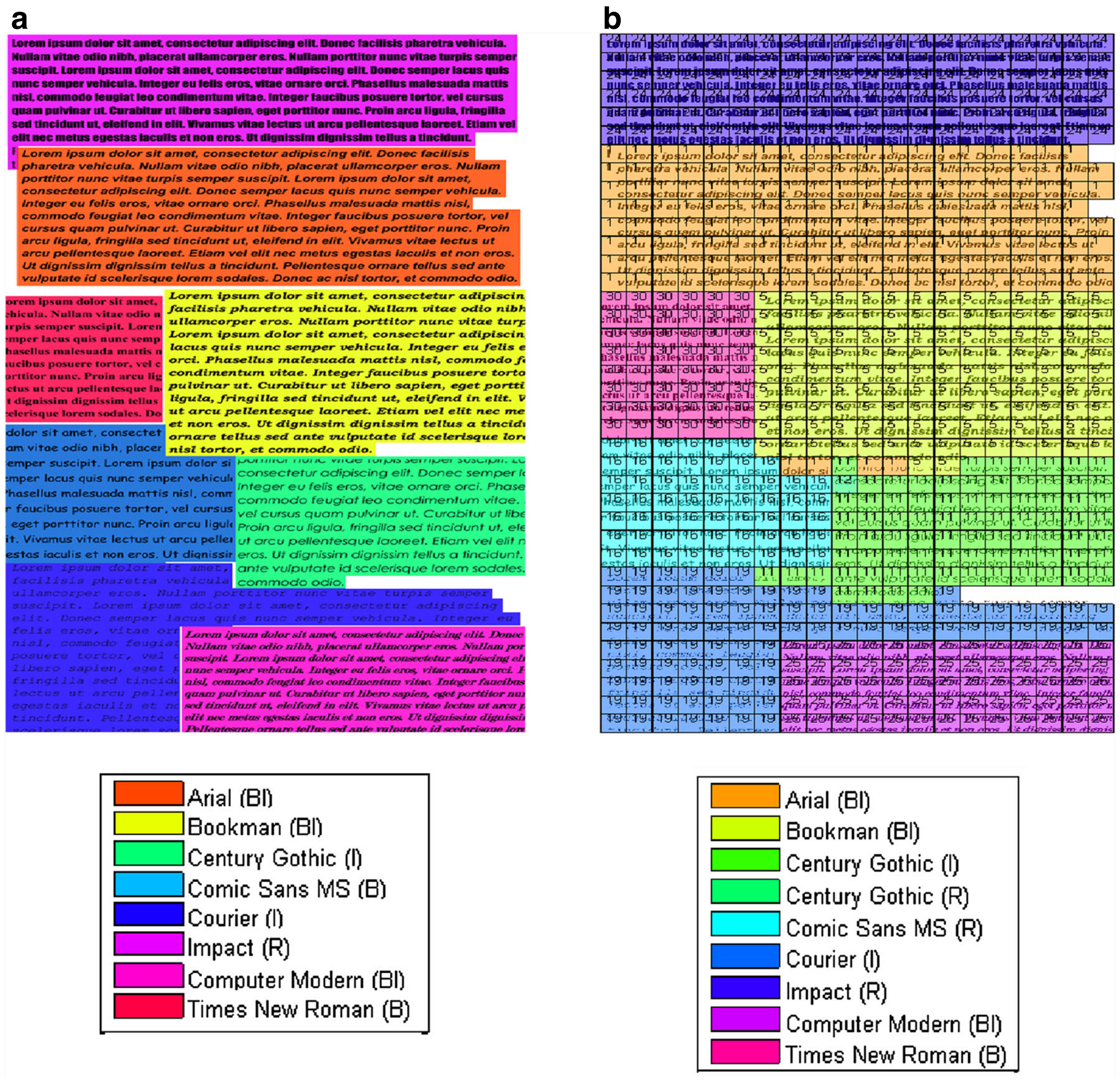

Fig. 2 A collage of multiple fonts: **a** ground truth and **b** estimated regions using our method

Table 6 Average recognition accuracy of the proposed method for each dataset compared with the combined dataset

Fonts	In single dataset	In the combined dataset
Farsi fonts	87.23	86.88
Arabic fonts	96.86	96.75
English fonts	96.78	96.62
Chinese fonts	91.34	91.34

Block size is 96 × 160

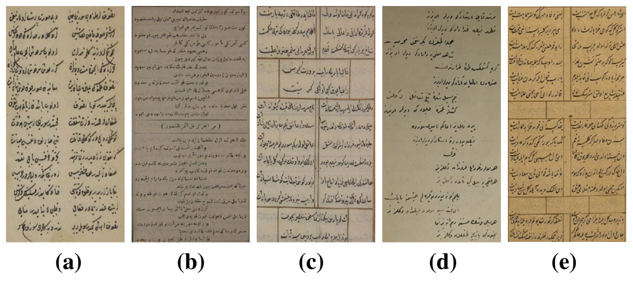

Fig. 3 Examples from different Ottoman calligraphy styles. **a** Divani. **b** Matbu. **c** Nesih. **d** Rika. **e** Talik

We performed two tests to classify calligraphy styles in Ottoman documents. In the first one, we used unmodified documents and extracted features from the entire image to make use of the page layout style. In the second one, we extracted blocks (as done in font recognition) from the dataset images to test the effect of writing style alone. Table 7 summarizes the performance of our method in the task of categorizing Ottoman calligraphy styles. Although there are large intra-class variations due to different handwriting characteristics, our method classifies the calligraphy styles almost perfectly. In the second test, we choose a block size of

Table 7 Confusion matrix (recognition percentages) of the proposed method on unedited Ottoman texts

True style	Estimated style				
	Divani	Matbu	Nesih	Rika	Talik
Divani	98.95	0.00	0.00	1.05	0.00
Matbu	0.00	100	0.00	0.00	0.00
Nesih	0.00	0.00	100	0.00	0.00
Rika	0.00	0.00	0.00	100	0.00
Talik	0.00	0.00	0.00	1.05	98.95

Table 8 Confusion matrix (recognition percentages) of the proposed method on cropped Ottoman texts

True style	Estimated style				
	Divani	Matbu	Nesih	Rika	Talik
Divani	99.30	0.00	0.70	0.00	0.00
Matbu	0.00	98.68	1.32	0.00	0.00
Nesih	3.26	2.17	94.57	0.00	0.00
Rika	0.00	0.00	0.00	87.50	12.50
Talik	0.00	0.00	0.00	0.00	100.00

Table 9 Confusion matrix using DWT

	Correct style	Wrong style
Correct font	84.18	6.25
Wrong font	5.27	4.30

With CWT all examples are correctly classified with 100 % accuracy

96 × 160. The confusion matrix for the cropped dataset is given in Table 8. The overall recognition accuracy is 96.01 %, and the results indicate that, although the layout has a positive effect on accuracy, the proposed method can classify calligraphy styles with high accuracy using only texture information.

4.3 Analysis of the proposed method

4.3.1 Comparison of CWT and DWT

To demonstrate the superiority of CWT over DWT in the task of font recognition, the features are extracted from the English "noise-free" texts using both CWT and DWT. The SVMs are trained and cross-validated using these features, and the SVM parameters are optimized for each case. The confusion matrices over all fonts and styles for DWT are presented in Table 9. DWT fail to differentiate between emphases of a font, especially bold/bold-italic and regular/italic, due to the lack of directional selectivity. CWT, on the other hand, perfectly discriminate among fonts and styles.

4.3.2 Choosing block size

Since decisions are generated per block, it is desirable that block size is as small as possible. However, the height of the blocks should be at least larger than the height of the characters in the sample, because smaller blocks would contain only parts of the characters and would not capture all characteristics of a given font. Recall that we use 36 dimensional features corresponding to the mean and standard deviations of the absolute values of the outputs of the CWT. Statistical features allow a degree of freedom once that lower bound is passed. These features capture a font-style pair's characteristics very similarly, regardless of its block size. Figure 4 shows the features extracted from three different sizes of Arial bold blocks, where the similarity of features of a 96×96 block and a 192×192 or 288×288 block are apparent. Thus, a classifier trained with 96×96 blocks can easily classify 192×192 or 288×288 blocks. To demonstrate, we performed several tests on English "low-noise" dataset. A classifier was trained with 96×96 blocks and was used to classify square blocks of side length 96, 144, 192, 240, 288 and 336 pixels (see Table 10). Numbers are chosen as integer multiples of 48, which is found to be

4.3.3 Computational complexity

We compared the efficiency of our system with the other methods in terms of complexity. All tests were done in MATLAB ®2011b on a 32 bit Windows 7-installed PC with an Intel i7 1.6 GHz CPU and 4 GB RAM. Our feature extraction stage is faster in MATLAB than with Gabor, and SRF implementations summarized in Table 11. Our method has a lower complexity compared to other methods because the DT-CWT algorithm uses $\mathcal{O}(N)$ multiplications and additions [22].

5 Conclusions and future work

We present a novel and computationally efficient method for language and script-independent font and calligraphy style recognition. Experimental results indicate that our method outperforms the respective methods for all datasets. Our CWT-based features are computationally the most efficient among MATLAB implementations of the other feature extraction methods.

We also experimentally show that the proposed features are capable of capturing and discriminating among different font and calligraphic styles with high accuracy. Since the style of an Ottoman document is an indicator of the type of document, calligraphic style estimation is an important step for automatic processing of the millions of Ottoman documents in archives. It is possible to automate the entire

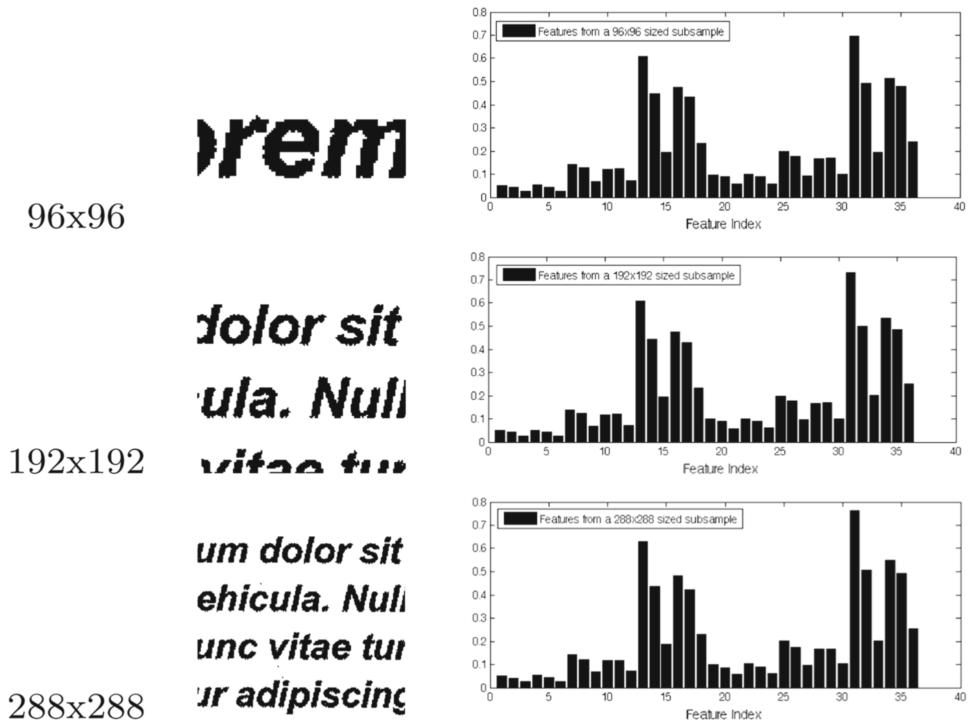

Fig. 4 Example image blocks and average features of Arial bold, for different block sizes

Table 10 Recognition rates of a classifier trained with features extracted from 96 × 96 blocks

Font	Emphasis	Block size					
		96 × 96	144 × 144	192 × 192	240 × 240	288 × 288	336 × 336
Correct	Correct	93.14	95.98	98.78	99.48	100	100
Correct	Wrong	1.76	1.55	0.41	0.00	0.00	0.00
Wrong	Correct	3.52	1.99	0.65	0.52	0.00	0.00
Wrong	Wrong	1.57	0.49	0.16	0.00	0.00	0.00

Table 11 Time to extract each feature from 128 × 128 and 256 × 256 sample

Feature	Implementation	Required time per sample (ms)	
		128 × 128	256 × 256
DT-CWT	MATLAB[a]	4.40	10.40
SRF	MATLAB	9.40	13.70
	C	3.78[b]	–
Skewness and Kurtosis	MATLAB	8.60	39.30
Gabor	MATLAB	29.30	100.70

DT-CWT [12], SRF [20], Skewness and Kurtosis [5], Gabor [27,30]
[a] With precompiled C kernels, [b] value taken from [20]

Ottoman style recognition method by incorporating region segmentation as a stage of pre-processing. Image region segmentation can remove blank regions of a document, which will not only speed up the entire font recognition system but increase its accuracy. In this study, we used our own Ottoman dataset, comprised of 60 documents. We are planning to enlarge our dataset by including other resources from the Internet and by scanning more Ottoman documents, with the aim of preparing a public database for automated Ottoman document processing.

We select the texture analysis block size as a multiple of the size of a single character. In general, each block contains several characters and character portions. It is also possible to automate block size selection according to the size of a character by automatically detecting character sizes. We can also process a given document in overlapping blocks, which will increase the computational cost but improve recognition results.

References

1. Abdelnour, A.F., Selesnick, I.W.: Nearly symmetric orthogonal wavelet bases. In: Proceedings IEEE International Conference on Acoustics, Speech, Signal Processing (ICASSP), vol. 6 (2001)
2. Abuhaiba, I.S.I.: Arabic font recognition using decision trees built from common words. J. Comput. Inf. Technol. **13**(3), 211–224 (2004)
3. Aiolli, F., Simi, M., Sona, D., Sperduti, A., Starita, A., Zaccagnini, G.: SPI: A system for paleographic inspections. In: AI IA Notizie, vol. 4, pp. 34–48 (1999)
4. Amin, A.: Off-line arabic character recognition: the state of the art. Pattern Recognit. **31**(5), 517–530 (1998)
5. Aviles-Cruz, C., Rangel-Kuoppa, R., Reyes-Ayala, M., Andrade-Gonzalez, A., Escarela-Perez, R.: High-order statistical texture analysis-font recognition applied. Pattern Recognit. Lett. **26**(2), 135–145 (2005)
6. Azmi, M.S., Omar, K., Nasrudin, M.F., Muda, A.K., Abdullah, A.: Arabic calligraphy classification using triangle model for digital jawi paleography analysis. In: 11th International Conference on Hybrid Intelligent Systems (HIS), pp. 704–708 (2011)
7. Bataineh, B., Abdullah, S.N.H.S., Omar, K.: A novel statistical feature extraction method for textual images: optical font recognition. Expert Syst. Appl. **39**(5), 5470–5477 (2012)
8. Ben Moussa, S., Zahour, A., Benabdelhafid, A., Alimi, A.M.: New fractal-based system for arabic/latin, printed/handwritten script identification. In: 19th International Conference on Pattern Recognition (ICPR 2008), pp. 1–4 (2008)
9. Ben Moussa, S., Zahour, A., Benabdelhafid, A., Alimi, A.M.: New features using fractal multi-dimensions for generalized arabic font recognition. Pattern Recognit. Lett. **31**(5), 361–371 (2010)
10. Borji, A., Hamidi, M.: Support vector machine for persian font recognition. Int. J. Intell. Syst. Technol. 184–197 (2007)
11. Bozkurt, A., Suhre, A., Cetin, A.E.: Multi-scale directional-filtering-based method for follicular lymphoma grading. Signal Image Video Process. **8**(1), 63–70 (2014)
12. Cai, S., Li, K., Selesnick, I.: Matlab implementation of wavelet transforms. Polytechnic University (2010)
13. Celik, T., Tjahjadi, T.: Multiscale texture classification using dual-tree complex wavelet transform. Pattern Recognit. Lett. **30**(3), 331–339 (2009)
14. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)
15. Chaudhuri, B.B., Garain, U.: Automatic detection of italic, bold and all-capital words in document images. In: Proceedings of Fourteenth International Conference on Pattern Recognition, vol. 1, pp. 610–612. IEEE (1998)
16. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
17. Hatipoglu, S., Mitra, S.K., Kingsbury, N.: Texture classification using dual-tree complex wavelet transform. In: Seventh International Conference on Image Processing and its Applications (Conf. Publ. No. 465), vol. 1, pp. 344–347. IET (1999)
18. Hill, P.R., Bull, D.R., Canagarajah, C.N.: Rotationally invariant texture features using the dual-tree complex wavelet transform. In: Proceedings of International Conference on Image Processing, vol. 3, pp. 901–904. IEEE (2000)
19. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. National Taiwan University (2003)
20. Khosravi, H., Kabir, E.: Farsi font recognition based on sobel-roberts features. Pattern Recognit. Lett. **31**(1), 75–82 (2010)
21. Kingsbury, N.: A dual-tree complex wavelet transform with improved orthogonality and symmetry properties. In: Proceedings of International Conference on Image Processing, vol. 2, pp. 375–378. IEEE (2000)
22. Kingsbury, N.G.: The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement. In: Proceedings of EUSIPCO, vol. 98, pp. 319–322 (1998)
23. Ma, H., Doermann, D.: Gabor filter based multi-class classifier for scanned document images. In: 7th International Conference on Document Analysis and Recognition (ICDAR), pp. 968–972 (2003)
24. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)
25. Ozturk, A., Gunes, S., Ozbay, Y.: Multifont ottoman character recognition. In: The 7th IEEE International Conference on Electronics, Circuits and Systems (ICECS) (2000)
26. Ozturk, S., Toygar Abak, A., Sankur, B.: Font clustering and cluster identification in document images. J. Electron. Imaging **10**(2), 418–430 (2001)
27. Petkov, N., Wieling, M.B.: Gabor filter for image processing and computer vision. University of Groningen (2008)
28. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. J. Comput. Vis. **40**(1), 49–70 (2000)
29. Rado, S.: Turk Hattatlari:XV. yzyildan gnmze kadar gelmis nl hattatlarin hayatlari ve yazilarindan rnekler. Yayin Matbaacilik Ticaret (1983)
30. Ramanathan, R., Soman, K.P., Thaneshwaran, L., Viknesh, V., Arunkumar, T., Yuvaraj, P.: A novel technique for english font recognition using support vector machines. In: International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom '09), pp. 766–769 (2009)
31. Senobari, E.M., Khosravi, H.: Farsi font recognition based on combination of wavelet transform and sobel-robert operator features. In: 2nd International eConference on Computer and Knowledge Engineering (ICCKE), pp. 29–33. IEEE (2012)
32. Slimane, F., Kanoun, S., Hennebert, J., Alimi, A.M., Ingold, R.: A study on font-family and font-size recognition applied to arabic word images at ultra-low resolution. Pattern Recognit. Lett. **34**, 209–218 (2013)
33. Villegas-Cortez, J., Aviles-Cruz, C.: Font recognition by invariant moments of global textures. In: Proceedings of International Workshop VLBV05 (very low bit-rate video-coding 2005), pp. 15–16 (2005)
34. Yang, Z., Yang, L., Qi, D., Suen, C.Y.: An EMD-based recognition method for chinese fonts and styles. Pattern Recognit. Lett. **27**(14), 1692–1701 (2006)
35. Yosef, I.B., Beckman, I., Kedem, K., Dinstein, I.: Binarization, character extraction, and writer identification of historical hebrew calligraphy documents. Int. J. Doc. Anal. Recognit. **9**(2–4), 89–99 (2007)
36. Zahedi, M., Eslami, S.: Farsi/arabic optical font recognition using sift features. Procedia Comput. Sci. **3**, 1055–1059 (2011)
37. Zhu, Y., Tan, T., Wang, Y.: Font recognition based on global texture analysis. IEEE Trans. Pattern Anal. Mach. Intell. (**10**), 1192–1200 (2001)
38. Zhuang, Y., Weiming, L., Jiangqin, W.: Latent style model: discovering writing styles for calligraphy works. J. Visual Commun. Image Represent. **20**, 84–96 (2009)
39. Zramdini, A., Ingold, R.: Optical font recognition using typographical features. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 877–882 (1998)