Journal of mathematics and computer science 15 (2015), 40-56

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

Farsi Font Recognition Based On the Fonts of Text Samples Extracted by SOM

Majid Ziaratban[1,*], Fatemeh Bagheri[1,+]

[1] Engineering Faculty, Golestan University, Gorgan, Iran

[*] *m.ziaratban@gu.ac.ir*

[+] *f.bagheri@gu.ac.ir*

Article history:
Received November 2014
Accepted January 2015
Available online January 2015

Abstract

A Farsi font recognition algorithm based on the fonts of some frequent text samples is proposed. Some features are extracted from the connected components of a text image. The feature vectors are clustered by using a Self-Organizing Map (SOM) clustering method. The clusters with more members determine the most frequent connected components (MFCCs). A number of members of these big clusters are extracted from the input image. This procedure is applied to both training and test images. Since the frequent samples in different Farsi texts are very similar, it can be guaranteed that a large number of samples of the detected MFCCs for a test image surely are in the extracted training samples set. The font type and font style of the extracted test samples are recognized by matching between them and the training samples. The most frequent recognized font of the extracted samples is considered as the font of the input text. To achieve a more accurate algorithm with lower complexity, the font size is determined in the second phase after the phase of the font type and style recognition. Using a lexicon reduction procedure reduces the complexities and processing time. The font size estimation is carried out based on the size of a particular MFCC in a text image. Experiments show that the proposed method outperforms other font recognition methods.

Keywords: Farsi font recognition; Most-frequent connected components; SOM**.**

1. Introduction

Optical character recognition (OCR) is an important and applicable subject of the pattern recognition field. The OCR is used in wide range of applications. Vehicle license plate recognition, video based document processing, automatic postal code recognition, making editable version of historical books and many other applications utilize OCR. Existence of a large number of fonts with various characteristics makes the machine-printed text recognition tasks very difficult. Applying an OCR to a machine-printed

text with known font causes more accurate result than when the font of the input text is unknown [1]. Several works have been done in optical font recognition in various languages such as Latin [2-10], Chinese [6, 11-13], Arabic [14-16], and Farsi [17-20]. In all of these works, the font of the whole text in a document image was assumed to be uniform. Thus, like all other previous works, we propose an algorithm to recognize an unknown uniform font of a machine-printed document.

Generally, three types of font recognition methods can be considered: Typographical feature-based methods, texture analysis-based approaches, and frequently used component-based methods.

Typographical feature-based algorithms [3-5, 7] extract some features, like character skews, between-characters and between-words space widths, and projections in upper, center and lower zones of the line from the printed texts. The main drawback of these approaches is that they require noise-free and high-quality text images [18].

Texture analysis was used in many researches [2,6,8,10,16-19] to recognize the font of a text block. Some methods such as Gabor filters are used in various texture based applications [21]. In the font recognition tasks, these approaches first normalize the spaces between text lines, words and characters to make a normalized text blocks. Texture features are extracted from the normalized text blocks.

In [17], Gabor filter was used to extract the global texture features from Farsi text images. The rate of 85% was achieved for the recognition of 7 font types and 4 font styles using a weighted Euclidean distance (WED) classifier.

Khosravi and Kabir [18] proposed a Farsi font recognition system based on Sobel-Roberts features. These statistical features describe the texture of the texts. Ten font types were considered and a font recognition rate of 94.16% was obtained.

Ziaratban and Bagheri [19] proposed DEG filters to describe curvedness of the components in printed texts. They showed that combining extracted texture features based on DEG filters and the Gabor based features improved the Farsi font recognition rate.

The approaches in the third category, i.e. frequently used component based methods have been designed for content-independent font recognition applications. In these methods, the learning set consists of a number of frequently used components in all font classes. These approaches determine the text font based on the fonts of the detected samples of the components in a text.

Abuhaiba in [14] and [15] proposed two methods based on most-frequent components for recognizing Arabic fonts. In [14], Arabic words of the training set for each font are segmented into symbols. Then, the templates are selected from the most frequent segmented symbols. An input word is segmented into its symbols and the symbols are checked to be matched with the templates. The fonts of the templates, for which the match score is greater than a preset threshold, are retained. The most-frequent font is considered as the font of the text. The error, the rejection and success rates of recognizing 36 font classes were obtained as 15%, 7.6% and 77.4%, respectively. In [15], the font recognition of Arabic texts is done using a decision tree built from the most-frequent words (MFWs). If the distance between a test word and a word in the leaf node is less than a predefined threshold, then the test word belongs to the set of most-frequent Arabic words; otherwise, it is rejected. One hundred MFWs were used to recognize 36 Arabic font classes and a 90.8% success rate was obtained.

Similar method was applied in [11] to recognize the Chinese fonts based on most-frequent Chinese characters. In this paper, we try to propose a font recognition method in the third category with more accuracy and lower complexity. The proposed algorithm includes four main innovations.

Unlike the existing font recognition methods, to reduce the complexity, the font recognition is partitioned into two separate phases. In the first phase, the type and style of the font are recognized, irrespective to the font size. Then, by using the information about the recognized font, the font size is

determined in the second phase. Moreover, instead of most frequent words (MFWs) used in [15], in our algorithm, *Most-Frequent Connected Components* (MFCCs) are used. The reason is that there are much more connected components than words in Farsi texts. The conventional approaches in the third category [11,15] have to perform time-consuming recognition or matching trials to extract appropriate samples from a text. In our algorithm, the matching processes are not performed between all CCs of a test image and all predetermined MFCCs in all font classes. In the proposed method, most frequent samples for both training and test sets are determined for each text image separately by using the SOM clustering method. Hence, we do not need performing the recognition phase for all CCs in the image to find samples of predetermined CCs in test images which is very time consuming process.

In addition, a lexicon reduction method based on the aspect ratio of the samples in the training and test sets is used in the matching stage. In the matching stage, the font of the best matched training sample is considered as the font of the test sample. Using this lexicon reduction procedure reduces the complexities in the matching stage and decreases the processing time.

The rest of the paper is organized as follows: the methodology of the proposed Farsi font recognition approach is discussed in section 2. In section 3, the experimental results are presented and analyzed. Finally, conclusions are drawn in section 4.

2. Methodology

The first main stage in the proposed method is determination of the most frequent CCs for an input text image. Then some samples for each MFCC are extracted. The font of the input text is recognized based on the recognized fonts of the extracted samples. In the following, various parts of the proposed method are discussed.

Since Farsi is a cursive writing language, some characters may connect to each other and make bigger components. Therefore, Farsi texts usually contain CCs in a wide range of sizes.

A *scale* parameter, h_S, is defined to have a scale independent system. The value of h_s is computed from the horizontal projections of text lines as follows:

$$h_s = \frac{1}{n_L}\sum_{i=1}^{n_L} d_i \qquad (1)$$

where d_i is the number of HP_i cells with values greater than 0.15 max(HP_i). HP_i is the horizontal projection of the i-th row. The value of h_s for the instance text image in Figure 1 is 86 pixels (h_s=86). The font size and the scanning resolution of the text image in this figure are 20 points and 300 dpi, respectively.

2.1 SOM-based CC clustering

To determine most frequent CCs and extracting samples of the detected MFCCs, CCs of a text image are clustered. Various clustering methods [22-27] have been proposed and used. In our experiments, CCs are clustered by using the SOM method [28]. A self-organizing map (SOM) is an artificial neural network. Learning procedure of SOM is unsupervised. In other words, it can classify input samples without external helps. Hence, SOM can be useful for data clustering.

Some features are extracted from CCs of a text image to be used for clustering. The goal of the clustering is that the samples of a CC locate in the same cluster. The number of members of each cluster determines the frequency of the CC in the text image. Hence, members of the bigger clusters are considered as the samples of most frequent CCs. Totally, 28 features are extracted for each CC image. The first three features are the height, width, and number of pixels of the CC. Other 25 features are the

number of pixels of 25 correlated sub-image of the normalized CC image. Normalized CC image is obtained by concatenating sufficient empty rows or columns to right or bottom of the CC image to have a square CC image. In the normalized CC image, the numbers of rows and columns are equal.

Consecutive sub-images are 50% correlated. The size of the sub-images is $\frac{h}{3} \times \frac{h}{3}$, and h is the value of height and width of the normalized CC image. In Figure 2-a, a sample CC image including Farsi character "س" (Sin) is shown. Figure 2-b shows the normalized CC image. Three sub-images of the normalized CC image are illustrated by red, blue, and green squares in figure2-c.

Before feature extraction and clustering CCs, dots and small diacritics do not present significant changes in different fonts should be ignored for clustering and also for font matching. A CC is considered as dots or small diacritics if both following conditions are satisfied:

$$h_i < 0.35\ h_s \tag{2}$$

$$np_i < 3\ h_s \tag{3}$$

where h_i and np_i are the height and the number of pixels of the i-th CC, respectively.

آقای ناظم دستمالم را دادم که آن عکس ها را پاک کند و بعد هم راه افتادم که بروم سراغ اتاق خودم . در اتاقم را که باز کردم ، داشتم دماغم با بوی خاک نم کشیده ه - اش اخت می کردم که آخرین معلم هم آمد . آمدم توی ایوان و با صدای بلند ، جوری که در تمام مدرسه بشنوند ، ناظم را صدازدم و گفتم با قلم قرمز برا ی آقا یک ساعت تاخیر بگذارند. روز سوم باز اول وقت مدرسه بودم . هنوز از پشت دیوار نپیچیده بودم که صدای سوز و بریز بچه ها به پیشبازم آمد . تند کردم . پنج تا از بچه ها توی ایوان به خودشان می پیچیدند و

Figure 1. A sample Farsi text image

(a) (b) (c)

Figure 2. (a) A sample CC image, (b) normalized CC image, (c) sub-images of the normalized CC image

The sizes (heights and widths) of the detected most frequent CCs are stored in a new matrix $\boldsymbol{S}$, of size $N_{MFCC} \times 2$. N_{MFCC} is the number of selected MFCCs after the dot elimination stage. $\boldsymbol{S}$ will be used in the font size estimation phase. $\boldsymbol{S}(j,1)$ and $\boldsymbol{S}(j,2)$ determine the height and width of the $MFCC_j$, respectively and j is the priority of the remaining MFCCs.

A 6×6 clusters topology is considered for the SOM. The topology and the number of members of clusters for the text in Figure 3 are shown in Figure 4.

After CC clustering, samples of most frequent CCs are the members of the bigger clusters. Bigger clusters are the clusters containing larger number of members. A number of MFCC samples are extracted from the text image. In Table 1, six detected MFCCs of the instant text and the number of samples in the text are given. Twelve extracted samples for each MFCC are shown in this table. The fonts of these extracted samples are determined by matching with the training samples. The font of the text is determined based on the recognized fonts of the MFCC samples. The overall diagram of the proposed font recognition method is illustrated in Figure 5.

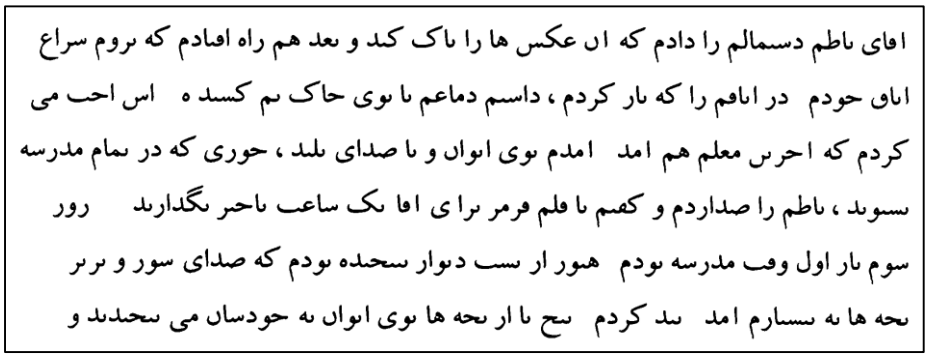


Figure 3. The sample Farsi text image after removing dots and small diacritics

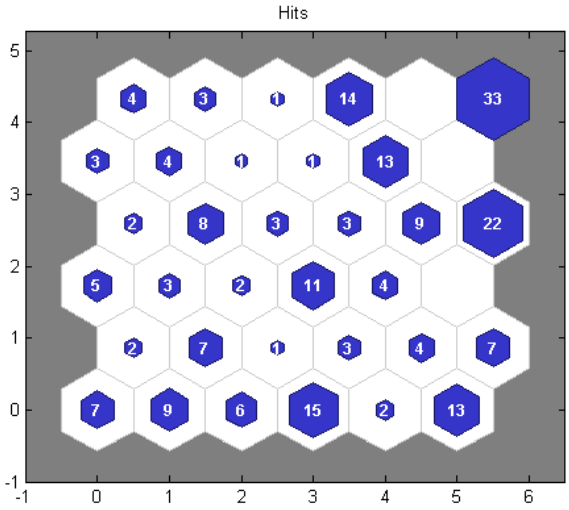


Figure 4. Clusters topology of the SOM and the number of members of clusters for the text in Figure 3

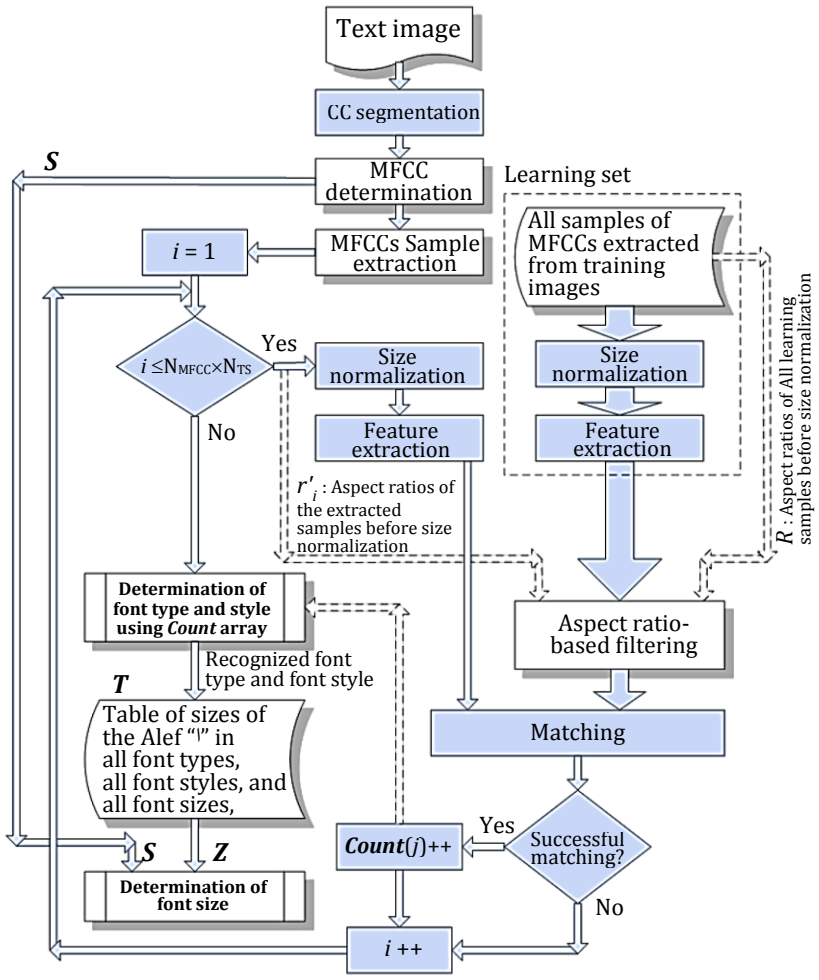


Figure 5. The flowchart of the proposed Farsi font recognition algorithm

Table 1. Ten extracted samples for each of the four mfccs of the text image in Figure 3

Index of the cluster	Twelve extracted MFCC samples	Number of members
1	ا ا ا ا ا ا ا ا ا ا ا ا	33
2	ر ر ر ر ر ر ر ر ر ر ر ر	22
3	ر ر ر ر ر ر ر ر ر ر ر ر	15
4	م م م م م م م م م م م م	14
5	د د د د د د د د د د د د	13
6	ا ا ا ا ا ا ا ا ا ا ا ا	13

2.2 Font recognition

To determine the font of each sample extracted from a test image, the best matched sample from the learning set have to be find. The font of the best matched learning sample is considered as the font of the test sample. The learning set samples are the samples extracted from the text images in the training set.

As mentioned before, to have a higher recognition rate, the estimation of the font size is carried out in a separate phase, after the determination of the font type and font style (in the first phase). Thus, the sizes of the extracted CCs are not important in the first module. Consequently, the extracted CCs are normalized and the results are sent to the feature extraction stage. But the sizes of MFCCs will be used for the font size estimation and are very important in the second module. Therefore, as shown in Figure 5, before the size normalization and after the MFCCs determination, the sizes of MFCCs are stored in ***S*** to be used in the font size estimation module.

Furthermore, in the matching stage of the first module, an extracted test CC is checked to be matched with only the samples in the learning set, for which the *aspect ratios* are close to the aspect ratio of the test CC. Consequently, in the learning set, before the CC size normalization, the aspect ratios of the training samples are calculated and stored in vector ***R***. It is shown in the flowchart of the algorithm in Figure 5. In the size normalization stage, the resizing factor RF_i, for CC_i is calculated as follows:

$$RF_i = \frac{q}{max\{h_i, w_i\}} \tag{4}$$

Some empty rows or columns are added to the bottom or to the right side of each resized CC to obtain a normalized qxq pixels image (The value of q will be set in the following). Any feature extraction approaches such as the methods based on wavelet coefficients, Zernike moments, invariant moments, and structural feature extraction methods can be used to extract suitable features from the normalized images. In our approach, a wavelet transform-based feature extraction method is used. Since in the wavelet transformation, the image is down-sampled in each stage by a factor of 2, q is better to be equal to 2^n. In our experiments, like in [22], the normalized 64x64 pixels images are suitable for feature extraction. Therefore, the value of q is set to 64. By applying three stages of Haar wavelet transform to a normalized image, 64 wavelet features are extracted. For more details, please refer to [2]. A minimum Euclidean distance classifier is used to perform the matching process between the feature vectors of the test and the learning sample.

Totally, 34 different classes are considered irrespective to the font sizes. The font of a text image is the most frequent font of the extracted test MFCC samples. The most frequent font is determined by voting among the recognized fonts of the extracted MFCC samples of a test image. To perform the voting procedure, an empty vector, ***Count***, containing 34 cells is considered. For each extracted test samples, the value of the cell in the ***Count*** vector corresponding to the recognized font type and font style of the matched CC is incremented.

$$\boldsymbol{Count}(f_j) = \boldsymbol{Count}(f_j) + 1 \tag{5}$$

where f_j is the index corresponding to the recognized font type and font style of the j-th extracted test sample. After determining the fonts of all extracted samples of a test image, the font (font type and font style) of the text image is obtained by:

$$Font = \arg\max_c(Count(c)) \ , c=1, \ldots, 34 \tag{6}$$

where *Font* indicates the font type and font style of the text. The confidence values of the matched samples are used to determine the font of a text for which more than one cell of ***Count*** have the maximum

value. m is the maximum value of ***Count*** and ***Count*** (K) = m where K is a set of two or more font indexes. The confidence value is calculated as follows:

$$Confidence(i) = \frac{1}{\sum_{j=1}^{m} d_{i,j}} \quad , i \in K \tag{7}$$

where $d_{i,j}$ is the Euclidean distance between the feature vectors of the j-th MFCC sample extracted from test image and its corresponding best matched learning sample. The text font is recognized based on the maximum confidence value.

2.3 Font size estimation

In our algorithm, the font size of the text image is estimated when the font type and font style is determined. The font size is estimated based on the size of a particular MFCC. This MFCC is "ا" which is called *Alef*. The first MFCC in Table 1 is Alef. Since Alef is the most-frequent CC (after neglecting dots) in Farsi texts, we can be sure that the size of this CC exists in $\boldsymbol{S}$ which includes the sizes of MFCCs of the test image.

In our algorithm, the sizes (heights and widths) of Alef in all font types, font styles and font sizes are obtained and stored in a table called $\boldsymbol{Z_{ALL}}$ (Table 2). After recognizing the type and style of the font of a text image, the heights and widths of Alef in all font sizes of the recognized font type and style are extracted from the table $\boldsymbol{Z_{ALL}}$ and stored in matrix $\boldsymbol{Z}$. $\boldsymbol{Z}$ is a N_{fs}×2 matrix and N_{fs} is the number of different font sizes. The k-th row of $\boldsymbol{Z}$ contains the height and width of Alef in the k-th font size of the recognized font type and style. The font size determination procedure for the instance text image in Figure 3 is illustrated in Figure 6. The font index of the example text (Figure 3) is equal to 23. Hence, $\boldsymbol{Z}$ includes the 23rd column of $\boldsymbol{Z_{ALL}}$. The transpose matrix of $\boldsymbol{S}$ which includes the sizes of the MFCCs of the instance text image is given in Figure 6.

$$\boldsymbol{Dist}(j,k) = \sqrt{(\boldsymbol{Z}(k,1) - \boldsymbol{S}(j,1))^2 + (\boldsymbol{Z}(k,2) - \boldsymbol{S}(j,2))^2} \quad , k=1,2,3,\ldots,N_{fs} \quad , j=1,2,3,\ldots,N_{MFCC} \tag{8}$$

$$\boldsymbol{D}(k) = \min_{j}(\boldsymbol{Dist}(j,k)) \tag{9}$$

$$FontSize = \underset{k}{arg}\{\min(\boldsymbol{D}(k))\} \tag{10}$$

The Euclidean distances between the rows of $\boldsymbol{S}$ and $\boldsymbol{Z}$ rows are calculated and stored in matrix ***Dist***. Vector $\boldsymbol{D}$ includes the least value of each row of the ***Dist*** matrix and indicates the minimum distance of the $\boldsymbol{Z}$ rows from the rows of $\boldsymbol{S}$ matrix. In Figure6, the minimum value of $\boldsymbol{D}$ is in the fifth cell. It shows that the fifth row of $\boldsymbol{Z}$ has the minimum distance from the first row of $\boldsymbol{S}$. It means that the first MFCC of the instance text image is Alef and the fifth font size (which is equal to 20 points in our experiment) is determined as the font size of the text.

3. Experimental Results

3.1 Datasets

Two datasets were used to evaluate performances of various methods: our dataset, *DB1*, and the dataset introduced and used in [18], *DB2*. Totally, 374 font classes including 11 Farsi font types, 4 font styles, and 11 font sizes were considered in DB1. Font types consisted of 'Andalus', 'Arial', 'Jadid', 'Koodak', 'Lotus', 'Nasim', 'Nazanin', 'Sina', 'Tahoma', 'Titr' and 'Zar'. Font styles were 'Normal' (Regular), 'Italic', 'Bold' and 'Bold Italic'. An instant text in all font types and styles is shown in Figure 7. Five font types ('Jadid', 'Koodak', 'Nasim', 'Sina' and 'Titr') do not have Bold and Bold-Italic styles. Font sizes

included 9, 12, 14, 16, 20, 24, 28, 32, 36, 40 and 48 points. Thus, totally 34x11=374 font classes were considered. All texts were printed with an hp LaserJet 1320 printer. 978 out of 1700 pages were scanned with an hp scanjet 5590 scanner and used for training.

Table. 2. The table Z_{ALL} that includes the heights and widths of the MFCC "ا" (Alef) in all font types, font styles and font sizes. Columns and rows correspond to 34 font indexes and 11 font sizes, respectively.

Height	*The index corresponding to the recognized font type and font style*																															
Font sizes	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*
9	23	23	23	23	23	23	24	24	23	23	29	29	29	29	23	23	24	24	22	22	23	23	21	21	24	24	28	28	27	27	28	28	29	29
12	31	31	31	31	31	30	31	31	31	31	38	38	38	38	31	31	32	32	29	29	30	30	28	28	32	32	37	37	36	36	37	37	38	38
14	36	36	36	36	36	36	37	36	36	36	45	45	44	44	36	36	38	38	34	34	35	35	32	32	37	37	43	43	42	42	43	43	45	45
16	41	41	41	41	41	41	42	42	41	41	51	51	51	51	41	41	43	43	39	39	40	40	37	37	42	42	49	49	48	48	49	49	51	51
20	51	51	52	52	51	51	52	52	52	52	64	64	64	64	51	51	54	54	48	48	50	50	46	46	53	53	62	62	60	60	61	61	64	64
24	61	61	62	62	61	61	63	63	62	62	77	77	76	76	61	61	65	65	58	58	60	60	56	56	63	63	74	74	72	72	74	74	77	77
28	72	72	72	72	71	71	73	73	72	72	89	89	89	89	71	71	75	75	68	68	70	70	65	65	74	74	87	87	84	84	86	86	89	89
32	82	82	83	83	82	81	84	83	83	83	102	102	102	102	82	82	86	86	77	77	80	80	74	74	84	84	99	99	96	96	98	98	102	102
36	92	92	93	93	92	91	94	94	93	93	115	115	114	114	92	92	97	97	87	87	90	90	83	83	95	95	111	111	108	108	111	111	115	115
40	102	102	103	103	102	102	105	104	103	103	128	128	127	127	102	102	108	108	97	97	100	100	93	93	106	106	124	124	120	120	123	123	128	128
48	123	123	124	124	122	122	126	125	124	124	153	153	153	153	122	122	129	129	116	116	120	120	111	111	127	127	148	148	144	144	147	147	153	153

Width	*The index corresponding to the recognized font type and font style*																															
Font sizes	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*
9	9	9	10	9	4	8	4	9	4	10	3	13	5	15	3	9	4	10	2	8	4	9	3	8	6	11	8	12	8	14	14	8	8	16
12	12	12	13	12	5	11	6	12	5	13	4	17	7	20	4	11	5	13	3	10	5	12	4	10	8	15	10	16	10	19	19	11	10	22
14	15	14	15	14	5	13	7	13	6	15	5	20	8	23	4	13	6	16	4	12	5	14	4	12	9	17	12	19	12	22	22	13	12	25
16	17	16	17	16	6	14	8	15	7	18	5	23	9	26	5	15	7	18	4	13	6	16	5	14	10	20	13	21	13	25	25	15	13	29
20	21	19	21	20	8	18	10	19	8	22	7	29	11	33	6	19	9	22	5	17	8	21	6	17	13	25	17	27	17	31	31	18	17	36
24	25	23	25	24	9	22	11	23	10	26	8	34	13	40	7	23	11	27	6	20	9	25	7	21	15	30	20	32	20	37	37	22	20	43
28	29	27	30	28	11	25	13	27	12	31	9	40	16	46	8	27	12	31	7	24	11	29	8	24	18	35	23	37	23	44	43	26	23	51
32	33	31	34	32	12	29	15	31	13	35	11	46	18	53	9	30	14	36	8	27	12	33	10	28	20	39	27	43	27	50	49	29	27	58
36	37	35	38	36	14	32	17	35	15	40	12	51	20	60	11	34	16	40	9	30	14	37	11	31	23	44	30	48	30	56	56	33	30	65
40	42	39	42	40	15	36	19	38	17	44	13	57	22	66	12	38	18	45	10	34	15	41	12	35	25	49	33	53	33	62	62	37	33	72
48	50	47	51	48	18	43	23	46	20	53	16	68	27	79	14	46	21	54	12	40	18	49	15	42	30	59	40	64	40	75	74	44	40	87

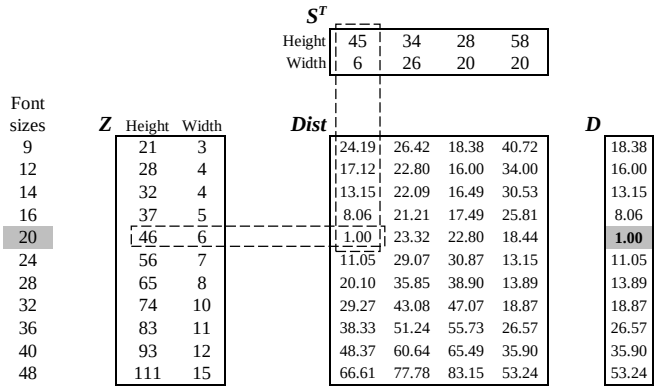


Figure 6. Font size estimation of the instance text image in Figure 3

The remaining pages were used for the test. Test pages were scanned with an hp scanjet 7400 scanner. The pages were scanned with 300 dpi resolution. The contents of text pages in the training and test sets are completely different. In the learning set, the most-frequent CCs of each training text image were determined and samples of each MFCC were extracted. Totally, 167843 MFCC samples were extracted from 978 training images of DB1.

DB2 is the dataset which was gathered by Khosravi and Kabir and used in [18]. This dataset consists of 20,000 images of Farsi text lines: 15000 text blocks for the training and 5000 blocks for the test. The scanning resolution was set to 100 dpi. Ten font types, a font style and a font size were considered in this dataset. Font types are: 'Mitra', 'Traffic', 'Yaghut, 'Homa', 'Lotus', 'Times New Roman', 'Nazanin', 'Tahoma', 'Titr' and 'Zar'. Totally, 134218 MFCC samples were extracted from 15000 training images of DB2.

3.2 Parameter tuning

MFCCs are sorted based on their frequency so that the index of the most frequent CC is 1. Experiments showed that by using more than 5 MFCCs, the accuracy may decrease. The reason is that for the MFCCs with higher index values, the frequency value decreases. Furthermore, by increasing the index of MFCCs, their varieties increase significantly. Hence, there may not be similar samples in the learning set. In our approach, to have a faster font recognition system, we used 4 MFCCs.

The number of matchings and the algorithm complexity are proportional to the number of samples. Experiments showed that in big text pages, only ten samples for each MFCCs were sufficient to reach the accuracy rate of 100%. Hence, only ten samples for each MFCC were used (N_{TS} was set to 10).

To have more reliable font voting procedure, a maximum allowable distance, *Dmax*, is considered. A successful matching occurs when the minimum distance between samples in the test and learning sets is lower than *Dmax*. Otherwise, the matching is not successful and the test sample is rejected and not allowed to participate in the font voting procedure.

Our algorithm is in the third category of font recognition approaches which are based on frequently used components. This type of algorithms such as MFW-based method is very time-consuming. We try to reduce the complexity of our method by using a lexicon reduction procedure. The lexicon reduction is performed by defining and using the aspect ratios parameter for extracted samples.

Figure 7. All the 34 font classes. The numbers under the sample texts are the font class indexes.

In the matching stage, to reduce the number of redundant matching trials, an extracted test CC is checked to be matched with only the CCs in the learning set, for which the aspect ratios are close to the aspect ratio of the test CC. A learning sample is placed in the reduced lexicon for the matching if the following condition is satisfied:

$$r_i'(1-\Delta r) \le r_k \le r_i'(1+\Delta r) \qquad (11)$$

where r_i' and r_k are the aspect ratios of the *i*-th extracted test sample and that of the *k*-th learning sample, respectively. Δr is a non-negative constant. Very large value of Δr is not suitable for lexicon reduction; because most of the learning samples are allowed to be used the matching stage and hence, the complexity will not reduce.

Small value of Δr causes small learning subset and decreases the number of matchings; however the sample rejection rate increases. Figure 8 shows the rejection, recognition and error rates of the extracted samples versus different values of *Dmax* and Δr over DB1 and DB2 datasets. Small values of *Dmax* and Δr will increase the sample rejection rate. For small *Dmax* values, the training samples should be more similar to the extracted test

samples. For very large values of Δr and *Dmax*, a large number of learning samples are selected for matching. In these cases, some samples with low matching scores enter into the voting procedure and may cause incorrect font recognition results.

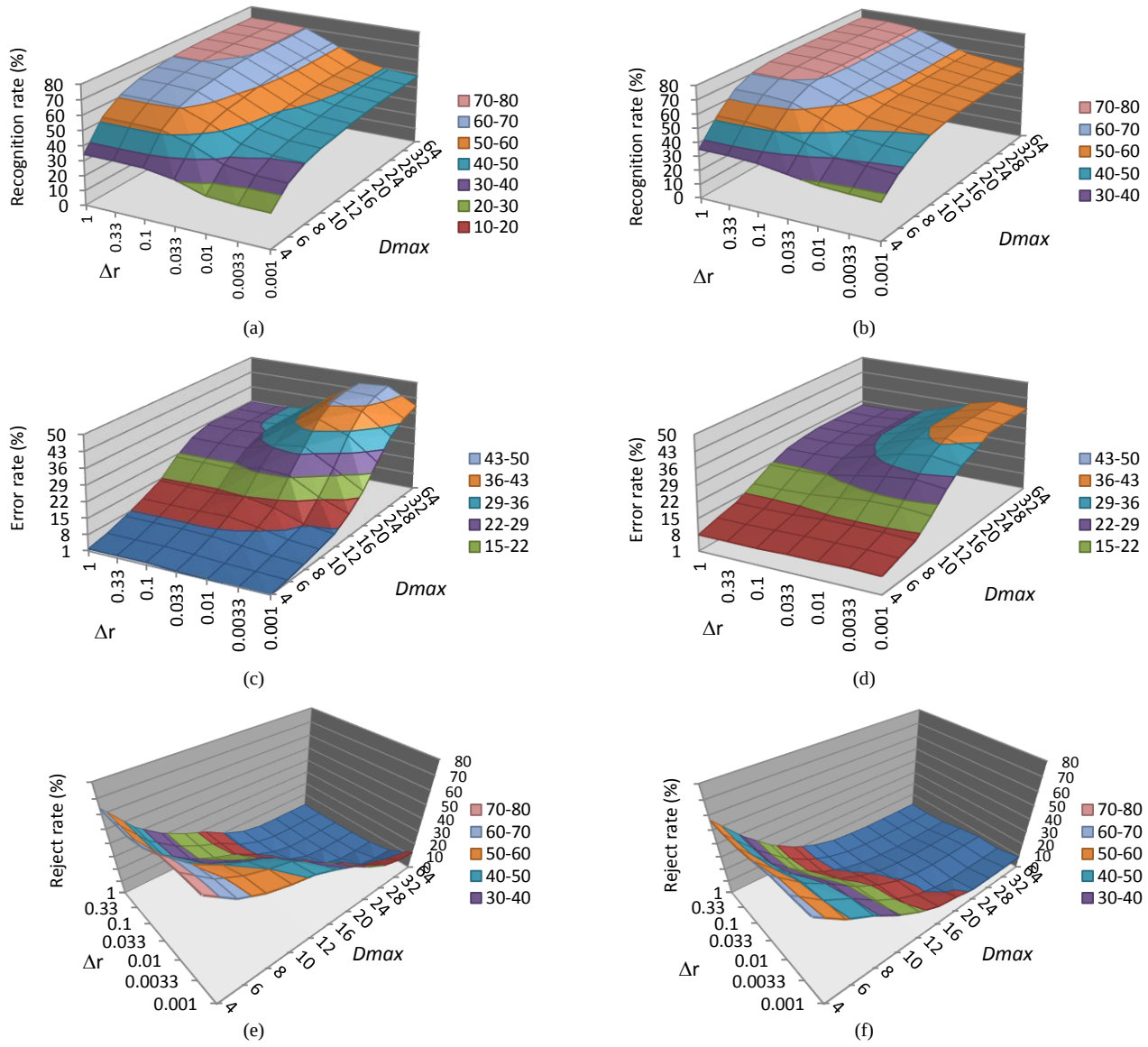


Figure 8. Sample rejection, correct matching and error rates versus different values of *Dmax* and Δr. The left and right plots in each row deal with the results of the experiment on DB1 and DB2, respectively.

For very large values of *Dmax* and very small Δr values, there are not enough learning samples to be accurately matched with an extracted sample, but the matching is not rejected because of large *Dmax* value. Therefore, in these cases, the error rate increases considerably.

The variations of font recognition rate versus *Dmax* and Δr are shown in Figure 9. From this figure, the values of Δr and *Dmax* were set to 0.1 and 10, respectively.

Table 3 shows the lexicon reduction rate versus different values of Δr. The lexicon reduction rate is equal to the total number of the training samples in the learning set divided by the number of samples in the selected training subset with respect to the value of Δr. Since in our experiments, the value of Δr was set to 0.1, the number of matchings was obtained at least about 26 times lower than that when the whole learning set was used for matching process (Table 3).

3.3 Experimental results analysis

Two methods in the second font recognition category (Gabor filter-based and Sobel-Roberts-based [18] algorithms) and an approach in the third category (MFW-based [15]) were implemented for evaluating. In the implemented MFW-based algorithm, 100 most-frequent Farsi words were considered. In the Gabor filter-based method, 32 Gabor filters in 16 directions and two values of λ were used. Values of λ were set to 3 and 5. The value of σ in Gabor filters was set to 0.56λ. The size of Gabor filters was 17x17 pixels. For Sobel-Roberts-based method, the features were extracted and 4 MLP neural networks were used with AdaBoost M2 training method [23] just like in [18].

In the experiment, 722 text pages of DB1 were used for test. These text pages included 200 words per page on average. Test set in DB2 consisted of 5000 images of text lines. Results are given in Table 4. The best font recognition rate on both datasets was obtained by using the proposed algorithm. The second row of Table 4 shows the recognition rates of font type and font style irrespective to font size. Comparison of results in first and second rows of Table 4 shows that the font recognition rates for other approaches except the proposed method reduces significantly when a large number of different font sizes are considered. It demonstrates that most of errors in other approaches except the proposed method occurred in recognizing font sizes.

This reduction in recognition rate did not occur in the proposed method, because we recognize the font type and style irrespective to font size in the first module and then the font size is estimated in the second module. In the second row of Table 4, the accuracy of texture-based approaches is lower than that of MFW-based and MFCC-based methods. Most of errors in texture-based approaches occurred in classifying very similar fonts.

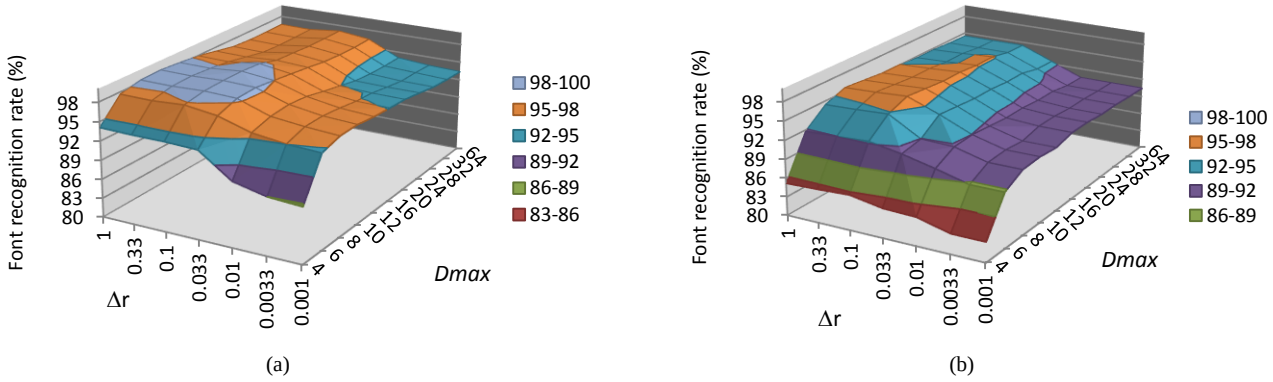


Figure 9. The variations of font recognition rate versus *Dmax* and Δr (a) DB1 and (b) DB2

Table 3. Lexicon reduction rate versus different values of Δr

Δr	0.001	0.0033	0.01	0.033	**0.1**	0.33	1
lexicon reduction rate in DB1	815.94	471.48	216.82	77.03	**26.88**	9.73	3.56
lexicon reduction rate in DB2	1032.4	589.65	253.06	85.13	**29.37**	10.49	3.82

In several cases, three similar fonts (‘Lotus’, ‘Nazanin’, and ‘Zar’) in DB1 and four similar fonts (‘Mitra’, ‘Lotus’, ‘Nazanin’, and ‘Zar’) in DB2 were misclassified by using texture-based methods. The reason is that for these similar fonts, the textures of the normalized blocks are very similar to each other. If the texts of various blocks are identical, the corresponding textures have slight differences and can be recognized. But in the experiments, the texts of documents in the test and training sets were not the same. Therefore, discrimination between the texture features of the document images with different contents in similar fonts was very difficult.

In the proposed algorithm the font recognition is based on the fonts of the text components which occur frequently. These frequent components usually exist in documents even with different contents. The accuracy of the proposed algorithm was greater than that of the MFW-based method. One reason is that the number of samples of the frequent CCs in a document image is much more than the number of MFW samples and the decision based on more samples is more reliable. Another reason is that the MFW-based method must separate 374 font classes in a single matching phase, while the proposed algorithm, first determines the font type and font

style of a document among 34 font classes. Then, in the second module, the font size is estimated. Lower total number of font classes in the matching stage of the proposed method rather than the MFW-based algorithm causes more discrimination power and accuracy.

Although, the resolution of the images in DB2 is lower than that in DB1, the font recognition rates of the Gabor filter-based and Sobel-Roberts-based approaches on DB2 are higher than that on DB1 (first row of Table 4). The reason is that the number of font classes in DB2 (10 font classes) is much smaller than that in DB1 (374 font classes).

The accuracies of these texture-based methods on DB2 are lower than those in the second row of Table 4 on DB1. One reason is the lower qualities and resolutions of document images in DB2 versus those in DB1. Moreover, the number of words per text image in DB2 is lower than that in DB1.

The performances of the proposed algorithm and the MFW-based method on DB2 were lower than texture-based methods. Because text blocks in DB2 are very small and the number of text components in the text blocks of DB2 is much lower than those of DB1. Furthermore, since the quality and scanning resolution of the images of DB2 much lower than those of DB1, the sample matching accuracy reduced. Some errors of the proposed method on DB2 are illustrated in Figure 10. As this figure shows, errors occurred in texts including broken samples of one long word Figure 10 (a-c), the text samples including numerals or English characters Figure 10 (d,e), and also low quality text blocks Figure 10 (f).

The processing times for various approaches are reported in Table 5. The most time-consuming approach is the MFW-based method; because in this method, all words of a text image are checked to be matched with all predetermined MFWs in all font classes. In our experiments, for implementing the MFW-based approach, the average number of required matching trials for recognizing the font of a sample text page of the DB1 dataset (including 300 Farsi words) was:

300 (number of words in the sample test image) × 100 (number of predetermined MFWs) × 20 (number of training samples for each MFW) × 374 (number of font classes) = 224,400,000.

In the proposed algorithm, the number of required matching trials was:

4 (number of MFCCs in a test image) × 10 (N_{TS} : number of required samples for each MFCC in a test page) × 167843 (total number of training samples in the learning set) × $\frac{1}{26}$ (from Table 3, the complexity reduction rate for Δr=0.1 is about 26. That means each CC in a test image is checked to be matched with only $\frac{1}{26}$ of all training samples, on average) = 258,220 which is more than 860 times lower than that of the MFW-based algorithm.

Reducing the number of font classes in the matching phase by splitting the font recognition into two modules and also reducing the search space by using the aspect ratios are the two main reasons which increased the speed of the font recognition of the proposed method versus the MFW-based approach.

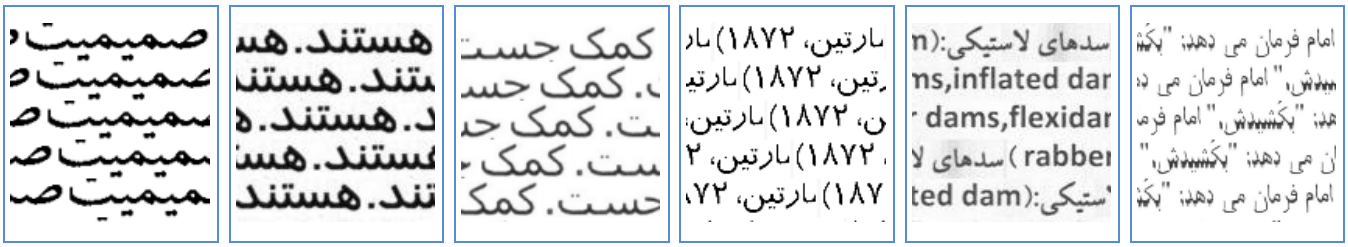
Sobel-Roberts-based algorithm was the fastest method; because in this method, the size of both Sobel and Roberts filters is 3×3. While in the Gabor filter-based algorithm, 32 Gabor filters are used. Furthermore, the sizes of Gabor filters are larger than the sizes of Sobel and Roberts filters.

Table 4. Font recognition rate (%)

	# of font classes	Gabor filter-based	MFW-based [15]	Sobel-Roberts-based [18]	Proposed MFCC-based
DB1	374	81.72	85.32	83.24	**100**
DB1	34	93.21	97.37	95.29	**100**
DB2	10	91.08	74.68	94.16	**96.90**

Table 5. Processing time (second per text image)

	Gabor filter-based	MFW-based [15]	Sobel-Roberts-based [18]	Proposed MFCC-based
DB1	7.093	35.737	0.331	0.917
DB2	0.468	3.182	0.023	0.184



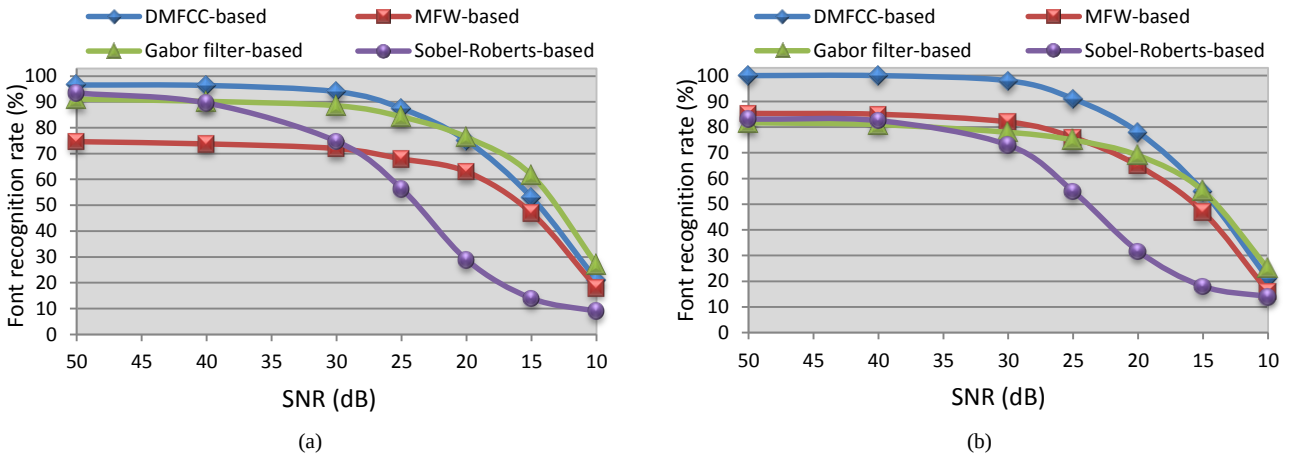
(a) (b) (c) (d) (e) (f)

Figure 10. Some misclassified samples by using the proposed method on DB2

3.4 Noise robustness analysis

In the literature, only some researchers tested their approaches against noisy document images [6,8,12]. To generate noisy text images, they artificially added noise with different SNR (signal to noise ratio) values to the test document images. Like them, Gaussian noises with different values of SNR were added to test images. The results of font recognition on noisy text images of DB1 and DB2 are shown in Figure 11(a) and (b), respectively.

From this figure, of the proposed method is accurate enough when the SNR value is not lower than 20 dB. The matching stage is the most sensitive part of the proposed method against noise. The accuracy of the Sobel-Roberts-based method reduces by increasing noise. The reason is that both Sobel and Roberts filters are high-pass filters. Gabor based method is not very noise sensitive; because Gabor filters smooth text images at least in one direction. The smoothing amount is proportional to the width of the Gaussian part of the filter, σ.



(a) (b)

Figure 11. Font recognition accuracy for noisy text images on (a) DB1 and (b) DB2

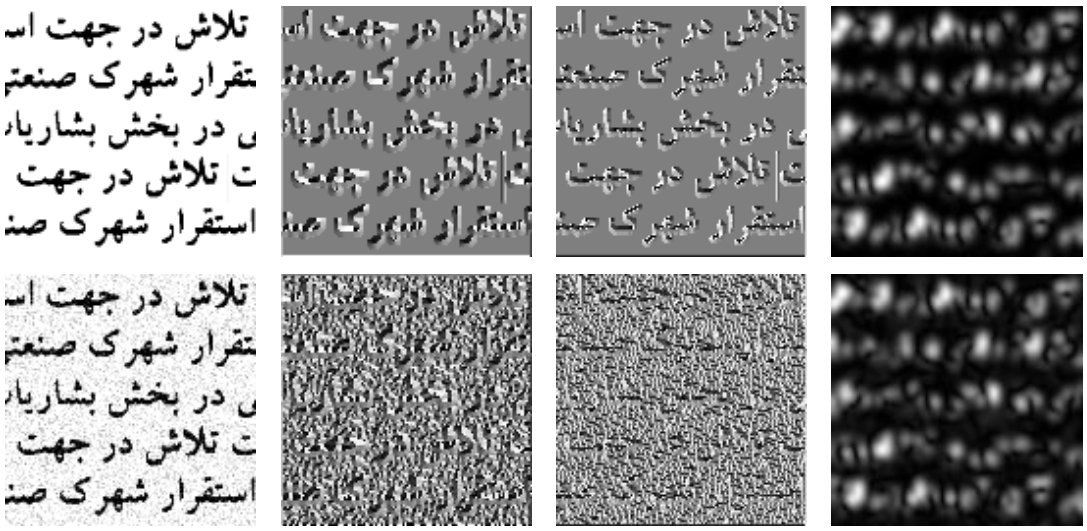


Figure 12. Noise sensitivity in texture-based methods: (1st column) a noise-free and a noisy text blocks, (2nd, 3rd, and 4th columns) corresponding Sobel phase image, Roberts phase image, and Gabor filtered image, respectively.

In this approach, noises are smoothed. Figure 12 shows the effect of noise in Sobel, Roberts, and Gabor filters. The second and third columns in Figure 12 are the phase images of the filtered image by applying Sobel and Roberts filters, respectively. As shown in Figure 12, Sobel and Roberts filters are highly affected by noise. Unlike these filters, the Gabor filter is not very sensitive to noise.

Since word recognition stage is highly sensitive to noise, in the MFW-based algorithm, the accuracy of words sample extraction stage reduces significantly by increasing noise. Consequently, the font recognition rate of the MFW-based method decreased.

4. Conclusion

In this paper, a Farsi font recognition algorithm based on most frequent connected components was proposed. In the proposed method, since the number of CCs is much higher than the number of words in Farsi texts, samples of MFCCs are used instead of samples of MFWs. Furthermore, words are constructed by one or more CCs. Hence, the variety of words is much higher than the variety of CCs. The proposed method has much lower complexity than the other approaches in the third font recognition category. A reason is that in the proposed method, MFCCs are not predetermined. They are obtained for each text image by using the SOM clustering algorithm. Hence, performing a time consuming recognition phase for detecting samples of the frequent CCs in a text image is not required. In addition, by using an aspect ratio-based lexicon reduction procedure, total number of matchings required for MFCC samples font recognition decreased significantly. Font of an input text image is the most frequent recognized font of the extracted MFCC samples.

Estimating the font size in a separate phase after recognizing the font type and font style reduced the number of candidate font classes (and consequently, the classification complexity) in the matching stage and made the recognition more accurate and fast. The font size was estimated based on the size of a particular MFCC (Alef) in the test images. The proposed method outperformed other font recognition approaches.

References

[1] H.S. Baird, G. Nagy, "*A Self-Correcting 100-Font Classifier*", In Proc. of SPIE, Vol. 2181, (1994), pp. 106-115.

[2] H. Ma, D. Doermann, "*Font Identification Using the Grating Cell Texture Operator*", In Proc. of DRR, (2005), pp. 148-156.

[3] A. Zramdini, R. Ingold, "*Optical Font Recognition from Projection Profiles*", Electronic Publishing, Vol. 6, No. 3, (1993) pp. 249-260.

[4] A. Zramdini, R. Ingold, "*Optical Font Recognition Using Typographical Features*", IEEE Trans. on PAMI, Vol. 20, No. 8, (1998) pp. 877-882.

[5] M.C. Jung, Y.C. Shin, S.N. Srihari, "*Multifont Classification using Typographical Attributes*", In Proc. of ICDAR, India, (1999) pp. 353-356.
[6] Y. Zhu, T. Tan, Y. Wang, "*Font Recognition Based on Global Texture Analysis*", IEEE Trans. on PAMI, Vol. 23, No. 10, (2001) pp. 1192-1200.
[7] S.H. Kim, "*Word-Level Optical Font Recognition Using Typographical Features*", IJPRAI, Vol. 18, No. 4, (2004), pp. 541-561.
[8] C.A. Cruz, R.R. Kuoppa, M.R. Ayala, A.A. Gonzalez, R.E. Perez, "*High-order Statistical Texture Analysis-Font Recognition Applied*", Pattern Recognition Letters, Vol. 26, (2005), pp. 135-145.
[9] B.B. Chaudhuri, U. Garain, "*Extraction of Type Style-based Meta-information from Image Documents*", IJDAR, Vol. 3, (2001), pp. 138-149.
[10] B. Allier, H. Emptoz, "*Font Type Extraction and Character Prototyping Using Gabor Filters*", In Proc. of ICDAR, (2003), pp. 799-803.
[11] C.F. Lin, Y.F. Fang, Y.T. Juang, "*Chinese text distinction and font identification by recognizing most frequently used characters*" Image and Vision Computing, Vol. 19, (2001), pp. 329-338.
[12] Z. Yang, L. Yang, D. Qi, C.Y. Suen, "*An EMD-based Recognition Method for Chinese Fonts and Styles*", Pattern Recognition Letters, Vol. 27, (2006), pp. 1692-1701.
[13] X. Ding, L. Chen, T. Wu, "*Character Independent Font Recognition on a Single Chinese Character*", IEEE Trans. on PAMI, Vol. 29, No. 2, (2007), pp. 197-204.
[14] I.S.I. Abuhaiba, "*Arabic Font Recognition Based on Templates*", Int. Arab Journal of Information Technology, Vol. 1, (2003), pp. 33-39.
[15] I.S.I. Abuhaiba, "*Arabic Font Recognition Using Decision Trees Built from Common Words*", Journal of Computing and Information Technology (CIT), Vol. 13, No. 3, (2005), pp. 211-223.
[16] B. Moussa, A. Zahour, M.A. Alimi, A. Benabdelhafid, "*Can Fractal Dimension Be Used in Font Classification*", In Proc. of ICDAR, (2005), pp. 146-150.
[17] A. Borji, M. Hamidi, "*Support Vector Machine for Persian Font Recognition*", Int. Journal of Intelligent Systems and Technologies, Vol. 2, (2007), pp. 178-183.
[18] H. Khosravi, E. Kabir, "*Farsi font recognition based on Sobel–Roberts features*", Pattern Recognition Letters, Vol. 31, (2010), pp. 75–82.
[19] M. Ziaratban, F. Bagheri, "*Improving Farsi font recognition accuracy by using proposed directional elliptic Gabor filters*", First Iranian Conference on Pattern Recognition and Image Analysis (PRIA), (2013), pp. 1 – 5.
[20] M. Ziaratban, K. Faez, F. Bagheri, "*Content-Independent Farsi Font Recognition Based on Dynamic Most-Frequent Connected Components*", 21st International Conference on Pattern Recognition (ICPR 2012) Tsukuba, Japan, November 11-15, (2012), pp. 729-733.
[21] S.M. Lajevardi, Z.M. Hussain, "*Feature Extraction for Facial Expression Recognition based on Hybrid Face Regions*", Advances in Electrical and Computer Engineering, Vol. 9, No. 3, (2009), pp. 63-67.
[22] R. Maghsoudi, A. Ghorbannia Delavar, S. Hoseyny, R. Asgari, Y. Heidari, "*Representing the New Model for Improving K-Means Clustering Algorithm based on Genetic Algorithm*", The Journal of Mathematics and Computer Science Vol .2 No.2 (2011) 329-336.
[23] J. Rajaie, B. Fakhar, "*A Novel Method for Document Clustering using Ant-Fuzzy Algorithm*", The Journal of Mathematics and Computer Science Vol. 4 No.2 (2012), pp. 182 – 196.
[24] K. Azaryuon, B. Fakhar, "*A Novel Document Clustering Algorithm Based on Ant Colony Optimization Algorithm*", The Journal of mathematics and computer Science Vol. 7 (2013), pp. 171-180.
[25] J. Vahidi, S. Mirpour, "*Introduce a New Algorithm for Data Clustering by Genetic Algorithm*", The Journal of Mathematics and Computer Science Vol. 10 (2014) pp. 144 – 156.
[26] Gh. H. Mohebpour, A. Ghorbannia Delavar, "*Some new mutation operators for genetic data clustering*", The Journal of Mathematics and Computer Science Vol. 12 (2014), pp. 282-294.
[27] Gh. H. Mohebpour, A. Ghorbannia Delavar, "*CCGDC: A new crossover operator for genetic data clustering*", The Journal of Mathematics and Computer Science Vol. 11 (2014) pp. 191-208.
[28] T. Kohonen, "*The self-organizing map*", Proc. IEEE, Vol. 78(9), Sept. (1990), pp. 1464-1480.
[29] A. Mowlaei, K. Faez, A. T. Haghighat, "*Feature Extraction with Wavelet Transform for Recognition of Isolated Handwritten Farsi/Arabic Characters and Numerals*", In Proc. of Int. Conf. on Digital Signal Processing, Vol. 2, (2002), pp. 923-926.
[30] Y. Freund, R.E. Schapire, "*Experiments with a new boosting algorithm*", In Proc. of Int. Conf. on Machine Learning, Bari, Italy, (1996), pp. 148–156.