Ottoman Archives Explorer: A Retrieval System for Digital Ottoman Archives

ISMET ZEKI YALNIZ, ISMAIL SENGOR ALTINGOVDE, UĞUR GÜDÜKBAY, and ÖZGÜR ULUSOY
Bilkent University

This article presents Ottoman Archives Explorer, a Content-Based Retrieval (CBR) system based on character recognition for printed and handwritten historical documents. Several methods for character segmentation and recognition stages are investigated. In particular, sliding-window and histogram segmentation methods are coupled with recognition approaches using spatial features, neural networks, and a graph-based model. The prototype system provides CBR of document images using both example-based queries and a virtual keyboard to construct query words.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*; I.7.5 [**Document and Text Processing**]: Document Capture—*Document analysis; graphics recognition and interpretation; optical character recognition; scanning*; I.5.1 [**Pattern Recognition**]: Models—*Neural nets; statistical; structural*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms: Algorithms

Additional Key Words and Phrases: Optical Character Recognition (OCR), historical document analysis, Information Retrieval (IR), Content-Based Retrieval (CBR)

ACM Reference Format:

Yalniz, I. Z., Altingovde, I. S., Güdükbay, U., and Ulusoy, Ö. 2009. Ottoman archives explorer: A retrieval system for digital Ottoman archives. ACM J. Comput. Cult. Herit. 2, 3, Article 8 (December 2009), 20 pages.
DOI = 10.1145/1658346.1658348 http://doi.acm.org/10.1145/1658346.1658348

1. INTRODUCTION

As one of the most powerful states of its time, the Ottoman Empire produced a huge amount of written documents. They are now housed in several countries, but most of them reside in Turkey. Ottoman archives include millions of documents, ranging from title deeds to letters and orders. Preparing manual catalogs for such a cultural treasure is a laborious and demanding task, only some of which has been completed. This means that only a subset of the information buried in these archives is available to researchers, and information that may assist in resolving several issues of controversy between scholars still remains undiscovered.

In the last few years, sections of the Ottoman archives have been digitized for the purposes of efficient and durable storage as well as to provide the infrastructure for electronic access to these documents in the near future. Clearly, a system that allows automatic access with sophisticated retrieval

This work is supported in part by the European Commission's Sixth Framework Program, with Grant No: 507752 (MUSCLE Network of Excellence Project) and the Turkish Scientific and Technical Research Council (TUBITAK), with Grant No: 105E065.
Authors' address: I. Z. Yalniz (corresponding author), I. S. Altingovde, U. Güdükbay, Ö. Ulusoy, Computer Engineering Department, Bilkent University, Ankara, Turkey; email: zeki@cs.umass.edu.

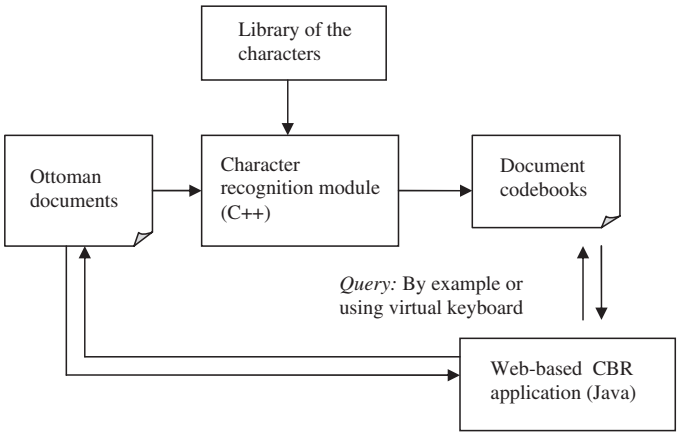


Fig. 1. The architecture of Ottoman Archives Explorer.

functionalities for the archives would be an invaluable tool for researchers. Such a system would not only facilitate their work but may also lead to the discovery of new knowledge.

In this article, we describe a Content-Based Retrieval (CBR) system for digital Ottoman archives. The heart of the system contains a character recognition module which is used to recognize Ottoman characters in the digitized images. Once the characters are recognized, the retrieval functionality is supported both by example-based queries and by a virtual keyboard allowing users to construct query words. The overall system architecture is shown in Figure 1.

The retrieval of Ottoman documents at its current stage is more like a CBR problem (i.e., based on matching character sequences) than an Information Retrieval (IR) problem, as in typical document libraries. Specifically, all modern IR systems are based on the vector-space model, in which a document is represented as a (weighted) vector of "words" and query evaluation involves computing a similarity function of the query vector and document vectors, a process that is usually facilitated by the use of an inverted index [Witten et al. 1994]. However, due to language-specific reasons discussed in the following section, it is hard to determine "boundaries" for Ottoman words and to then represent documents as vectors of words. Instead, what we propose is to represent documents as a list of recognized characters—so-called "codebooks" [Witten et al. 1994; Şaykol et al. 2004]— extracted from each connected component encountered in a document. Such a representation would alleviate the burden of determining word boundaries, which is a complex process. In addition, it may still be possible to use state-of-the art data structures (i.e., an inverted index) for efficient retrieval.

For the character recognition module, we systematically employ and evaluate several different approaches. Recall that the number of documents in the archives is in the order of millions, and a practical solution should employ fast and reasonably effective techniques, rather than (possibly) more accurate yet prohibitively slower methods. With this necessity in mind, we select a set of relatively time-efficient and well-known Optical Character Recognition (OCR) techniques from dozens of alternatives in the literature. We classify these character recognition techniques according to three essential stages, described next.

(a) *Creation of the Character Library*. This can be done in a supervised manner, that is, by simply providing a character set that includes distinct Ottoman characters in various fonts, sizes, etc. Alternatively, an unsupervised symbol library can be created, where the system has no prior knowledge of Ottoman characters and learns them from the documents that are processed. We call the former

a *static library* approach and the latter a *dynamic library* approach. The former one is more like a traditional OCR approach, whereas the latter is more suitable for textual image compression and reconstruction [Witten et al. 1994, Şaykol et al. 2004]. In our work, we use character recognition approaches that employ an initial static character library.

(b) *Segmentation of the Isolated Components*. For connected scripts such as Ottoman, segmentation is a crucial step. In this article, we consider two approaches, namely a *sliding-window-based segmentation* and a *histogram-based segmentation*.

(c) *Similarity Computation Between Characters and Segments*. Several methods are proposed in the literature for comparing characters in the library to the candidate segments. In this work, we employ techniques from three different classes of algorithms. We compare the approaches using: (*i*) spatial domain features, (*ii*) neural networks, and (*iii*) graph-based models.

The contributions of this article are as follows: (*i*) We propose a character-recognition-based retrieval system for Ottoman archives, which is expected to be an invaluable tool for researchers in history and related disciplines. (*ii*) We systematically adapt and/or combine several techniques in the literature into the character recognition component to find the most effective strategy for Ottoman scripts. (*iii*) We provide an evaluation framework that also includes a practical document annotation tool for obtaining datasets that can be contributed by and shared between several researchers. (*iv*) We present initial experimental results for the discussed approaches, and finally, (*v*) we present a Web-based prototype for scholarly use.

In the next section, we start by reviewing the segmentation methods generally used for connected scripts. Afterwards, we summarize the characteristics of Ottoman scripts and discuss earlier studies concentrating on OCR and automatic retrieval of historical Ottoman documents. The approaches proposed for Arabic OCR have some common properties with Ottoman OCR; thus, Arabic character recognition is also briefly discussed. In Sections 3 and 4, we describe the approaches employed for the Ottoman character recognition task in detail and evaluate their performances experimentally. Section 5 presents the architecture and functionality of the CBR system prototype. Finally, we conclude and point to future research topics in Section 6.

2. RELATED WORK

Text recognition is the automatic reading of the text written in digital images. The ultimate aim is to imitate human ability and accuracy to read printed text, but with greater speed [Khorsheed 2002]. Text recognition is widely discussed in the literature and various methods are proposed for interpreting different scripts, fonts, noise levels, etc.

There are various problems involved with text recognition, such as segmenting connected characters. Because of noise or low resolution, digital documents may contain characters that have been erroneously connected. There are also some scripts, such as Ottoman, Arabic, and Hebrew, where certain characters are connected in certain ways according to syntactic rules. In such cases, recognition of individual characters in documents becomes quite hard. Advanced techniques should be developed in order to handle such problems. In Section 2.1, character segmentation methods in the literature are classified and briefly explained. In Section 2.2, related work regarding recognizing text in Ottoman documents is elaborated upon.

2.1 Character Segmentation Methods

There are two major unknowns in the recognizing connected characters: the number of characters in a word or connected component and the character boundaries. There may also be unexpected dissections or connections between characters due to noise in the document image. In such cases, recognition

of individual characters becomes even harder. The aim of segmentation methods is to find ways of determining accurate character boundaries so that characters can be classified in a straightforward manner. Once the characters are segmented correctly, it is possible to recognize these isolated characters with a high degree of accuracy. Character segmentation is a critical stage of the OCR process. The systems that employ any character segmentation method are called "segmentation-based systems." Various segmentation methods are proposed in the literature [Arica and Yarman-Vural 2001; Casey 1996; Khorsheed 2002]. The systems using such methods can be classified as follows.

2.1.1 *Isolated/Presegmented Characters.* The focus of such systems is not on the segmentation of characters; it is assumed that characters are obtained from a reliable segmentation algorithm. The aim is to recognize isolated or segmented characters with a degree of high accuracy. These systems are not practical, as they cannot be deployed and used directly, especially for connected scripts. As an example of such systems, see Öztürk [2000].

2.1.2 *Segmenting a Word into Characters.* The letters in a connected component can be divided into characters by using an explicit segmentation algorithm prior to recognition. In this way, segmented characters can be classified directly, as in Latin OCR systems. However, overall success heavily depends on the accuracy of the segmentation algorithm used. It is not easy to build a robust segmentation algorithm, which tries to segment a connected component by using only features like vertical and horizontal projection vectors [Amin 1998; Altingovde 2006], connected component contours, etc. It should also be noted that; explicit segmentation methods do not use outputs of any character classifier to establish character boundaries.

2.1.3 *Segmenting a Word into Primitives.* There are methods that try to segment a connected component into symbols that may represent characters, fractions of characters, or ligatures. One way to obtain these symbols can be through oversegmenting the connected component by tracing its contour [Allam 1995]. Associated symbols that form particular characters can later be learned and used for recognition in further stages.

2.1.4 *Integration of Segmentation and Recognition.* The segmentation and recognition stages can be interleaved so that more accurate character boundaries and, consequently, recognition rates can be obtained. In such systems, the connected component is systematically divided into many overlapping segments without regard for their contents. Each such segment is later classified and used to determine a coherent segmentation and recognition. The aim of such methods is to eliminate the problems caused by segmentation. Such methods can also be called segmentation free, since they do not employ any feature-based segmentation method. In such systems, recognition errors are mostly due to failures in classification [Casey 1996]. See Chan et al. [2006] as an example of recognizing Arabic script in this manner.

There are also text recognition systems that do not employ any segmentation method. These are called "segmentation-free systems." In principle, segmentation-free systems try to recognize words as a whole instead of as individual characters or primitives. This is achieved by extracting certain features from each word and matching these features to a database of features extracted from the words in the vocabulary. The most likely word is selected as the match. One drawback to such systems is their limited vocabulary. Rare words may not be included in the vocabulary. Moreover, broader vocabulary may also degrade the system performance significantly [Yalniz et al. 2009]. Examples of segmentation-free systems include Ataer and Duygulu [2006; 2007].

2.2 Character Recognition and Automatic Retrieval for Ottoman Documents

Ottoman, handwritten and printed, is similar to Arabic and Persian in that the text is cursive and connected. A character can be written in four different shapes according to its position in the text, that

Isolated	Final	Medial	Initial
ا	ـا	—	
ء	—		
ب	ـب	ـبـ	بـ
پ	ـپ	ـپـ	پـ
ت	ـت	ـتـ	تـ
ث	ـث	ـثـ	ثـ
ج	ـج	ـجـ	جـ
چ	ـچ	ـچـ	چـ
ح	ـح	ـحـ	حـ
خ	ـخ	ـخـ	خـ
د	ـد	—	
ذ	ـذ	—	
ر	ـر	—	
ز	ـز	—	
ژ	ـژ	—	
س	ـس	ـسـ	سـ
ش	ـش	ـشـ	شـ
ص	ـص	ـصـ	صـ

Isolated	Final	Medial	Initial
ض	ـض	ـضـ	ضـ
ط	ـط	ـطـ	طـ
ظ	ـظ	ـظـ	ظـ
ع	ـع	ـعـ	عـ
غ	ـغ	ـغـ	غـ
ف	ـف	ـفـ	فـ
ق	ـق	ـقـ	قـ
ك	ـك	ـكـ	كـ
گ	ـگ	ـگـ	گـ
ڭ	ـڭ	ـڭـ	ڭـ
ل	ـل	ـلـ	لـ
م	ـم	ـمـ	مـ
ن	ـن	ـنـ	نـ
و	ـو	—	
ه	ـه	ـهـ	هـ
لا	ـلا	—	
ى	ـى	ـيـ	يـ

Fig. 2. Basic Ottoman alphabet and character forms at different positions in text.

is, a character may be either isolated or at the beginning, middle, or end of a word. In Figure 2, the Ottoman alphabet and possible forms of each character are shown (adapted from Ottoman [2006]). It should be noted that the shapes of different characters in particular positions may overlap and their meaning must be inferred from the context of the sentence. In this study, we use a character library of 48 shapes that is a subset of those in Figure 2 and covers 98% of shape occurrences for the test dataset used in the experiments. The remaining 2% of shapes consist of characters that are written on top of each other according to the Ottoman syntactic rules.

Ottoman words can also include several components (also called as subwords [Amin 1998]) that are separated by spaces. Moreover, historical handwritten Ottoman documents convey a variety of writing styles ranging from the causal notes of local officers to calligraphic styles which were enthusiastically encouraged by the empires of the time [Alparslan 1999]. Ottoman archives, mostly located at the Turkish State Archives Office today, include millions of handwritten and printed documents that show great variability in writing style and document quality (i.e., due to deterioration effects of time). Thus, it is an important and challenging task to provide sophisticated retrieval mechanisms for the digitized sections of the archives, which, through recent efforts, is quickly growing [Archives 2006].

Despite its great historical and cultural importance, automatic character recognition and retrieval systems for digitized Ottoman documents is a mostly unexplored research topic. In one of the earliest works, a Neural-Network (NN)-based recognition approach is applied to Ottoman characters [Öztürk et al. 2000]. Although the recognition rate is said to be high, both the training and testing stages are

applied on manually segmented characters. In this article, we focus on automatic segmentation and recognition of characters from connected components as they are typically found in historical Ottoman documents.

In Ataer and Duygulu [2006], a retrieval system for Ottoman documents is proposed that involves first segmenting lines and words in a document, and then comparing whole words while querying. In their approach, word comparisons are performed by using quantized vertical projection profiles. In Ataer and Duygulu [2007], querying Ottoman documents is considered as an image retrieval problem. More precisely, each word image in a document is represented by a set of visual terms obtained by the vector quantization of SIFT descriptors extracted from salient points. Words are later matched by comparing the similarity of these visual terms. In Kilic et al. [2008], Ottoman characters are recognized with a support vector machine. Image segmentation, normalization, and edge detection stages are employed for feature extraction. In a recent work [Yalniz et al. 2009], segmentation and recognition stages are integrated for recognizing Ottoman script so that the weaknesses of the classifier can be compensated for by taking into account possible character sequences for a particular connected component. The sequence of candidate characters that formulates a connected component is found by tracing the longest path over the acyclic graph constructed for the bigram approximation of the score function.

Şaykol et al. [2004] propose an effective method for compressing digital Ottoman documents and applying a CBR process on them. In their work, instead of using a static character library as in the typical practice of OCR methods, a dynamic library of symbols is constructed. The construction process begins with an empty library and each extracted component is compared to the current elements of the library to determine if it is (or, it contains) an already discovered symbol. If so, the location of the occurrence is recorded to the document's codebook and the symbol is removed from the document. If it is a new symbol, it is added to the library. Of course, this comparison stage includes further complexities to ensure that the symbols occurring in the connected components are being correctly deduced. Once the codebooks are constructed, they are used to process user queries.

Our work extends Şaykol et al.'s [2004] work in several directions and differs from it in many aspects. First, instead of the dynamic library construction approach, which is essentially proposed for textual image compression [Witten et al. 1994], we adapt the more traditional OCR practice of using a static character library. Of course, both approaches have advantages and disadvantages. In the static case, there are a fixed number of characters in the library with particular fonts and styles, which make similarity comparisons efficient. However, the accuracy of the system may suffer if a high portion of documents include characters that are completely different from those in the library.

In the case of dynamic library creation, the system is more flexible and adaptive so that variety in fonts, sizes, and writing styles may be accommodated. However, if the library size grows significantly, the system may experience efficiency bottlenecks, as each unknown symbol extracted from a document must now be compared to a very large set of candidates. A compromise between these two approaches is to start with a prefilled library, and to extend the library by those symbols that are frequently encountered in the collection but could not be matched to any of the characters in the library. In this article, we use the static library approach.

Another difference of our study is in the segmentation of characters; in addition to the sliding-window approach used in Şaykol et al. [2004], a histogram-based segmentation is employed. Third, the similarity computation stage includes three different methods, based on spatial features, neural networks, and graphs. Fourth, our experimental evaluation involves an annotation framework so that system performance can be measured in terms of both the recognition of single characters and of connected components. Fifth, the prototype CBR system is extended to allow user queries with a virtual keyboard of Ottoman characters in addition to example-based queries.

Input: A character library CL, A document D

1. Extract all isolated components in D
2. **For** each isolated component IC in document D
3. **Repeat**
4. Obtain a segment S of IC
5. Compare S with each character C in CL
6. **If** for some C, Similarity (S, C) > threshold
7. Record (S, C, similarity)
8. Save the location of S yielding the highest similarity value, remove it from IC
10. **Until** (IC is consumed or no segment S in IC has a similarity > threshold)

Fig. 3. A general approach to the segmentation and recognition of connected characters.

2.3 Character Recognition and Automatic Retrieval for Arabic Documents

As Arabic is currently spoken by millions of people all over the world, OCR and information retrieval for Arabic documents have been researched more than for Ottoman, a language surviving mainly in historical texts. Still, it is hard to say whether solutions to any of these problems could be devised that are as satisfactory as for other languages, such as English. Amin [1998] provides a comprehensive survey of offline Arabic character recognition and discusses a variety of approaches adapted for segmentation and recognition, involving methods based on local and global features, neural networks, graphs, and stochastic methods such as HMMs. He concludes that Arabic OCR is still an open research area and that most of the research results are inconclusive because they are provided on small datasets that are not available to others.

Information retrieval systems for Arabic have also garnered particular attention. Unlike for Ottoman, several Arabic documents exist (e.g., on the Web) created by typical word processor programs, etc. Thus, IR of electronic Arabic texts is a more attainable task and is investigated in various works (e.g., Oard and Gey [2002]). On the other hand, for automatic retrieval of textual images of Arabic documents, success is probably limited to the problem of character recognition; some of the most recent studies report encouraging results to this end (e.g., Chan et al. [2006]).

3. OTTOMAN CHARACTER RECOGNITION

Consider the basic character recognition algorithm based on Şaykol et al. [2004] in Figure 3. In this algorithm, an *isolated component* refers to a connected group of black pixels in the document image (i.e., textual image). We prefer to use the term "component" as it can correspond to a single Ottoman character or to a number of combined characters, as is typically encountered in Ottoman script. Isolated components are extracted following the approach outlined in Witten et al. [1994]; that is, the document is scanned from left to right and top to bottom, and whenever a black pixel is encountered, a four-connected boundary detection algorithm is employed to obtain the bitmap of the component and remove it from the original document. It should be noted that the nested components or dots are detected separately; that is, each extracted component is connected. Two crucial points in the preceding algorithm are segmentation (line 4) and similarity computation (line 6), which are discussed in the following sections.

3.1 Segmentation of Isolated Components

Once the set of isolated components is obtained, a set of segments in each component is extracted. This can be achieved using one of the two methods described in the literature, either by placing a

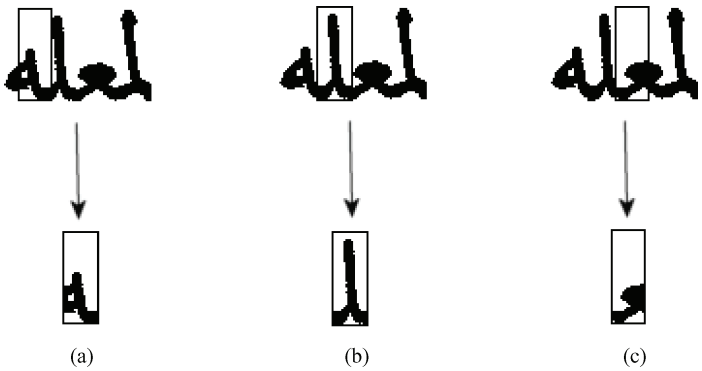


Fig. 4. Segmentation using the sliding-window approach. Cases shown in (a) and (c) are nonmatches as the similarity to any character is below the threshold, whereas the case in (b) is a match.

sliding window over the isolated component or by segmenting the component from the points that are probable character boundaries. In the sliding-window approach, the system does not attempt to determine character borders, but simply slides windows of varying sizes over the bitmap of the isolated component. Specifically, the window is applied in all possible sizes between the component width and a predefined minimum width. This process is necessary because there is no fixed font size in the historical documents. Moreover, the characters in the library are not necessarily the same size, either.

The sliding-window approach may be especially useful for causally handwritten documents for which a heuristic-based segmentation algorithm would not work successfully. The issue of window size is only related to the process of obtaining a candidate segment of the isolated component to be compared with the characters and has nothing to do with the scale invariance, which is satisfied by the all similarity comparison techniques discussed in this article. Figure 4 illustrates how the sliding-window approach works.

Another commonly used technique for character recognition in connected scripts is segmenting a component into a several probable characters. In this article, we adapt a version of the histogram-based approach described in Amin [1998]. This approach first obtains a vertical histogram vector of the pixels in the isolated component. Next, the points that have a density under a specified threshold are determined to be character connection points, due to the observation that, generally, in connected scripts character connection lines are thinner than the characters themselves. A histogram-based segmentation is illustrated in Figure 5.

Clearly, the histogram segmentation is more efficient than the sliding-window approach, as the former usually performs only a few similarity comparisons for a component, whereas the latter considers a much greater number of possibilities. However, our experiment results reveal that the accuracy of the sliding-window technique is better than the accuracy of the histogram segmentation. Nevertheless, for very large document collections where efficiency is a concern, histogram segmentation may be a viable option if satisfactory results can be obtained.

3.2 Similarity Computation

For each segment extracted from an isolated component, the similarity of the segment to the characters in the library should be computed. However, most of the time, there is no need to compute all the similarities. First of all, a segment extracted from a specific position in the isolated component can include only a certain form of the character. For example, in Figure 4, the leftmost window can include only the end-form characters, whereas the other two windows can include only the medial-form of characters.

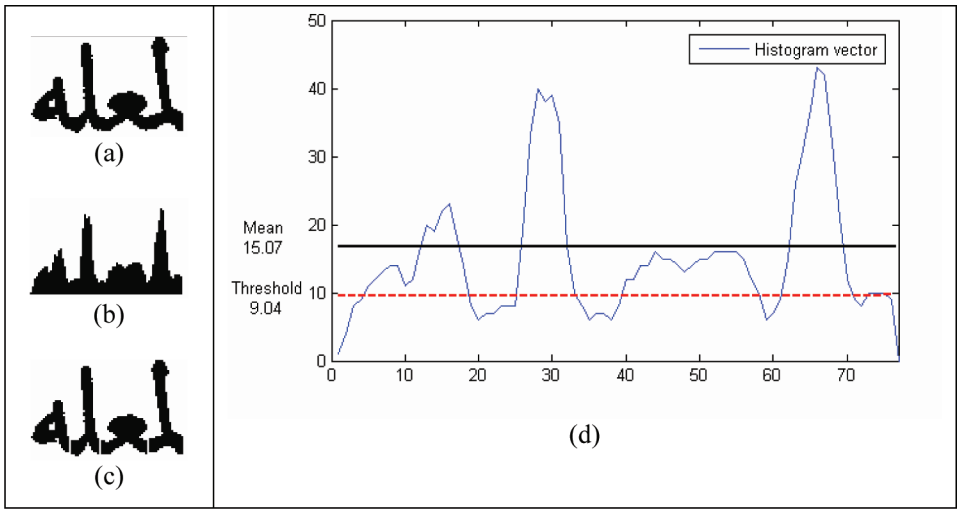


Fig. 5. (a) The original isolated component; (b) its vertical histogram; (c) the segmented characters; and (d) a larger view of (b). The threshold is computed to be 60% of the histogram mean. Each segment is required to be larger than a predefined minimum width.

Secondly, if the aspect ratios of the segment and the character to be compared do not match to some extent, there is no need to make further similarity calculations because the segment will not be able to be matched to the character.

Similarity computation also uses the ratio of segment height to line height. This feature can be used as a filter, as well. If the ratios for a segment and a character in the library do not match, they are not considered similar. After these filtering stages, typically fewer than a dozen out of 48 characters in the library remain to be compared. In this way, it is more likely that the correct match is found with a more sophisticated classifier. In Section 4, we further discuss how these simple filters improve the overall recognition accuracy.

In the literature, several features (e.g., geometric, global) or techniques (such as decision trees, neural networks, hidden Markov models, etc.) are proposed for character recognition both for Latin characters and for connected scripts such as Arabic [Amin 1998; Casey 1996; Khorsheed 2002]. In this study, we adapt and compare three strategies to match an unlabeled segment to a character.

—*Similarity Computation Using Spatial Domain Features*. A similarity computation strategy should include the important feature of scale-invariant and rotation-tolerant comparisons between segments and characters. Scale invariance is crucial; historical documents, most of which are handwritten, include text in varying font sizes. As it is quite tedious to provide characters in all possible sizes, features that allow scale invariance are required. On the other hand, while full-rotation invariance is not required (e.g., some characters may be vertical opposites of each other) a degree of tolerance is required to handle slanted scripts.

In Saykol et al. [2004], spatial domain features are used to provide both scale invariance and rotation tolerance during the similarity computation process. Distance and angular spans are computed for all characters in the library and for each extracted segment. The *angular span vector* is computed by obtaining the number of black pixels in θ-degree slices centered at the character's center of mass with respect to the x-axis. The entries of the vector are normalized by the area (the total number of black pixels) of the character. For instance, in Figure 6(b) the angular span vector is computed for eight slices of $\theta = 45^{\circ}$. The *distance span vector* is computed by obtaining the number of black pixels between

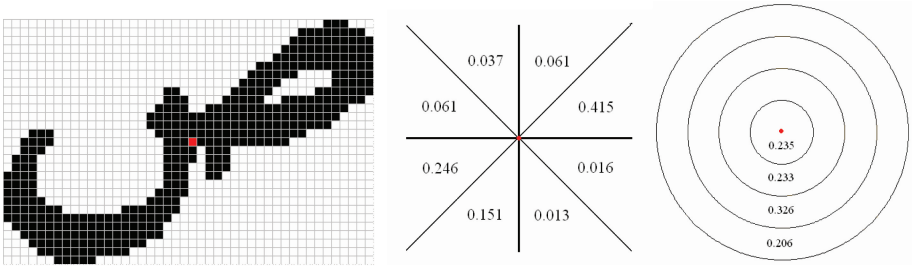


Fig. 6. (a) A character from the library (the red spot is the center of mass); (b) angular span of the character for $\theta = 45°$; and (c) distance span of the character with four entries.

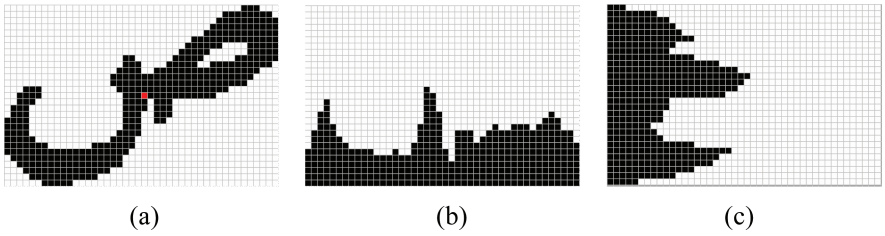


Fig. 7. (a) A character from the library; (b) its horizontal histogram; (c) its vertical histogram.

the concentric circles with radii r, 2r, 3r, etc., centered at the character's center of mass. The entries of the vector are normalized by the distance of the farthest two pixels in the character. Figure 6(c) illustrates the distance span vector for four circles. Both vectors are scale invariant. Rotation tolerance can be achieved by simply allowing the angular span vector to rotate θ degrees in either direction, which means shifting the vector entries by one in the corresponding direction. During the similarity computation, the angular span vector of a library character should be compared to the original vector of the extracted segment, and to the two variants that are obtained by shifting its entries to the right and left by, for example, one slice.

As an alternative to these span vectors, we also obtain horizontal and vertical histograms as feature vectors, as shown in Figure 7.

Similarity Computation Using Neural Networks (NN). Character recognition is one of the areas to which neural networks are widely and successfully applied. To achieve Arabic character recognition, neural networks are used with varying input features, including the original bitmap images of characters [Öztürk et al. 2000], graph-based features such as lines and curves, and various moments and global features (see Amin [1998] for an extensive survey). In our work, we use angular and distance span vectors as the inputs to a back-propagation NN. In particular, four different neural networks are trained for each character form (isolated, beginning, medial, and end) with different numbers of neurons in the hidden and output layers (further details are given in Section 4). In Figure 8, a block diagram of the NN is provided for characters in the medial form.

Moreover, instead of directly using the NN's decision for the recognition, we use the NN simply as a filter; that is, characters that are most similar to the given segment are determined and further compared with the segment by using features such as aspect ratio, line height ratio, and the span vectors as described earlier. NNs are known as robust classifiers and they are widely applied for multifont character recognition [Avi-Itzhak et al. 1995]. Therefore, such a filtering is expected to improve both

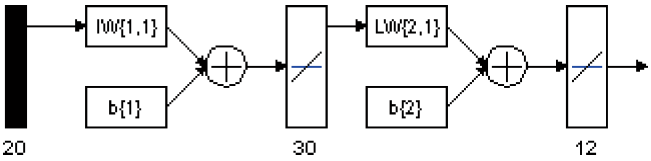


Fig. 8. Feed-forward back-propagation neural-network topology for recognizing the medial form of characters.

the recognition accuracy and system efficiency if, for example, Ottoman characters in various font types are added to the library.

Similarity Computation Using Graph-Based Models. In this case, characters and isolated components are both represented as labeled graphs. During the recognition process, each isolated component is first segmented and then converted to a graph using the thinning algorithm described in the Stentiford Library [2006]. Once the thinned graph is obtained, we compare it with the graphs of characters in the library. This comparison is achieved by extracting a number of features from the graphs and calculating an overall similarity score by weighting the similarity scores between corresponding features of the graphs.

The features are extracted by using the coordinate points of the nodes on the document image. This is possible because the nodes obtained after thinning are like representative pixels in the bitmap and thus have real coordinates. The first set of features is the angular and distance span vectors for the distribution of the number of nodes and node degrees of a graph around the center of mass. Similarity between feature vectors is calculated with the histogram intersection technique [Swain and Ballard 1991]. The second set of features is extracted by projecting the nodes in the graph onto the X and Y axes and obtaining two sequences of degrees of the nodes. The angular and distance span vectors have fixed sizes, whereas the degrees series of projected nodes is a list of numbers of size K, where K is the total number of nodes in the graph.

$$Similarity(S_1, S_2) = \frac{Cost(LCS(S_1, S_2))}{Max(Cost(S_1), Cost(S_2))} \tag{1}$$

The similarity between two series of node degrees is calculated by using Eq. (1). Here, S_1 and S_2 are the two series to be compared and LCS is the longest common subsequence of two series of degree values. Each degree value in the series has a cost associated with it, and the total cost is calculated by summing the individual costs. Dividing the cost of the longest common subsequence of series S_1 and S_2 by the maximum cost of these series gives a similarity score between 0 and 1. The final similarity score between two graphs is calculated by weighting similarity scores of the four features. After a set of experiments for different values of weights, it is observed that the best results are obtained by assigning weights 0.35, 0.35, 0.15, and 0.15 to the angular span, distance span, vertical projection of node degrees, and horizontal projection of node degrees, respectively.

Figure 9 illustrates the graph representation and matching process. After thinning, a refinement could also be performed to reduce the number of nodes, but this is not necessary in our case since we do not use a graph-matching algorithm (to compute, for example, isomorphism or edit-distance [Bunke 1999]), but simply extract a number of features for graph matching. These features exploit not only the spatial distribution of nodes, but also the connectivity information between nodes.

4. EXPERIMENTAL RESULTS

In Table I, we list eight different recognition approaches that correspond to the techniques described in Section 3. The performance of these approaches is evaluated by using 100 printed Ottoman document images scanned from Kurt [2000] and Develi [2006] at 300 dpi. The average resolution of the scanned

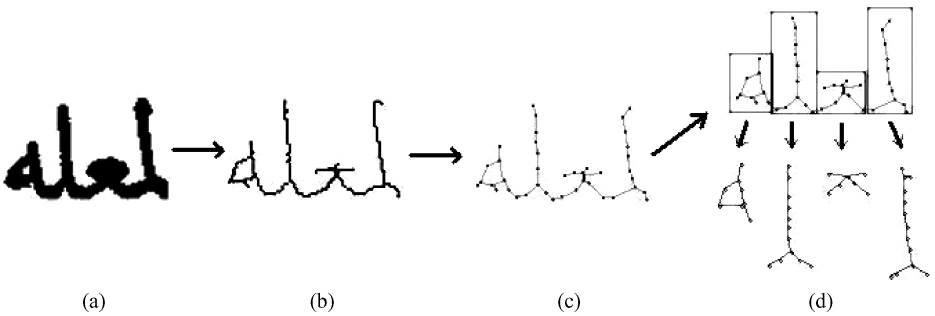


Fig. 9. (a) The original isolated component; (b) its thinned graph; (c) the refined graph; (d) segmentation and graph matching with the graphs in the character library.

Table I. Character Recognition Approaches

Approach	Segmentation	Similarity Computation
WINDOW-SPAN	Sliding-Window	Angular-distance span
WINDOW-NN	Sliding-Window	Neural network
WINDOW-GRAPH	Sliding-Window	Graph matching
WINDOW-PROJ	Sliding-Window	Horizontal and vertical histogram
HIST-SPAN	Histogram-based	Angular-distance span
HIST-NN	Histogram-based	Neural network
HIST-GRAPH	Histogram-based	Graph matching
HIST-PROJ	Histogram-based	Horizontal and vertical histogram

document images is 1300 x 1600 pixels. Page skew is avoided manually at this step. The documents include 34,298 annotated isolated components including a total of 70,041 characters and yield, on average, 2.04 characters per component. No specific noise removal procedure is applied, but during the extraction of isolated components from the documents, those that are smaller than the predefined width, height, or area thresholds are discarded. All results are obtained using a 2.0 GHz Pentium Core2 Duo PC with 2GB RAM.

For the angular and distance span vectors, sizes of 12 and 8 are used following the practice in Şaykol et al. [2004]. For the vertical and horizontal histogram vectors, the size is set to 10. For all vector-based similarity computation methods, two vectors are compared using the histogram intersection technique [Swain and Ballard 1991].

For the NN-based similarity calculation, we trained four different back-propagation neural networks, each of which has only one hidden layer. Each neural network is designed for recognizing a particular form of the characters in the library. There are 15 isolated, 13 beginning, 12 medial, and 16 end forms of characters identified in our library. (It should be noted that a character in the library may have the same shape for its different forms or two distinct characters may have the same shape at different positions). The input size for these networks is 20, corresponding to the angular and distance span vectors of a segment.

Ten training samples are selected manually from a set of correctly recognized samples for each character. At the point of convergence, a success rate of 100% is achieved for the training set. To test the isolated performance of the NNs, perturbed versions of the training samples are automatically generated. That is, bitmaps of the training samples are perturbed by a fixed number of pixels. The neural networks have correctly recognized 90% of perturbed characters, when the perturbation ratio is 10%.

In Table II, we first provide results for the vector-based approaches that use either span or histogram vectors. It should be noted that the *precision measure* reveals the percentage of correctly recognized

Table II. Precision-Recall Analysis of Spatial Vector-Based Recognition Approaches

Approach	Precision	Recall	Processing time per document (sec)
WINDOW-SPAN	0.91	0.89	5.98
WINDOW-PROJ	0.82	0.86	5.11
HIST-SPAN	0.67	0.52	3.46
HIST-PROJ	0.60	0.51	3.10

Table III. Precision-Recall Analysis of NN-Based Recognition Approaches

Approach	Precision	Recall	Processing time per document (sec)
WINDOW-NN	0.91	0.89	9.74
HIST-NN	0.58	0.45	7.61

Table IV. Precision-Recall Analysis of Graph-Based Recognition Approaches

Approach	Precision	Recall	Processing time per document (sec)
WINDOW-GRAPH	0.70	0.55	498.72
HIST-GRAPH	0.54	0.33	283.43

characters among all recognitions, and *recall* reveals the percentage of correct recognitions among all characters in the documents. In other words, precision represents accuracy and recall represents coverage. However, dots can still be used in the prototype system described in Section 5. As seen in Table II, the sliding-window approach outperforms histogram segmentation for all similarity measures. In terms of recall, regardless of the segmentation method, both character recognition techniques that use span vectors and histogram projection vectors perform similarly. However, in terms of precision, character recognition using span vectors is superior to recognition using projection vectors.

We also observe that filtering based on the aspect-ratio and line-height ratio comparisons during the actual recognition stage greatly influence on the overall system's success. Without these filters, the recognition figures decrease about 20%–30% for the same approaches. For example, WINDOW-SPAN's precision and recall figures fall down to 0.60 and 0.71, respectively.

In Table III, the results of NN-based recognition approaches are given. In this process, the characters most similar to the segment are first obtained from one of the NNs in descending order. The list of characters is examined until a character with the same aspect ratio and line-height ratio is found. The similarity of the segment to the corresponding character is then calculated by using the angular and distance span vectors. If the similarity value is also greater than a specified threshold, it is written to the codebook as a match.

Table III reveals that, as in the previous case, segmentation using sliding windows is significantly more successful than using histogram segmentation. Moreover, for the former segmentation method, NN-based character recognition achieves the same precision and recall figures with WINDOW-SPAN, which yields the best results for this dataset. However, we emphasize that the NN method experimented with here still makes use of the span vectors for the final decision.

In Table IV, findings for the graph-based recognition approach show that this approach is inferior to all others. Furthermore, processing a document takes considerably more time compared to the other approaches. In other experiments that are not reported here, we try refining the graphs but we observe almost no improvements and even see decreases in precision and recall figures. It is determined that

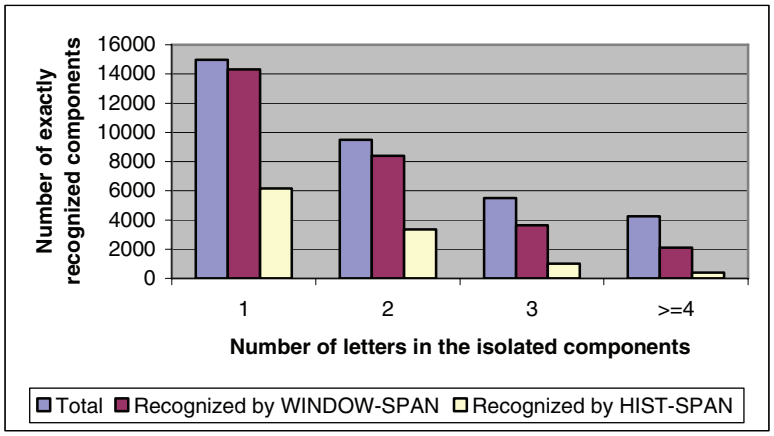


Fig. 10. Number of exactly recognized components by each approach for varying number of characters in the isolated components.

Table V. Recognition Figures for Isolated Components

Approach	# exact matches	# subset matches	# superset matches	# annotated components
WINDOW-SPAN	28438	1277	14	34298
HIST-SPAN	10887	3967	5	34298

the success of the graph-matching method mentioned in this article is sensitive to the artifacts of the thinning process. Therefore its performance is heavily dependent on the success of the thinning algorithm.

In what follows, we select one of the best-performing approaches for each segmentation method, namely WINDOW-SPAN and HIST-SPAN, and compare the results for *exact* recognition of isolated components. More specifically, an isolated component is said to be recognized exactly if all its characters are correctly recognized by the system without any false recognition. It is important to measure component recognition success because we expect the users of a CBR system to submit query words including one or more components. Thus, unlike systems that consider only a single character recognition rate we evaluate the percentage of components that are correctly recognized.

In Figure 10, the first column in the first group represents the true number of isolated components that consist of one character. The second and third columns show the number of correct recognitions for those components using by WINDOW-SPAN and HIST-SPAN approaches, respectively. The remaining groups illustrate recognition rates for isolated components containing two, three, and four (or more) characters.

In Table V, we also provide recognition rates for isolated components, and not only for exact matches but also for the cases where all the characters of a component are recognized and some additional ones are found (i.e., a superset), or when only a subset of the characters in a component is found.

From the previous results, we draw the following observations.

—Spatial features, especially angular and distance span, which are introduced as scale-invariant and rotation-tolerant features in Saykol et al. [2004], are found to be the most successful features of the character recognition task. Coupled with the sliding-windows segmentation, these features yield precision and recall figures of about 90%, which enables the effective use of the OCR output for automatic retrieval [Doerman 1998].

—The NN-based approach coupled with the sliding-window segmentation also works equally well (again with about a 90% success rate). However, it should be noted that this approach also exploits span

vectors. In particular, the span vectors are fed to NNs as input and further employed for direct comparison during the final recognition. The performance of NNs may improve with a larger training set of the number of character examples with various fonts.

—For the graph-based approach, recognition success is much lower than expected. The significant cause of this failure may be the thinning algorithm employed, which has a crucial impact on the performance. Thus, alternative thinning methods from the literature may be applied. Additionally, the graph-matching method used in this article is inherently sensitive to the geometry of the nodes, that is, to their coordinates on the document image. Thus, graph-matching algorithms that do not heavily depend on the coordinates of the nodes may be more useful. Our future work includes experimenting with these alternatives.

—Our experiments reveal that a sliding-window-based segmentation approach usually outperforms a histogram-based one in terms of recognition effectiveness. This may be attributed to the fact that the former method tries almost all possible segments that are greater than a minimum size, whereas the latter method relies on more general parameters to determine the segmentation points of a component. Moreover, in the sliding-windows segmentation method, segmentation is integrated with recognition. However, the histogram segmentation method does not get any feedback from a classifier, thus it is hard to obtain optimal character boundaries. On the other hand, the histogram-segmentation-based method is more efficient than the sliding-windows approach, as it produces a lesser number of segments to be compared with the library.

—In the experiments, we focus not only on the success of single-character recognition but also evaluate the recognition of components as a whole, either exactly or partly. We believe that this is important because the users of a CBR system would specify queries involving one or more such components, therefore the overall success of the system may be measured more realistically by considering the recognition of components.

5. PROTOTYPE CBR SYSTEM FOR OTTOMAN ARCHIVES

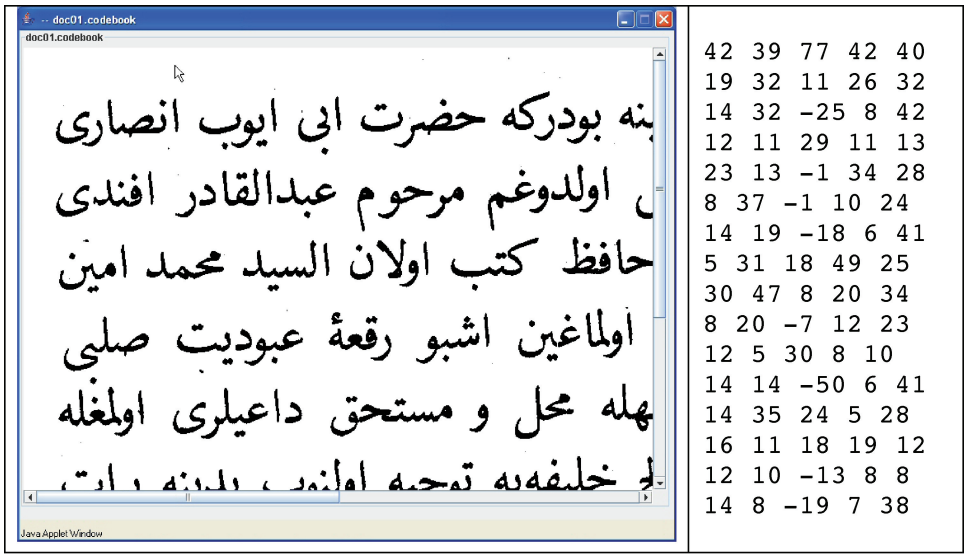
Our prototype system is based on an earlier system [Şaykol et al. 2004], but extends it by constructing a static character library, recognizing characters and specifying queries with a virtual Ottoman keyboard.

In Figure 11, we present a sample document and its corresponding codebook, which includes the ID of the recognized character and its position in the document expressed by four parameters, that is, relative x and y coordinates with respect to the previous character, and width and height. The codebooks are constructed following the practices in earlier works [Witten et al. 1994; Şaykol et al. 2004].

For example, according to Figure 11, the first isolated component extracted from the document includes only one character (similar to character “s”) with the ID 42 in the library. In the Appendix, we describe a codebook-viewer tool that is used to visualize the character recognition output, that is, which component in a document is associated to which character(s) in the library. We envision that this tool may also be used to gather relevance feedback from the users, to improve the success of recognition; this is our current work.

In Figure 12, we provide the Web interface of Ottoman Explorer, the prototype CBR system for Ottoman archives. The user can specify a query in two ways, either by browsing through the available documents and selecting an example region to be searched for, or directly entering the query word in the virtual keyboard, which is on the right-hand side frame of the GUI frame.

In Figure 13, the former method, example-based querying, is exemplified. The user selects a rectangular area including one or more components. The system consults the corresponding document’s codebook to obtain the ID numbers of the recognized characters within the coordinates of the given region, and then looks for the same or similar character sequences in the other codebooks. Finally,



(a) (b)

Fig. 11. (a) An example Ottoman document; and (b) the corresponding codebook.

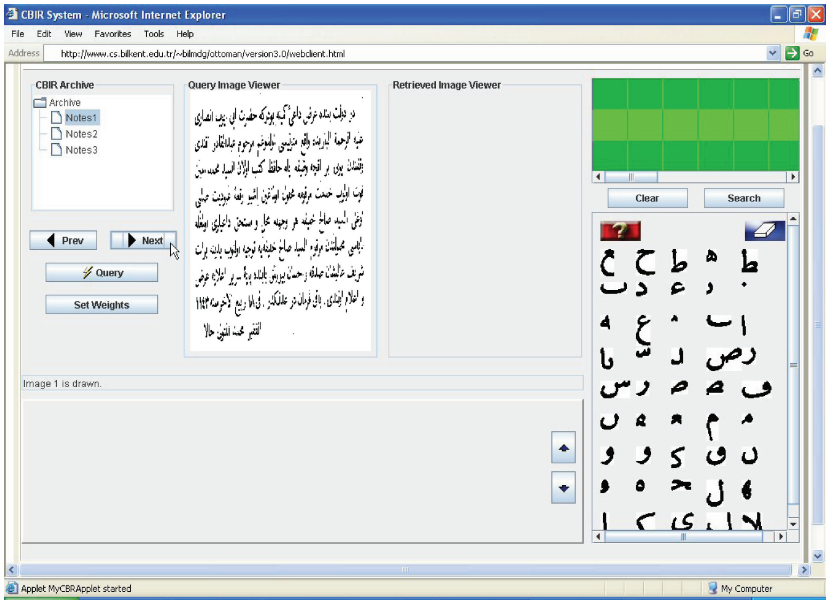
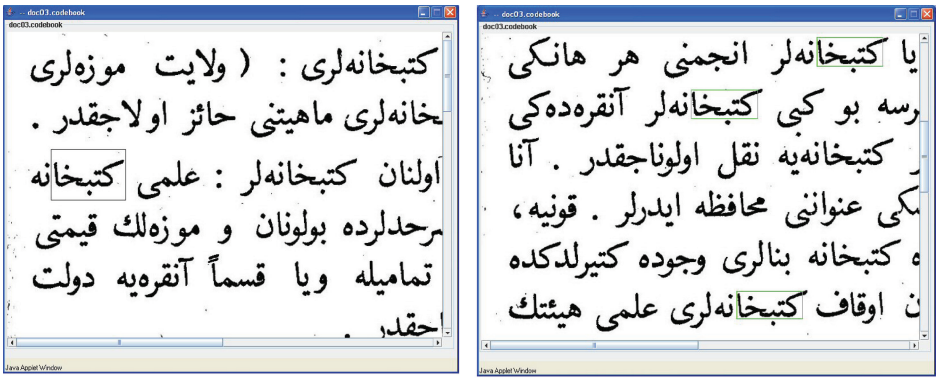


Fig. 12. Web user interface of Ottoman Archives Explorer.

matches are returned to the user in descending order of similarity. While computing the similarity score, each exact matching component is weighted by 1; each component that is a superset or subset of the query component is weighted by 0.5; and each component that has overlapping character(s) with the query components is weighted by 0.25.



(a) (b)

Fig. 13. (a) An example-based query involving an isolated component including multiple characters; (b) one of the result pages including three exact occurrences of the query component.

For a system with acceptable precision and recall figures of single-character recognition and exact-component recognition it is possible to create an inverted index structure to facilitate the search process. In particular, an n-gram index; that is, an index that stores the location of every sequence of n characters, is found to be a useful approach for OCRed documents (see, for instance Beitzel et al. [2003], Parapar et al. [2009], Coetzee [2008]). For our case, it is also possible to index each extracted component separately, or create n-grams of these components for search efficiency. This direction is left as a future work.

Figure 14 illustrates the use of the virtual keyboard. The user enters the query using the alphabet and including all characters forms (isolated, beginning, middle, and end) so that the word may be constructed with the correct character sequence (the connecting lines or curves between the characters are not shown). The query grid includes three lines: the middle one is used for entering the actual query word and the upper and lower lines are for the diacritics.

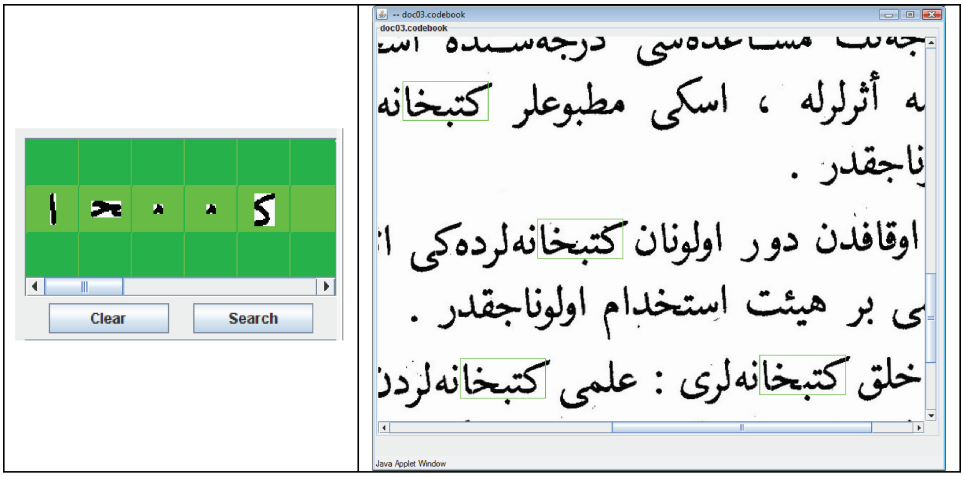
It is important to create an annotated dataset when evaluating and comparing recognition and retrieval systems. In our framework, an annotation tool (presented in the Appendix) is also used to facilitate the creation and exchange of ground-truth data.[1]

6. CONCLUSION

In this article, we present Ottoman Archives Explorer, a content-based retrieval system based on character recognition. The system uses a predefined library of Ottoman characters and employs several segmentation and similarity computation methods to recognize characters. In particular, sliding-window and histogram-segmentation methods are coupled with recognition approaches using spatial features, neural networks, and graphs. The experimental framework includes textual images of historical Ottoman documents that are manually annotated for evaluation purposes, and includes 70,000 characters and 34,000 isolated components, in total.

The recognition performance of the sliding-window segmentation with spatial features is found to be the best, reaching a precision figure of 91% and a recall figure of 89%. The prototype system, which allows querying by example and by using a virtual Ottoman keyboard, is available on the Web. Given

[1]The Ottoman Archives Explorer prototype system is accessible at http://www.cs.bilkent.edu.tr/~bilmdg/ottoman/, as are the datasets and auxiliary tools developed during this study.



(a) (b)

Fig. 14. (a) The same query word as in Figure 13 is specified using the virtual keyboard; (b) one of the returned documents containing occurrences of the query word.

the importance and difficulty of gaining automatic access to Ottoman archives, which include millions of documents with varying styles, this study and prototype system can be seen as one of the first attempts to tackle the problem.

In future work, we plan to: (*i*) improve character recognition performance by adapting more sophisticated graph-matching techniques and gathering user feedback through the GUI; (*ii*) extend the library to other widely used writing styles in Ottoman documents; (*iii*) investigate the use of an inverted index of codebooks for efficient searches on components; (*iv*) obtain a larger annotated dataset for further experimentation; and (*v*) employ a postprocessing stage for correcting recognition errors in OCRed documents.

ACKNOWLEDGMENTS

We are grateful to R. Nelson for proofreading and suggestions.

APPENDIX

CODEBOOK VIEWER AND ANNOTATION TOOLS

Figure 15 depicts the codebook viewer; the characters in the library are shown in the right-hand frame and the segmented document with the character IDs on top of each recognized segment is shown in the main frame. This tool is primarily used to visualize the OCR output during the development process. The user is able to see recognized letters along with their bounding boxes. This enables users to locate resulting OCR errors on documents and change respective parameters of the OCR module for improving the recognition success.

Figure 16 shows the annotation tool for Ottoman documents. At the first step of the annotation, the document image is processed to extract the isolated components automatically. Next, the annotator clicks on a component and selects the characters that are within this component from the library.

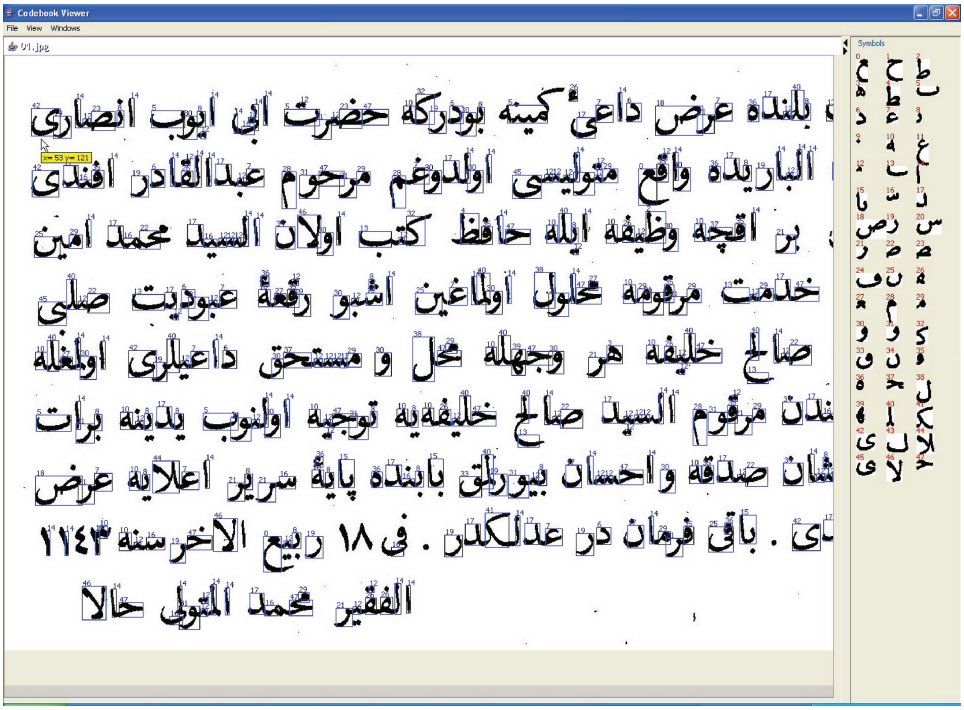


Fig. 15. The user interface of the codebook viewer.

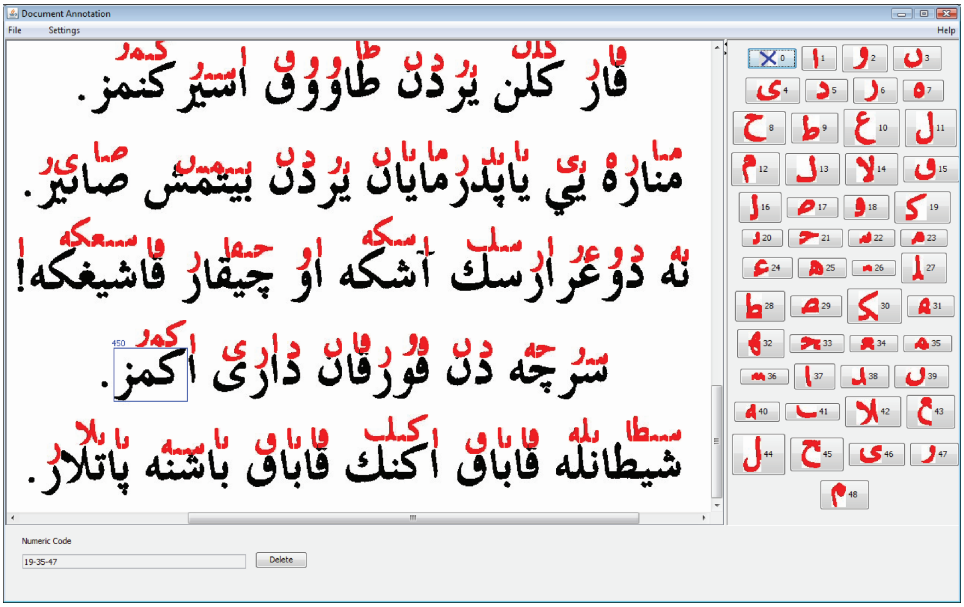


Fig. 16. The user interface of the annotation tool. The selected component (the rectangular area) is annotated with character IDs 19, 35, and 47 as entered in the textbox at the bottom. These characters are drawn as they are recorded in the character library on top of the annotated components for better visualization.

REFERENCES

ALPARSLAN, A. 1999. Osmanlı hat sanatı tarihi. *Yapı Kredi Yayınları*.

ALLAM, M. 1995. Segmentation vs. segmentation-free for recognizing Arabic text. In *Proceedings of the International Society for Optical Engineering*. Vol. 2422, 228–235.

ALTINGOVDE, I. S., SAYKOL, E., ULUSOY, Ö., GUDUKBAY, U., ÇETIN, E., AND GOCMEN, M. 2006. Content-Based retrieval (CBR) system for Ottoman archives (in Turkish). In *Proceedings of the IEEE Sinyal İşleme ve Uygulamaları Kurultayı*.

AMIN, A. 1998. Off-Line Arabic character recognition: The state of the art. *Pattern Recog. 31*, 5, 517–530.

ARCHIVES. 2006. Turkish State Archives Office Web Site. Available at www.devletarsivleri.gov.tr

ARICA, N., AND YARMAN-VURAL, F. T. 2001. An overview of character recognition focused on off-line handwriting. *IEEE Trans. Syst. Man, Cybern Part C: Appl. Rev. 31*, 2, 216–233.

ATAER, E., AND DUYGULU, P. 2006. Retrieval of Ottoman documents. In *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR'06)*.

ATAER, E., AND DUYGULU, P. 2007. Matching Ottoman words: An image retrieval approach to historical document indexing. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval (CIVR'07)*. 341–347.

AVI-ITZHAK, H. I., DIEP, T. A., AND GARLAND, H. 1995. High accuracy optical character recognition using neural networks with centroid dithering. *IEEE Trans. Pattern Anal. Mach. Intell. 17*, 2, 218—224.

BEITZEL, S. M., JENSEN, E. C., AND GROSSMAN, D. A. 2003. Retrieving OCR text: A survey of current approaches. In *Proceedings of the Symposium on Document Image Understanding Technologies (SDUIT)*.

BUNKE, H. 1999. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Anal. Mach. Intell. 21*, 9, 917–922.

CASEY, R. G. 1996. A survey of methods and strategies in character segmentation. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 7, 690–706.

CHAN, J., ZIFTCI, C., AND FORSYTH, D. 2006. Searching off-line Arabic documents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

COETZEE, D. 2008. TinyLex: Static n-gram index pruning with perfect recall. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (CIKM). 409–418.

DEVELI, H. 2006. *Osmanlı Türkçesi Kılavuzu 1*. 3F Yayınevi.

DOERMANN, D. 1998. The indexing and retrieval of document images: A survey. *Comput. Vis. Image Understand. 70*, 287–298.

GOOGLE PRINT. 2006. Google print project. http://print.google.com/googleprint/about.html

KHORSHEED, M. S. 2002. Off-Line Arabic character recognition—A review. *Pattern Anal. Appl. 5*, 31–45.

KILIC, N., GORGEL, P., UCAN, O. N., AND KALA, A. 2008. Multifont Ottoman character recognition using support vector machine. In *Proceedings of the IEEE International Symposium on Control Communication and Signal Processing* (ISCCSP'08). 328–333.

KURT, Y. 2000. *Osmanlıca Dersleri 1*. Akcağ Yayınları.

OARD, D. AND GEY, F. 2002. The TREC Arabic/English CLIR Track. In *Proceedings of the Text Retrieval Conference* (*TREC)*.

OTTOMAN. 2006. Ottoman Turkish language. At http://en.wikipedia.org/wiki/Ottoman_Turkish_language

ÖZTÜRK, A., GÜNES, S., AND ÖZBAY, Y. 2000. Multifont Ottoman character recognition. In *Proceedings of the 7th IEEE International Conference on Electronics Circuits and Systems (ICECS)*. 945–949.

PAPAPAR, J., FREIRE, A., AND BARREIRO, A. 2009. Revisiting n-gram based models for retrieval in degraded large collections. In *Proceedings of the 31th European Conference on IR (ECIR)*. 680–684.

STENTIFORD LIBRARY. 2006. Stentiford thinning Java library. http://www.geocities.com/yccheok81/thinning/index.html

SWAIN, M. J., AND BALLARD, D. H. 1991. Color indexing. *Int. J. Comput. Vis. 7*, 1, 11–32.

ŞAYKOL, E., SINOP, A. K., GÜDÜKBAY, U., ULUSOY, Ö., AND ÇETIN, A. E. 2004. Content-Based retrieval of historical Ottoman documents stored as textual image. *IEEE Trans. Image Proc. 13*, 3, 314–325.

WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1994. *Managing Gigabytes*. International Thompson.

YALNIZ, I. Z., ALTINGOVDE, I. S., GÜDÜKBAY, U., AND ULUSOY, Ö. 2009. Integrated segmentation and recognition of connected Ottoman script. *Optical Engin. 48*, 11, Article 117205.

Received December 2006; revised July 2009; accepted October 2009