A LINE-BASED REPRESENTATION FOR MATCHING WORDS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ethem Fatih Can

December, 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Pınar Duygulu-Şahin (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fazlı Can

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Fatoş T. Yarman-Vural

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute

ABSTRACT

A LINE-BASED REPRESENTATION FOR MATCHING WORDS

Ethem Fatih Can
M.S. in Computer Engineering
Supervisor: Assist. Prof. Dr. Pınar Duygulu-Şahin
December, 2009

With the increase of the number of documents available in the digital environment, efficient access to the documents becomes crucial. Manual indexing of the documents is costly; however, and can be carried out only in limited amounts. Therefore, automatic analysis of documents is crucial. Although plenty of effort has been spent on optical character recognition (OCR), most of the existing OCR systems fail to address the challenge of recognizing the characters in historical documents on account of the poor quality of old documents, the high level of noise factors, and the variety of scripts. More importantly, OCR systems are usually language dependent and not available for all languages. Word spotting techniques have been proposed recently to access the historical documents with the idea that humans read whole words at a time. In these studies the words rather than the characters are considered as the basic units. Due to the poor quality of historical documents, the representation and matching of words continue to be challenging problems for word spotting. In this study we address these challenges and propose a simple but effective method for the representation of word images by a set of line descriptors. Then two different matching criteria making use of the line-based representation are proposed. We apply our methods on the word spotting and *redif* extraction tasks. The proposed line-based representation does not require any specific pre-processing steps, and is applicable to different languages and scripts. In word spotting task, our results are better than the existing word spotting studies in terms of retrieval and recognition performances. In the *redif* extraction task, we obtain promising results providing us a motivation for further and advanced studies on Ottoman literary texts.

Keywords: Historical Manuscripts, Ottoman Texts, Word Image Matching, Word Retrieval, Word Spotting.

ÖZET

KELİMELE EŞLENMESİ İÇİN ÇİZGİ TABANLI BİR NİTELEME

Ethem Fatih Can
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assist. Prof. Dr. Pınar Duygulu-Şahin
Aralık, 2009

Tarihi dokümanların sayısal ortama aktarılması ile, bu dokümanlara hızlı erişim daha çok önem kazanmıştır. Sayısal ortamdaki tarihi dokümanların elle dizinlenmesi çok zaman almanın yanı sıra ancak sınırlı sayıda dokümanlar üzerinde yapılabilmektedir. Bu yüzden otomatik dizinleme önem kazanmaktadır. Optik karakter tanıma (OPT) sistemleri yıllardır çalısılan bir konu olmakla beraber, tarihi dokümanlar üzerinde çoğunlukla istenilen sonuçları vermemektedir. Buna neden olarak, tarihi dokümanların yıpranmış olması, ve yazım şekillerinin farklılıkları gösterilebilir. Daha da önemlisi, OPT sistemleri genellikle tek bir dile odaklı olarak çalışmaktadır, ve farklı diller için çalışan sistemler nadir olarak bulunmaktadır. Kelime tabanlı arama teknikleri, OPT çalışmalarına alternatif olarak sunulmuştur ve kelimelerin tek bir seferde okunduğu prensibine dayanır. Bu tip çalısmalarda, kelimenin harflerini ayrı ayrı incelemek yerine kelimenin bütün olarak incelenmesi esasına dayanır. Yıpranmış ve lekeli dokümanlar, ve farklı yazım şekilleri gibi etkenlerden dolayı, tarihi dokümanlarda tanımlama ve arama, kelime tabanlı arama çalışmalarında da henüz tam olarak çözülememiştir. Bu çalışmada, bu problemlere çözüm olarak basit fakat etkili bir yöntem sunulmuştur; bu yöntemde kelimeler çizgi tabanlı bir niteleme yöntemiyle ifade edilmiştir. Buna ek olarak, iki farklı eşleme yöntemi sunulmuş, ve bu yöntemler kelime eşlemek ve redif bulmak için kullanılmıştır. Çizgi tabanlı niteleme yöntemini kullanan sunduğumuz yaklaşımlar, önceki çalışmaların aksine karmaşık ön işleme safhalarına ihtiyaç duymamaktadır. Kelime eşleme için yapılan deneylerin sonuçlarının, daha önceki çalişmalarda elde edilen sonuçlardan daha iyi olduğu gözlemlenmiştir. Redif bulma işlemi göz önünde bulundurulduğunda ise deneylerin sonuçları, daha detaylı çalışmalar için ümit vaat edicidir.

Anahtar sözcükler: Tarihi Metinler, Osmanlıca Metinler, Kelime Resimlerinde Eşleştirme, Kelime Erişimi, Kelime Tabanlı Arama.

Acknowledgement

First of all, I would like to express my gratitude to Dr. Pınar Duygulu-Şahin, from whom I have learned a lot, due to her supervision, patient guidance, and support during this research.

I am also indebted to Dr.Fazlı Can and Dr.Fatoş T. Yarman-Vural for showing keen interest to the subject matter and accepting to read and review this thesis.

I thank to Dr. Mehmet Kalpaklı for his support.

I am thankful to Dr. Fazlı Can for his continuous guidance, invaluable and generous assistance, encouragement, and support.

Finally, I wish to express my love and gratitude to my beloved family (Kâzım, Güzin, Selçuk, Gülşah, and Aysu) for their understanding and endless love through the duration of my study.

This thesis is supported by TÜBİTAK Grant 104E065, and 109E006.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Efficient access to historical documents becomes crucial with the increase in the number of texts available in the digital environment. Manual indexing of the documents is costly; however, and can be carried out only in limited amounts. As a result, automatic indexing and retrieval systems should be built to make the content available to users.

Even though there are plenty of optical character recognition (OCR) studies [2, 6, 11, 20, 27, 28, 29] in the literature, most of the existing OCR systems fail to address the challenge of recognizing the characters in historical documents because of the poor quality of old documents, the high level of noise factors, and the variety of scripts. Furthermore, OCR systems are usually language dependent and not available for all languages.

Recently, word spotting techniques [1, 3, 4, 16, 19, 23, 22], an alternative to the character-based studies, have been proposed to deal with the problem by matching the words rather than characters. The motivation is the tendency of humans to read whole words at a time [18]. The common approach in the studies based upon word spotting techniques is first to segment the documents into word images, and then apply matching methods to those words for word retrieval or word recognition purposes.

Due to poor quality of historical documents, and variety of scripts the task of recognition and retrieval continues to be a challenging problem for matching the words. In this work, these challenges are addressed by proposing a simple but effective method for line-based representation and matching of word images.

The line-based representation schema is inspired from the idea that words consist of lines and curves (the latter of which can also be approximated by lines) and encouraged by the success of using line segments as descriptors in the task of object recognition [8].

For matching words using the proposed line-based representation, we propose two different matching criteria. In the first matching criterion, we describe words by using line segments extracted from contours of word images. Then the distances between the corresponding line descriptors determine the degree of similarity of the word images. In the latter criterion, we consider un-segmented images and work on contour segments rather than word images. We represent the contour segments as sequence of code words obtained from line descriptors. Then the distances between the sequence of code words determine the degree of similarity of the images.

We focus on two tasks, word spotting and *redif* extraction, to show the effectiveness of the proposed line-based representation and matching criteria.

- **Word Spotting Task**

  To retrieve or recognize the word images in historical documents, we use the line-based representation for word spotting with the application of the first matching criterion. For word matching, we make use of the word images, and represent those images with a set of approximated lines. Then the lines of the word images are compared to build a relationship with other images. We call this method Word Image matching using Line Descriptors (WILD). Going beyond the George Washington data set, which has become a benchmark in the word spotting literature, by applying our method on Ottoman documents provided in [3], we also address the challenge of working on different alphabets and different writing styles. Within the context

of the data sets, we evaluate the performance of our method by computing precision-recall, and word-error-rate scores for retrieval and recognition purposes respectively.

- ***Redif* Extraction Task**

  Redifs are repeated patterns in Ottoman poetry. As a challenging application for word matching, we also work on another task, which we call as *Redif* Extraction using Contour Segments (RECS). In order to extract the *redif* in handwritten Ottoman literary texts, we first find the repeated sequences and then eliminate the ones which are not *redif* using the *redif* rules or constraints. This approach, unlike the other one, works on un-segmented texts, and does not need to have a feed image to query. For testing we constructed a collection of 100 images containing handwritten Ottoman literary texts.

The main contributions of this work can be summarized as follows: (a) We propose an effective and efficient representation of word images based on line descriptors, and (b) Two new matching criteria using descriptors. (c) In word spotting task, we test our method not only on English, as do most of the previous word spotting studies, but also on Ottoman documents. For both sets our method provides promising results without the need for complicated pre-processing steps such as normalization and artifact removal. (d) We also analyze our method in a multi-scaled way by considering different line approximation accuracies in which the distances at different approximation accuracies are combined to compute a final distance. (e) We present a pioneering image-based automatic *redif* extraction method (RECS) for handwritten Ottoman literary texts. To the best of our knowledge, it is a first in the literature.

In the following, previous studies are reviewed firstly. Then the line-based representation is presented, followed by a detailed discussions for the two proposed matching criteria. Next, word spotting and *redif* extraction tasks are explained together with their extensive analysis of experimental results.

Chapter 2

Related Work

Word spotting techniques are introduced as an alternative to OCR systems to access historical manuscripts. With the assumption that multiple instances of a word are written similar to each other by a single author, words are represented by simple image properties.

In the studies of Manmatha et al. [19, 23, 22], words are represented by image properties such as projection profiles, word profiles and background/ink transitions. Compared to other techniques such as sum of squared differences (SSD), and Euclidean distance mapping (EDM), dynamic time warping (DTW) is shown to be the best method for matching words.

Even though word spotting studies mostly focus on the task of word retrieval, in recent studies [24, 1] the task of word recognition is also considered. Rath and Manmatha [24] use clustering to recognize words, and the authors state that clustering is better than techniques based on hidden Markov model (HMM) when word error rate (WER) is considered. In [5], they use DTW to match the words in printed documents using profile-based and structural features. In another study, [26], they use a similarity measure based on corresponding interest points.

In [1], so as to eliminate the limitations of profile-based or structural features that depend on slant angle and skew normalizations, Adamek et al. propose a

contour-based approach to match the image words. They extract the contours of the image after several processes, including binarization with adaptive pixel-based thresholding, as well as removing artifacts (e.g. segmentation errors) and diacritical marks, and produce a single closed contour. Then they employ the multi-scale convexity/concavity (MCC) representation, which stores the convexity/concavity information and utilizes DTW for matching.

Most of the word spotting studies make use of DTW in the computation of the dissimilarity between the image words. The main issue with DTW-based studies is the complexity of running time. Kumar et al. [13] make use of the locality sensitive hashing (LSH) technique for increasing the speed, and focus on documents in Indian.

There are studies which do not employ DTW in the word spotting literature. Leydier et al. [16] use gradient angles as features and variations of elastic distance. In the study they search for a template word in the whole document instead of using the segmented word images. However, speed remains a problem for this study as well.

A statistical framework on a multi-writer corpus is proposed by Rodriguez-Serrano and Perronnin [25]. They make use of the continuous Hidden Markov Model (C-HMM) and semi-continuous Hidden Markov Model (SC-HMM) and demonstrate that their method outperforms the approaches based on DTW for word image distance computation.

Konidaris et al. [12] propose an algorithm for the word spotting task that combines synthetic data and user feedback. They focus on the printed Greek documents in their study.

Ataer and Duygulu [3] extract the interest points from word images by using Scale Invariant Feature Transform (SIFT) operator [17]. A codebook obtained by vector quantization of SIFT descriptors is then used to represent and match the words. The method is tested on Ottoman documents.

Chapter 3

Line-based Word Representation

In this thesis, a line-based representation schema for matching words in historical manuscripts is proposed. Starting from the idea that the words consist of lines and curves, and encouraged by the success of using line segments as descriptors for object recognition [8], the words are described using line segments extracted from the contours of images.

In what follows the details of the proposed line-based representation is provided. The schema consists of four steps; binarization, contour extraction, line approximation, and line description.

3.1 Binarization

Most of the existing studies employ complex and costly pre-processing steps including binarization. In this work, we prefer to use simple thresholding methods to binarize the images.

For word spotting task, we use the global thresholding in which the image is thresholded with a global value that is the average intensity value of the pixels of the image. In Fig. 3.1 original gray-scale images and their binarized versions are provided for some sample word images used in word spotting task.

Figure 3.1: Original gray-scale images (left) used in word spotting task and their binarized versions (right) are provided.

For *redif* extraction task, we prefer to use more advanced thresholding method, namely the Otsu method [21], than the global thresholding. Since the images used in this task are in worse shape than the ones used in the word matching task. In Fig. 3.2 original gray-scale images and their binarized versions are provided for some sample images used in *redif* extraction task.

Figure 3.2: Original gray-scale images (left) used in *redif* extraction task and their binarized versions (right) are provided.

3.2 Extraction of Contour Segments

As the next step, the connected components are found using 8-neighbors and the contour segments are extracted from these connected components. In Fig. 3.3, and Fig. 3.4 extracted contour segments are provided for word spotting, and *redif* extraction tasks.

Figure 3.3: The contour segments extracted from the binarized images for word spotting task.

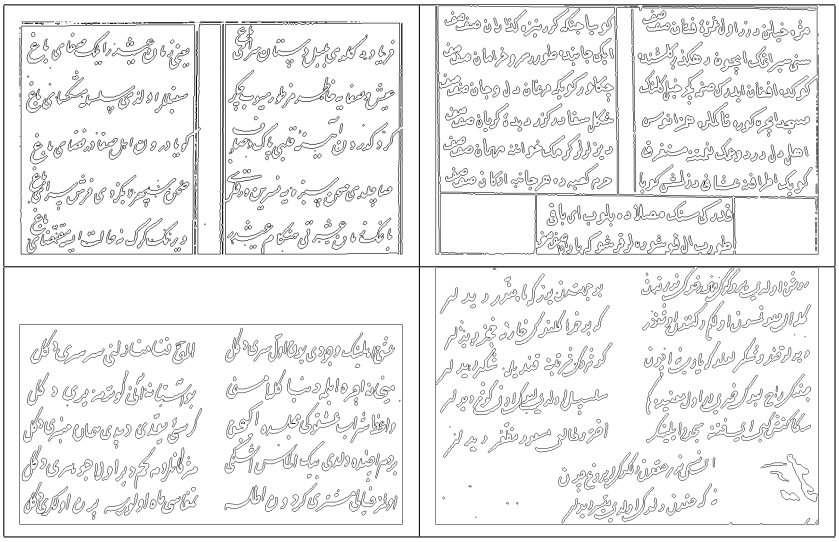

Figure 3.4: The contour segments extracted from the binarized images for *redif* extraction task.

3.3 Line Approximation

We do not use the contour itself for matching but fit lines to the points of the contour segments. The points on the contours are approximated into lines with the method summarized in Algorithm 1. For each contour segment in a word image, we make use of Douglas-Peucker method to fit lines to the points of the contour segments. The approximation method returns number of points representing the start and end points of the fitted lines.

Let $C = \{c_1, c_2, ...\}$ *be extracted contour segments and*
ζ *be set of approximated lines on contour segments;*
$\zeta = \emptyset$;
foreach *contour segment* $c_i \in C$ **do**
  ψ_i = points on c_i;
  ζ_i = Douglas-Peucker(ψ_i, τ);
  $\zeta = \zeta \cup \zeta_i$;
end

Algorithm 1: Pseudo code of line approximation on contour segments.

Line approximation is performed using the Douglas-Peucker algorithm which was first proposed in [7], and then improved by Hershberger and Snoeyink [9] in terms of cost for worst-case running time from quadratic n to $nlog_2(n)$ where n is the number of points. The line approximation method can be summarized in the following way. The Douglas-Peucker algorithm gets a set of points ψ, and τ as input parameters. It first finds the furthest point to the line in which start, and end points are $\psi[first]$, and $\psi[last]$ respectively. Then the algorithm checks whether the distance from furthest point to the line is greater or equal to the value of τ or not. If the condition is satisfied, the algorithm calls itself with the parameters; $(\psi[start, ..., index\ of\ furthest\ point], \tau)$, and $(\psi[index\ of\ furthest\ point, ..., last], \tau)$ (these recursive calls last when all the points in ψ are visited.) Otherwise, the points are kept. These recursive calls last when all the points in ψ are visited, and the points kept are returned as the start, and end points of the approximated lines.

The resulting approximated lines on the points of the contour segments extracted from the images are provided in Fig. 3.5 for word spotting task, and in

Fig. 3.6 for *redif* extraction task.

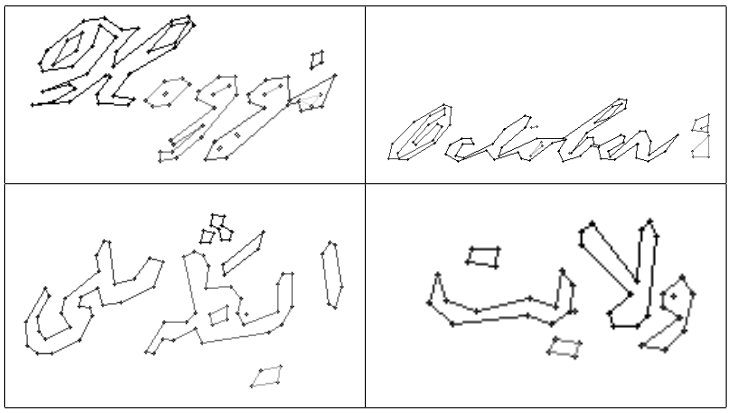

Figure 3.5: The approximated lines on the points of the contour segments for word spotting task. The points on the lines are the start and end points of the lines.

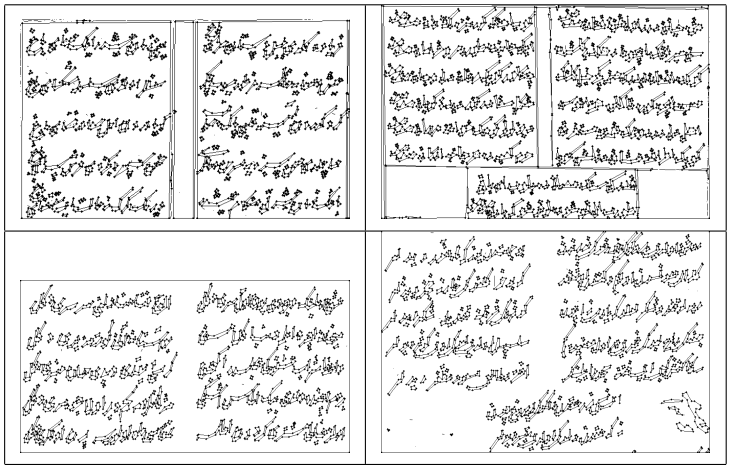

Figure 3.6: The approximated lines on the points of the contour segments for *redif* extraction. The points on the lines are the start and end points of the lines.

Note that the Douglas-Peucker algorithm may return more than one line for a single contour segment. The parameter τ in the Douglas-Peucker algorithm can be defined as approximation accuracy, tolerance value, or compression factor. It serves the determination of key points when fitting lines into points. The greater values of τ result in a smaller number of lines and sharper segments, while smaller values of τ result in greater number of lines and smoother segments. The effect of different τ values on word retrieval and recognition will be later explained in detail.

3.4 Line Description

We describe a line ℓ using the position, orientation, and length information as in [8]:

$$\ell = \{p_s, p_m, p_e, \theta, \rho\} \tag{3.1}$$

As illustrated in Fig. 3.7; $p_s = (x_s, y_s)$ is the start point, $p_m = (x_m, y_m)$ is the mid-point, $p_e = (x_e, y_e)$ is the end point, θ is the orientation, and ρ is the length of the line.

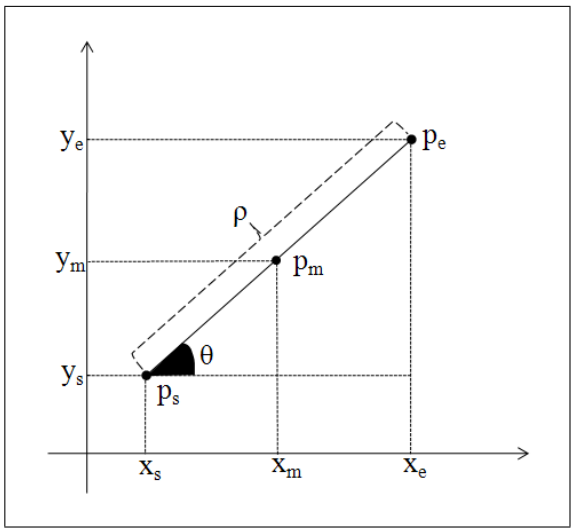

Figure 3.7: Start point (p_s), mid-point ($p_m = r$), and end point (p_e), orientation (θ) and length (ρ) of a line that is approximated on the points of a contour segment.

Chapter 4

Word Matching

Using the proposed line-based representation, two criteria are proposed for matching words. The first matching criterion requires the segmented word images to be provided and matches words based on corresponding line descriptors. The latter one is proposed for the un-segmented documents. In this case the contour segments are found and represented as sequence of code words obtained by vector quantization of line segments. The contour segments are then matched using string matching techniques.

4.1 Matching Segmented Word Images using Corresponding Line Descriptors

In this method, each word image I is represented as a set of line descriptors as $I = \{\ell_1, \ell_2, ..., \ell_N\}$ where N is the number of lines approximated for the word image. Then we normalize the line descriptors of each word image by re-arranging the positions of the lines depending upon the center point (X_I,Y_I) of the word image I.

The center point (X_I,Y_I) is computed using the mid-points of the lines in the word image as:

$$X_I = \frac{\sum x_m^i}{N},\ Y_I = \frac{\sum y_m^i}{N}, i = 1 \dots N, \tag{4.1}$$

where x_m^i and y_m^i are the x and y coordinates of the mid-point of line ℓ_i. Representative points of each line descriptor are re-arranged as defined below.

$$\begin{aligned} p'_s &= (x_s - X_I, y_s - Y_I) \\ p'_m &= (x_m - X_I, y_m - Y_I) \\ p'_e &= (x_e - X_I, y_e - Y_I) \end{aligned} \tag{4.2}$$

Having re-arranged the position information of the line descriptors, we also normalize the line descriptors by dividing the position parameters to the farthest distance to the center point of the line descriptors.

We prefer to use only p'_m and refer to it as r when comparing the positions of the line descriptors to compute the distance between the descriptors. Since using the mid-point information is sufficient to determine the similarity between the line descriptors in terms of position.

In order to find a matching score, we first find corresponding line descriptors in two word images and compute the distances between them.

The distance between the two line descriptors, ℓ_a and ℓ_b are computed by the dissimilarity function introduced in the study of Ferrari et al. [8] as:

$$d(\ell_a, \ell_b) = 4d_r + 2d_\theta + d_l \tag{4.3}$$

where $d_r = |r_a - r_b|$, $d_\theta = |\theta_a - \theta_b|$, and $d_l = |log(\rho_a/\rho_b)|$.

The first term is the difference of the relative positions of the mid-points of lines (r_a and r_b). The second term is the difference between the orientations of the lines where $\theta_a, \theta_b \in [0, \pi]$. The third term is the logarithmic difference between the lengths of the lines (ρ_a and ρ_b). For each line segment, we take the one with

minimum distance as the corresponding line segment and construct matched pair of lines.

Having computed the distance between the line descriptors, we make use of the matched pair of line descriptors to compute the distance between the word images.

The similarity of two word images I^a and I^b which are described as $I^a = \{\ell_1^a, \ell_2^a, ..., \ell_{N_a}^a\}$ and $I^b = \{\ell_1^b, \ell_2^b, ..., \ell_{N_b}^b\}$, are computed based on the values $d(\ell_i^a, \ell_j^b)$ where $i = 1, 2, ..., N_a$ and $j = 1, 2, ..., N_b$. For each line ℓ_i^a in I^a, we search for the best matching line ℓ_j^b in I^b by finding the line with minimum distance. That is; (ℓ_i^a, ℓ_j^b) is a matching pair, if $d(\ell_i^a, \ell_j^b) < d(\ell_i^a, \ell_k^b)\ \forall_k$, $j \neq k$, $k = 1, 2, ..., N_b$. If two or more lines in I^a match to a single line in I^b then we choose the one with the minimum distance and eliminate the others. The final distance between two images is then computed as the sum of dissimilarity score of some of the best matches. For example; $I^a = \{\ell_1^a, \ell_2^a, \ell_3^a\}$ and $I^b = \{\ell_1^b, \ell_2^b, \ell_3^b, \ell_4^b\}$ and the minimum matches are $\{(\ell_1^a, \ell_3^b), (\ell_2^a, \ell_2^b), (\ell_3^a, \ell_2^b)\}$, in this case the total dissimilarity value of I^a and I^b is computed from the matches as $D_{a,b} = d(\ell_1^a, \ell_3^b) + min(d(\ell_2^a, \ell_2^b), d(\ell_3^a, \ell_2^b))$. Note that $D_{a,b} \neq D_{b,a}$. The dissimilarity score is defined below:

$$(D_{a,b}) = \sum d(\ell_i^a, \ell_j^b) \tag{4.4}$$

where $\ell_j^b = match(\ell_i^a)$.

Considering the example given in Fig. 4.1; $I^a = \{\ell_1^a, \ell_2^a, \ell_3^a\}$ and $I^b = \{\ell_1^b, \ell_2^b, \ell_3^b, \ell_4^b\}$ and the minimum matches are $\{(\ell_1^a, \ell_3^b), (\ell_2^a, \ell_2^b), (\ell_3^a, \ell_2^b)\}$, in this case the total dissimilarity value of I^a and I^b is computed from the matches as $D_{a,b} = d(\ell_1^a, \ell_3^b) + min(d(\ell_2^a, \ell_2^b), d(\ell_3^a, \ell_2^b))$.

In order to compute the final matching score, instead of using only the distance between the images, we prefer considering other values as well: the number of hits $h_{a,b}$ as the number of matches between two images (in the example above the number of hits is 2, $h_{a,b} = 2$), and the number of lines in the images N_a and N_b. We normalize the dissimilarity value $D_{a,b}$ between two images I^a and I^b as

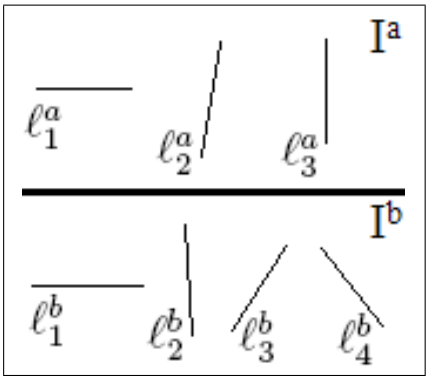

Figure 4.1: Illustration of matching pairs of line descriptors of the images I^a and I^b to compute the dissimilarity score.

defined in Eq. 4.5.

$$f(I^a, I^b) = D_{a,b}\left(\frac{(N_a - h_{a,b})^2 + (N_b - h_{a,b})^2}{\sqrt{[(N_a)^2 + (h_{a,b})^2][(N_b)^2 + (h_{a,b})^2]}}\right) \tag{4.5}$$

In the equation above $h_{a,b}$ is the number of hits while matching word images I^a and I^b. N_a and N_b are the number of line descriptors, $D_{a,b}$ is the sum of the total distances of the matched line descriptors of I^a and I^b, and $\mathbf{f}(I^a,I^b)$ is the final score while comparing the images I^a and I^b. The equation above changes the value $D_{a,b}$ in that images with a small difference between the number of line descriptors and the number of hits have more chance of being matched than images in which the difference is greater.

Finally, we construct a global distance matrix $\mathbf{F}$ with the size of $Q \times Q$, where Q is the number of word images in the test bed, using $f(I^a, I^b)$ values that are the dissimilarity values between the images, so that $F(a, b) = f(I^a, I^b)$. For instance, $F(1, 3)$ is the dissimilarity value between the first and third image in the data set.

The only parameter introduced in our approach, τ, has an important role in determining the lines in the approximation process. In other words, for different values of τ, the points of the contour segments are approximated into lines in

different scales. We empirically observe that considering different values of τ affects the results of the experiments. We also carry out experiments based on multi-scale line descriptors by simply taking into account the line descriptors for different values of τ. The issue is covered in detail in the next section.

4.2 Matching Contour Segments Represented as Sequence of Code Words

The second matching criterion focus on contour segments for matching. When segmented words are not available the most basic unit, unlike the first matching criterion, may not be the word, instead a contour segment C, which may represent a word, a character, or a sequence of characters. This matching criterion removes the dependency of segmented documents for matching.

Each contour segment consists of number of line descriptors. By quantizing the information of the line descriptors of the contour segments, we represent each contour segments with a sequence of code words in which these code words are then used to find the similarity between them.

4.2.1 Contour Segment Description

A line descriptor ℓ contains position, length, and orientation information as stated before. The position information of a line descriptor gives the global x and y coordinates of the descriptor in the image. This information is not a discriminative criterion when matching two contour segments in the case of un-segmented documents. Similarly, length and orientation may not be a discriminative criteria since the texts are handwritten; words might have different sizes and orientations than others even in the same image. Considering the different writing styles, relative length and orientation information can provide more discriminative criterion than actual information. In order to have a better discriminative criterion, we normalize the line descriptors. We first find the center point (X_C, Y_C) of

all line descriptors in a contour segment, and define a reference line descriptor $\ell^r = (p_m^r, \theta^r, \rho^r)$ (similar to WILD we only consider the mid-point for the position information) which is the line descriptor having the minimum distance to the center point (X_C, Y_C). Then, we represent each line descriptor of a contour segment depending upon their reference line descriptor, and define a contour segment descriptor C' using the normalized lines $\ell' = (p'_m, \theta', \rho')$ of the contour segment. The center point (X_C, Y_C) of the line descriptors of a contour segment is computed in the following way.

$$X_C = \frac{\sum x_m^i}{n}, Y_C = \frac{\sum y_m^i}{n}, i = 1, 2, ..., n \tag{4.6}$$

The normalization is performed using the reference line descriptor in the following way.

$$\begin{aligned} x'_m &= x_m - x_m^r \\ y'_m &= y_m - y_m^r \\ \theta' &= \theta - \theta^r \\ \rho' &= \rho / \rho^r \end{aligned} \tag{4.7}$$

Having defined the contour segment descriptors, we construct a codebook B, and represent each contour segment descriptor with a sequence of code words. Similarities among the contour segment descriptors are obtained with a string matching algorithm by using the code words.

4.2.2 Finding similarity of contour segment descriptors

In order to compute the distance $dist(C'_i, C'_j)$ between two contour segment descriptors, C'_i, and C'_j represented by the elements of the generated codebook, we find the amount of difference between sequences of codes of the contour segment descriptors [15]. The difference is the sum of insertions, deletions, and

substitutions of a single label in codes of the contour segment to transform codes of one descriptor to the other. For example; the distance $dist(C'_i, C'_j)$ of $C'_i = \{b_{10}, b_{21}, b_{33}, b_7\}$, and $C'_j = \{b_{10}, b_{33}, b_{33}\}$ is 2. Since the second code of the C'_i should be deleted, and b_7 should be substituted with b_{33}. With two operations; a deletion and a substitution, the distance turns out to be 2. We use the distances between contour segment descriptors to rank the matching images for a given contour segment descriptor.

Chapter 5

Word Spotting Task

To retrieve and recognize the word images in historical documents we use the line-based representation for word spotting. In this task, we make use of segmented documents with the first matching criterion, matching segmented word images using corresponding line descriptors.

In this chapter, we first give details of experimental environment, and provide the results of the experiments of the word spotting task.

5.1 Experimental Environment

5.1.1 Data sets

In word spotting task, we focus on data sets used in previous studies, for which the segmented word images and annotations are available. The first set of the images is from the George Washington (GW) Collection in Library of Congress, which is used as a benchmark data set in word spotting literature. We used the two available data sets constructed from the GW Collection. The first set of data used in this study is ten pages with 2381 words [23, 22, 26] (hereby referred to as GW10), and the second one is twenty pages with 4860 words [1, 23] (hereby

referred to as GW20) from the George Washington collection in the Library of Congress. The documents in GW sets form part of books of letters written between 1754 and 1799. The documents are of acceptable quality; however, some word images have artifacts or do not have any words at all due to segmentation errors (see Fig. 5.1).

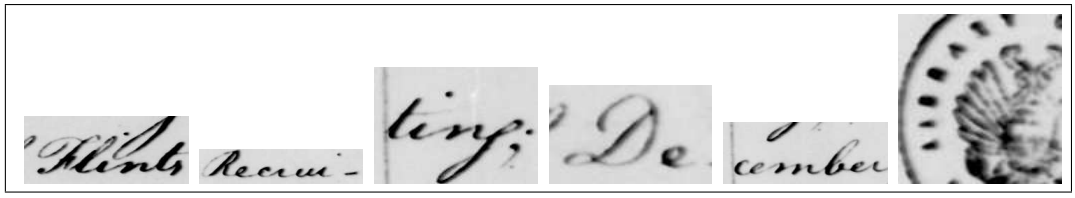

Figure 5.1: Example word images from GW collections.

In order to test the effectiveness of our approach on other documents with different alphabets, especially on those with diacritical marks, we prefer to use the Ottoman sets provided by Ataer and Duygulu [3, 4]. The first collection in the Ottoman sets consists of 257 words in three pages of text (hereby referred to as OTM1), and the second one consists of 823 words in six pages of texts (hereby referred to as OTM2). The third one is the combination of OTM1 and OTM2 sets (hereby referred to as OTM1+2). While the documents in OTM2 set are printed, those in OTM1 set are handwritten. OTM1 set is written with a commonly encountered calligraphy style called Riqqa, which is used in official documents. OTM2 set consists of printed documents on the list of books in the library [4]. The documents are of acceptable quality; however, the segmented images have artifacts (see Fig. 5.2).

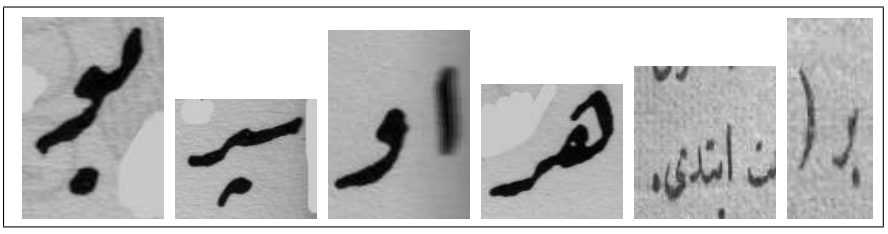

Figure 5.2: Example word images from Ottoman sets.

5.1.2 Evaluation Criteria

In our study we mainly focus on retrieval; therefore, the results are mostly provided in terms of precision scores and analyzed for retrieval. However, some previous studies test their methods in terms of recognition rate. Thus, in order to compare our results with the studies offering recognition rates we also provide these rates.

In order to obtain the precision and recall values we use the `trec_eval` package provided by the National Institute of Standards and Technology (NIST), which is a common tool used in the literature. All the precision values given in this study are the average precision scores computed using `trec_eval`,as in [23].

We also use the score of word error rate to compare our results with other studies that provide WER. In most of those studies, researchers use 20-fold cross validation by choosing the number of folds as the number of pages. In other words, the words on a page are tested with words on other pages to compute the recognition rates. The final recognition rate is provided as the average of the recognition rates of each iteration in the cross validation process. For each page the recognition rate is computed by taking the ratio of the total number of correct recognitions and the total number of words on that page. WER is computed for the words in a test page as follows:

$$WER = 1 - (\frac{\#correct\ matches\ in\ test\ page}{\#words\ in\ test\ page})$$

In order to determine the number of correct matches on a test page, one-nearest neighbor approach is used. We provide two different types of WER; the first one considers the Out of Vocabulary (OOV) words, and the latter does not consider OOV words. Note that a word is called an "Out of Vocabulary" word when the word appears on the test page but not in the other pages.

5.2 Experimental Results

5.2.1 Evaluation of the parameter τ

We deploy the parameter τ in our approach as mentioned before. For different values of τ we employ a global distance matrix such that $F_{\tau=2.0}$ is the distance matrix for $\tau = 2.0$. Having constructed the matrices, we run tests on each collection by setting various values of τ ranging from 0.5 to 5.0, with an increment of 0.5. We empirically determine the best value for each set, which turns out to be 2.5, as it provides the highest precision score. The precision scores for different τ values of GW10, GW20 and OTM1+2 collections are given in Fig. 5.3, all the recall scores are 1.0 in our experiments in our method. Note that the precision scores in Fig. 5.3 represent the results of different τ values.

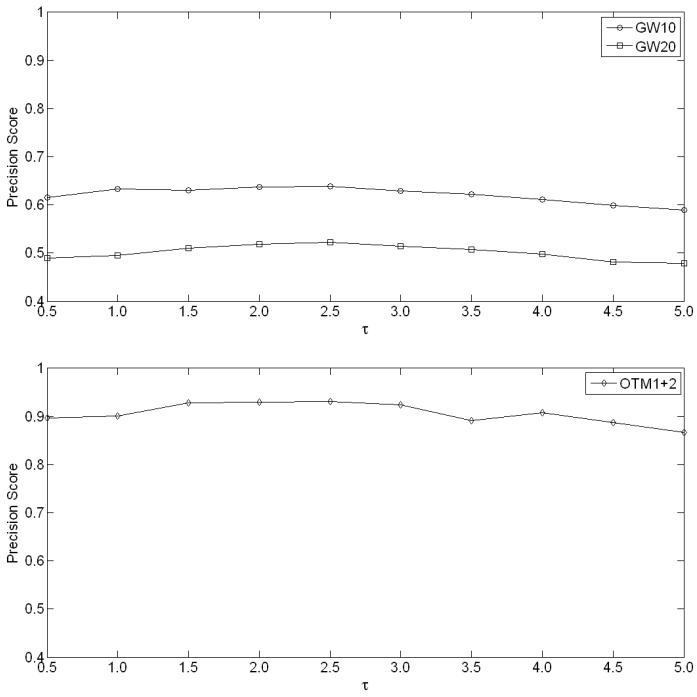

Figure 5.3: The precision scores for different τ values of GW10, GW20 and OTM1+2 sets. The figure on the top is the results of GW collections, and the other one is the results of OTM1+2 collection.

For $\tau = 2.5$ we obtain the precision scores of 0.638 and 0.523 for GW10 and GW20 sets respectively. A precision score of 0.931 is obtained for OTM1+2 set in our experiments. Even though $\tau = 2.5$ provides the highest precision scores, the results at $\tau = 2.0$ and $\tau = 3.0$ have close precision scores.

In addition to experiments considering single τ values, we also carry out experiments in which we combine the results of different τ values by summing the dissimilarity scores. We empirically test different coefficients while adding these scores, and observe that using coefficients does not change the results significantly. In order to combine two or more results at different tolerance values we simply sum the matrices, such that $F' = F_{\tau=2.0} + F_{\tau=2.5}$, where the matrix F' is the distance matrix constructed by combining the distance matrices for $\tau = 2.0$ and $\tau = 2.5$. Then, we use the distance matrix F' to compute the precision or recognition scores.

We observe that combining the distance matrices of individual τ values allows us a multi-scale approach and helps to obtain higher precision scores and recognition rates than using the distance matrices individually. Experiments considering the dissimilarity values at different approximation accuracies (as stated before, τ is called approximation accuracy, tolerance value, or compression factor) mostly eliminate false line descriptor matches, which may appear in experiments considering a single approximation scale.

We construct the distance matrix F' for all combinations of τ values; { ($\tau = 0.5$, $\tau = 1.0$),...,($\tau = 0.5$, $\tau = 1.0$, $\tau = 2.5$),...,($\tau = 0.5$, $\tau = 1.0$, $\tau = 1.5$, $\tau = 2.0$, $\tau = 2.5$, $\tau = 3.0$, $\tau = 3.5$, $\tau = 4.0$, $\tau = 4.5$, $\tau = 5.0$) }. We empirically observe that results of τ values greater than 5.0 display lower precision and recognition rates. For this reason, we do not consider the results of those τ values. In Table 5.1 we provide the results of some of the combinations that consider distance matrix F' constructed with various combinations of τ values. The distance matrix for the case on the top row of the table is constructed as $F' = F_{\tau=0.5} + F_{\tau=1.0}$, and the distance matrices for other cases are constructed in a similar way. The row with a † sign provides the highest precision scores for the sets; hereby the results are provided by considering the distance matrix of that combinations which is defined

Table 5.1: Precision scores of some of the experiments that combine distance matrices of various τ values for GW10, GW20 and OTM1+2 sets. P: Precision, OTM1+2: The combination of OTM1 and OTM2 sets. Recall scores are always 1.0.

τ										GW10	GW20	OTM1+2
0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	P	P	P
✓	✓									0.652	0.534	0.940
✓	✓	✓								0.662	0.549	0.937
✓	✓	✓	✓							0.673	0.535	0.939
✓	✓	✓	✓	✓						0.679	0.543	0.947
✓	✓	✓	✓	✓	✓					0.683	0.547	0.950
✓	✓	✓	✓	✓	✓	✓				0.686	0.551	0.952
† ✓	✓	✓	✓	✓	✓	✓	✓			0.688	0.566	0.957
✓	✓	✓	✓	✓	✓	✓	✓	✓		0.688	0.564	0.957
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0.687	0.565	0.956
◇ ✓	✓				✓					0.675	0.542	0.955
◇✓			✓				✓			0.675	0.541	0.948
◇✓	✓		✓			✓				0.684	0.549	0.950
◇	✓			✓	✓				✓	0.680	0.545	0.948
◇✓	✓		✓		✓				✓	0.686	0.552	0.953

as $F' = F_{\tau=0.5} + F_{\tau=1.0} + F_{\tau=1.5} + F_{\tau=2.0} + F_{\tau=2.5} + F_{\tau=3.0} + F_{\tau=3.5} + F_{\tau=4.0}$. Even though the combination of more distance matrices of different τ values provide higher precision scores, there are some distance matrices that yields closer results (See the rows with ◇). Our system preserves its multi-scale characteristic by choosing some sample scales corresponding to the combinations of results in a few τ values rather than using all of the τ values. The results where we combine four different τ values are close to each other and approach the best score obtained.

The following subsection covers the results of the experiments. The results of the experiments on GW and OTM sets are provided separately.

Figure 5.4: The first 10 retrieval results for querying the keywords “December”, “Instructions”, “should”, and “1755” in the GW10 set. The order is top to bottom and the images on the most top position are the keywords.

5.2.2 Results on GW data sets

In Fig. 5.4 we provide the retrieval results for the keywords “December”, “Instructions”, “should”, and “1755.” and show the first ten matches. Note that the results retrieved by the algorithm for the keyword “should” display character mismatches and the query of the words “December”, and “1755.” also yield some partially matching results.

For the query of the word “December” five exact matches, two partially matched words (“Vc.Decembe” and “Decembe”), and two false matches (“Recruits” and “Buckner”) are retrieved. The two partially matched words are almost the same as the query word. As the characteristics of the lines of the false matches are very close to the lines of the query word, our method retrieves these

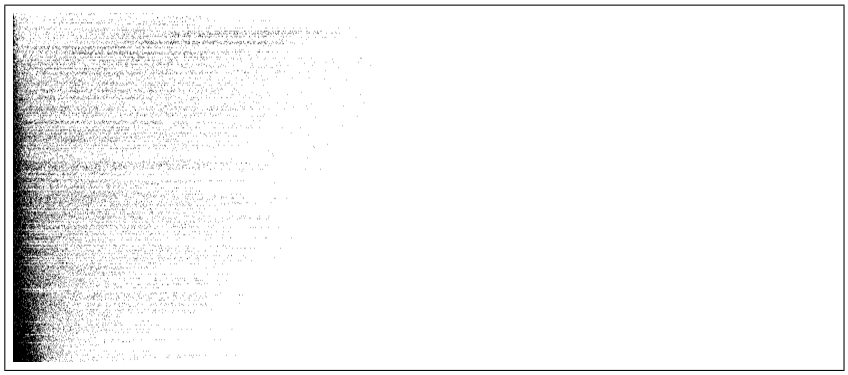

Figure 5.5: Word-Rank representation for the words in GW10 data set which have forty or more relevant images. Each row is a result of query for a different word. A black point means a correct match.

words in the initial ranks. Similarly, in the query of the word "1755." our method retrieves partially matched words as well as exact matches. Eight words out of ten exactly match, whereas "3,1755" partially match the query word. The situation holds for other queries such as "particular-particularly", "he-the", "you-your", "recruit-recruits", and 'me-men".

In this study, the best precision score we obtain for the GW10 set is 0.688. As stated before, in this experiment the distance matrix is constructed as $F' = F_{\tau=0.5}+F_{\tau=1.0}+F_{\tau=1.5}+F_{\tau=2.0}+F_{\tau=2.5}+F_{\tau=3.0}+F_{\tau=4.0}$, and the final precision score is computed by considering the retrievals based upon the dissimilarity matrix constructed above (this case appears in the row with a † sign in Table 5.1). Experimental results show that the distance matrix constructed by combining the distance matrices of τ values from 0.5 to 4.0 also yields the highest precision and recognition results for also the other sets used in this study.

In Fig. 5.5 the word-rank representation of the GW10 set is provided. The queries appearing on Fig. 5.5 are for the words which have forty or more relevant images in the data set. Our method manages to retrieve most of the relevant images in the initial ranks, with the consequence that few images remain to be retrieved in the following ranks - a situation reflected as a large white area occupying most of the image, beginning from the right side, and all blacks are on

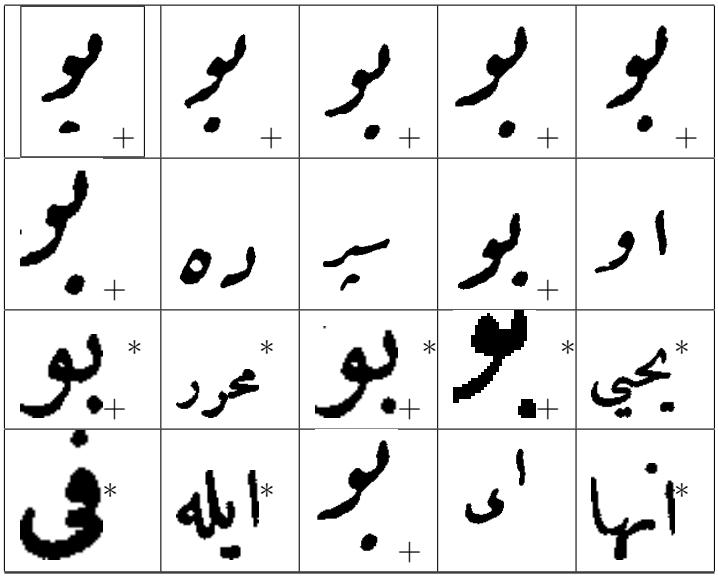

Figure 5.6: The first 20 retrieval results for querying keyword “bu (this)” in OTM1+2 set. The order is top to bottom, left to right. The image on the top-left most position is the keyword. Images with a plus sign are the correct matches. Images with a star sign are from the OTM2 set and the others are from OTM1.

the left side.

The highest precision score for the GW20 set is 0.566 (See Table 5.1). We also provide WER for GW20 set since some of the previous studies employ WER in testing their methods. In terms of WER we obtain a score of 0.303 considering out of vocabulary (OOV) words, and 0.189 when disregarding the OOV words.

5.2.3 Results on OTM data sets

In Fig. 5.6 the retrieval results of the query for the keyword “bu” (meaning “this”) is displayed. The keyword is searched in the set, which is a combination of the OTM1 and OTM2 sets, the OTM1+2 set. Note that the images have different sizes, as illustrated in the figure.

In Fig. 5.7, the word-rank representation of the OTM1 and OTM2 data sets is provided. Our method manages to retrieve most of the relevant images in the

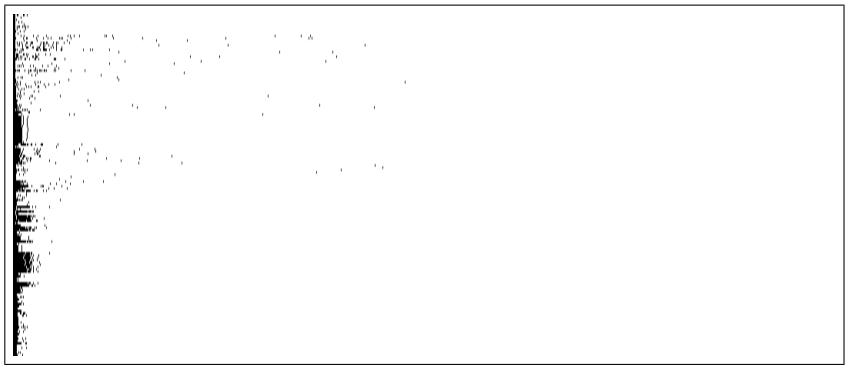

Figure 5.7: The Word-Rank representation for the words in the combination of the OTM1 and OTM2 data sets which have five or more relevant images. Each row represents a query for a different word. A black point means a correct match.

initial ranks, with the consequence that few images remain to be retrieved in the later ranks. Fig. 5.7 represents the queries for words having five or more relevant images in the combination of OTM1 and OTM2 data sets.

For the sets in Ottoman language the best scores we obtain is 0.987 and 0.944 for OTM1 and OTM2 respectively. Since the characteristics of the sets are different we construct a new set by combining the words of OTM1 and OTM2 sets, the OTM1+2 set. The highest precision score we obtain on this combined set is 0.957.

5.2.4 Analysis of the Method

Our matching technique considers not only the total dissimilarity value, but also the number of hits and number of lines of the images. The motivation behind considering parameters other than the dissimilarity value is that the number of lines of the word images are different which is a situation that may alter the total dissimilarity value. Considering the other factors helps to obtain a better similarity criterion between the images. For example a precision score of 0.415 is obtained on GW10 test for $\tau = 2.5$ using only dissimilarity value, whereas considering other factors the precision score turns out to be 0.638.

Considering the lines of the images at different approximation accuracies also yields better results than considering the lines of the images at a single approximation accuracy. Our matching technique with normalized distance values allows us to add the results of the experiments to consider a single approximation accuracy. Correct matches at different tolerance factors provide almost similar results; however, false matches may not provide the same results. In this way some false matches are eliminated, which yields higher precision scores.

Our approach of line approximation runs in $m.nlog_2(n)$, where m is the number of contour segments having more points than zero and n is the number of points on that contour segment. Matching the two word images requires the time $O(kN_aN_b)$, where N_a and N_b are the number of line descriptors for the images and k is the number of τ results combined.

The proposed method does not handle rotation invariance; however, we empirically test that our method can handle the rotation invariance of [-19,24] degrees for GW sets, and [-14,18] degrees for OTM sets. In order to find these numbers, we manually rotate the words and compute the distance between the original image and the rotated images, and then we check the distance between the rotated images and first retrieved image (not rotated) for querying the original word. The limit degrees provided above are the average values of each rotation test.

Next, we discuss the results of our method as well as the comparison with other studies. Since we provide two types of test score, precision score and word error rate, we analyze each of them separately.

5.2.5 Comparisons with other studies for the task of retrieval

In Table 5.2 we provide our results as well as the results of the existing studies in terms of precision and recall scores. We carry out experiments using all words in the collections; therefore, we provide the precision scores in which the recall scores are 1.0. The studies providing a recall value less than 1.0, pay attention to

Table 5.2: Precision scores of our and the other approaches. OTM1+2: the combination of OTM1 and OTM2 data sets. [1]: Ataer and Duygulu [4] provide their own implementation of DTW for the OTM1 set.

Method	Data set	Precision	Recall
our approach	GW10	0.688	1.000
our approach	GW10	0.774	0.770
DTW (Rath and Manmatha [23])	GW10	0.653	0.711
DTW (Rath and Manmatha [22])	GW10	0.726	0.652
our approach	GW20	0.566	1.000
our approach	GW20	0.667	0.673
DTW (Rath and Manmatha [22])	GW20	0.518	0.550
our approach	OTM1	0.987	1.000
bag-of-words (Ataer and Duygulu [4])	OTM1	0.910	1.000
DTW (Ataer and Duygulu [4][1])	OTM1	0.940	1.000
our approach	OTM2	0.944	1.000
bag-of-words (Ataer and Duygulu [4])	OTM2	0.840	1.000
our approach	OTM1+2	0.957	1.000
bag-of-words (Ataer and Duygulu [4])	OTM1+2	0.810	1.000

pruning step which eliminates a set of likely wrong matches by analyzing different criteria such as aspect ratio - a process that requires extra effort and run tests on smaller sets; therefore, they obtain low recall values. Even though we do not pay attention to pruning step, we also provide precision scores at recall scores less than 1.0 to have a better comparison with such studies. For this purpose, we only take into account the first x percent of the retrievals. Precision and recall scores for different x values in the GW10 and GW20 collections are shown in Fig 5.8.

The precision score of our approach for the GW10 set is 0.688, with a recall score of 1.0. Rath and Manmatha [23] obtain 0.653 as the precision score. Our approach is better than their study in terms of precision score. However, the same authors obtain higher precision scores with lower recall scores in another study of theirs [22]. Considering the precision scores the study has better results than our method in which the recall score is 1.0; however, when we consider the precision score of our study, with a recall score of 0.770, it outperforms that study as well.

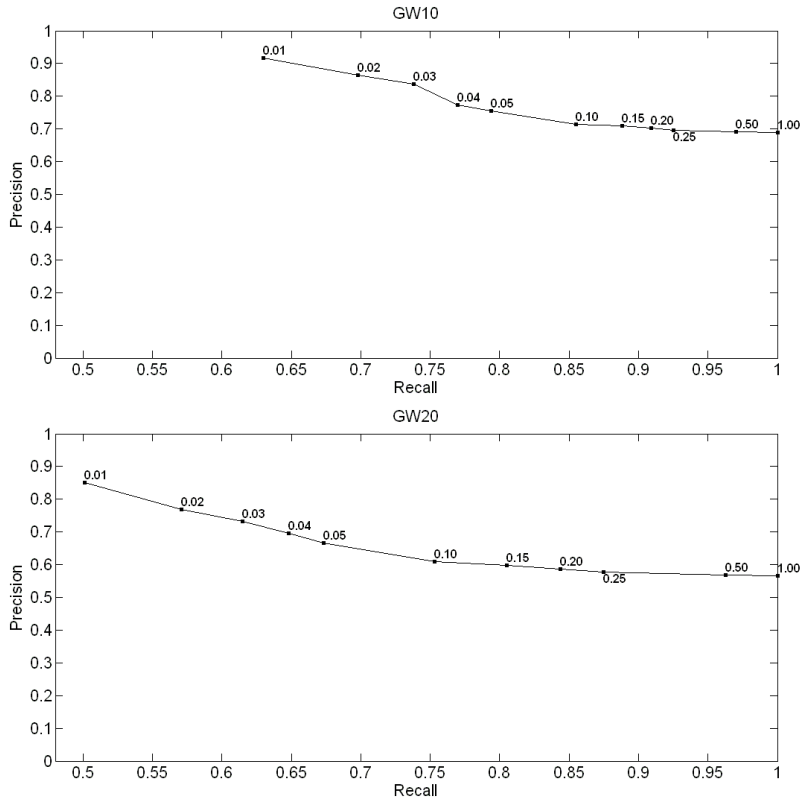

Figure 5.8: Precision and Recall scores for different x values in GW10 and GW20 sets.

On the GW20 set, we obtain a precision score of 0.566 when the recall score is 1.0, and 0.667 when the recall score is 0.673. In both cases, our results turn out to be higher than the results of the studies [23, 22].

Ataer and Duygulu [4] run their method on the OTM1 and OTM2 sets. They also compare their algorithm with the DTW method. Our promising method has a better performance than theirs as well as the DTW method as far as the OTM1 and OTM2 sets are concerned. They also test their method on a set which is a combination of the OTM1 and OTM2 sets (OTM1+2) in order to prove the script independence, whereby they also obtain high precision scores. Our approach, however, performs better on the OTM1+2 set than theirs as well as their implementation of the DTW method.

Table 5.3: Results of our and other methods in terms of WER for GW20 set.

Method	WER	WER w/o OOV words	Language model post-processing
our approach	0.303	0.189	-
Adamek et al. [1]	0.306	0.174	-
Lavrenko et al. [14]	0.449	0.349	+

5.2.6 Comparisons with other studies for the task of recognition

In Table 5.3, the WER with and without OOV words yielded by our method as well by the other studies are given for GW20 set. Our results outperform those of the study of Lavrenko et al. [14] in terms of WER with and without OOV. However, Lavrenko et al. [14] use a language model post-processing, and Adamek et al. [1] state that removing the language model post-processing causes a dramatic decrease in the recognition rate. Even though the results of Howe et al. [10] are better than our results, without language model post-processing our results turn out to be better than their scores.

Adamek et al. [1] test their method on the GW20 set and provide the results in the form of Word Error Rate (WER), as 0.306 and 0.174. Their score excluding OOV words is better than the score of our method, whereas our rate outperforms their score in the experiments including OOV words. However, the method of [1] does not work on scripts in which the diacritical marks are important, as in the case of Ottoman. Moreover, they make use of complex preprocessing steps before matching the word images. Our implementation of MCC-DCT algorithm without the preprocessing steps stated in the paper provides lower rates. The decrease on the rates shows that their approach has a high degree of dependence on the preprocessing steps. The method of Adamek et al. runs in $O(N^3)$ where N is the number of contour points used. They also provide modifications on the method to reduce the running time complexity.

Considering the time complexity of pre and post-processing steps, our method

is better than most of the existing studies. Since our method does not have steps such as linking disconnected contours, skew and slant correction, and removal of artifacts. In the task of word matching, the time complexity of our matching technique is also better than most of the DTW based and other techniques. Furthermore, our method is more efficient than most of the studies in the literature when we consider the final time complexity in which pre-processing steps, matching technique, and post-processing steps are taken into account.

Chapter 6

Redif Extraction Task

In Ottoman (Divan) poetry, most of the poems are based on a pair of lines, i.e., distich or couple. A distich contains two hemistichs (lines). In poems, hemistichs of the same distich completes each other. The rhyme and *redif* are used to provide the integrity of the distichs of a poem and provide a melody to its voice. The *redif* can be explained as the repeated patterns following the rhyme in a poem.

In Fig. 6.1 an original text in Ottoman language and its corresponding transcription are provided. The circled parts are the *redifs*, and the letters in gray are the rhymes in the original script. The boxed parts are the *redifs*, and the underlined letters are the rhymes in the transcription.

In this thesis, we also propose a method to automatically extract the *redifs* in handwritten Ottoman literary texts. We make use of the second matching criterion, matching using contour segments, for matching. In this task, unlike word spotting task, we use un-segmented images to extract the *redifs* which is a more challenging task compared with word spotting. In oreder to automatically extract the *redifs* in handwritten Ottoman literary text, we first find the ranking table for contour segment descriptors of the images by computing the distance between them, then we apply constraints to select the contour segments descriptors which are actually *redifs*. Before providing the details of the constraints and *redif* extraction, we first give details of the experimental environment, and

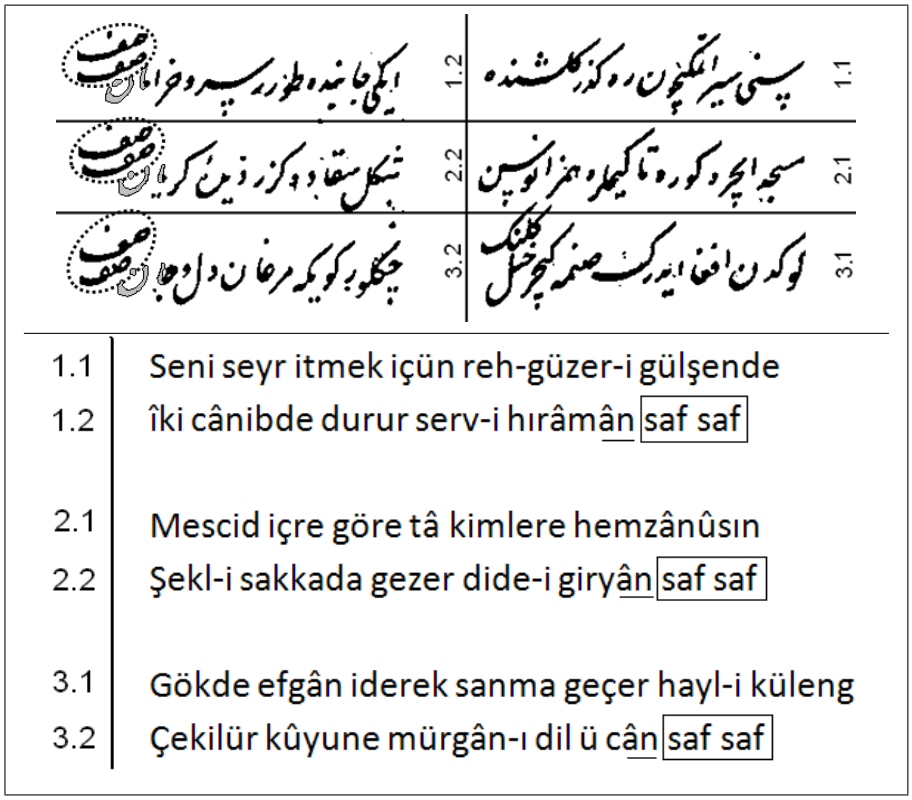

Figure 6.1: A part of a *gazel* by Bâkî (16^{th} century). The image on the top is the original text in Ottoman script, it is followed by its transcription. The circled parts of the original text are the *redifs*, and the letters in gray are the rhymes. The boxed parts and underlined letters of the transcribed version are, respectively, the corresponding *redifs* and rhymes.

provide results of experiments of the *redif* extraction task.

6.1 Experimental Environment

6.1.1 Data sets

Since there is not an available set consisting of Ottoman literary texts, we construct a collection of Ottoman literary texts specifically poems that consist of 100 poems or part of poems for testing purposes. The test collection contains works

of twenty different poets from the 15^{th} to 19^{th} centuries. The images of the poems are obtained from the "Turkey Manuscripts" web page of the Ministry of Culture and Tourism of Turkey[1] in which the digital copies of the handwritten Ottoman literary texts are publicly available, and Ottoman Text Archive Project (OTAP)[2] which collects the digital copies of the handwritten manuscripts and transcribes them (OTAP is an international project between University of Washington and Bilkent University.) For each of the poem in the collection, a ground truth table containing the number of contour segments of the *redifs* are recorded. The correction of the extracted contour segments as *redifs* is performed manually.

6.1.2 Evaluation Criteria

The correctness of the proposed method is computed by "extraction rate (ER)," and in the following way.

$R : extracted\ redifs$
$R_{gt} : ground\ truth\ for\ redifs$
$ER : extraction\ rate\ \in\ [0, 1]$
$ER = \frac{\#correct\ extractions\ in\ R}{max(sizeof(R_{gt}), sizeof(R))}$

Algorithm 2: Pseudo code of line approximation on contour segments.

Here, $sizeof(R)$ returns the number of contour segments in R, and similarly $sizeof(R_{gt})$ returns the number of contour segments in the ground truth table. We compute the ER score for each poem in the collection used in this study, and the final ER score is the average value of the ER scores of all poems.

[1] http://www.yazmalar.gov.tr
[2] http://courses.washington.edu/otap

6.2 *Redif* Extraction using Contour Segments (RECS)

In this task, we propose a method to automatically extract the *redifs* in handwritten Ottoman literary texts. In order to extract the *redifs*, we first describe the contour segments C, and then, normalize the line descriptors of the contour segments depending upon their reference line descriptor ℓ_r. Having normalized the line descriptors, we define the contour segment descriptors C' using the normalized line descriptors. The codebook B is generated over all line descriptors of all contour segments, and each contour segment descriptor then can be represented with a set of elements of the codebook as visual words. String matching algorithm is employed to find the distances the visual words which provides a ranking table for each contour segment descriptor. We employ two constraint calling the rules of the *redif* to extract the *redifs* among the contour segment descriptors by using the computed ranking table.

In order to extract the *redifs* we take into account the rules of the *redif*. Considering the rules of *redif* we add constraints to differentiate the *redifs* from other repeated patterns since all repeated sequences are not *redif*. A *redif* must appear:

- at the end of the second hemistich -line- of a distich -couple- (constraint 1)
- in every distich (constraint 2)

According to constraint number 1, the x positions of the *redifs* should roughly be the same and they should be close to the left border (end of the last hemistich) of the poem images. In order to satisfy this constraint, we first eliminate the contour segments that do not appear in the left (last) part of the distichs. A contour segment is in the last part of the distich if its x position is less than $\alpha_1 \times w$ where w is the width of image of the poem and $\alpha_1 \in [0, 1]$. Among the remaining ones, a contour segment and its matches are need to be vertically aligned to be counted as a *redif*. For a contour segment, we check each of its

matches whether they are vertically aligned. When the segment and a match are not vertically aligned, we ignore rest of the matches for the segment. Two contour segments are referred to as vertically aligned if the distance in x positions (i.e., the pre-condition is that they are in the left part of the image) are less than $\alpha_2 \times w$ where w is the width of image of the poem and $\alpha_2 \in [0, 1]$. We performed experiments with different values of α_1 and α_2, and empirically determined the values of these parameters as 0.25 and 0.15 respectively.

Among the remaining contour segments and their matches, we check the number of matches for each remaining contour segment to satisfy constraint two. The number of matches for a contour segment should be the same with the number of distichs in a literary text. However, determination of the number of distichs in an image of a poem is a challenging task and left as a future work. Instead we use five as the minimum number of matches should be extracted for a contour segment where five is the minimum number of distichs that a poem must have in Ottoman literature. Furthermore, in our collection, the poems have at least five distichs.

We check the remaining contour segments and their matches in the case of two contour segments having the same match. In other words, we search for the contour segments that have one or more common matches and take the union of the matches of those contour segments, and we perform this operation until any pair of contour segments has a common match. We take the union of the contour segments and matches in order not to extract the same contour segment as *redif* more than once. Finally, we check the remaining contour segments whether they have a minimum of five matches or not. In the case a contour segment does not have more than four matches, we eliminate the contour segment. The remaining contour segments are extracted as *redifs*. If any contour segment is not extracted as *redif*, then the text does not have *redif* at all.

In order to understand the proposed method let's consider a poem with ten contour segments $\{C_1, C_2, ..., C_{10}\}$. Assume that after eliminating the contour segments not satisfying the constraint 1 only four contour segments are left and they are $\{C_1, C_6, C_7, C_9\}$ and their matches are as follows: for C_1: (C_1, C_3, C_5, C_7),

for C_6: (C_6, C_4), for C_7: (C_7, C_2, C_3, C_5), and for C_9: (C_9, C_{10}). The contour segments having one or more common matches are combined by taking the union of the matches. The resulting matches turn out to $(C_1, C_2, C_3, C_5, C_7)$, (C_6, C_4), and (C_9, C_{10}). C_1 has five matches, and C_6 and C_9 has two. Depending upon the constraint 2, we eliminate the contour segment C_9 and C_6 since the numbers of matches are less than five. Finally, $(C_1, C_2, C_3, C_5, C_7)$ are extracted as the *redifs* in distichs. We select the one among the matches having minimum distance to the left border of the poem image, and return it as the *redif* of the poem.

Note that most of the Ottoman literary texts contain *redifs* following rhymes. The differentiation of *redifs* and rhymes requires lexical information. In order to differentiate the *redifs* and rhymes most of the words of the line should be recognized which is another challenging task. Therefore, we consider rhymes and *redifs* together in a poem. In most of the cases, a rhyme consists of two characters corresponding to one isolated contour segment and part of another contour segment. The character corresponding to part of a contour segment has different shapes depending upon the previous character; hence, the extraction and recognition of that character is almost impossible, and such characters are disregarded.

6.3 Experimental Results

Depending upon the successful results in word spotting task, we set the parameter τ is set to 2.5 in the experiments of RECS. In RECS we focus on the parameter k as stated before. Moreover, the extensive analysis of different τ values for different values of k is left as future work.

Figure 6.2: Some extraction results: the *redifs* are circled, and extracted *redifs* are in white boxes. Poet (century) information for the images: (a) Hamza (18-19), (b) Hayâlî (16), (c) Nihânî (16), (d) Mihrî (16), (e) Nesîmî (15), (f) Ümîdî Ahmed (16), (g) Bâkî (16), (h) Ümidî (17), (i) Baki Mahmud (17).

In Fig. 6.2 we provide nine sample extraction results. In Fig. 6.2(a), 6.2(b), and 6.2(c) the *redifs* are extracted correctly; therefore, the extraction rates are 1.0. In Fig. 6.2(d) three segments of the *redif* extracted correctly while four should be extracted and the ER value is 0.75. In Fig. 6.2(e) and 6.2(f) *redifs* as well as one extra contour segment are extracted in which the ER values turn out to be 0.75. Two out of three are extracted correctly in Fig. 6.2(g) and in Fig. 6.2(h); therefore, the ER value is 0.67. According to the ground truth table for 6.2(h) its *redif* consists of seven contour segments whereas our method extracts three correct segments, and the extraction rate is 0.43. For the images of Fig. 6.2 the overall (average) ER value is 0.78 which is the average of the above ER values. Note that the number of contour segments for the *redifs* in the collection is roughly the same. For this reason, we do not prefer a weighted average computation.

For the whole test collection our method extracts the *redifs* correctly with an ER score of 0.682. The score of 0.682 is obtained when k is set to 45. However, as stated before, we do experiments on different values of k which affects the extraction rate. Fig. 6.3 gives the extraction rates for different k values.

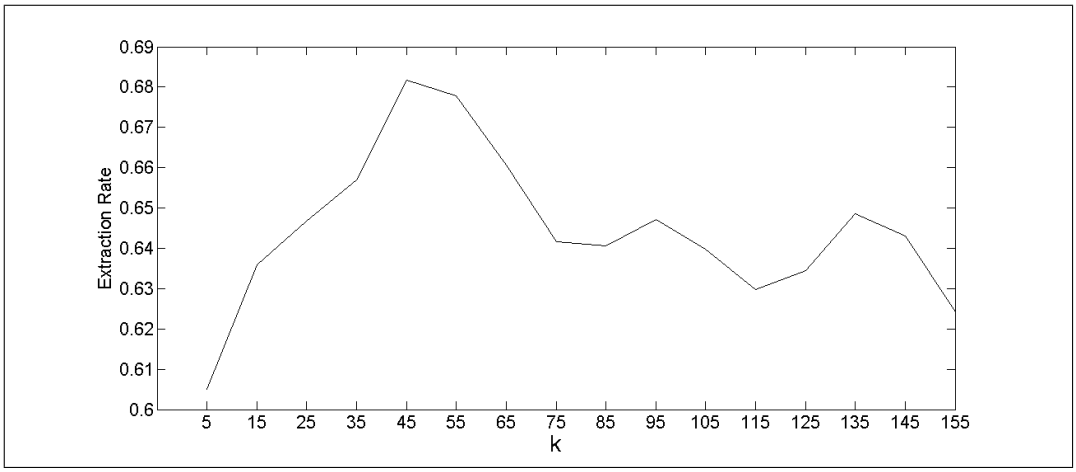

Figure 6.3: Extraction rates for different values of k.

We observe that as the value of k increases, sensitivity of the method increases. In other words, for large values of k, our method is able to extract more complicated *redifs*; however, at the same time it misses more number of *redifs*.

The experimental results imply that we have room for improvement.

As stated before, in constraint 2, the minimum number of matches of a contour segment that should be counted as *redif* is set to five. If it is decreased, the method extracts more number of correct contour segments as *redifs*; however, the number of false matches also increases. For higher values of the same parameter, the number of false matches decreases; however, in this case the number of misses increases. It is experimentally determined that the best value for this parameter is five.

Since there is no previous study like ours in the literature, we are not able to compare our results with the others. However, in order to increase the success of the method we plan to do an extensive analysis of τ values while the value of k is changing. The other future pointer for RECS method is to consider the results of different k values as we did for the parameter τ in criterion, matching using line descriptors.

Chapter 7

Conclusion

In this work, we propose a new representation to match the words in historical documents. We represent the words using the approximated lines on the points of the extracted contours. We make use of the representation in two different matching criteria, matching word images using corresponding line descriptors, and matching contour segments using sequence of code words.

The criterion, matching using line descriptors, is employed in word spotting task in terms of word retrieval and recognition. Going beyond the George Washington data set, which has become a benchmark in the word spotting literature, by applying our method on Ottoman documents provided in [3], we also address the challenge of working on different alphabets and different writing styles. In word spotting task, we also consider the results of different τ values and obtain a multi-scale analysis. According to the experimental results we obtain precision scores of 0.774, 0.667, 0.987, 0.944, and 0.957 for the sets GW10, GW20, OTM1, OTM2, and OTM1+2 respectively in the task of word retrieval. In the case of word recognition we obtain a WER (Word Error Rate) of 0.303 excluding the OOV (out of vocabulary) words in GW20 set. The results are mostly higher than the existing studies in the literature.

The criterion, matching using contour segments is employed in *redif* extraction of handwritten Ottoman literary texts. We correctly extract the *redifs* with

a score of 0.682 when the parameter k is set to 45. The results of the *redif* extraction provide motivation for further and advanced studies based on the matching criterion. Our work can be used in different ways and has several implications within the context of Ottoman literary studies.

Bibliography

[1] T. Adamek, N. E. O'Connor, and A. F. Smeaton. Word matching using single closed contours for indexing handwritten historical documents. *International Journal on Document Analysis and Recognition*, 9:153–165, 2007.

[2] A. Amin. Off line Arabic character recognition - a survey. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 596–599, 1997.

[3] E. Ataer and P. Duygulu. Retrieval of Ottoman documents. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 155–162, 2006.

[4] E. Ataer and P. Duygulu. Matching Ottoman words: An image retrieval approach to historical document indexing. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 341–347, 2007.

[5] A. Balasubramanian, M. Meshesha, and C. V. Jawahar. Retrieval from document image collections. In *Proceedings of the Conference on Document Analysis Systems*, pages 1–12, 2006.

[6] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.

[7] D. Douglas and T. Peucker. Algorithms for reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[8] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):36–51, 2008.

[9] J. Hershberger and J. Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. Technical report, University of British Columbia, Vancouver, BC, Canada, Canada, 1992.

[10] N. R. Howe, S. Feng, and R. Manmatha. Finding words in alphabet soup: Inference on free form character recognition for historical scripts. *Pattern Recognition*, 42(12):3338–3347, 2009.

[11] S. Impedovo, L. Ottaviano, and S. Occhinegro. Optical character recognition - a survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 5:1–24, 1991.

[12] T. Konidaris, B. Gatos, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal of Document Analysis and Recognition*, 9(2):167–177, 2007.

[13] A. Kumar, C. V. Jawahar, and R. Manmatha. Efficient search in document image collections. In *Proceedings of the 8th Asian Conference on Computer Vision*, pages 586–595, 2007.

[14] V. Lavrenko, T. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings Document Image Analysis for Libraries*, pages 278–287, 2004.

[15] I. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[16] Y. Leydier, F. Lebourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recognition.*, 40:3552–3567, 2007.

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[18] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2):149–164, 2001.

[19] R. Manmatha, C. Han, and E. M. Riseman. Word spotting: A new approach to indexing handwriting. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 631–637, 1996.

[20] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE*, 80(7):1029–1058, 1992.

[21] N. Otsu. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1), 1979.

[22] T. M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pages 218–223, 2003.

[23] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2:521, 2003.

[24] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9:139–152, 2007.

[25] J. Rodriguez-Serrano and F. Perronnin. Handwritten word-spotting using hidden Markov models and universal vocabularies. *Pattern Recognition*, 42(9):2106–2116, 2009.

[26] J. L. Rothfeder, S. Feng, and T. M. Rath. Using corner feature correspondences to rank word images by similarity. *Computer Vision and Pattern Recognition Workshop*, 3:30, 2003.

[27] C. Y. Suen, M. Berthod, and S. Mori. Automatic recognition of handprinted charactersthe state of the art. *Proceedings of IEEE*, 68(4):469–487, 1980.

[28] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[29] O. D. Trier, A. K. Jain, and T. Taxt. Feature extraction methods for character recognition-a survey. *Pattern Recognition*, 29(4):641–662, 1996.