SEGMENTATION BASED OTTOMAN TEXT AND MATCHING BASED KUFIC IMAGE ANALYSIS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hande Adıgüzel

July, 2013

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.

Asst. 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.

Vis. 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.

Asst. Prof. Dr. Sinan Kalkan

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

SEGMENTATION BASED OTTOMAN TEXT AND MATCHING BASED KUFIC IMAGE ANALYSIS

Hande Adıgüzel
M.S. in Computer Engineering
Supervisor: Asst. Prof. Dr. Pınar Duygulu Şahin
July, 2013

Large archives of historical documents attract many researchers from all around the world. The increasing demand to access those archives makes automatic retrieval and recognition of historical documents crucial. Ottoman archives are one of the largest collections of historical documents. Although Ottoman is not a currently spoken language, many researchers from all around the world are interested in accessing the archived material. This thesis proposes two Ottoman document analysis studies; first one is a crucial pre-processing task for retrieval and recognition which is segmentation of documents. Second one is a more specific retrieval and recognition problem which aims matching Islamic patterns is Kufic images. For the first segmentation task, layout, line and word segmentation is studied. Layout segmentation is obtained via Log-Gabor filtering. Four different algorithms are proposed for line segmentation and finally a simple morphological method is preferred for word segmentation. Datasets are constructed with documents from both Ottoman and other languages (English, Greek and Bangla) to test the script-independency of the methods. Experiments show that our segmentation steps give satisfactory results. The second task aims to detect Islamic patterns in Kufic images. The sub-patterns are considered as basic units and matching is used for the analysis. Graphs are preferred to represent sub-patterns where graph and sub-graph isomorphism are used for matching them. Kufic images are analyzed in three different ways. Given a query pattern, all the instances of the query can be found through retrieval. Going further, through known patterns images can be automatically labeled in the entire dataset. Finally, patterns that repeat inside an image can be automatically discovered. As there is no existing Kufic dataset, a new one is constructed by collecting images from the Internet and promising results are obtained on this dataset.

Keywords: Historical Manuscripts, Ottoman Documents, Layout Segmentation, Line Segmentation, Word Segmentation, Islamic Pattern Matching.

ÖZET

BÖLÜTLEME TABANLI OSMANLICA METİN VE EŞLEŞTİRME TABANLI KUFİ RESİM ANALİZİ

Hande Adıgüzel
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Asst. Prof. Dr. Pınar Duygulu
Temmuz, 2012

Tarihsel arşivler dünyanın pek çok yerinden akademisyenlerin ve konuyla ilgilenen araştırmacıların ilgisini çekmektedir. Bu belgelere erişim isteğinin artması otomatik erişim ve tanıma sistemlerini zorunlu kılmaktadır. Osmanlıca belgeler tarihsel belgeler arasında önemli ve büyük bir yer kaplamaktadır. Osmanlıca günümüzde halen konuşulan bir dil olmamasına rağmen bir çok tarihçinin ilgisini çekmektedir. Bu tezde de iki adet Osmanlıca belge analizi çalışması sunulmaktadır. İlki Osmanlıca belgelerin bölütlenmesi olup; bölge, satır ve kelime bölütleme çalışılmıştır. Bölgelere ayırma Log-Gabor filtreleme yöntemi ile sağlanmıştır. Satırlara bölütleme içinse 4 farklı yöntem sunulmaktadır. Son olarak ise belgeler morfolojik yöntemler ile kelimelere ayrılmıştır. Veri kümelerine Osmanlıcanın yanında farklı dillerden oluşan belgeler (İngilizce, Yunanca ve Bangla) da eklenmiştir. Deneylerden elde edilen sonuçlar bölütleme algoritmalarının iyi çalıştığını göstermektedir. Tezin ikinci kısmında ise Kufi resimlerinde İslami motiflerin tespiti amaçlanmıştır. Motiflerin temsili için grafikler kullanılmıştır. Eşyapılı grafikler ve altçizgeler incelenerek motifler eşleştirilmeye çalışılmıştır. Kufi imgeleri farklı deneyler ile incelenmiştir. İlki verilen bir sorgu motifinin veri kümesinden geri getirilmesidir. İkinci deney Kufi resimlerinin otomatik etiketlenmesidir. Son olarak, her resimdeki tekrarlanan motifler incelenmiştir. İnternet üzerinden toplanan resimlerle bir veri kümesi oluşturulmuştur. Önerilen yöntem bu veri kümesi ile test edilmiş ve umut verici sonuçlar elde edilmiştir.

Anahtar sözcükler: Tarihi Metinler, Osmanlıca Belgeler, Bölge Bölütleme, Satır Bölütleme, Kelime Bölütleme, İslami Motif Eşleştirme.

Acknowledgement

First of all, I would like to express my gratitude to my supervisor Dr. Pınar Duygulu from whom I have learned a lot due to her supervision, patient guidance, and support during this research. Without her invaluable assistance and encouragement, this thesis would not be possible.

I am indebted to the members of my thesis committee Prof. Dr. Fazlı Can and Asst. Prof. Dr. Sinan Kalkan for accepting to review my thesis and their valuable comments.

I would like to express my special thanks to my friends Burcu and Zeren for always being so supportive and cheerful towards me.

I am thankful to all my friends from the RETINA group especially Fadime, Nermin, Gokhan, Sermetcan and Caner. Conference days or Quick China meetings would not be so memorable and enjoying without them.

The biggest of my love goes to my beloved family, for their endless support and love. None of this would be possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Historical documents constitute a large heritage that needs to be preserved. Many researchers from all around the world are interested in accessing, analyzing and studying them. Ottoman archives are one of the largest collections of historical documents; they include more than 150 million documents ranging from military reports to economic and political correspondences belonging to the Ottoman era [10]. Ottoman empire, which had lasted for more than 6 centuries and spread over 3 continents, shaped the history of the old world for several countries [9]. Although Ottoman is not a currently spoken language, many researchers from all around the world are interested in accessing the archived material.

Until recently, access to historical documents were provided only by manual indexing which can be considered costly because excessive amount of human effort is required. Lately, digital environments became available for keeping historical documents in image format. After this technological progress, demand to access digital historical archives has been increased. To make the historical contents available, automatic indexing and retrieval systems are required. Since, most of the historical documents are kept in image format, analyzing their visual content is suggested to build automatic systems.

Optical character recognition (OCR) can be used to provide automatic document indexing and retrieval [11, 12, 13, 14]. However, applying OCR techniques on old historical documents is nearly impossible because of the poor quality of the documents, the variety of scripts and the high-level noise factors like faded ink and stained paper caused by deterioration. Moreover, existing OCR systems

are language dependent and not available for every language. Thus, retrieval and indexing of historical documents problems are usually solved by word spotting approaches [15, 16, 17].

As a pre-processing step, segmentation provides several benefits for retrieval and indexing tasks by supporting fast and easy navigation. In the first part of this thesis, segmentation of Ottoman documents is studied. First layout segmentation which aims to detect regions consisting of text lines written with different orientations is explored. A Log-Gabor filtering based approach is used to segment documents with multi-oriented lines. Experiments are performed on an Ottoman dataset constructed with documents with multi-oriented lines.

The second segmentation task is line segmentation where four different approaches are proposed. The first one is a traditional line segmentation method, called projection profile, used for printed documents with straight lines and it is used as a baseline system. The second method is a hybrid line segmentation, combining projection profile and connected component based methods. This method is proposed for segmenting lines from Ottoman documents. The last 2 methods are based on Log-Gabor filtering like the layout segmentation and they support script independency. The first one uses region intersection while line fitting is preferred for the second one.

The hybrid line segmentation method is tested on both printed and handwritten Ottoman documents to compare the algorithm's performance for different type of texts. Also, another handwritten Ottoman dataset is constructed with different books and authors to evaluate the performance under different writing styles and writers. The four line segmentation methods are tested and compared on a mixed dataset including Ottoman, English, Greek and Bangla documents.

The last segmentation task is word segmentation where a simple morphological method is applied. The tests are performed on English, Greek and Bangla documents. Ottoman word segmentation is not in our study's scope because it can be only performed by language based rules which requires Ottoman language experts.

The second part of this thesis concentrates on Islamic pattern matching in Kufic images. The approach involves four main steps: (i) foreground extraction, (ii) sub-pattern extraction, (iii) representation and matching, (iv) analysis. The

sub-patterns are considered as basic units and extracted connected components are used as sub-patterns. The relationships between straight lines in specific orientations are observed and thought to be important for the identification of Kufic patterns. Based on this observation, a line-based method for representing and matching sub-patterns is proposed. Each sub-pattern is represented with a graph and then graph and sub-graph isomorphism are applied to match the patterns.

Sub-pattern matching is used for the analysis of Kufic images in three different ways. Given a query pattern, all the instances can be found through retrieval. Going further, through known patterns images can be automatically labeled in the entire dataset. Finally, patterns that repeat inside an image can be automatically discovered.

The rest of the thesis is arranged as follows. First segmentation of documents is explained in the order: layout, line and word segmentation. Line segmentation section is divided into four, where each section explains a different algorithm. Afterwards, the segmentation experiments are given; first one is Ottoman document segmentation where layout and line segmentation results are discussed for Ottoman documents. Second experiment section is script-independent segmentation where line and word segmentation results for different languages are given.

Second part of the thesis focuses on matching Islamic patterns in Kufic images. First, extraction of foreground pixels and labeling of sub-patterns is explained. Then in the next section, graph isomorphism is proposed for sub-pattern matching. Finally, analysis is done with three different experiments; query retrieval, image indexing and repeating pattern detection.

Chapter 2

Segmentation of Ottoman Documents

2.1 Motivation

Large archives of historical documents attract many researchers from all around the world. The increasing demand to access those archives makes automatic retrieval and recognition of these documents crucial. Ottoman archives are one of the largest collections of historical documents; they include more than 150 million documents ranging from military reports to economic and political correspondences belonging to the Ottoman era [10]. Researchers from all around the world are interested in accessing the archived material [18]. Unfortunately, many documents are in poor condition due to age or recorded in manuscript format.

Line segmentation is usually a crucial pre-processing step in most of the document analysis systems. Although text line segmentation is a long standing problem, it is still challenging for hand-written degraded documents. The problems of handwritten texts can be categorized into 2 parts: (i) line-based problems such as, variance of interline distances, inconsistent baseline skews, multi-oriented text lines and high degrees of curl; and (ii) character-based problems such as, broken characters due to degradation, touching and overlapping text lines, small-sized diacritical components, noisy components like ornamentation and variance of character size.

Even though there are many advanced methods [19, 20, 21, 22, 23] designed

for complex datasets, the studies in text line segmentation are dominated by projection profile and connected component-based approaches. Projection profile based methods are usually successful on machine printed documents [24], nevertheless, they can be extended to deal with slightly curved text lines [23]. Besides, projection profile based methods are easy to implement and fast thru the basic intuition of straightness of text lines.

Connected component based methods are appropriate for more complex documents where interline distances and baseline skews change. However, most of the connected component based methods work directly on the input image where each pixel is treated equally and a change of one pixel may result in a different result [23].

In this study, we first propose a layout segmentation algorithm based on Log-Gabor filtering to obtain line segmentation for documents with multi-oriented lines. First, the document image is convolved with a Log-Gabor filter bank with different scales and orientations and then the convolution results are divided into cells and maximum Gabor response per cell is computed.

Secondly, four different line segmentation algorithms are explained. First one is based on vertical projection profile which is used as a baseline system. Second one is a new segmentation algorithm where we use a hybrid approach which combines both connected components and projection information. Rather than obtaining the projection profile directly from the input image or straightly using connected components for line detection, baselines of connected components are extracted and passed to second phase where projection profiles are used. This process also allows some skew tolerance.

The last two line segmentation algorithms are based on Log-Gabor filtering like the layout segmentation algorithm. Firstly, document image is pre-processed to suppress language based characteristics and emphasize the line structure. Then pre-processed image is convolved with different Log-Gabor filters with different scales to detect the correct character scale. Afterwards, convolution result is binarized to obtain the line regions. The first algorithm intersects connected components with detected regions while the other one fits lines to boundaries and computes the closest baseline for each component.

Finally, a simple morphology based word segmentation algorithm is proposed.

Figure 2.1 shows the overall system design.

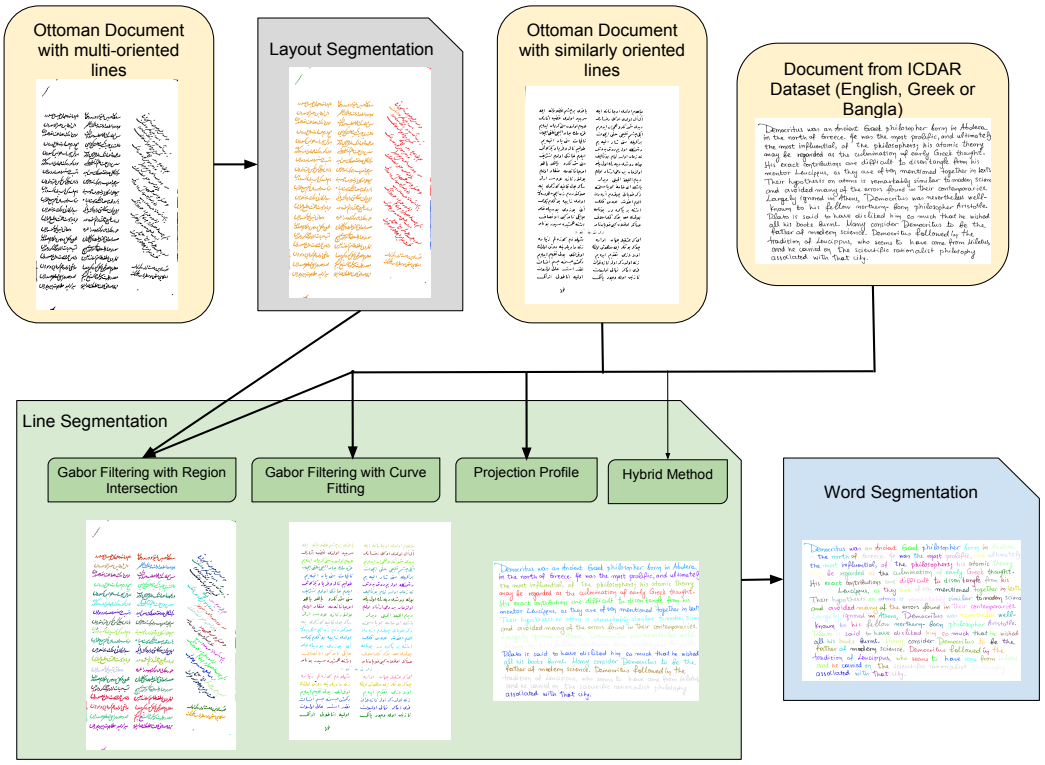


Figure 2.1: The overall organization of our segmentation system.

The contributions of this study are threefold. First we address the layout and line segmentation problem for historical Ottoman documents which are rarely studied. We intend to apply our method to considerably large and complex historical datasets with multiple authors from various time periods including documents with multi-oriented lines. To achieve a simple but effective line segmentation method, bottlenecks of projection profile based and connected component based methods are prevented through a hybrid approach. Second, Fourier curve fitting is suggested for determining the peaks and valleys in projection profile analysis which is still considered as a problematic issue [23]. Additionally, script independent line segmentation and words segmentation methods are proposed. As far as we know, our study is the first one to use Gabor filtering for the text line segmentation problem.

2.2 Related Work

Text line segmentation algorithms can be mainly categorized as projection profile based [24, 25] and connected component based [26, 27]. Projection based methods makes the assumption of text lines being parallel and straight thus, they are effective for machine printed documents.

For handwritten documents where interline gaps are small or lines have considerably high skew, piece-wise projection approaches are used [25]. In these approaches documents are divided into vertical strips and vertical projection profiles of strips are combined to obtain the results.

Connected component based methods [26, 27] extracts geometrical information such as shape, orientation, position and size from connected components to group or merge them into lines. They are more appropriate for complex documents than the projection profile based methods. However, they are sensitive to small changes in connected component structures. Another disadvantage is that they may be script dependent. For example, there is a space between neighboring words in English, but a Chinese text line is composed of a string of characters without word spaces [28].

There are also studies which propose script-independent line segmentation methods with deformable models [20, 22, 28]. The paper [28] first enhances text line structure using Gaussian window and then uses level set method to evolve text line boundaries. With the prior knowledge that a text line is a horizontally elongated shape, the text line boundaries are forced to grow faster in the horizontal direction. However, the approach is sensitive to the number of level set evolution iterations.

Another study that segments lines with deformable models is [20], which uses the Mumford-Shah Model. The approaches presented in [20, 28], depend on skew-correction and zone segmentation before text line segmentation. Also, they are sensitive to large number of touching and overlapping components and they use heuristic post-processing rules for splitting and joining segmented text lines to handle these cases [22].

The authors of the paper [22] solve these problems by using active contours to detect curved lines. First, snakes are deformed in horizontal direction and then

neighboring baby snakes are joined together. Since, they preferred to use image smoothing using multi-oriented Gaussian filters that enhance the line structure even on high curvature; the algorithm does not depend on skew correction or zone segmentation.

Another script-independent line segmentation study is [29]. In this study, text line segmentation is achieved by applying Hough transform on a subset of the document image connected components. A post-processing step includes the correction of possible false alarms, the detection of text lines that Hough transform failed to create and finally the efficient separation of vertically connected characters using a novel method based on skeletonization. Although, Hough-based methods can handle documents with variation in the skew angle between text lines, they are not very effective when the skew of a text line varies along its width [30].

There are few studies [10, 31] that apply line segmentation on Ottoman datasets. In [10], it is assumed that baselines will have more number of black pixels than the other rows. With this intuition projection profile of the documents are analyzed and peaks of the profile are detected according to some predefined threshold. However, due to inconsistent baseline skews, multi-oriented text lines and small interline gaps observed in Ottoman documents; directly applying projection profile method is likely to fail. Further, different threshold values need to be set for different types of writing styles or writers.

Another study that demonstrate their results on Ottoman documents, constructs a Repulsive-Attractive Network for line segmentation [31]. In this network, attractive and repulsive forces are defined and baseline units' y-coordinates are iteratively changed according to these forces until local convergence is obtained. Nevertheless, the lines must have similar lengths and each baseline is detected according to previously examined one where a detection error can trigger other ones.

There are also papers which work on line segmentation of documents with multi-oriented lines [22, 32, 33, 34]. In paper [34], multi-oriented text line extraction from handwritten Arabic documents is studied. The local orientations are determined using small windows obtained by image paving. The orientation of the text within each window is estimated using the projection profile technique

considering several projection angles.

Another study [32] focuses on multi-oriented line segmentation on English documents. Their method is based on foreground and background information of the characters of the text. In the proposed scheme at first, individual components are detected and grouped into 3-character clusters using their inter-component distance, size and positional information. Then clusters are merged to obtain individual lines. Another study that segments multi-oriented text lines [33], uses a similar strategy of clustering connected components. First they obtain word groups from the clusters and then, text lines of arbitrary orientation are segmented from the estimation of these word groups.

Word segmentation is applied on Ottoman documents in a few studies [9, 10]. In [9] a version of a document, in which word segmentation is easy, is used as a source data set and another version in a different writing style, which is more difficult to segment into words, is used as a target data set. The source data set is segmented into words by a simple method and extracted words are used as queries to be spotted in the target data set. In [10] a more simpler method is preferred. To find the boundaries between the words, they apply a threshold value on the length of the space in between the words.

There are also statistical methods for word segmentation [35, 36, 37, 38]. However, they are usually preferred for Chinese documents where sentences are written as characters strings with no spaces between 2-character words. Thus, statistical features that capture the dependency among connected components of a word such as mutual information and context dependency are used to extract words.

2.3 Methodology

Our segmentation process consists of four main tasks whose details are given in the following sections. First, pre-processing of documents is explained which consists of binarization, simple repairment and page segmentation steps. These steps are only applied to Ottoman documents. ICDAR dataset which consists of English, Greek and Bangla documents were already binarized and page segmented when it was obtained.

The second task is layout segmentation, which is applied to Ottoman documents that include multi-oriented text lines in a single page. Within the layout segmentation, how to segment lines from documents with multi-oriented lines is also explained.

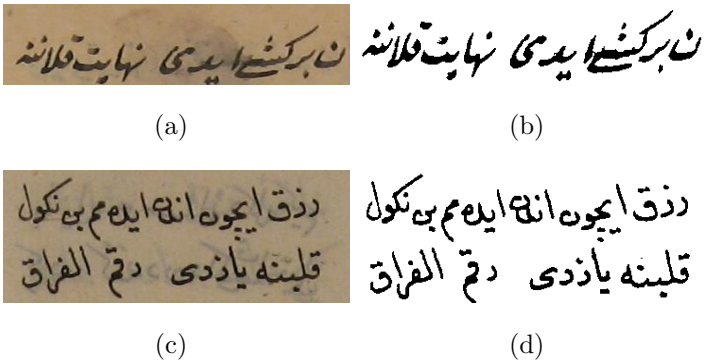
The third task is line segmentation, where we present 4 different approaches. First one being projection profile is a traditional method which we use as a baseline system. Second method is a hybrid line segmentation, combining projection profile and connected component based methods. This method is proposed for segmenting lines from Ottoman documents. Third and fourth line segmentation methods are designed for script-independent documents and both of them are based on Gabor filtering, first one uses region intersection while line fitting is preferred for the second one.

The last segmentation task is word segmentation where we apply a simple morphological method whose parameters are optimized based on simple characteristics of different languages. Ottoman word segmentation is not in our study's scope because of the fact that it can be only done by language based rules which requires Ottoman language experts.

2.3.1 Pre-processing

Binarization is one of the important pre-processing steps of segmentation. Global binarization methods use a single threshold value to classify pixels into foreground or background classes. However, they do not always yield satisfactory results especially on historical documents that are degraded, deformed and not in good quality due to faded ink and stained paper and may be noisy because of deterioration (see Figure 2.2 (a) and (c)).

After the original documents are converted into gray scale, adaptive binarization method [39], which calculates multiple threshold values according to the local areas, is used for binarization (see Figure 2.2 (b) and (d)). Then, small noise such as dots and other blobs are cleaned by removing connected components which are smaller than a predefined threshold.



(a) (b)

(c) (d)

Figure 2.2: (a) and (c): Some examples from the historical degraded documents, the ink is faded, paper is stained and the images are noisy; (b) and (d): Adaptive binarization manages to overcome those difficulties.

After binarization, to connect broken characters; first Manhattan distance between adjacent foreground pixels are calculated then pixels are connected if the measured distance is smaller than a predefined threshold.

The documents in our datasets are scanned in 2-page format. Therefore, before segmentation documents must be segmented into pages. The horizontal projection profile of each document is calculated and then the two largest peaks of the profile are observed for segmenting the two pages. To detect the widest peaks, a Fourier curve [40] is fitted to the horizontal projection profile and then the image is cropped according to the smallest value of the profile that lie between the two peaks.

2.3.2 Layout Segmentation

For layout segmentation task we preferred to use Gabor filters to detect regions which include similarly oriented lines. Gabor filtering, which is named after Denis Gabor is basically a linear filter used to detect edges [41]. Besides, the theory proposed in [42] explains that the retinal image is decomposed to a number of filtered images with different sizes and orientations in the human visual system. This theory indicates that Gabor functions are similar to perception in the human visual system and thus, Gabor filters are found to be appropriate for texture representation, optical character recognition, iris recognition and fingerprint recognition

[41].

A Gabor filter is a Gaussian Kernel function multiplied by a complex sinusoid which are known as the envelope and the carrier respectively. The formula of a complex Gabor function in space domain is [43]:

$$g(x, y) = s(x, y)w_r(x, y); \tag{2.1}$$

where $s(x, y)$ is a complex sinusoid, and $w_r(x, y)$ is a 2-D Gaussian-shaped function. The complex sinusoid is defined as follows:

$$s(x, y) = exp(j(2\pi(u_0x + v_0y) + P)); \tag{2.2}$$

where (u_0, v_0) and P define the spatial frequency and the phase of the sinusoid respectively. This sinusoid can be thought as two separate components, conveniently allocated in the real and imaginary part of a complex function. Besides, the two components may be formed into a complex number or used individually [41].

The Gaussian envelope is as follows:

$$w_r(x, y) = Kexp(-\pi(a^2(x - x_0)_r^2 + b^2(y - y_0)_r^2)); \tag{2.3}$$

where K scales the magnitude of the Gaussian envelope, (x_0, y_0) is the peak of the function, a and b are scaling parameters of the Gaussian, and the r subscript stands for a rotation operation.

For segmentation tasks usually a Gabor filter bank is constructed with filters of different scales and orientations. Then the filters are convolved with the image and the response in Gabor space is analyzed. This process is very similar to the process in the human primary visual cortex [44]. Another study [45] proposes that the real parts of the Gabor function is a good approximation of a receptive function belonging to cats' striate cortex. For our study, we adapt a similar process of constructing a Gabor filter bank and analyzing the real components of different responses generated by Gabor filters with different orientations and scales.

We preferred to use Log-Gabor filters [46] instead of Gabor filters, which

eliminates some traditional disadvantages such as DC-bias. They are basically constructed with the logarithmic transformation of the Gabor domain.

Firstly for a binarized document (see Figure 2.3), connected components are extracted and average connected component height ($avgH$) is found. This metric is used as a parameter for the wavelet scale of the Gabor filter. Then, the binary image is convolved with a bank of Log-Gabor filters with 4 orientations and 4 scales, resulting in 16 different filters (see Figure 2.4). Peter Kovesi's Gabor Convolve software is used for the convolution task [1]. The filters with different orientations are used to detect the text line regions with different orientations and different scales are used to obtain accurate results for the documents which has characters with varying sizes.

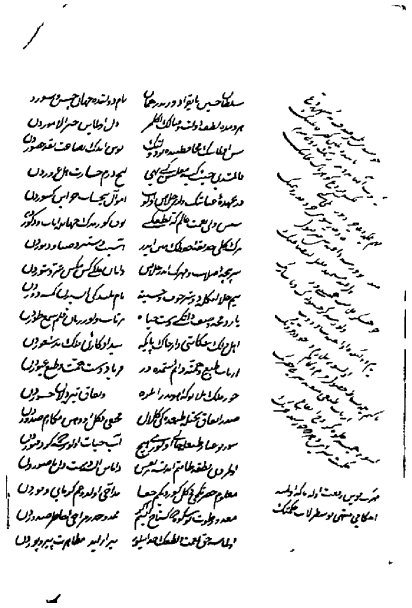


Figure 2.3: A binarized document with multi-oriented lines.

[1] http://www.csse.uwa.edu.au/~pk/research/matlabfns/

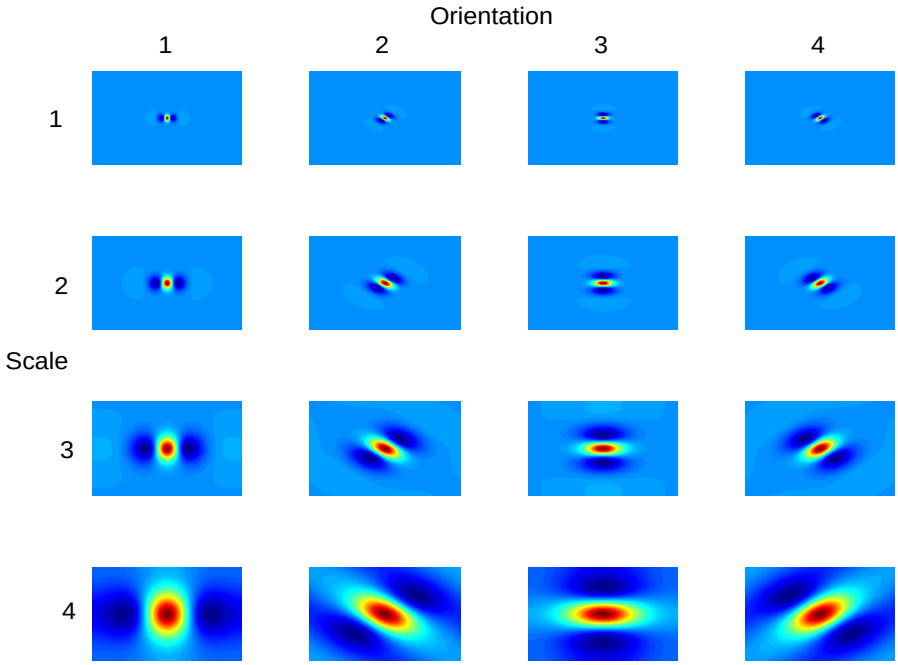


Figure 2.4: A bank of Log-Gabor filters with 4 different orientation and scales for a minimum wavelength of 40 and scaling factor of 2.

The wavelength of the smallest scale filter is set to half of the average connected component height ($avgH/2$) and the scaling factor between successive filters is set to 2, which results in filters with scales $avgH/2$, $avgH$, $2avgH$, $4avgH$. The ratio of the standard deviation of the Gaussian describing the log-Gabor filter's transfer function in the frequency domain to the filter center frequency is set to 0.65 and the ratio of angular interval between filter orientations and the standard deviation of the angular Gaussian function used to construct filters in the frequency plane is set to $1, 3$. Figure 2.5 shows the real part of the convolution responses to Log-Gabor filters of 4 orientations and a single scale.

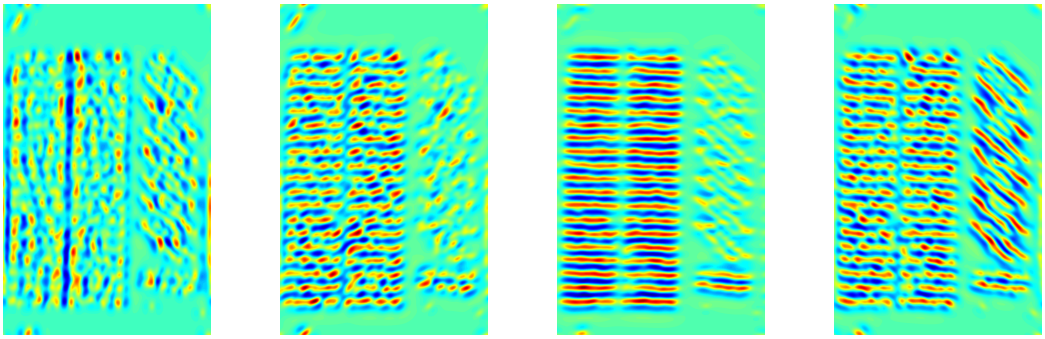
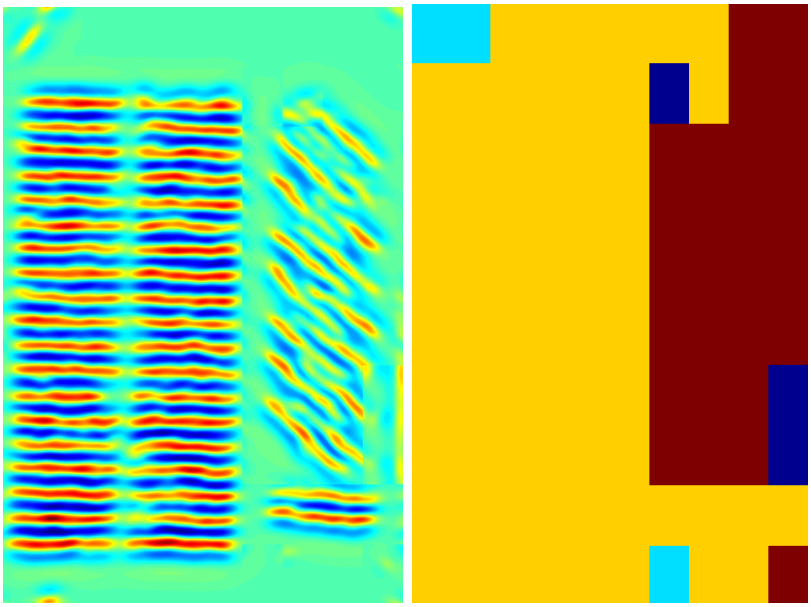


Figure 2.5: Convolution response images to Log-Gabor filters with 4 different orientations. Only a single scale is shown here.

Afterwards, an empty image is constructed with the same size as the original image and it is divided into $n * m$ cells. For each cell, the maximum convolution response that reside in the same location in 16 response images is extracted which makes 16 responses for each cell grid. To compute the maximum of these 16 responses, the result of convolving with the even symmetric filter, which are the real components; are summed and then sorted. After the maximum sum is found, the cell is assigned to the response which was extracted as a grid from the response image. Figure 2.6 (a) shows an example image obtained with this approach.

Besides, each maximum response's orientation is saved and each cell is tagged with that orientation. Figure 2.6 (b) shows an example image with $10 * 10$ cells, each tagged with 4 possible different orientations. As a result, two images are constructed from different Log-Gabor filters, resulting one with responses ($ResponseImage$) and the other with regions ($RegionImage$) tagged with integers indicating different orientations (see Figure 2.6).

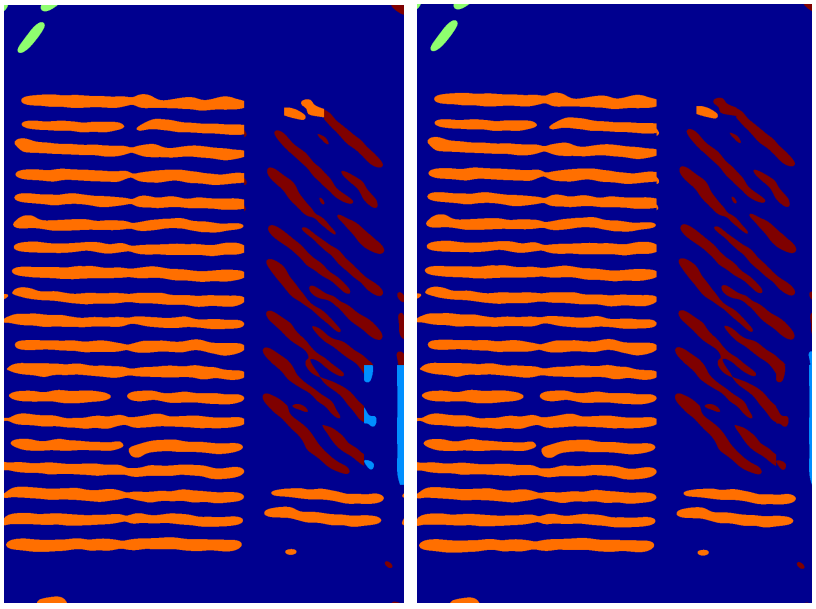


(a) (b)

Figure 2.6: (a) Response image constructed from $4 * 4$ Gabor filtering results with finding the maximum response for each cell ($ResponseImage$). (b) Region image with cells tagged with 4 possible values indicating orientations computed from maximum responses of different filters ($RegionImage$).

Afterwards, to detect the approximation of the line boundaries, binarization is applied to the computed response image (RI). A predefined threshold is used for binarization, which is $max(ResponseImage)/10$. The resulting image indicates the approximate of line boundaries, where boundaries are tagged as 1 and background pixels are tagged as 0 ($BoundaryImage$).

To approximately find the regions which have differently oriented lines, the $BoundaryImage$ computed from the $ResponseImage$ is combined with the $RegionImage$. Each pixel of the binary image with the value 1, is assigned to the value of the pixel which is at the same location in $RegionImage$. Figure 2.7 (a) shows the resulting image.



(a) (b)

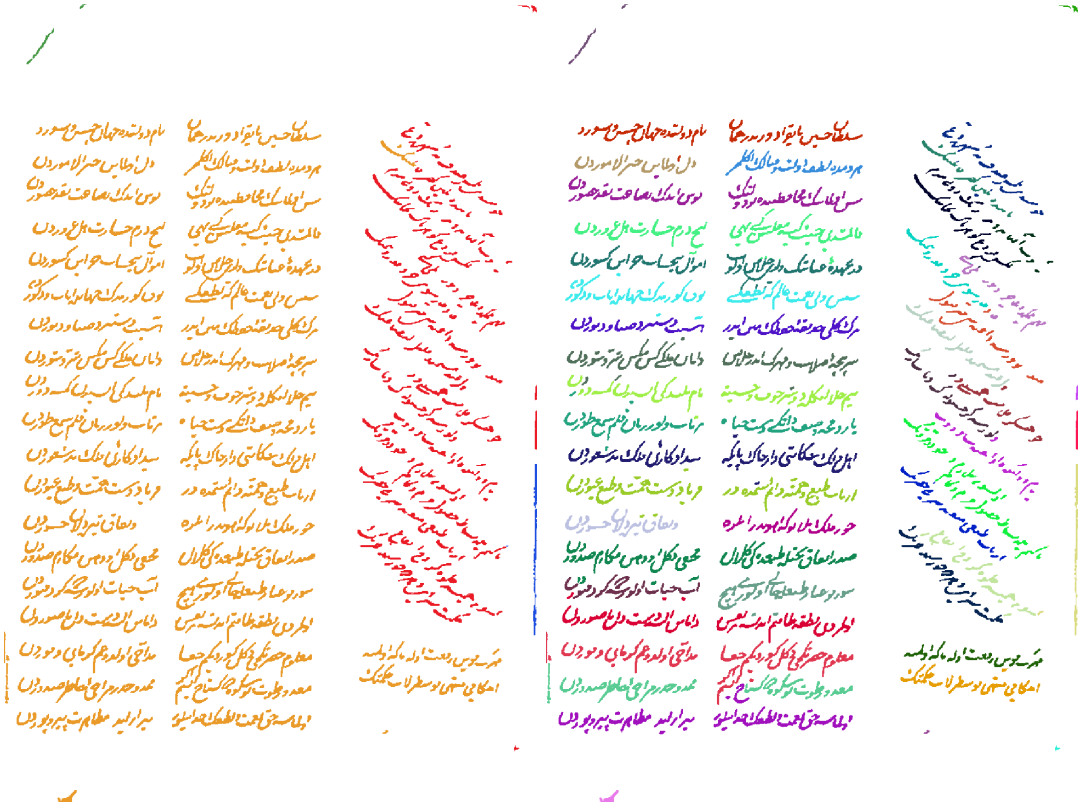
Figure 2.7: (a) *BoundaryImage* combined with the *RegionImage* showing approximate line boundaries with orientation tags. (b) Post-processed image of (a).

After this step, post-processing is applied to correct some results. Firstly, connected components that have multiple orientation tags are found and if the ratio of the pixels with different values are lower than 50%, the pixel group which are the minority are assigned to the majority pixels' value. The reason we are using a threshold like 50% is to eliminate assigning a connected component to a single orientation while it contains multiple regions. Figure 2.7 b shows the resulting post-processed image.

To segment the image, we prefer tagging each connected component with a value indicating the orientation of the line that it belongs. To obtain this, first we intersect the original binary image with the image that indicates boundaries with orientations. The ink pixels intersecting with a boundary are tagged with the boundary's orientation. There remains some ink pixels in the original image which do not intersect with any of the boundaries. Thus, we find those pixels and for each of them, we count the votes using a grid whose size is predefined and

centered on the pixel. Therefore, each unassigned pixel is tagged with a value indicating the orientation.

Final step is computing the majority of the tags for each connected component. What we mean by this statement is that, sometimes connected components of the original image are assigned to multiple tags cause they were partly intersecting with multiple boundaries. A similar approach we explained in the post-processing step is used except there is no threshold. Thus, all connected components of the original image are assigned to single values. Figure 2.8 a shows the result of layout segmentation.



(a) (b)

Figure 2.8: (a) Layout segmentation result. (b) Line segmentation result.

Line segmentation of these kind of documents, documents that include multi-oriented lines, can be done in a very similar approach to layout segmentation.

Instead of assigning orientations to line boundaries which is computed by binarizing the maximum Gabor response per cell, each line boundary can have a different tag indicating the line identification number. Similar post-processing techniques can be used such as using voting for correcting the boundary tags or connected component tags. Figure 2.8 b shows an example line segmentation result done in a similar fashion.

2.3.3 Line Segmentation

We propose 4 different line segmentation algorithms. First one is projection profile, which is a traditional line segmentation method generally preferred for printed documents with straight lines. Second one is a new method that we designed specifically for Ottoman documents which uses advantages of both projection profile and connected component based methods. Third and fourth algorithms are based on Gabor filtering where the first one uses region intersection after the line boundaries are extracted, while the other one fits lines to boundaries as baselines. These 2 algorithms were designed so that they will suppress language based features and emphasize the line structure and because of these characteristics they prove to be successful on script-independent documents.

2.3.3.1 Projection Profile

Projection profile is applied for segmenting lines of Ottoman documents in other studies too. The authors of [10] indicates that finding positions of baselines and segmenting lines according to character sizes is a better solution than finding spaces between lines. The reason comes from the fact that, Ottoman language includes many characters with long ascender and descender parts resulting in narrow spaces between lines. They propose a method, where vertical projection profile of the image is calculated and then peaks of the profile are extracted as lines. To detect the peaks, projection profile values are compared to a threshold value with the intuition of lines should have greater number of black pixels in the profile. However, using a single threshold value may produce extra or missing lines in the results. To eliminate that, we propose a new idea for detecting the peaks of the profile. First a Fourier curve [40] is fitted to the profile and local maxima

points are found. These points are thought to be the location of baselines.

Fourier curve can capture the repetitive pattern of lines. A Fourier series is defined as:

$$y = a_0 + \sum_{i=1}^{n} a_i \cos(nwx) + b_i \sin(nwx); \tag{2.4}$$

where the function is a weighted sum of sine and cosine functions that describes a periodic signal, a_i's are the weights, n is the number of terms and w is the fundamental frequency of the curve.

Finally, connected components are assigned to closest baselines.

2.3.3.2 A Hybrid Approach

Ottoman language has some common properties with Arabic; most notably the alphabet and the writing style which relies on dots and diacritics heavily. However, these small-sized components may produce ambiguous results for line segmentation since they usually lie between the text lines. In [47, 48] it is mentioned that diacritical points can generate false separating or redundant lines.

Some line segmentation studies applied on languages that include diacritical symbols [48, 49] does not filter these small connected components during line segmentation and then apply a post processing step for correcting the approximate results. On the other hand, some studies [34, 19] eliminate those small-sized components during segmentation and reconsider them to generate the final line segmentation results.

We propose a method that ignores small-sized components during line segmentation to obtain results more accurately without post processing. After we detect all connected components, the small ones are marked so that they will not be used during detection of the lines.

To find the small-sized components, each connected component's filled area is calculated and then components which have a smaller filled area than a predefined threshold are marked as small. After the lines are detected, small-sized components are reconsidered and assigned to related lines. Figure 2.9 b shows the document image without small-sized components. As it can be observed, the text line structure is enhanced.

(a) (b)

Figure 2.9: (a) Binarized document image, (b) image without small-sized components.

Baseline is the fictitious line which follows and joins the lower part of the character bodies in a text line [50]. Thus, each connected component has baseline pixels that fit or come close on its baseline. In this study, for baseline extraction first each connected component's baseline pixels are found approximately. To find those pixels, contour image of the connected component is obtained (see Figure 2.10 (b)). Then, left-to-right (see Figure 2.10 (c)) and right-to-left (see Figure 2.10 (d)) gradient image, measuring the horizontal change in both left and right directions are calculated.

Also, the bottom-to-top (see Figure 2.10 (e)) gradient image which shows the vertical change in the upward direction is obtained. Then these 3 gradient images are subtracted from the contour image which results in the group of pixels that approximately lie on the baseline (see Figure 2.10 (f)).

To obtain the exact baseline pixels, first the y-coordinates' standard deviation (s) and mean (m) values are calculated. Then the pixels where $|p - m| > s$ are considered as outliers and removed from the group. The rest of the pixels are used as baseline pixels (see Figure 2.10 (g)).

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

Figure 2.10: Procedure of extracting baseline pixels. (a) connected component, (b) contour image, (c) left-to-right, (d) right-to-left, (e) bottom-to-top gradient images, (f) approximate baseline pixels, (g) exact baseline pixels.

This procedure is applied for each connected component and a new image is reconstructed from these obtained baseline pixels which can be seen from Figure 2.11. Then to detect the baselines of each line, vertical projection profile of the reconstructed image consisting of baseline pixels is obtained (see Figure 2.11). The peaks of this profile can be interpreted as lines and the valleys as interline gaps.

To detect the peaks, a Fourier curve [40] is fitted to the profile and local maxima points are found. Fourier curve can capture the repetitive pattern of lines.

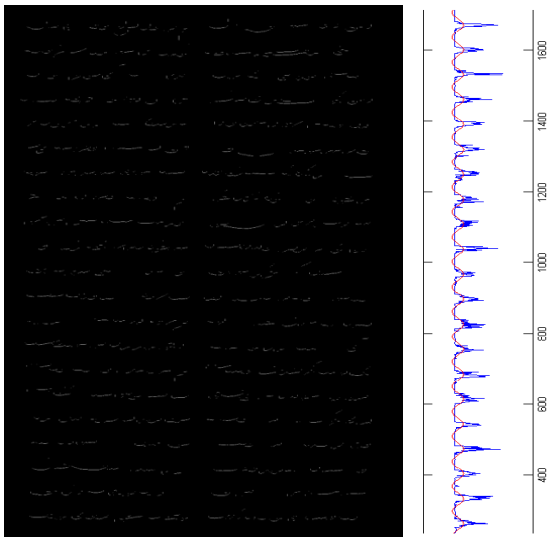
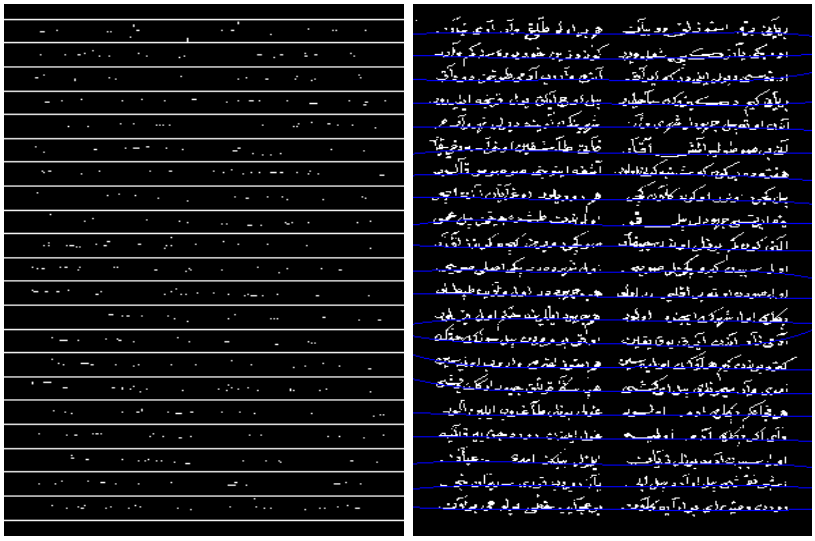


Figure 2.11: Left part shows the image reconstructed from the baseline pixels of connected components, right part is the vertical projection profile of the reconstructed image. A Fourier curve is fitted to the projection.

Then, for each two adjacent peaks of the curve, the smallest value in the profile that lie between these peaks, which is usually zero, is obtained as a cut point. Thus, for each gap a cut point is calculated respectively. These points can be considered as an approximate of the interline gaps and are used for separating the baseline pixels that belong to different adjacent lines (see Figure 2.12).

After obtaining the baseline pixels that belong to each line (see Figure 2.12 (a)), polynomial curves are fitted to each group of those pixels to calculate the actual baselines (see Figure 2.12 (b)). Line fitting can also be used however; we preferred to use a 4th degree polynomial to tolerate some amount of curvature.



(a) (b)

Figure 2.12: (a) Approximation of interline gaps, (b) computed baselines.

After the baseline curves are extracted, the connected components which are not marked as small are assigned to their closest curves. To find the closest curve of a component, the distance function is obtained from the curve's equation and the component's midpoint. Then, the derivative of the distance function is computed to find the closest distance between the midpoint and the curve.

Finally, the component is assigned to the curve which has the minimum distance to its midpoint.

To finalize the results, removed diacritical components are assigned to lines. First each small-sized connected component's nearest neighbors in 4 directions (right, left, up, down) are found. The nearest neighboring components should not be small-sized thus, must be assigned to some line.

The 4 nearest neighbors' assigned lines are voted accordingly to their distances to the small-sized component. To, illustrate if the nearest neighboring component in some particular direction is closest to the small-sized component, its line id gets the highest vote. With this voting scheme each small-sized connected component has at most 4 different line candidates with their votes calculated according to the distances. Finally, each small-sized component is assigned to the line which has the highest vote.

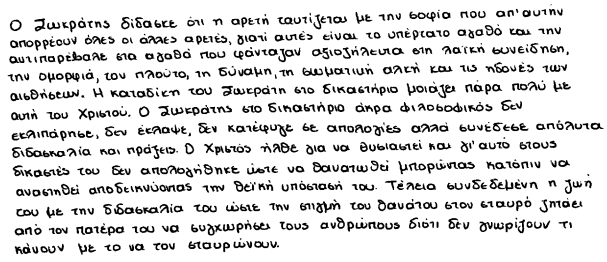
2.3.3.3 Gabor Filtering with Region Intersection

To achieve script-independency for line segmentation, an algorithm not only enhances the line structure but suppresses the language based characteristics should be used. Thus, we designed and algorithm that applies morphological and preprocessing operations on the image before the line segmentation step, to eliminate the mislead of language based properties. Also, Log-Gabor filter bank that we used for layout segmentation is also used here to detect line boundaries.

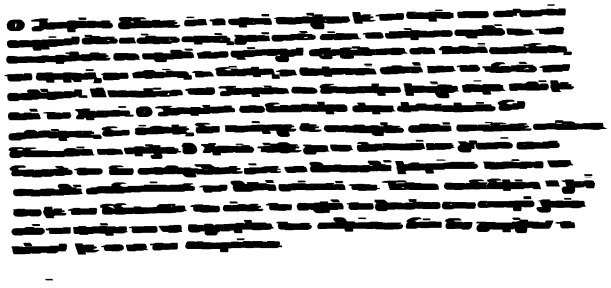
First of all, average connected component width ($avgW$) is calculated. The image is dilated with a disk structuring element of size average width ($avgW$). Then, opening and closing is applied with a disk of size 2. Small connected components are removed from the image in the same fashion explained in section 2.3.3.2 and they are left to be assigned after the lines are detected.

Since we are studying hand written documents many of them include characters with long ascender and descender parts although they are in different languages. By intuition these parts may cause problems during line segmentation since they complicate the process of discriminating lines with gaps. To eliminate this problem, extremes of connected components are smoothed. First, connected components whose rate of filled over convex area larger than 70% is found. These

connected components are thought to have extremes. The y-axis contour coordinates of the connected components are found and their mean (m) and standard deviation (s) is computed. Finally the pixels whose y-values lie in the range of $(m-s, m+s)$ is cropped from the connected component and the others are considered as outliers and removed from the component. Figure 2.13 shows the so far pre-processed image with the processes: morphological operations, small sized connected component removal and extreme smoothing. As you can see, the line structure is enhanced and language based properties are less significant.



(a)



(b)

Figure 2.13: (a) Original binary image to be line segmented. (b) Pre-processing applied to (a) with processes: morphological operations, small sized connected component removal and extreme smoothing.

From the pre-processed image connected components are extracted and baseline's of components are computed with the same approach explained in section 2.3.3.2. As an additional step instead of using baseline pixels, lines are fitted

to each connected component's baseline pixels and actual baseline formulas are used. Afterwards, an image is constructed from those baselines which can be seen in Figure 2.14.

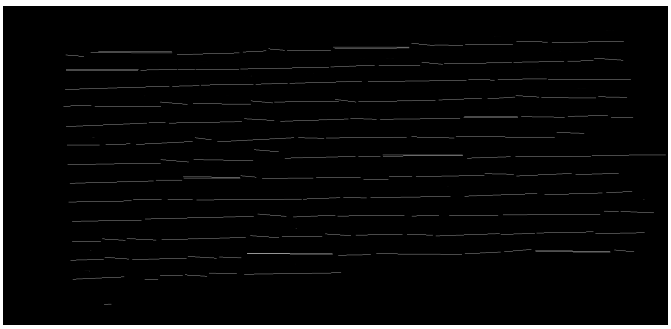


Figure 2.14: Image constructed from the baselines of connected components.

The projection profile method we explained in section 2.3.3.1 is applied to the constructed baseline image to compute the frequency of the Fourier curve which indicates an approximate of gap and line distance together (GaL). This metric is used as a parameter for the wavelet scale of the Gabor filter. A filter bank is constructed from Gabor filters with a single orientation and 4 scales. Peter Kovesi's Gabor Convolve software is used for the task [2].

Since we explained how to segment lines from documents with multi-oriented lines in section 2.3.2 and our line segmentation datasets include documents with similarly oriented lines, we used single orientation for the Gabor filters. Different scales are used to decrease the error rate caused by computing the gap and line distance (GaL). Thus, multiple scales are used to find the best approximation of gap distance.

Wavelength of the smallest scale filter is set to half of the gap and line length ($GaL/2$) and scaling factor between successive filters is set to 2, which results in filters with scales $GaL/2$, GaL, $2GaL$, $4GaL$. Ratio of the standard deviation of the Gaussian describing the log Gabor filter's transfer function in the frequency domain to the filter center frequency is set to $0,65$ and ratio of angular interval

[2] `http://www.csse.uwa.edu.au/~pk/research/matlabfns/`

between filter orientations and the standard deviation of the angular Gaussian function used to construct filters in the frequency plane is set to $1, 3$.

The filter bank is convolved with the constructed image and the maximum convolution response image is chosen. To find the maximum, the result of convolving with the even symmetric filter, which are the real components are summed and then sorted.

Afterwards, to detect the approximate of the line boundaries, binarization is applied to the maximum convolution response image ($ConvolutionImage$). A predefined threshold is used for binarization, which is $max(ConvolutionImage)/10$. The resulting image indicates the approximate of line boundaries, where boundaries are tagged as 1 and background pixels are tagged as 0. An example output can be seen in Figure 2.15.

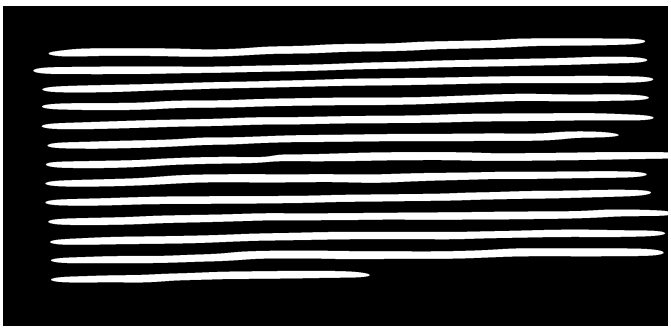
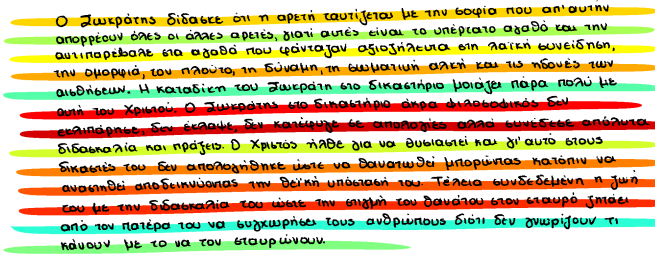
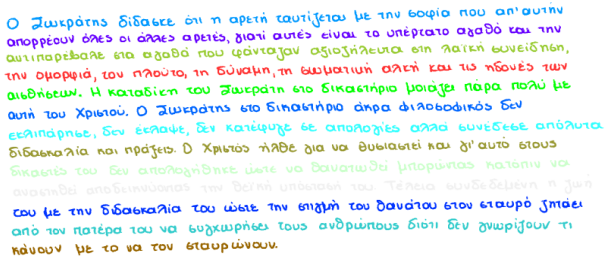


Figure 2.15: Approximate line boundaries computed by binarizing the maximum convolution response.

The intersection of line boundaries and connected components of the original image is computed (see Figure 2.16 a). For each connected component, the largest intersecting line is found and the component is labeled with that line's id. There remains unlabeled connected components which do not intersect with any of the boundaries and components which are labeled as small before. Each of them is assigned to its closest labeled connected component's line. Figure 2.16 b shows the final line segmentation result.



(a)



(b)

Figure 2.16: (a) Intersection of line boundaries with ink pixels is visualized. (b) Final line segmentation result.

2.3.3.4 Gabor Filtering with Line Fitting

Our last line segmentation method uses the same steps with region intersection algorithm to detect the line boundaries which are explained in section 2.3.3.3. Afterwards, instead of intersecting connected components with regions, lines are fitted to each boundary and connected components are assigned to their closest lines. Figure 2.17 shows an example image with fitted lines to each boundary.

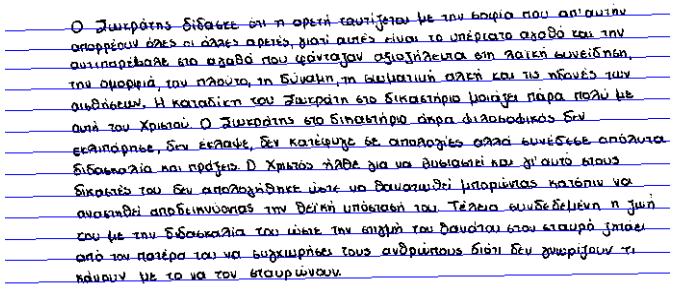


Figure 2.17: Baselines shown on the original binary image, computed by fitting lines to line boundaries.

2.3.4 Word Segmentation

Word segmentation is a difficult task for Ottoman documents because words consist of one or more sub-words (see Figure 2.18) and a sub-word means a connected group of characters or letters, which may be meaningful individually or only meaningful when it comes together with other sub-words [9]. This indicates that there are both inter and intra-word gaps and when intra-word gaps are as large as inter-word gaps or when both gaps are very little the words can not be discriminated. To apply word segmentation to Ottoman documents language based rules can be used which requires language experts. Also, supervised techniques can be applied but usually Ottoman archives do not contain segmented documents and word segmentation is usually required before recognition.

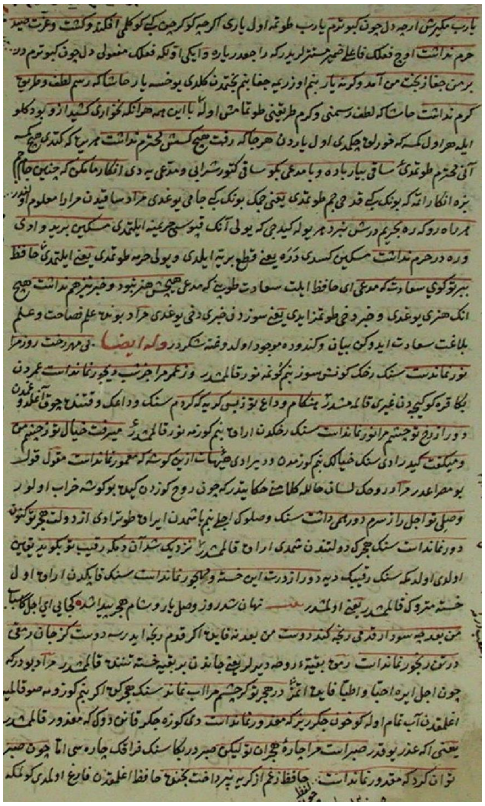


Figure 2.18: Example page in Ottoman.

Therefore, Ottoman word segmentation is not in our study's scope. We designed a simple morphology based word segmentation algorithm for script-independent documents. Most of the word segmentation algorithms are based on analysis of character space distances. Thus, we use basic morphological operations like dilation, opening and closing to generate an output where each connected component will be a single word.

After line segmentation for each line image, average connected component height ($avgH$) and width ($avgW$) is found. Then a structuring element of size proportional to average component sizes ($avgH$ and $avgW$) is generated and the source image is dilated with that structure. Opening and closing is applied with a disk of size 2. Each connected component in the resulting image is thought to be a word. So, the intersection of the resulting image and original image is found and each word is tagged with a different label. Figure 2.19 shows an example

word segmentation result for a Greek document.

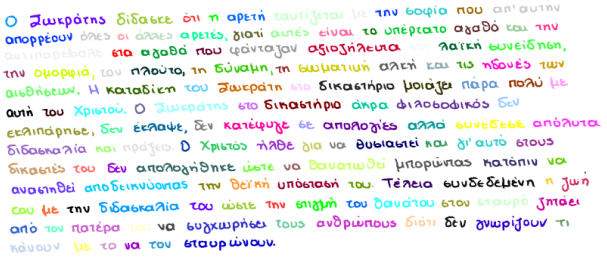


Figure 2.19: Word segmentation result for a Greek document.

Chapter 3

Segmentation Experiments

3.1 Dataset Descriptions

3.1.1 Ottoman Dataset with Multi-Oriented Lines

The first Ottoman dataset contains 50 handwritten documents from different books with total number of 44973 connected components. It was generated specifically to test layout segmentation. Thus, it only contains documents with multi-oriented lines and a document in this dataset should at least include 2 line groups with different orientations. Also, there are cases where different character sizes, line gaps and even 4 different orientations can be observed on a single page. Examples can be seen in Figure 3.1.

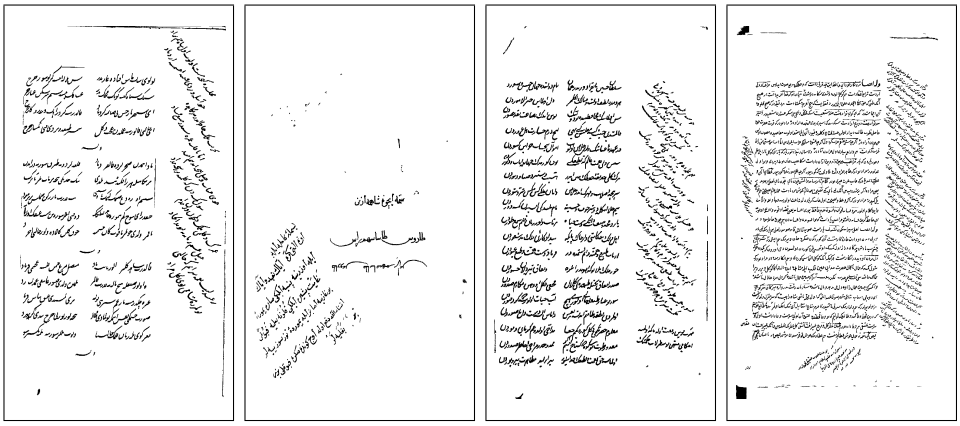


Figure 3.1: Example images from the first Ottoman dataset with multi-oriented lines.

3.1.2 Ottoman Dataset with Similarly Oriented Lines

Second Ottoman dataset is generated to evaluate the performance of the hybrid line segmentation method which was designed specifically for Ottoman documents. This dataset was constructed with 3 different parts. First one and the second one consist of text pages belonging to single books. First part is printed (*Printed*) and includes 120 pages while the second one is handwritten (*Handwritten*) and includes 50 pages. These 2 parts are constructed to compare the algorithm's performance for handwritten and printed texts.

The third part includes 240 pages taken from 6 different books ($Book1-6$), 20 pages from each and the documents are all handwritten. This part is constructed to evaluate the performance under different writing styles and writers. Figure 3.2 shows examples from the dataset. Number of connected components and lines in the dataset are given in Table 3.1.

Figure 3.2: Example images from the second Ottoman dataset with similarly oriented lines. The leftmost one is printed while the others are handwritten.

3.1.3 ICDAR Dataset

This dataset is released on ICDAR 2013 Handwriting Segmentation Contest. It includes 125 English, 125 Greek as well as 100 Bangla documents. The documents are all handwritten and in binary format. We also converted 50 Ottoman handwritten documents to the dataset's format and added them to obtain comparative results with different languages. Thus, the whole dataset includes 400 pages. Figure 3.3 shows example images from the dataset.

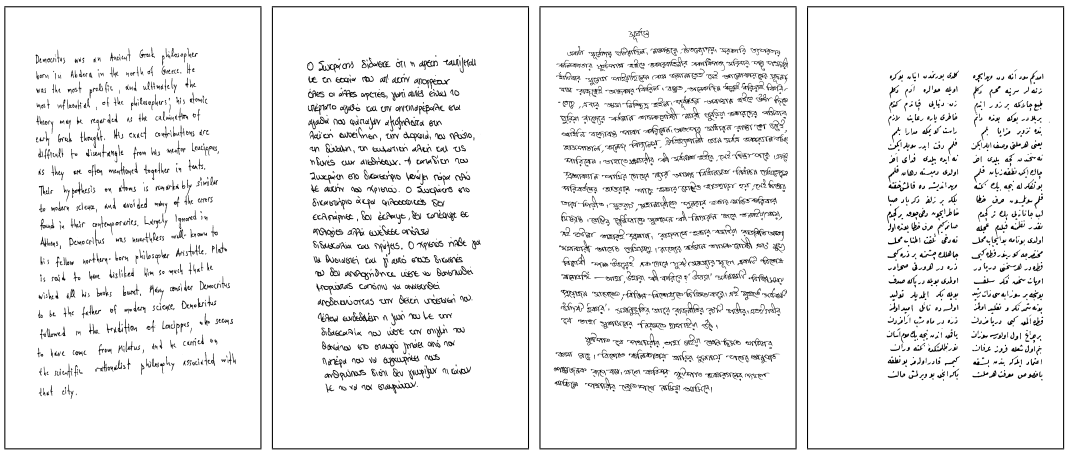


Figure 3.3: Example images from the ICDAR dataset combined with Ottoman documents. From left to right: English, Greek, Bangla and Ottoman document.

3.2 Evaluation Strategies

Segmentation results are evaluated according to the ICDAR 2013 Handwriting Segmentation Contest's evaluation strategies [1].

The performance evaluation method is based on counting the number of one-to-one matches between the areas detected by the algorithm and the areas in the ground truth. A *MatchScore* table is used whose values are calculated according to the intersection of the ON pixel sets of the result and the ground truth.

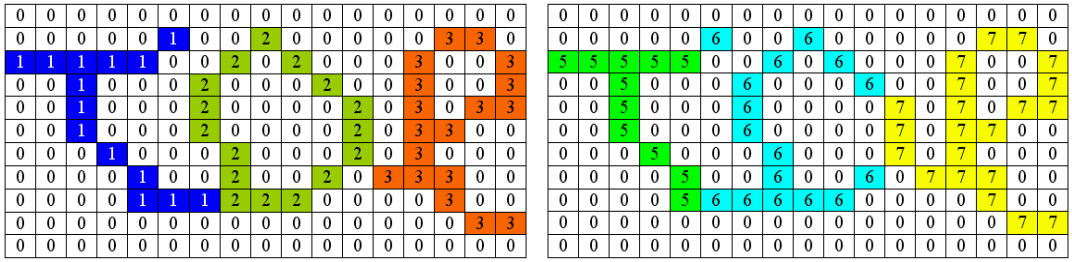
Let G_i be the set of all points of the i^{th} ground truth region and R_j be the set of all points of the j^{th} result region. T is an operator that counts the number

[1] `http://users.iit.demokritos.gr/~nstam/ICDAR2013HandSegmCont/index.html`

of pixels in the zone. Table $MatchScore(i, j)$ represents the matching results of i^{th} ground truth region and the j^{th} result region as follows:

$$MatchScore(i, j) = T(G_i \cap R_j)/T(G_i \cup R_j). \tag{3.1}$$

An example is illustrated in Figure 3.4.



(a) (b)

GT \ RES	5	6	7
1	.79	.11	0
2	0	.68	.08
3	0	0	.86

(c)

Figure 3.4: (a) Ground truth. (b) Segmentation result. (c) Matchscore table.

The matching-scores between all the result zones and the ground-truth zones are obtained. If the matching score is above a predefined threshold ($MS - Threshold$) then the result zone is counted as a $TruePositive$ (TP). Result zones which are not matched to any ground truth zones are $FalsePositives$ (FP) and the ground truth zones which are not matched to any result zones are $FalseNegatives$ (FN). $Precision$, $Recall$ and the $F1 - Score$ are calculated as follows:

$$Precision = \frac{TP}{TP + FP}, \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN}, \tag{3.3}$$

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall}. \tag{3.4}$$

3.3 Experiments and Discussion

3.3.1 Ottoman Document Segmentation Experiments

3.3.1.1 Layout Segmentation Experiments

Ottoman dataset with multi-oriented lines which is explained in section 3.1.1 is used for layout segmentation experiments. All layout segmentation results can be found in layout segmentation results section of Appendix A. Some example results are shown in Figure 3.5. The components which are tagged with same colors are thought to have the same orientation. As it can be observed, each document has more than one region that has a different orientation. Some has regions with changing character sizes and some has connected components which are written so close that their region boundaries are difficult to discriminate.

Layout segmentation algorithm computes the maximum Gabor response per cell (see Section 2.3.2). The number of cells are defined by two parameters n and m and during experiments both of them are set to 10. Thus, each image is divided into 100 cells. These two parameters should not be too large or too small. The cell size should be small enough to discriminate each region while it should be big enough to capture the scale and orientation relationship between components.

To evaluate the results we counted the number of connected components which are labeled wrong. As it was mentioned before, there are total 44973 connected components in this dataset and according to results only 1794 of them are labeled wrong which implies an accuracy rate of 96.01%.

Figure 3.5: Layout segmentation results.

Different types of errors can be observed from wrong labeled connected components. Figure 3.6 shows example error types from results. First type of error is caused by regions which are much smaller compared to others. Figure 3.6 (a) is an example and as it can be observed, the green part has the same writing orientation with the blue part however according to results they are labeled with different orientations. This mistake is caused by the narrowness of the green region. Since those connected components construct a vertical narrow shape, during Gabor filtering a larger scale than their true character scale with a vertical orientation gives higher convolution response than the correct scale and orientation. This problem can be solved by using a single scale but then, the algorithm will loose its character scale-independency and it will lack of detecting regions with different character sizes which will result in a decrease in accuracy.

Figure 3.6 (b) and (c) shows example errors of the second type. They are caused by connected components that are written very closely to other connected components with different orientations. Thus, those adjacent connected components are tagged with the wrong region's orientation. One solution might be searching the document and looking for connected components that are shaped in the same way without rotation. After finding them, their orientations can be observed, if the majority of the orientations is not same with the result, the result can be changed. However, finding similarly shaped connected components is not an easy task for handwritten documents especially for historical Ottoman documents. Therefore, this solution can be preferred for printed documents. Another solution might be using language based rules because it is even hard to discriminate the closely written regions for a person who does not know the language. Therefore, an unsupervised technique would not solve the problem for handwritten documents.

Third type of error is caused by components that are shaped in a way which looks like they are written with a different orientation. Figure 3.6 (d) shows an example. The component that is shaped on the vertical axis is labeled with the vertical orientation's tag which is incorrect. Also, it's incorrect label misguided closely located components' results. The same solutions suggested for the second type of error which is explained in the above paragraph can also be used to eliminate this error.

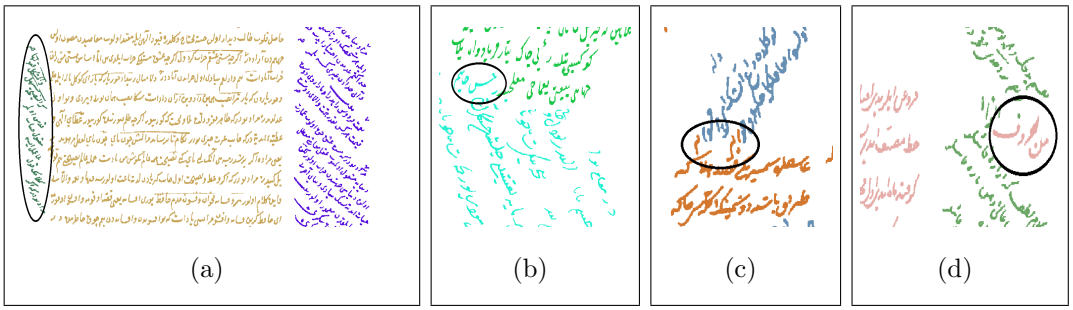


Figure 3.6: Layout segmentation errors showed with black circles. (a) Type 1 error caused by small regions. (b) and (c) Type 2 error caused by closely located components. (d) Type 3 error caused by shape illusions.

As explained in section 2.3.2, 4 different orientations and scales are used resulting in 16 different Gabor filters. These parameters can be changed according to the dataset that layout segmentation is applied. If the character size inside a single document does not change, then a single scale can be used. Similarly, number of orientations can be chosen according to the number of different orientations observed inside the dataset. However, although an increase in the number of different filters will result in better segmentation for some documents, the total accuracy might decrease because of the fact that more components will fall into error categories (see Figure 3.6).

It is explained in section 2.3.2 that line segmentation can be easily applied to multi-oriented documents after layout segmentation. Figure 3.7 shows some example results for the multi-oriented Ottoman dataset generated by the line segmentation algorithm explained in section 2.3.2.

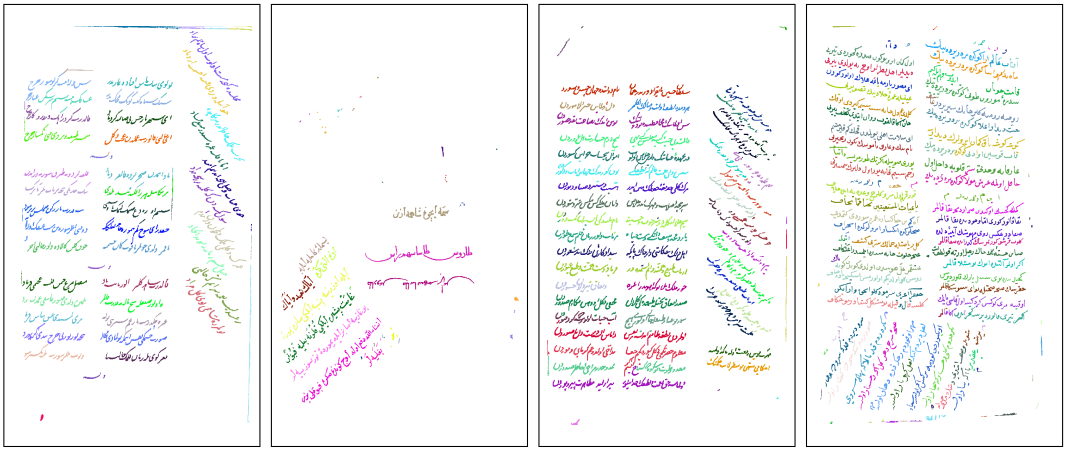


Figure 3.7: Line segmentation results of Ottoman dataset with multi-oriented lines.

3.3.1.2 Line Segmentation Experiments

For this experiment Ottoman dataset with similarly oriented lines is used which is explained in section 3.1.2. Table 3.1 shows the number of lines and connected components in the ground truth and the results for each part of the dataset.

	Lines in GT	Detected Lines	CCs	Correct Detected CCs
Book1	923	921	5208	4847
Book2	880	879	4634	4315
Book3	871	869	4489	4154
Book4	795	793	4132	3884
Book5	764	763	3795	3375
Book6	836	834	4208	3824
Printed	3210	3210	16157	15510
Handwritten	1068	1041	5573	5245

Table 3.1: Results obtained with different printed and handwritten datasets with different writing styles and writers.

As it can be observed from Tables 3.2, 3.3, 3.4 and 3.5 the line segmentation results for both printed and handwritten datasets are considerably high at $MSthresholds$ 0.95 and 0.90. The $F1 - Score$ for the 6 different handwritten books is nearly 0.93 and 0.94 for $MSthreshold$ 0.95 and 0.90 respectively. The segmentation accuracy is nearly same for different books. Thus, it can be concluded that, the change in the writing styles or writers which means different interline spacing, character sizes and line skews does not have an impact on segmentation results.

0.95	**Precision**	**Recall**	**F1-Score**
Book1	0.9665	0.9067	0.9357
Book2	0.9703	0.9101	0.9392
Book3	0.9552	0.9030	0.9283
Book4	0.9710	0.9024	0.9354
Book5	0.9568	0.8937	0.9242
Book6	0.9544	0.8973	0.9250

Table 3.2: Results obtained on 6 different handwritten books with MS threshold of 0.95.

0.90	Precision	Recall	F1-Score
Book1	0.9893	0.9085	0.9472
Book2	0.9932	0.9120	0.9509
Book3	0.9875	0.9133	0.9490
Book4	0.9926	0.9085	0.9487
Book5	0.9792	0.9057	0.9410
Book6	0.9858	0.9069	0.9447

Table 3.3: Results obtained on 6 different handwritten books with MS threshold of 0.90.

Also, as it was expected Tables 3.4 and 3.5 show that the segmentation accuracy increases for printed texts. However, there is not much difference between them and the results of handwritten documents, which means the algorithm is successful for segmenting both printed and handwritten documents.

We observed that most of the errors are due to inadequate binarization, noisy components such as page numbers or ornamentation and assigning small-sized connected components to wrong lines. Binarization errors are due to dark stains that cannot be separated from the ink pixels and multiple ink colors used in the document. The noisy components can be detected as a separate process or removed manually before segmentation. Moreover, most of the wrong assigned small components are very difficult to classify without language dependent metrics. Figure 3.8 shows example results of the algorithm.

0.95	Precision	Recall	F1-Score
Printed	0.9864	0.9456	0.9656
Handwritten	0.9775	0.9229	0.9494

Table 3.4: Results obtained on 2 different printed and handwritten datasets with MS threshold of 0.95.

0.90	Precision	Recall	F1-Score
Printed	0.9956	0.9503	0.9724
Handwritten	0.9933	0.9237	0.9572

Table 3.5: Results obtained on 2 different printed and handwritten datasets with MS threshold of 0.90.

Figure 3.8: Line segmentation results of the hybrid method applied to Ottoman dataset with similarly oriented lines.

3.3.2 Script-Independent Document Segmentation Experiments

ICDAR dataset combined with 50 Ottoman documents in the same format is used for script-independent segmentation tests. The dataset is explained in section 3.1.3.

3.3.2.1 Line Segmentation Experiments

For line segmentation experiments 4 different methods are used to obtain comparative results which are projection profile (PP), hybrid method (HM), Gabor filtering with region intersection ($GFRI$) and Gabor filtering with line fitting ($GFLF$). The dataset contains documents with 4 different languages: English,

Greek, Bangla and Ottoman. Figure 3.9 shows line segmentation results of 4 methods categorized according to languages and Figure 3.10 shows the results for all dataset categorized according to methods.

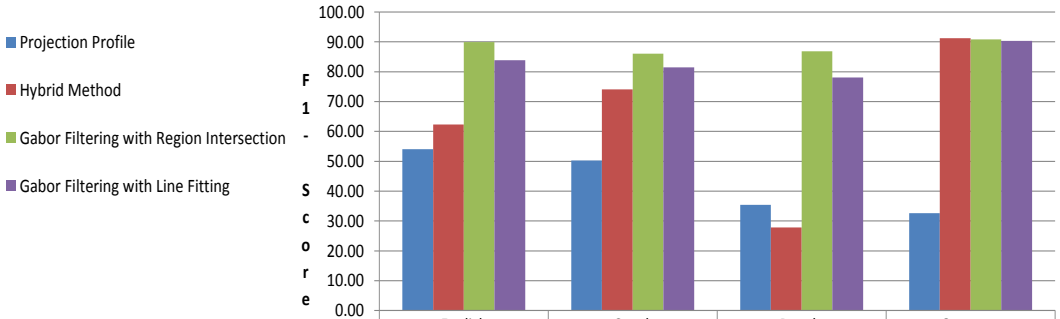
It can be observed that projection profile has the worst performance (see Figure 3.9 and 3.10) because as we mentioned before, projection profile method is appropriate for printed documents with straight lines. The method shows even lower performance for cursive scripts like Bangla and Ottoman (see Figure 3.9).

The best performing method is Gabor filtering with region intersection and it has similar results with Gabor filtering with line fitting method (see Figure 3.9 and 3.10). The reason might be because of having similar pre-processing steps. Only line fitting method fits lines to regions while the other directly uses the regions to segment the lines. Although their performance is similar, region intersection beats line fitting for each language which implies that correctly extracting the baselines of the document is not sufficient for line segmentation. The connected components should be also matched correctly to baselines. Region intersection automatically skips this step so it is more advantageous.

The other line segmentation method which combines connected component based methods and projection profile has average results for English and Greek (see Figure 3.9). It is successful for segmenting the lines of Ottoman documents since it was designed to do so. However, the performance for Bangla documents can be considered really low. The reason comes from the fact that Bangla script has a distinctive horizontal line running along the tops of the graphemes that links them together which is called matra [51]. The hybrid method uses gradients of contours of the components (see Section 2.3.3.2) and excludes bottom-to-top gradient to find the baselines. During this procedure, the matra information is lost and very few pixels are detected as baseline pixels which results in bad segmentation. To solve the problem; instead of extracting the baselines, matra lines can be detected for Bangla documents.

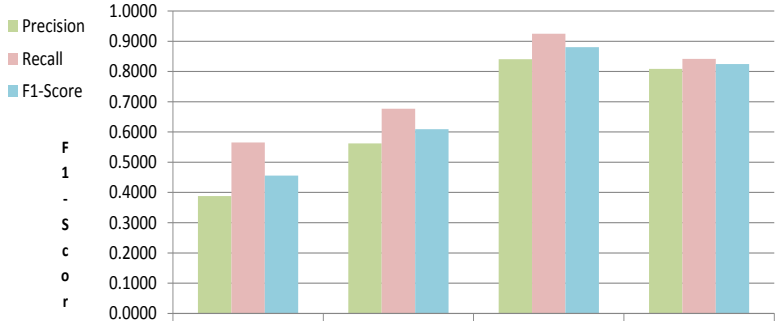
Another thing to mentioned is that, the reason why English and Greek documents have very similar results for each method (see Figure 3.9) is that they are both in Latin. Also, the same patterns can be observed for Ottoman and Bangla document's results, except for the hybrid method which the reason is explained in above paragraph. This is also because of the similarity of the languages which

is being cursive.



	English	Greek	Bangla	Ottoman
Projection Profile	54.07	50.32	35.44	32.62
Hybrid Method	62.38	74.06	27.80	91.20
Gabor Filtering with Region Intersection	89.92	86.02	86.81	90.86
Gabor Filtering with Line Fitting	83.85	81.50	78.10	90.30

Figure 3.9: Line segmentation results of 4 methods categorized according to languages. PP: Projection Profile, HM: Hybrid Method, GFRI: Gabor Filtering with Region Intersection, GFLF: Gabor Filtering with Line Fitting.



	Projection Profile	Hybrid Method	Gabor Filtering with Region Intersection	Gabor Filtering with Line Fitting
Precision	0.3879	0.5623	0.8408	0.8087
Recall	0.5653	0.6770	0.9250	0.8417
F1-Score	0.4556	0.6090	0.8804	0.8248

Figure 3.10: Line segmentation results of 4 methods for the whole dataset. PP: Projection Profile, HM: Hybrid Method, GFRI: Gabor Filtering with Region Intersection, GFLF: Gabor Filtering with Line Fitting.

Table 3.6 shows the results of the best performing method, Gabor filtering with region intersection, categorized according to languages. The table shows that most results are similar for different languages. This implies that the algorithm is script-independent. Also, the changing in writing styles or authors which means different interline spacing, character sizes and line skews does not have an impact on segmentation results. Figure 3.11 shows example outputs generated by the method.

0.90	**Precision**	**Recall**	**F1-Score**
English	0.8628	0.9388	0.8992
Greek	0.7968	0.9343	0.8602
Bangla	0.8365	0.9022	0.8681
Ottoman	0.9043	0.9129	0.9086

Table 3.6: Line segmentation results of GFRI: Gabor Filtering with Region Intersection method categorized by languages. $MS-Threshold$ is taken as 0.90.

Figure 3.11: Line segmentation results of GFRI: Gabor filtering with region intersection for Greek, Bangla, English and Ottoman documents.

3.3.2.2 Word Segmentation Experiments

Last segmentation experiment is word segmentation and as we mentioned before, Ottoman word segmentation is not in our study's scope because of the fact that it can be only done by language based rules which requires Ottoman language experts. Thus only results in English, Greek and Bangla documents are given (see Table 3.7). English and Greek documents have similar results because they are both in Latin.

For all of these 3 languages, the letters run from left to right and spaces are used to separate words unlike Ottoman, where letters run from right to left and spaces are both used as inter and intra word gaps. Although, the linguistic of

using spaces only as inter word gaps can be violated for handwritten documents, still the gaps between words are going to be more obvious than others. Thus, basic morphological operations gives promising results for segmenting words from English, Greek and Bangla documents (see Table 3.7). Figure 3.12 shows example outputs of the algorithm. To increase the results, language based rules or supervised techniques can be applied.

	Precision	**Recall**	**F1-Score**
English	0.6954	0.8921	0.7675
Greek	0.6034	0.9317	0.7237
Bangla	0.6183	0.7126	0.6455
Total	0.6405	0.8449	0.7170

Table 3.7: Word segmentation results categorized by languages.

Figure 3.12: Word segmentation results. From top row to bottom row: English, Greek and Bangla documents.

Chapter 4

Matching Islamic Patterns in Kufic Images

4.1 Motivation

Islamic calligraphy, also known as Arabic calligraphy, has been the main form of artistic expression in Islamic cultures throughout the history [1]. Kufic is one of the oldest calligraphic forms of the various Islamic scripts. Kufic derives its name from the city of Kufa, where it was developed around the eighth century, and until about the eleventh century it was the main script used to copy Qur'ans. Although it has been mainly used as a decorative element in manuscripts, pottery, coins, architecture, stone inscriptions and wooden work for several centuries [52, 53, 54] (see Figure 4.1 for examples), the proverbs and passages from the Qur'an have continued to be used as dominant sources. Its influence on European art during the Middle Ages or the Renaissance can also be recognized and the resulting style is known as pseudo-Kufic or Western-Kufic.

[1] `http://en.wikipedia.org/wiki/Islamic_calligraphy`

Figure 4.1: Some decorative Kufic patterns. Left: Gudi Khatun Mausoleum in Karabaghlar, Azerbaijan, 1335-1338 (Image taken from [1]). Middle: Coin of the Hafsids, with ornamental Kufic script, from Bejaia, 1249-1276 (Image taken from [2]). Right: Tombstone of Abbas, with floriated Kufic, 9th century (Image taken from [3]).

Cultural heritage is a legacy from the past, which should be passed on to current and future generations. Analysis of Kufic scripts, may shed light to a relatively unknown era in history. However, even for a person whose native language is Arabic, it is difficult to determine the meaning of Kufic scripts due to a set of challenges inherent in Kufic calligraphy (see Section 4.2). Therefore, scholarly work on Kufic scripts is limited.

Providing automatic tools for the discovery, documentation and organization of Kufic designs may assist scholars working in this area and help for long-term preservation of this heritage. With the automatic analysis of Kufic designs, one can learn specific stretches of Kufic motifs and similar designs in other places can signify some similar cultural perspectives, at a scale that no human could physically perform. Moreover, automatic analysis and classification of Kufic images may help understanding of their characteristics and design rules, and may lead to the automatic generation of new designs [4].

Despite the need, based on our knowledge, automatic analysis of Kufic images has not been addressed previously. In this study, we aim to fill this gap by developing tools for indexing and retrieval of Kufic images towards documentation and preservation.

In the rest of the paper, first we briefly describe the challenges of studying Kufic images, and review related studies on the automatic analysis of other calligraphy images. Then, we describe our approach towards the matching of sub-patterns in Kufic images, and report the results of experiments. Finally, we discuss the results and conclude.

4.2 Challenges in Kufic patterns

Kufic script is grouped into three categories: written, ornamental and Ma'qeli. Ma'qeli Kufic is known as square Kufic and is also called geometric, rectangular, quadrangular or rectilinear Kufic. It is one of the most common Kufic types used in decoration [1]. Letters in square Kufic are in the form of a square or a rectangle. Geometric shapes consisting of various straight lines [1, 54] can be elongated by 45 or 90 degree angles to compose different motifs [1, 52, 4, 5].

In Arabic, a character may have different shapes depending on whether it is at the beginning, middle or end of a sentence or whether it is in an isolated form [55, 56, 57]. Most characters are only distinguished by the attached dots or zigzags, called diacritics. Moreover, because of the consonantal nature of Arabic, vowels are omitted [58].

Kufic calligraphy images involve additional challenges [1, 4]. Firstly, the direction of the words in Kufic images may change, unlike other calligraphy styles. Calligraphers who have to fill a specific space in a Kufic design, are forced to modify the letters to fit the space, whether by extending them or contracting them, or by changing their shapes. Therefore, a single word or letter can be modified in many different ways to create different motifs resulting in a wide variety of appearances and shapes of the same word or letter (Figure 4.3). Furthermore, there is very little distinction between the shapes of different letters (see Figure 4.2).

Moreover, texts in Kufic can be written in a spiral way, starting from a corner and ending at the center of an image. Bending may introduce new shapes to the letters. Zigzags crossing the design surface results in additional complications. The square shape of the Kufic image can be maintained by designing repeated

patterns around the square or at the center. Letters and their relative arrangements can be updated and the organization can be redefined when a new word is added to the composition [59]. Instead of writing a letter more than once, that letter may be used by two different words in a design, like a crossword puzzle, and similarly a word may be written just once and used by two different phrases in the same design. These challenges, related to the very nature of Kufic calligraphy, became even more daunting due to the differences of the calligraphy style by different cultures and at different periods.

All these challenges make the recognition and matching of Islamic patterns in Kufic images more interesting –and yet more difficult– to deal with and require techniques beyond usual text and handwriting recognition.

Figure 4.2: Square Kufic script letters. Note that, due to the nature of Arabic, the same character may have different shapes, depending on its position within the sentence. Characters may also have different shapes in different designs. (Image courtesy of [4])

Figure 4.3: The same sub-words in different shapes and different sub-words in similar shapes. In the first two images, the gray (sub-pattern *lillah*), light gray (sub-pattern *la*) and lighter gray sub-patterns have different shapes in both designs. For example, the gray ones are different designs of the letter *La*. In the last image, the gray (sub-pattern *lillah*), light gray and lighter gray ones are different sub-patterns but they share similar shapes.

4.3 Related work

In recent years, accessing and preserving cultural heritage has been considered in many different ways [60, 61, 62]. Computer vision techniques have been proposed for automatic indexing and retrieval of historical collections [63]. Here we focus on studies in indexing and generation of Arabic calligraphy [53, 59, 64, 65, 66, 67, 68, 69, 70, 71].

Dunham et al. [64] developed a method to generate a repeating pattern of a hyperbolic plane based on a tiling by any convex polygon. Their method draws patterns based on tilings by a polygon which is not necessarily regular and that polygon is assumed to be convex. In [65, 66], a method based on radially symmetric motifs is proposed to generate Islamic star patterns .

In [68], Aljamali and Banissi proposed a method to classify Islamic geometric patterns (IGPs) based on the minimum number of grids and lowest geometric shape methods necessary for the construction of the star pattern, while in [69], IGP images are described using discrete-symmetry-groups theory. Firstly, every pattern is classified into one of three major categories based on translation in one direction. Secondly, some symmetry features, such as the symmetry group and

the fundamental region, are extracted. As a last step, the fundamental region is described by a color histogram.

In [67], comprehensive analysis and cataloging of Islamic design patterns from digital images is done through plane-symmetry-group theory. By using image segmentation algorithms, these regular design patterns are then grouped by pixels to obtain pieces forming tiles. After vector representation is done, the objects are compared according to their contours and then classified by their shape and color.

In [59], a prototype (Interactive Calligraphy Exploration) is described to compose calligraphic images by manipulating symmetries to produce unusual visual effects. The approach introduces a novel method of interaction with compositional elements and demonstrates how controlled change propagation can be used to promote design exploration.

For Kufic calligraphy, there exists only a few studies for automatic generation [72]. Based on our knowledge, matching Kufic patterns has not been studied previously.

In [72], cellular automata with an extended Moor neighborhood is used to generate square Kufic script patterns, specifically *Muhammed*, by defining some transition rules. Their approach focuses on three most famous *Muhammed* patterns, which makes it applicable only to some models.

4.4 Our approach

As shown in Figure 4.4, our approach involves four steps: (i) foreground extraction, (ii) sub-pattern extraction, (iii) representation and matching, (iv) analysis.

Given a Kufic image, initially decorative elements in the background should be eliminated, and foreground pixels that are the elements of patterns should be extracted. In this study, we consider the sub-patterns as basic units and extracted connected components are used as sub-patterns.

We observe that the relationships between straight lines in specific orientations are important for the identification of square Kufic patterns. Based on this observation, we propose a line-based method for representing and matching sub-patterns. In our method, each sub-pattern is represented with a graph and then

graph and sub-graph isomorphism are applied to match the patterns.

Sub-pattern matching is used for the analysis of Kufic images in three different ways. Given a query pattern, all the instances can be found through retrieval. Going further, through known patterns images can be automatically labeled in the entire dataset. Finally, patterns that repeat inside an image can be automatically discovered.

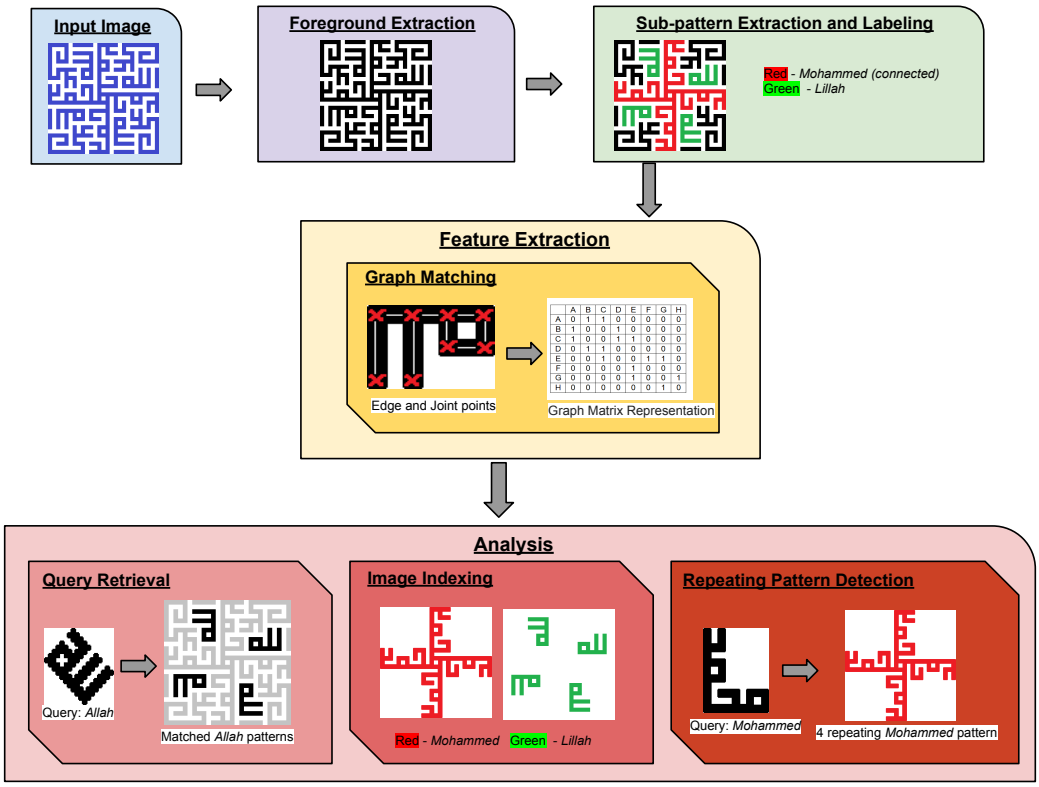


Figure 4.4: The overall organization of our system.

In the following, we will first describe the collection, and then labeling and decomposition of the dataset. In the next section, we present our methods for feature extraction and sub-pattern matching.

4.4.1 Extraction of foreground pixels

As it can be seen from Figure 4.5, due to the different origins of the images, the dataset includes examples in a large range of variety, from multi-color ornamental images to degraded and noisy images.

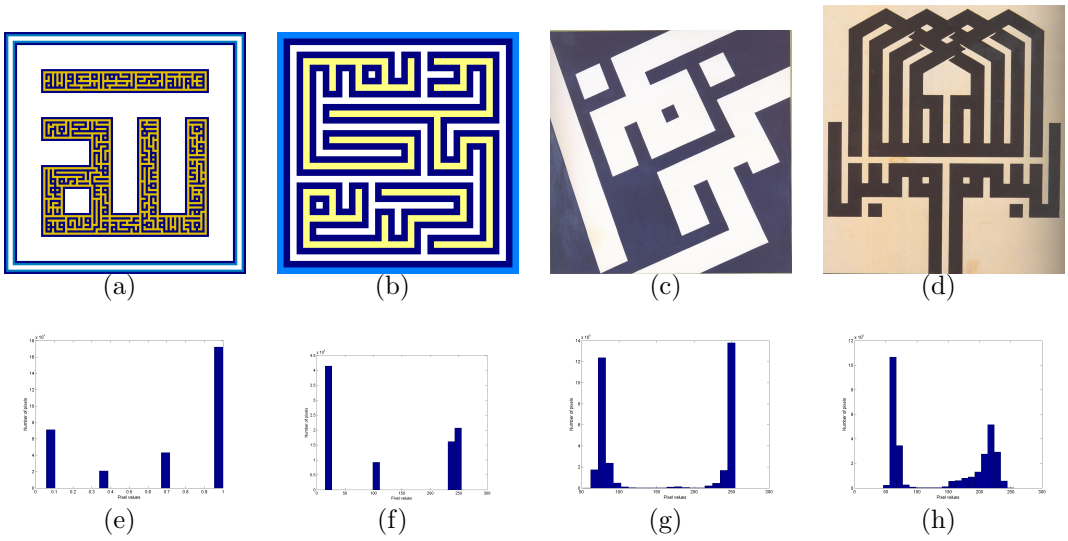


Figure 4.5: Top (a-d): Example Kufic images, bottom (e-h): their corresponding color histograms. As can be seen from the color histograms (a) and (b) has a few distinct colors, while the degradation in (c) and (d) result in multiple colors.

These different characteristics of images are reflected on their corresponding color histograms. We observe that, the clean images may have a few distinct colors –varying from two to eight colors for the images in our collection–, with clear peaks on the color histogram. On the other hand, noisy and degraded images, –which happen to be dominated by images with two main colors in our collection–, have likely to have shorter peaks where the strengths are reduced with similar colors around.

While standard binarization methods, when applied with adaptive thresholding, may resolve the problems in degraded images with two colors, on multi-color images they fail, and cannot extract the foreground pixels properly.

As a solution, we design a two stage method to extract the regions corresponding to foreground pixels. First, we divide the images into two sets according to their color distribution characteristics, and then apply different techniques to each set.

The first stage separates clean images from the degraded images by looking at the number of distinct colors in the color histogram. If this number is less than a predefined threshold, then the image is considered as a relatively clean multi-color image, otherwise it is considered as a degraded one. The threshold is selected as eight in our experiments, since we observed that there are at most eight different distinct colors in the images in our collection.

To extract the foreground pixels corresponding to Kufic scripts in multi-color images, and to eliminate the pixels corresponding to ornaments or frames, we propose a method based on color masking. Each peak value on the color histogram is selected and the image is masked with that value. The regions corresponding to selected color are marked with one, and the others are marked with zero in the output image. Then connected components are extracted on the binary image. This process is repeated for all the distinct colors. The color which results in the highest number of connected components is considered as the foreground color. Note that, the assumption in behind of this method is that, the image is dominated by the sub-patterns in Kufic script.

The second set of images includes degraded and noisy images. Since we observed that in most of the images backgrounds are distinguishable from the foreground, we applied Otsu's method for binarization without further processing.

4.4.2 Extraction and labeling of sub-patterns

We focus on four patterns that are very common across the square Kufic images. These patterns are *Allah*, *Muhammed*, *Resul*, and *Lailahe illallah*. We limit the retrieval and indexing experiments with these four patterns due to the difficulty in labeling. Note that, the proposed method is not restricted to these patterns only. It can match any other sub-pattern as will be shown by the experiments for automatic detection of repeating patterns.

Figure 4.6 shows some *Allah* patterns. This pattern is the most common one in the dataset and it is a combination of two sub-patterns, which are *Alif*

and *lillah*. *Muhammed* pattern is the second common pattern in our dataset and while it consists of a single sub-pattern it has a large variety among its instances (see Figure 4.8). *Resul* pattern is formed by three sub-patterns as it can be seen in Figure 4.7 and Figure 4.8. The longest pattern type in dataset is *La ilaha illa Allah* pattern (see Figure 4.8), which is composed of seven sub-patterns: one *lillah*, three *Alifs*, one *leh*, two *la*.

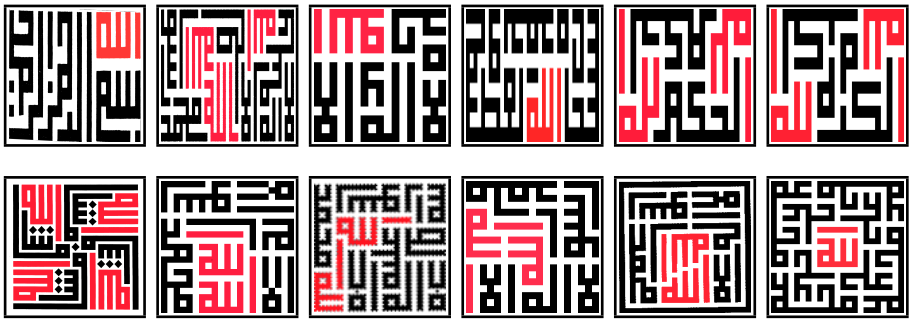


Figure 4.6: Some square Kufic images with *Allah* patterns are shown in red (gray). Note that this word has different shapes in different designs (Images 1,2,3 are taken from [4], 4-10 and 12 from [5]).

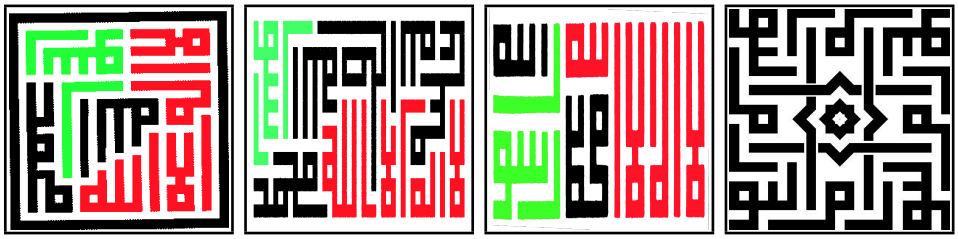


Figure 4.7: The patterns in green (light gray) are *Resul* patterns, which are formed by three sub-patterns. The red (gray) sub-patterns are from the *La ilaha illa Allah* pattern. The last image contains four *Resul* patterns at each corner (First image is taken from [6] and second one is from [1] and third-fourth ones are from [7]).

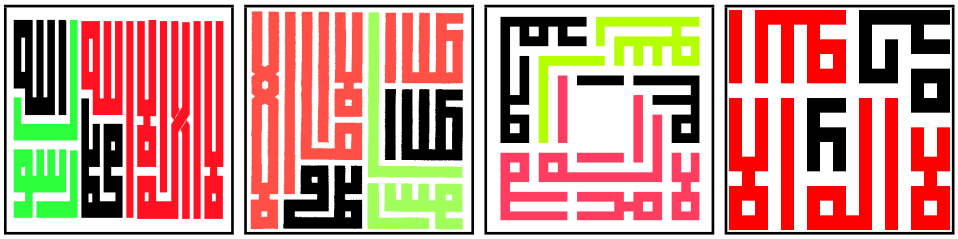


Figure 4.8: The patterns in red (gray) are *La ilaha illa Allah*. The green (light gray) patterns are *Resul* patterns and note that the last one's two sub-patterns are connected. In the first image, first black pattern is *Allah*, while second black one is *Muhammed* and in the second image first black sub-pattern is *Muhammed*, while the second one is *Allah* and the same for the third image. (The first and second images are taken from [7] and the third from [1]).

To extract the patterns, first all the connected components (CCs) from the binarized images are extracted [73] using OpenCV Library [74]. In total there are 8082 extracted CCs. We will refer to the CCs that are parts of the patterns as sub-patterns.

Rather than labeling all the sub-patterns for the four patterns used in our experiments, we choose only the discriminating ones. Similarly during querying we also aim to find these discriminative components. Note that, these sub-patterns still may also be placed in other patterns and have large variations in appearance.

Table 4.1 depicts the number of samples labeled for each sub-pattern. Components that are not labeled as one of the four patterns are put into the unlabeled class.

Label	Number of components	Number of images	Example
la	80	26	
leh	170	45	
lillah	938	129	
muh	184	37	
su	26	15	
unlabeled	6684	203	

Table 4.1: Number of components per class and number of images where these patterns are found. Note that an image may contain more than a single labeled pattern.

As shown in Figure 4.9 for the sub-pattern *su*, instances of a sub-pattern class can be in various sizes and rotations. In the overall dataset, height of the components vary between 14 pixels to 1918 pixels, median height is 50 pixels. Width of the components vary between 8 and 1840 pixels, median width is 47 pixels. Median aspect ratio is 1, 25% of the components are square, 35% are landscape and 40% are portrait.

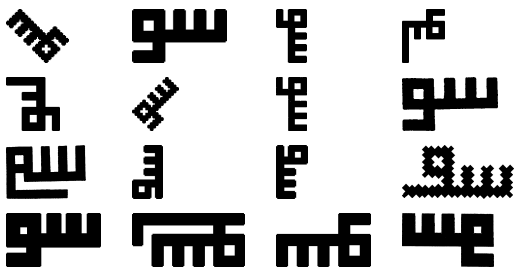


Figure 4.9: 16 sample images from *su* sub-pattern. This is the middle component of the word *Resul*.

4.4.3 Sub-pattern matching

As described in the previous section, all the sub-patterns are extracted automatically both for the query and for the dataset images. Then the discriminative

sub-patterns in query pattern are searched among all the sub-patterns in the dataset. However, sub-pattern matching is a challenging task due to large geometric variations within a class.

In this study, we addressed this problem and proposed a new descriptor and a matching approach for sub-pattern matching. Our method is based on graph isomorphism. In the following, first we describe how we represent sub-patterns as graphs, then we present the details of graph matching method.

Note that, the techniques in character recognition cannot be directly applied for our problem, since it is difficult, if not impossible, to segment the words or phrases into characters.

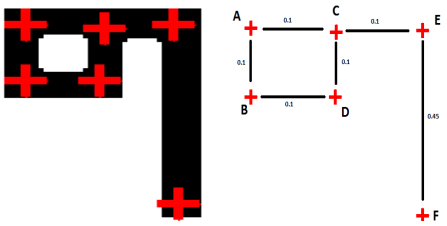
To represent the sub-patterns as graphs, we utilized the skeletons extracted from connected components. First we applied smoothing to get rid of knurls and noisy edges. Then, the endings and junctions of connected components are extracted using an available junction/ending extractor software [2].

The software produces many junction points for components with ragged edges. However, erroneous junction points may create extra nodes in the graph, and change the graph structure. To eliminate the unnecessary junction points, we checked the distances between each junction point pairs and only the ones that exceeds a pre-defined threshold are kept. This threshold is set relative to the minimum of width and height of the connected component. As seen in Figure 4.10, even for the complicated cases, junction and end points are extracted correctly.

Figure 4.10: Junction and end points of some example sub-patterns.

[2] http://www.csse.uwa.edu.au/~pk/research/matlabfns/

Then, the graph representation is obtained from the extracted points. Note that, graphs are undirected. We also prefer to keep them non-weighted to obtain scale invariance. Figure 4.11 shows the graph of a sub-pattern with the corresponding matrix representation.



	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	0	1	0	0
C	1	0	0	1	1	0
D	0	1	1	0	0	0
E	0	0	1	0	0	1
F	0	0	0	0	1	0

(a) (b) (c)

Figure 4.11: (a) An example sub-pattern (b) the sub-pattern's graph (c) matrix that represents the undirected, non-weighted graph

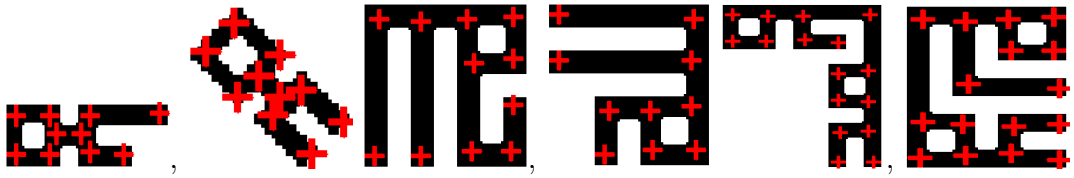
For graph matching, we first apply graph-isomorphism [75]. Two graphs are said to be isomorphic if their nodes can be one-to-one mapped with ensuring the adjacency of nodes. Given two graphs G_1 and G_2 there exists a function f such that:

$$\forall a, b \in V_1, (a, b) \in E_1 \Leftrightarrow (f(a), f(b))) \in E_2, \tag{4.1}$$

where V_1 is the vertex set of G1 and E_1,E_2 are the edge sets of G1 and G2 respectively [75].

The worst case of the algorithm is $O(n!)$, n being the number of nodes of G_1 or G_2 where they should be equal for the graphs to be isomorphic.

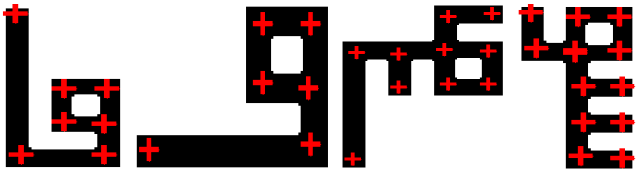
Figure 4.12 shows some different graph representations which are isomorphic. As it can be observed from the examples (such as Figure 4.12 (c)), graph matching is rotation and scale invariant.



(a) (b) (c)

Figure 4.12: Example sub-patterns with their graph representations, the graph pairs are isomorphic. (a) La sub-pattern (b) Lillah sub-pattern (c) Muhammed sub-pattern

Although graph isomorphism has many advantages, there are also some bottlenecks. Full graph isomorphism is based on a strict condition, where the two graphs should have the same number of nodes and there should be a one-to-one mapping between them. For our problem, the components which have the same number of limbs can be matched easily using isomorphism although they can differ in shape, scale or rotation. However, there are also sub-patterns with the same meaning but they differ in few limbs. This kind of sub-patterns cannot be matched with full isomorphism (see Figure 4.13).



(a) (b) (c) (d)

Figure 4.13: Example sub-patterns with their graph representations. Although the pairs are same sub-patterns their graphs are not isomorphic. (a) and (b) Leh sub-pattern (c) and (d) Su sub-pattern

To solve this problem, a partial graph matching approach is crucial. Thus, we applied a sub-graph isomorphism based approach. Although sub-graph isomorphism is NP-complete, it can be solved in polynomial time for certain cases such as when graphs are planar [76] as in our case.

In the sub-graph isomorphism problem, given two graphs G_1 and G_2, one must either detect an occurrence of G_1 as a sub-graph of G_2, or vice versa. For any two planar graphs, with n and m vertices, the decision problem can be solved in polynomial time $O(n^m)$ [76].

Directly applying sub-graph isomorphism to match the Kufic patterns rises some problems. Different sub-patterns can be matched due to one of them being part of another although they do not have the same meaning. To illustrate, the Figure 4.13 (a) and (c) are going to be sub-graph isomorphic although they are different sub-patterns. To eliminate this situation, we did not directly applied sub-graph isomorphism to match sub-patterns. We computed the difference between the numbers of nodes of the two graphs and checked if the difference exceeds a pre-defined threshold value. If the difference is smaller we applied sub-graph isomorphism else we said that the graphs are not isomorphic. With this kind of an approach Figure 4.13 (c) and (d) are going to be a true match and Figure (a) and (c) are not going to be matched.

Chapter 5

Kufic Pattern Matching Experiments

5.1 Dataset Description

As there is no existing Kufic dataset, we have constructed our own dataset by collecting images from the Internet (mostly from [1]) and from a book on calligraphy [7]. Some square Kufic images from our dataset can be seen In Figure 5.1. In total, there are 218 Kufic images in the dataset.

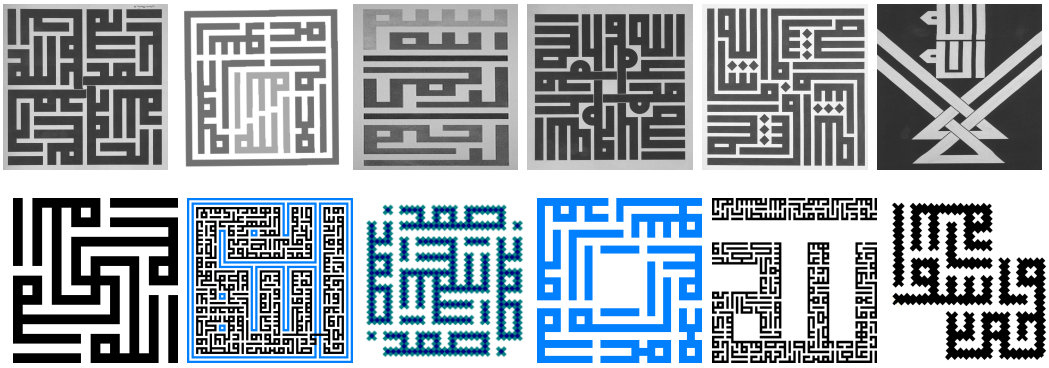


Figure 5.1: Some example square Kufic images. On the first row, the 4th image from the left has four *Allah* and *Muhammed* patterns, while the 5th image has four *Masaallah* patterns. Note that in the second row, the 2nd and 5th images have very small sub-patterns and their outer contours also form *Allah* patterns. The 3rd and 6th images in the second row have patterns which have some zig-zags on the contours, which make line extraction process difficult. On the first row, the 1st,4th,5th and 6th images are from [7], the 2nd image is from [6], and the 3rd image is from [8]. The second row images are from [1].

5.2 Other Approaches

To give comparative results, we examined previous methods that match patterns in Kufic Images. As a baseline system we experimented with profile based features [77] and we employed a second method which uses descriptors extracted from contours and exploits sequence matching [9]. In the following, we describe the details of each method.

5.2.1 Profile based features with DTW matching

Due to the difficulties in character segmentation, recently word spotting techniques have been proposed to match words as a whole. In word spotting, profile based features have been commonly used [78]. Since the two problems resemble to each other, profile based features from [77] are used as a baseline for comparison.

The baseline system utilizes the features used in [78]. Namely, the projection profile (that is the count of foreground pixels for each horizontal coordinate value), upper and lower profiles (which are similar to projection profile but they consider only the pixels above and below the figure baseline) and lastly the ink transitions (the number of foreground-background ink transitions) are used.

Profile features are compared using Dynamic Time Warping (DTW). Without normalization, DTW algorithm may favor the shorter signals. In the literature a post-normalization is performed after calculating the distance. In this study [77], signals are normalized before inputting them into the algorithm.

Since profile based features are not rotation invariant, registration is performed by rotating the query sub-patterns in 45 degrees, and the lowest dissimilarity value over all rotations is considered as the matching score.

5.2.2 Sequence matching based on contour representation

The method [9] uses a representation, based on lines to describe sub-patterns in square Kufic images. Firstly, contours are extracted from sub-patterns and points on these contours are approximated to lines using the Douglas-Peucker line approximation algorithm [79] as in [80] (see Figure 5.2).

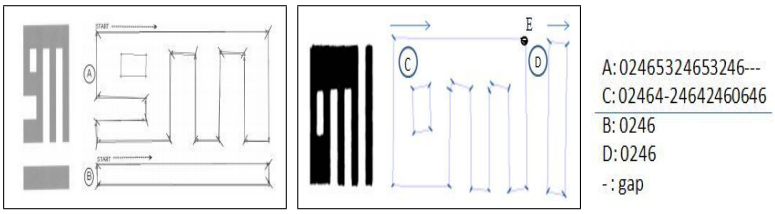
The Douglas-Peucker line approximation algorithm is a polygonal approximation method which is used for the description of the boundaries as a sequence of straight lines [81]. The Douglas-Peucker algorithm reduces the number of points in a curve by approximating it by a series of points. First, between a start and an end point, a sequence of points is approximated with a line segment. If the distance of the farthest point from the line is less than a threshold, the algorithm stops, otherwise it recursively divides the line into two from the farthest point [82]. The parameter τ used in the Douglas-Peucker algorithm can be defined as approximation accuracy, tolerance value, or compression factor. It serves for 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 a greater number of lines and smoother segments.

Can et al. [80] exploited Douglas-Peucker algorithm to describe words in handwritten documents as a set of lines. A line is described by its position, orientation and length as in [83]. They compute the matching score between two word images

as the sum of scores obtained from each matching line pair, normalized with the number of the matches and total number of lines in each word image. The lines with minimum dissimilarities are considered as the matching pairs.

In Kufic images, the composition of the lines in the sub-patterns are very important, while size and position of the lines may largely vary. Moreover, instances of a sub-pattern in different images may be approximated into different number of lines due to the variations in lighting conditions and sizes. Therefore, the method used in [80] is thought to be unfeasible for matching Kufic patterns [9].

As an alternative, in [9] they propose a new method for matching lines in sub-patterns based on chain code representation [84], by introducing a penalty for the gaps. Each Kufic sub-pattern 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 that sub-pattern. Then, using these lines as descriptors, for each extracted sub-pattern, an eight-connected chain code representation [85] is constructed. The proposed method is scale invariant because the length information of the lines is not used. In Figure 5.2, two sub-patterns and their chain code representations are given.



(a) (b) (c)

Figure 5.2: (a) and (b) Two *Allah* patterns in different shapes and the outputs of the line simplification process. Start-end points of lines are shown with small slashes. The chain code representation of sub-pattern A is 0246424642460646, B is 0246, C is 02465324653246 and D is 0246. (c) Output of the string matching algorithm for sub-patterns A-C and B-D. (The images are taken from [9])

Chain code representation depends on the start point. Circular movement algorithm, –where the start points are changed in a circular way, and the order in which chain codes form the possible smallest integer is taken–, is generally used

as a solution [85]. This method is thought to be unfeasible since the same sub-pattern may be approximated into different number of lines in different images, and the missing lines may result in incorrect matches by the choice of wrong starting point. Thus, in [9] it starts extracting chain codes at the upper left corner of each sub-pattern, but it rotates the sub-pattern by 45 degrees, and takes the match with the best score.

Chain code matching is performed utilizing a sequence matching algorithm [86]. Matching score of two chain code representations is calculated as follows:

$$D(I,J) = max \left\{ \begin{array}{l} D(I(i), J(j-1)) \\ D(I(i-1), J(j)) \\ D(I(i-1), J(j-1)) \end{array} \right\} + d(I(i), J(j)). \tag{5.1}$$

Here I and J are two chain code representations, and D is the score matrix; *I(i)* is the i_{th} element of chain code representation I (same for *J(j)*). $d(I(i), J(j))$ is the distance between *I(i)* and *J(j)*. It is 3 when *I(i)* and *J(j)* are the same, -2 when they are different and 1 when there is a gap. At the end of this step, the matching scores between each pair of sub-patterns are obtained and later used in experiments. A similar scoring approach is adapted as in [86], where aminoacid sequences are tried to be matched. However, when a chain code is matched a higher score is given, since a match is valuable, while a gap less than a mismatch is penalized, since there may be some additional lines in different instances of a same pattern because of drawbacks of dataset.

For example, in Figure 5.2, sub-patterns A and C are matched with score 20. At the end of this step, they have a matching score for each pair of sub-patterns and these scores are used in experiments.

5.3 Experiments

In the following, we will provide the experimental results for query retrieval, for indexing and for finding repeated patterns. In all experiments, True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) values

are obtained with respect to the parameters set (see Section 3.2). Throughout this section we will refer to the baseline method which matches profile based features as *profile*, the second comparative method which represents the lines extracted from contours as chain codes and utilizes sequence matching with penalties for gaps as *sequence matching*, and our own method which represents the sub-patterns as graphs and exploits graph isomorphism for matching as *graph matching*.

5.3.1 Query retrieval

We firstly perform experiments to find different instances of a query pattern in the entire collection. In this experiment, given a query pattern and a threshold, candidate patterns that have a matching score greater than this threshold are retrieved.

First, all of the 1398 labeled instances are used as the query sub-patterns. Recall that, we focus on four patterns, and use only discriminative sub-patterns to represent the patterns. *Muhammed* pattern has only one sub-pattern. Although *Allah* pattern has two sub-patterns, only the sub-pattern *lillah* is used as the discriminative sub-pattern, discarding *Alif* sub-pattern. For *Resul* pattern, *su* sub-pattern is the discriminative one. *La ilahe illa Allah* pattern, has *lillah*,and *leh* as discriminative sub-patterns, while again *Alif* sub-pattern being discarded. While *Allah*, *Resul* and *Muhammed* patterns can be retrieved through searching for a single sub-pattern, for *La ilahe illa allah* we count only the results containing all of the discriminative sub-patterns as correct.

Note that, sub-patterns in the query and dataset images are automatically extracted, and therefore the proposed approach can be applied to any pattern. The restriction for four query types in the experiments are due to the difficulty of labeling.

In Table 5.1, we compare the proposed method with *profile* and *sequence matching* on query retrieval task based on Area Under ROC Curve (AUC) and F1 scores. AUC value calculates the area between ROC curve and the x axis. F_1 metric is the harmonic mean of Precision and Recall values. As seen from the results our *graph matching* method outperforms both *profile* and *sequence matching* methods.

Feature	AUC	F1
Graph matching	0.85	0.79
Sequence matching	0.73	0.38
Profile	0.65	0.27

Table 5.1: Comparison of three methods on query retrieval based on Area Under ROC Curve (AUC) and F1 values.

In Figure 5.3, *sequence matching* and *graph matching* are compared based on their True Positive Rates (TPR) and False Positive Rates (FPR). Results show that graph matching method is better than sequence matching. In Table5.2, TPR and FPR values are given for each of the four patterns separately.

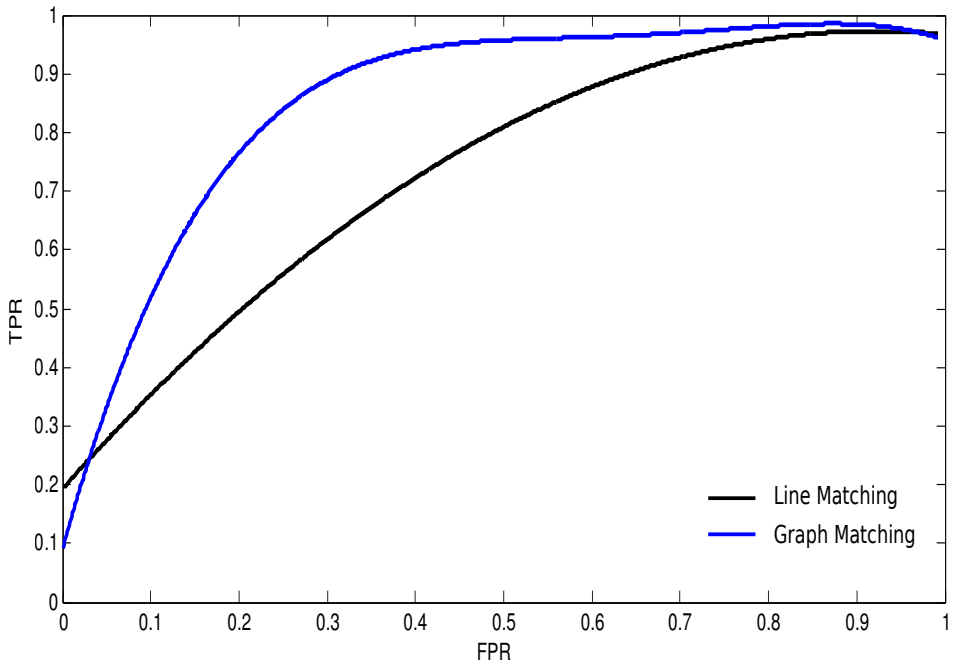


Figure 5.3: This Figure shows average TPR vs FPR results for all types of query patterns in dataset. Results show that sequence matching is good at finding instances of a pattern but it can not easily eliminate false matches, while graph matching can discriminate false matches.

	Sequence Matching		Graph Matching	
	TPR	**FPR**	**TPR**	**FPR**
Allah	0.5662	0.2066	0.9552	0.2933
Muhammed	0.5046	0.0429	0.4882	0.1107
LIIA	0.2961	0.0205	0.9267	0.4535
Resul	0.3016	0.5367	0.9833	0.3215

Table 5.2: Recall and precision values of query retrieval task performed by two different approaches: sequence matching and Graph matching.

In sequence matching method, the lowest score is retrieved with *La ilaha illa Allah* pattern due the number of sub-patterns it has. *Resul* pattern also has a low score since it is formed by sub-pattern *su*, which has a large variety between its instances. Note that graph matching method is good at discrimination of different pattern models, while at the same time it can successfully retrieve different instances of the same pattern. One other reason that graph matching outperforms sequence matching is the connected sub-patterns problem. Connected sub-patterns problem occur when more than one instance of a sub-pattern is connected to each other and they are extracted as only one sub-patterns (see Figure 5.4). In chain code representation, these connected sub-patterns and query sub-pattern have different representations and their sequence matching dissimilarity is large. We could also perform local matching in our sequence matching method, but in that case number of false matches would be much higher.

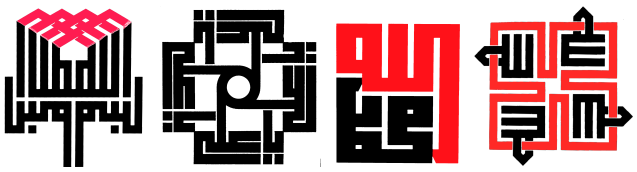


Figure 5.4: Connected pattern examples that our sequence matching method can not detect (The images are taken from [7]).

In graph matching method, graphs of connected patterns can be partially matched to query graphs with sub-graph isomorphism. To test our theory of graph matching detecting connected patterns more accurately, we performed a small test where we generated *Lillah* and *Muhammed* queries and searched them in images that contain the same patterns but in a connected form. Also, to understand the effect of sub-graph isomorphism we tried the same experiment with different threshold values explained in Section 4.4.3. The results are shown in Figure 5.5. When k is large enough sub-graph isomorphism is applied to every connected pattern instead of graph isomorphism and each query pattern is found inside the same pattern's connected graph.

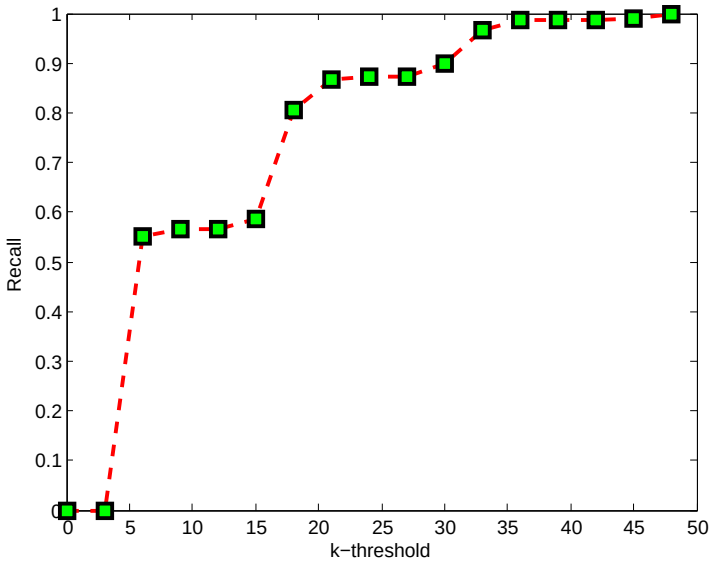


Figure 5.5: Connected pattern detection experiment results by graph matching.

Similarly, the ROC curve in Figure 5.3 shows that, as the threshold value increases true positive rate increases too. The reason is, the algorithm applies sub-graph isomorphism to more number of sub-patterns instead of graph isomorphism, which is a strict condition to obtain. Therefore, for high values of threshold, the number of matches increase which results in an increase for true positives and

false positives. As a result, the algorithm manages to find the true positives with full sub-graph isomorphism however, it fails for false positives.

The below Table 5.3 shows the 10-fold cross-validation results for subgraph-isomorphism applied on query retrieval task. The values show the accuracy computed for each run and for each pattern. Only the *Resul* pattern has unstable results but other patterns show stability in their detection rate. The overall accuracy is effected from the *Resul* pattern's accuracy. We believe that the low results of *Resul* comes from the fact that it has less number of examples in the dataset.

	Run1	**Run2**	**Run3**	**Run4**	**Run5**	**Run6**	**Run7**	**Run8**	**Run9**	**Run10**
Allah	0.9733	0.9467	0.9610	0.9481	0.9625	0.9853	0.9714	0.9851	0.9726	0.9677
Muhammed	1.0000	1.0000	0.9412	1.0000	1.0000	0.9130	0.9000	1.0000	1.0000	0.9200
LIIA	0.6578	0.6489	0.6537	0.6494	0.6542	0.6618	0.6571	0.6617	0.6575	0.6559
Resul	1.0000	0.3333	0.3333	0	0	0.3333	0	0.3333	0	0
Mean	0.9078	0.7322	0.7223	0.6494	0.6542	0.7233	0.6321	0.7450	0.6575	0.8859

Table 5.3: 10-fold cross validation, graph isomorphism accuracy results for query retrieval.

5.3.2 Image indexing

In another experiment, we relaxed the matching criteria, and when any instance in an image with the query sub-pattern is retrieved we assumed that the image is correctly indexed. This experiments is performed to show that the proposed approach could be used in indexing the images without localizing the patterns. Table5.4 shows TPR and FPR values for each of the four patterns separately.

	Sequence Matching		**Graph Matching**	
	TPR	**FPR**	**TPR**	**FPR**
Allah	0.5188	0.0762	0.9802	0.2855
Muhammed	0.8727	0.2240	0.5383	0.1065
LIIA	0.5875	0.2702	0.9053	0.4231
Resul	0.3728	0.3650	0.9734	0.2965

Table 5.4: Image categorization success rates with line and graph matching methods. Graph matching method again outperforms sequence matching method.

5.3.3 Repeating pattern detection

In the last experiment, we automatically detect repeating sub-patterns in a given image without using a query pattern. Any sub-pattern that exists at least twice in a Kufic image is accepted as a repeating sub-pattern. For example, in Figure 5.6, the image on the left has two repeating patterns and the others have more, because they are symmetrical.

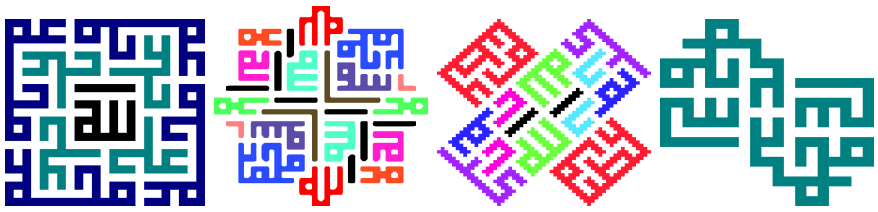


Figure 5.6: Repeating pattern examples (The images are taken from [1]).

Given a candidate image, all sub-patterns' in an image are assumed to be queries and searched in the same image. When the similarities are above some predefined threshold, then they are considered as repeating patterns.

This experiment is performed on a subset of our dataset, which has images having at least one of our four patterns (since other patterns are not labeled in our dataset). In Table 5.5, True Positive Rates (TPR) and False Positive Rates (FPR) are given for each category.

	Sequence Matching		**Graph Matching**	
	TPR	**FPR**	**TPR**	**FPR**
Allah	0.8223	0.2367	0.9113	0.0542
Muhammed	0.9125	0.2696	0.8069	0.0620
Resul	0.5714	0.0238	0.9813	0.0336

Table 5.5: Repeating pattern detection by sequence matching and graph matching methods. We didn't provide results for *La ilaha illa Allah*, because at most only one instance of that pattern in images, which makes it non-repeating pattern.

Repeating sub-patterns with different shapes in the same image can not be retrieved. For example, returning to Figure 4.6, in the second image from the left in the first row contains three *Allah* patterns (in gray), but as their shapes are different from each other, they can not be detected as repeating patterns.

The advantage of detecting repeating sub-patterns is that we can automatically find possible words in a given Kufic image without the usage of a query pattern. In this way, the meaningful patterns can be deciphered in these calligraphic images and a fully automatic indexing schema can be developed.

5.4 Discussion

In this study, we present a shape-based analysis of Kufic calligraphy images for indexing and retrieval of these image collections. The proposed method is based on graph representation and patterns are matched by a graph matching algorithm, also a detailed feature analysis is provided. We show that our graph matching algorithm gives promising results with matching Islamic patterns in Kufic images.

Although our method works well for most of the queries, querying less-common shapes is not as successful. Also, because two different letters may share the same shape in a Kufic design, precision rates in the experiments are low. Our method can not retrieve instances of a query pattern when the patterns are created in different shapes as in Figure 5.7.

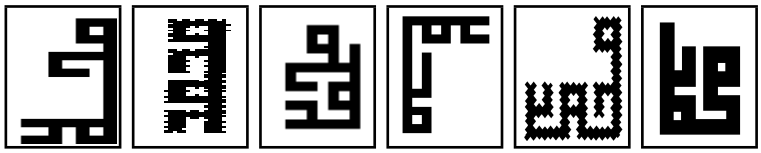


Figure 5.7: *Muhammed* patterns in different formats that our proposed method can not match.

Chapter 6

Conclusion

In the first part of this thesis, segmentation of Ottoman documents is studied. First layout segmentation which aims to detect regions consisting of text lines written with different orientations is explored. A Log-Gabor filtering based approach is used where maximum Gabor response per cell is computed for segmentation. Experiments are done on an Ottoman dataset constructed with documents which has multi-oriented lines. An accuracy rate of 96.01% is computed from the results. Wrong labeled connected components are discovered and different types of errors are observed. It is concluded that using different number of scales or orientations might solve some problems. However, it is argued that using less number of scales might bring up different issues such as loosing scale independency. Another solution suggested to eliminate problems is to use language based rules which requires language experts.

Second segmentation task is line segmentation where four different approaches are proposed. First one is a traditional line segmentation method, called projection profile. Second line segmentation algorithm is an hybrid approach based on both connected components and vertical projection profile. Projection profile based methods are simple, easy to implement and can deal with a certain amount of curve. Besides, connected component based approaches are successful for more complicated documents whose interline distances vary or baseline skews are inconsistent. Thus, by extracting baseline pixels from connected components and then using the projection profile information the algorithm manages to segment the lines from both handwritten and printed Ottoman documents.

The last 2 line segmentation methods are based on Log-Gabor filtering like the layout segmentation and they support script independency. First one uses region intersection while line fitting is preferred for the second one.

The 4 line segmentation algorithms are tested on a mixed dataset including English, Greek, Bangla and Ottoman documents to obtain comparative results on different languages. The results show that projection profile has the worst performance and Gabor filtering with region intersection has the best performance having similar results with line fitting. The effectiveness of the Gabor filtering with region intersection is demonstrated on different languages and it is shown that the algorithm is successful for different writing styles and writers. Also, the results showed that the algorithm is script-independent.

Last segmentation task is word segmentation where a simple morphological method is applied. Promising results are obtained on English, Greek and Bangla documents.

To increase the segmentation results, language based rules or supervised techniques can be applied. At the same time, we believe that using more advanced techniques for pre-processing steps such as binarization, diacritics detection and noisy component removal will improve the segmentation results.

Second part of this thesis focused on Islamic pattern matching on Kufic images. The approach involved four main steps: (i) foreground extraction, (ii) sub-pattern extraction, (iii) representation and matching, (iv) analysis. A new method, graph matching is proposed to match Islamic patterns in Kufic image collections. 3 different experiments are constructed. First experiment is to find different instances of a query pattern in the entire collection. In this experiment, given a query pattern and a threshold, candidate patterns that have a matching score greater than this threshold are retrieved. In the second experiment, the matching criteria is relaxed, and when any instance in an image with the query sub-pattern is retrieved it is assumed that the image is correctly indexed. The experiment showed that the proposed approach could be used in indexing the images without localizing the patterns. The last experiment is about automatically detecting repeating sub-patterns in a given image without using a query pattern.

The experiments showed that graph matching algorithms give promising results with matching Islamic patterns in square Kufic images. To increase the

success rates, the problem of connected sub-patterns can be solved. Using a sliding window approach that would detach patterns will make the matching easier. Besides, Kufic dataset which is constructed by images collected from the Internet can be extended in order to study different and longer words.

Bibliography

[1] "Kufic info," 2009. www.kufic.info/.

[2] "Coin image," As of 6 May 2013. http://en.wikipedia.org/wiki/File:Hafsids_Bougie_Algeria_1249_1276_ornemental_Kufic.JPG.

[3] "Tombstone image," As of 6 May 2013. http://www.smb.museum/roadsofarabia/index.php?id=17&L=1.

[4] "The art of arabic calligraphy," 1993. www.sakkal.com/ArtArabicCalligraphy.html.

[5] "Kufic," 2009. en.wikipedia.org/wiki/Kufic.

[6] "Kufic example 2," 2009. www.waterholes.com/~dennette/1995/islam/shahada.htm.

[7] S. Ozpalabiyiklar, *Bir Yazi Sevdalisi: Emin Barin*. Yapi Kredi, 2002.

[8] "Kufic example 1," 2009. www.farm4.static.flickr.com/3377/3318123762_ea07344f17.jpg?v=0.

[9] D. Arifoglu, "Historical Document Analysis Based On Word Matching," Master's thesis, Bilkent University, Turkey, 2011.

[10] E. Ataer and P. Duygulu, "Retrieval of ottoman documents," in *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pp. 155–162, 2006.

[11] A. Amin, "Off line arabic character recognition: a survey," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, vol. 2, pp. 596–599, IEEE, 1997.

[12] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 690–706, 1996.

[13] S. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical character recognition: a survey," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 01n02, pp. 1–24, 1991.

[14] S. Mori, C. Y. Suen, and K. Yamamoto, "Historical review of ocr research and development," *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1029–1058, 1992.

[15] N. Aouadi, "Word spotting for arabic handwritten historical document retrieval using generalized hough transform," *2011 The Third International Conferences on Pervasive Patterns and Applications*, vol. 1, no. c, pp. 67–71, 2011.

[16] A. Bhardwaj, S. Setlur, and V. Govindaraju, "Keyword spotting techniques for sanskrit documents," in *Sanskrit Computational Linguistics* (G. Huet, A. Kulkarni, and P. Scharf, eds.), vol. 5402 of *Lecture Notes in Computer Science*, pp. 403–416, Springer-Verlag, 2009.

[17] J. Lladós, P. Pratim-Roy, J. A. Rodríguez, and G. Sánchez, "Word spotting in archive documents using shape contexts," in *Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis, Part II*, (Berlin, Heidelberg), pp. 290–297, Springer-Verlag, 2007.

[18] "Ottoman text archive project (otap)," 2008. `http://courses.washington.edu/otap/`.

[19] R. Saabni and J. El-Sana, "Language-independent text lines extraction using seam carving," in *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 563–568, IEEE, 2011.

[20] X. Du, W. Pan, and T. D. Bui, "Text line segmentation in handwritten documents using mumford–shah model," *Pattern Recognition*, vol. 42, no. 12, pp. 3136–3145, 2009.

[21] A. Alaei, U. Pal, and P. Nagabhushan, "A new scheme for unconstrained handwritten text-line segmentation," *Pattern Recognition*, vol. 44, no. 4, pp. 917–928, 2011.

[22] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Script-independent handwritten textlines segmentation using active contours," in *10th International Conference on Document Analysis and Recognition*, pp. 446–450, IEEE, 2009.

[23] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "Script-independent text line segmentation in freestyle handwritten documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 8, pp. 1313–1329, 2008.

[24] Y. Lu, "Machine printed character segmentation: An overview," *Pattern Recognition*, vol. 28, no. 1, pp. 67–80, 1995.

[25] N. Tripathy and U. Pal, "Handwriting segmentation of unconstrained oriya text," in *Ninth International Workshop on Frontiers in Handwriting Recognition*, pp. 306–311, IEEE, 2004.

[26] S. Jaeger, G. Zhu, D. Doermann, K. Chen, and S. Sampat, "Doclib: a software library for document processing," in *Electronic Imaging 2006*, pp. 606709–606709, International Society for Optics and Photonics, 2006.

[27] A. Simon, J.-C. Pret, and A. P. Johnson, "A fast algorithm for bottom-up document layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 273–277, 1997.

[28] Y. Li, Y. Zheng, and D. Doermann, "Detecting text lines in handwritten documents," in *18th International Conference on Pattern Recognition*, vol. 2, pp. 1030–1033, IEEE, 2006.

[29] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line and word segmentation of handwritten documents," *Pattern Recognition*, vol. 42, no. 12, pp. 3169–3183, 2009.

[30] V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis, "Handwritten document image segmentation into text lines and words," *Pattern Recognition*, vol. 43, no. 1, pp. 369–377, 2010.

[31] E. Öztop, A. Y. Mülayim, V. Atalay, and F. Yarman-Vural, "Repulsive attractive network for baseline extraction on document images," *Signal Processing*, vol. 75, no. 1, pp. 1–10, 1999.

[32] P. P. Roy, U. Pal, J. Lladós, and F. Kimura, "Multi-oriented english text line extraction using background and foreground information," in *The Eighth IAPR International Workshop on Document Analysis Systems*, pp. 315–322, IEEE, 2008.

[33] U. Pal, S. Sinha, and B. B. Chaudhuri, "Multi-oriented text lines detection, their skew estimation.," in *ICVGIP*, 2002.

[34] N. Ouwayed, A. Belaïd, *et al.*, "Multi-oriented text line extraction from handwritten arabic documents," in *8th IAPR International Workshop on Document Analysis Systems-DAS'08*, pp. 339–346, 2008.

[35] J. Zhang, J. Gao, and M. Zhou, "Extraction of chinese compound words: an experimental study on a very large corpus," in *Proceedings of the second workshop on Chinese language processing: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 12*, pp. 132–139, Association for Computational Linguistics, 2000.

[36] J. H. Huang and D. Powers, "Chinese word segmentation based on contextual entropy," in *Proceedings of the 17th Asian Pacific conference on language, information and computation*, pp. 152–158, 2003.

[37] W.-Y. Ma and K.-J. Chen, "A bottom-up merging algorithm for chinese unknown word extraction," in *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, pp. 31–38, Association for Computational Linguistics, 2003.

[38] Y. Dai, T. E. Loh, and C. S. Khoo, "A new statistical formula for chinese text segmentation incorporating contextual information," in *Proceedings of*

the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pp. 82–89, ACM, 1999.

[39] A. Rosenfeld and A. C. Kak, *Digital picture processing*. Morgan Kaufmann, 1982.

[40] S. Bochner and K. Chandrasekharan, *Fourier Transforms.(Am-19)*, vol. 19. Princeton University Press, 1949.

[41] "Gabor filter," 2013. `http://en.wikipedia.org/wiki/Gabor_filter`.

[42] F. W. Campbell and J. Robson, "Application of fourier analysis to the visibility of gratings," *The Journal of Physiology*, vol. 197, no. 3, p. 551, 1968.

[43] J. R. Movellan, "Tutorial on gabor filters," *Open Source Document*, 2002.

[44] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision research*, vol. 20, no. 10, pp. 847–856, 1980.

[45] J. P. Jones and L. A. Palmer, "An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex," *Journal of Neurophysiology*, vol. 58, no. 6, pp. 1233–1258, 1987.

[46] D. J. Field *et al.*, "Relations between the statistics of natural images and the response properties of cortical cells," *J. Opt. Soc. Am. A*, vol. 4, no. 12, pp. 2379–2394, 1987.

[47] A. Zahour, L. Likforman-Sulem, W. Boussalaa, and B. Taconet, "Text line segmentation of historical arabic documents," in *Ninth International Conference on Document Analysis and Recognition*, vol. 1, pp. 138–142, IEEE, 2007.

[48] A. Zahour, B. Taconet, L. Likforman-Sulem, and W. Boussellaa, "Overlapping and multi-touching text-line segmentation by block covering analysis," *Pattern analysis and applications*, vol. 12, no. 4, pp. 335–351, 2009.

[49] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger, "Script-independent text line segmentation in freestyle handwritten documents," *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, vol. 30, no. 8, pp. 1313–1329, 2008.

[50] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2-4, pp. 123–138, 2007.

[51] "Bangla semantics," 2013. `http://banglasemantics.net/`.

[52] "Kuficpedia," 2009. `www.kuficpedia.com`.

[53] S. J. Abas, "Islamic geometrical patterns for the teaching of mathematics of symmetry," *Symmetry in ethnomathematics*, vol. 12, no. 1-2, pp. 53–65, 2001.

[54] "Maghribi kufic," 2009. `calligraphyqalam.com/styles/kufic-maghribi.html`.

[55] A. Amin, "Off Line Arabic Character Recognition - A Survey," in *Proceedings of the 4th International Conference on Document Analysis and Recognition*, (Washington, DC, USA), pp. 596–599, 1997.

[56] J. Chan, C. Ziftci, and D. Forsyth, "Searching off-line Arabic documents," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Washington, DC, USA), pp. 1455–1462, 2006.

[57] M. S. Khorsheed, "Off-line arabic character recognition: a review," *Pattern analysis & applications*, vol. 5, no. 1, pp. 31–45, 2002.

[58] A. Amin, "Segmentation of Printed Arabic Text," in *Proceedings of the Second International Conference on Advances in Pattern Recognition*, (London, UK), pp. 115–126, 2001.

[59] H. Moustapha and R. Krishnamurti, "Arabic calligraphy: A computational exploration," *Mathematics and Design*, pp. 294–306, 2001.

[60] C. Grana, D. Borghesani, and R. Cucchiara, "Picture extraction from digitized historical manuscripts," in *Proceeding of the ACM International Conference on Image and Video Retrieval*, (New York, NY, USA), pp. 1–8, 2009.

[61] J. Landre, F. Morain-Nicolier, and S. Ruan, "Ornamental letters image classification using local dissimilarity maps," in *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, (Washington, DC, USA), pp. 186–190, 2009.

[62] B. Zitova, J. Flusser, and F. Sroubek, "An application of image processing in the medieval mosaic conservation," *Pattern Anal. Appl.*, vol. 7, no. 1, pp. 18–25, 2004.

[63] E. Roman-Rangel, C. Pallan, J.-M. Odobez, and D. Gatica-Perez, "Analyzing ancient maya glyph collections with contextual shape descriptors," *Int. J. Comput. Vision*, vol. 94, pp. 101–117, August 2011.

[64] D. Dunham, "An algorithm to generate repeating hyperbolic patterns," in *Proceedings of ISAMA 2007*, pp. 111–118, 2007.

[65] C. S. Kaplan, "Computer generated islamic star patterns," in *Proc. Bridges 2000: Mathematical Connections in Art, Music and Science*, p. 4, 2000.

[66] C. S. Kaplan, *Computer graphics and geometric ornamental design.* PhD thesis, 2002.

[67] F. Albert, J. M. Gomis, and M. Valor, "Analysis and reconstruction of the tiling of Alcazar in Seville using computer vision tools," in *Proceedings of the 3rd International conference on Computer graphics and interactive techniques in Australasia and South East Asia*, (New York, NY, USA), pp. 127–130, 2005.

[68] A. M. Aljamali and E. Banissi, "Grid method classification of Islamic geometric patterns," *Geometric modeling: techniques, applications, systems and tools*, pp. 234–254, 2004.

[69] M. Djibril and R. Thami, "Islamic geometrical patterns indexing and classification using discrete symmetry groups," *Computing and Cultural Heritage*, 2008.

[70] V. Ostromoukhov, "Mathematical tools for computer-generated ornamental patterns," in *In Electronic Publishing, Artistic Imaging and Digital Typography. In Lecture Notes in Computer Science*, pp. 193–223, Springer-Verlag, 1998.

[71] M. Valor, F. Albert, J. M. Gomis, and M. Contero, "Textile and tile pattern design automatic cataloguing using detection of the plane symmetry group," *Computer Graphics International Conference*, vol. 0, p. 112, 2003.

[72] S. A. H. Minoofam and A. Bastanfard, "A novel algorithm for generating Mohammad pattern based on cellular automata," in *Proceedings of the 13th WSEAS International conference on Applied mathematics*, pp. 339–344, 2008.

[73] S. Suzuki and K. Abe, "Topological structural analysis of digitized binary images by border following," vol. 30, pp. 32–46, April 1985.

[74] "Intel opencv library." http://opencvlibrary.sourceforge.net/, August 2008.

[75] S. Fortin, "The graph isomorphism problem," tech. rep., MIT, 1996.

[76] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, (Philadelphia, PA, USA), pp. 632–640, Society for Industrial and Applied Mathematics, 1995.

[77] D. Arifoglu, E. Sahin, H.Adiguzel, P. Duygulu, and M. Kalpakli, "Matching islamic patterns in kufic images," *Pattern Analysis and Applications*, under review.

[78] 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*, pp. 218–223, 2003.

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

[80] E. Can and P. Duygulu, "A line-based representation for matching words in historical manuscripts," *Pattern Recognition Letters*, vol. 32, pp. 1126–1138, June 2011.

[81] P. K. Agarwal and K. R. Varadarajan, "Efficient algorithms for approximating polygonal chains," *Discrete and Computational Geometry*, vol. 23, pp. 273–291, 2000.

[82] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," tech. rep., School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 1997.

[83] 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.*, vol. 30, no. 1, pp. 36–51, 2008.

[84] G. Lu, "Chain code-based shape representation and similarity measure," in *Visual Information Systems*, (London, UK), pp. 135–150, Springer-Verlag, 1997.

[85] H. Freeman, "Computer processing of line-drawing images," *Computing Surveys*, pp. 6(1):57–97, March 1974.

[86] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins.," *Journal of molecular biology*, vol. 48, pp. 443–453, March 1970.

Appendix A

Layout Segmentation Results