Pattern Anal Applic (2015) 18:601–617
DOI 10.1007/s10044-014-0437-z

SHORT PAPER

Matching Islamic patterns in Kufic images

Damla Arifoglu · Emre Sahin · Hande Adiguzel · Pinar Duygulu · Mehmet Kalpakli

Received: 2 October 2011 / Accepted: 16 November 2014 / Published online: 21 January 2015

Abstract In this study, we address the problem of matching patterns in Kufic calligraphy images. Being used as a decorative element, Kufic images have been designed in a way that makes it difficult to be read by non-experts. Therefore, available methods for handwriting recognition are not easily applicable to the recognition of Kufic patterns. In this study, we propose two new methods for Kufic pattern matching. The first method approximates the contours of connected components into lines and then utilizes chain code representation. Sequence matching techniques with a penalty for gaps are exploited for handling the variations between different instances of sub-patterns. In the second method, skeletons of connected components are represented as a graph where junction and end points are considered as nodes. Graph isomorphism techniques are then relaxed for partial graph matching. Methods are evaluated over a collection of 270 square Kufic images with 8,941 sub-patterns. Experimental results indicate that, besides retrieval and indexing of known patterns, our method also allows the discovery of new patterns.

D. Arifoglu (✉) · E. Sahin · H. Adiguzel · P. Duygulu
Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey
e-mail: damla.arifoglu@gmail.com

E. Sahin
e-mail: i.emre.sahin@gmail.com

H. Adiguzel
e-mail: handee.a@gmail.com

P. Duygulu
e-mail: pinar.duygulu@gmail.com

M. Kalpakli
Department of History, Bilkent University, Ankara 06800, Turkey
e-mail: kalpakli@bilkent.edu.tr

Keywords Cultural heritage · Calligraphy · Kufic · Sequence matching · Graph matching

1 Introduction

Decorative inscriptions have been used both in Western and Eastern cultures throughout the history. As depicting humans in art is discouraged in Islam, geometric patterns and calligraphy have been the main form of artistic expression in Islamic cultures.[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. It has been mainly used as a decorative element in manuscripts, pottery, coins, architecture, stone inscriptions, and wooden work for several centuries [1–4], with the proverbs and passages from the Qur'an being 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.

Among the three Kufic scripts-written, Ma'qeli Kufic, also known as square Kufic, has been the most common Kufic types used in decoration and encountered in many different forms at historical buildings [5] (see Fig. 1).

Analysis of Kufic scripts may shed light to a relatively unknown era in history and therefore is of interest to scholars around the world. Moreover, for a regular person visiting a historical site, it is the taste of a treasure hunt to discover repeating words and phrases in complex Kufic patterns (see Fig. 2). However, even for a person whose

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

Fig. 1 Some decorative Kufic patterns on historical buildings. From *left to right*: **a** Gudi Khatun Mausoleum in Karabaghlar, Azerbaijan, 1335–1338 [5], **b** Al muayyad Mosque, Cairo, Egypt, 1415–1422 [6], **c** Sehzade Mosque, Istanbul, Turkey, 1543–1548 [7], **d** Isfahan Friday Mosque, Isfahan, Iran, 771 [5], **e** Royal Mosque, Isfahan, Iran, 1611 [8]

native language is Arabic, it is difficult to decipher Kufic scripts due to challenges inherent in Kufic calligraphy. Furthermore, there may be very large number of different words hidden making it difficult to track even for an expert. An application, possibly on mobile phones, to recognize words in Kufic scripts would have important effect on passing this tremendous heritage to future generations.

Going beyond analysis of historical Kufic patterns, studying Kufic images is also interesting due to the recent trend of using Kufic images as decorative elements at today's houses and accessories with its modern and geometric look as a design pattern [13–17] (see Fig. 2). Recent workshops organized to teach Kufic are getting attendance from different cultures and backgrounds.[2] Cataloging these increasing number of new designs, as well as their counterparts in historical buildings and manuscripts, is important but difficult to do with the available systems.

In this study, we attack the challenging problem of automatic identification of Kufic scripts which has not been previously tackled based on our knowledge. 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 of Kufic images may help in understanding of their characteristics and design rules, and lead to the automatic generation of new designs [13–17].

In the following, first we describe the challenges of studying Kufic images and review related studies on the automatic analysis of other calligraphy images. Then, we describe our approach toward the matching of sub-patterns in Kufic images and report the results of experiments. Finally, we discuss the results and the future work.

[2] http://blog.29lt.com/2007/09/01/kufi-workshop-for-non-arab-participants-in-amsterdam/; http://blog.29lt.com/2012/10/09/nuqat-conference-kuwait-2012/.

2 Challenges in Kufic patterns

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 [18–20]. Most characters are only distinguished by the attached dots or zigzags called diacritics. Moreover, because of the consonantal nature of Arabic, vowels are omitted [21].

Letters in square Kufic are in the form of a square or a rectangle (see Fig. 3). Geometric shapes consisting of various straight lines [2, 4, 5] can be elongated by 45° or 90° angles to compose different motifs [1, 5, 13, 22, 23]. These style characteristics of Kufic calligraphy bring additional challenges over regular Arabic text [5, 13]. 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 (see Fig. 4). There is also very little distinction between the shapes of different letters.

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 result 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 [24]. 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.

Last but not least, many different forms of square Kufic patterns may be mixed on a single image (see Figs. 5, 6).

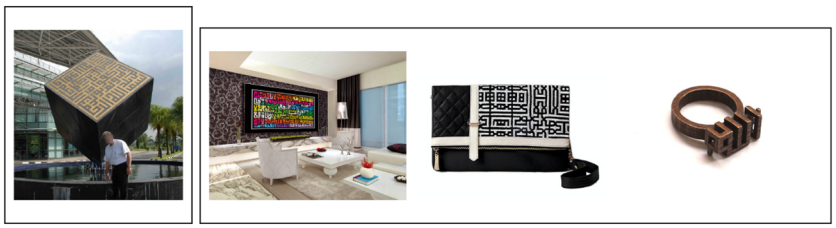


Fig. 2 *Left* An example photograph. A mobile application developed to process these photos and to decipher Kufic patterns would be beneficial for the visitors. (Image is taken from [9]). *Right* Some recent Kufic designs to be used in daily modern life (Images are taken from [10–12])

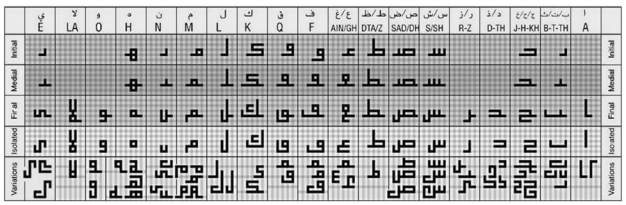


Fig. 3 Square Kufic script letters. Note that, due to the nature of Arabic, the same character may have different shapes, depending on its position within the sentence. Characters may also have different shapes in different designs. (Image courtesy of [13])

Fig. 4 The same sub-word may be in different shapes and different sub-words may be in similar shapes. In the first two images, sub-patterns *lillah* (*gray*), *la* (*light gray*) have different shapes in both designs. On the last image, three different sub-words, shown in *different shades of gray*, share similar shapes

Although the patterns are simple themselves, combination with large number of other (clutter) patterns makes their recognition very difficult. 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 hand-writing recognition.

3 Related work

In recent years, accessing and preserving cultural heritage has been considered in many different ways [26–28]. Computer vision techniques have been proposed for automatic indexing and retrieval of historical collections [29]. Here, we focus on studies in indexing and generation of Arabic calligraphy [3, 24, 30–37].

Dunham et al. [33] 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

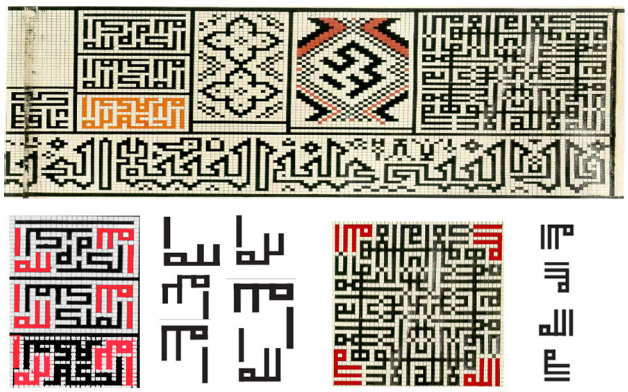


Fig. 5 *Top* Kufic calligraphic panels, Topkapi Palace, Istanbul, Turkey, Mid-15th Century [25]. *Bottom Allah* patterns extracted using our method

tilings by a polygon which is not necessarily regular and that polygon is assumed to be convex. In [34, 35], a method based on radially symmetric motifs is proposed to generate Islamic star patterns .

In [31], 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 [32], 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 [30], comprehensive analysis and cataloging of Islamic design patterns from digital images are 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 [24], Interactive Calligraphy Exploration prototype is described to compose calligraphic images by manipulating symmetries to produce unusual visual effects. This study

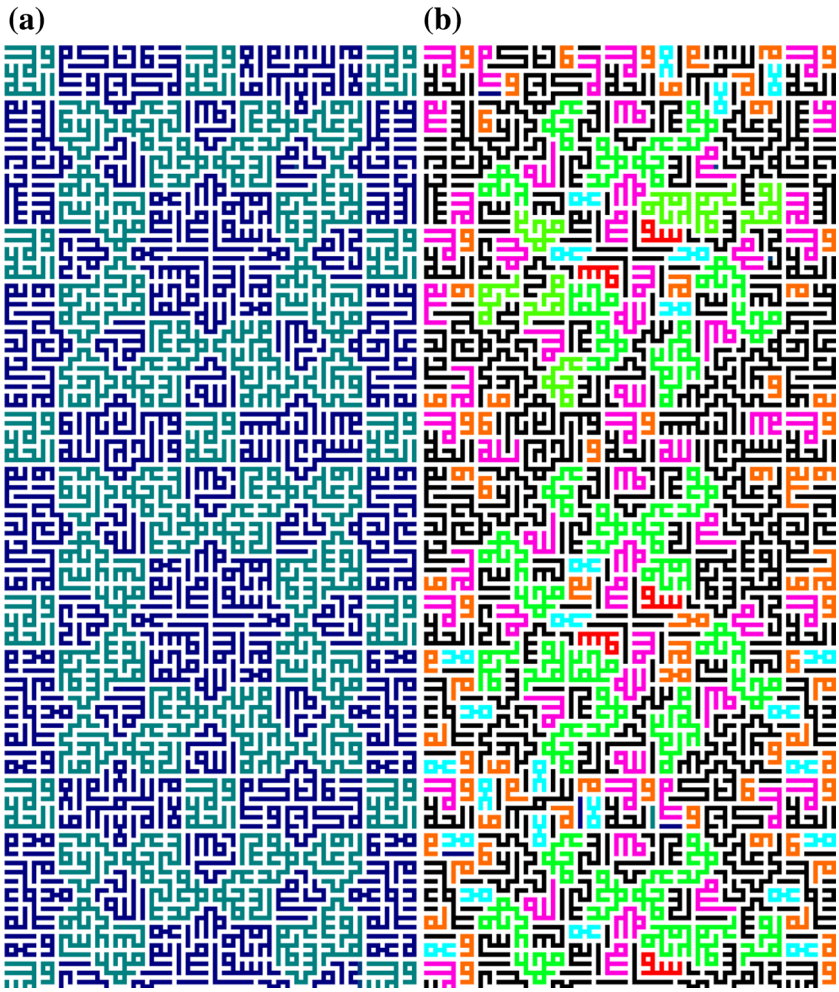


Fig. 6 Individual patterns are simple yet their use in decoration is hard to recognize. *Left* Original images, *Right* Our detections

introduces 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 [38]. In [38], 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, while our method is not specific to any pattern. An internationally recognized Arabic calligrapher Mamoun Sakkal has demonstrated the principles of creating Kufic script for new designs [13–17].

To the best of our knowledge, Kufic pattern matching has not been studied yet.

4 Our approach

For identification of Kufic patterns, we propose a retrieval-based approach. An automatic morphological segmentation is performed to break down Kufic words or sentences into their minimal grammatical constituents. Segmented words are then used to index and decipher of the entire text of a Kufic design, which enable the patterns to be cataloged, and thus browsed. As shown in Fig. 7, our approach involves four steps: (i) foreground extraction, (ii) sub-pattern extraction, (iii) representation and matching, and (iv) analysis. 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.

4.1 Dataset generation

Collecting the dataset images: As there is no existing Kufic dataset to be used for automatic analysis, we have constructed our own dataset by collecting all the images that we have found on the Internet and from a book on calligraphy [39]. Some square Kufic images from our dataset can be seen in Fig. 8. In total, there are 270 Kufic images in the dataset. Although there are an increasing number of

available images on the web that include square Kufic patterns, some of the pictures are taken in very bad conditions with extreme perspective deformations or occlusions, and there are many duplicates. As will be discussed, as a future work, we plan to handle these problems but this is out of scope for this study. Here, given a Kufic image with orthogonal lines our goal is to identify the patterns as one of the known instances. Inclusion of the new examples to the dataset is straightforward, and since we follow a retrieval-based approach, it could be done for increasing numbers.

Extraction of foreground pixels As it can be seen in Fig. 9, due to the different origins of the images, the dataset includes examples in a large range of varieties, from multi-color ornamental images to degraded and noisy images.

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,

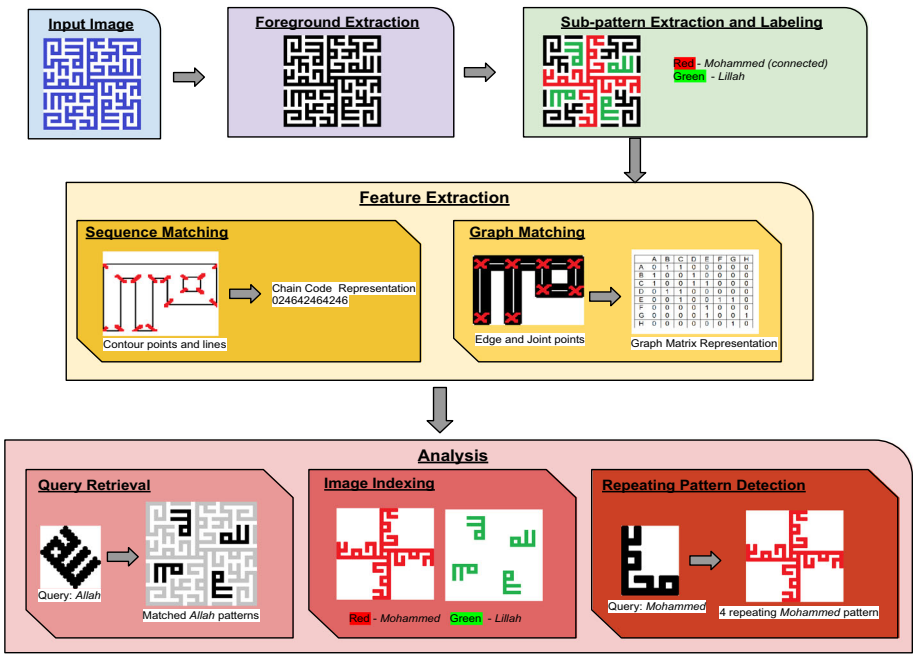


Fig. 7 The overall organization of our system

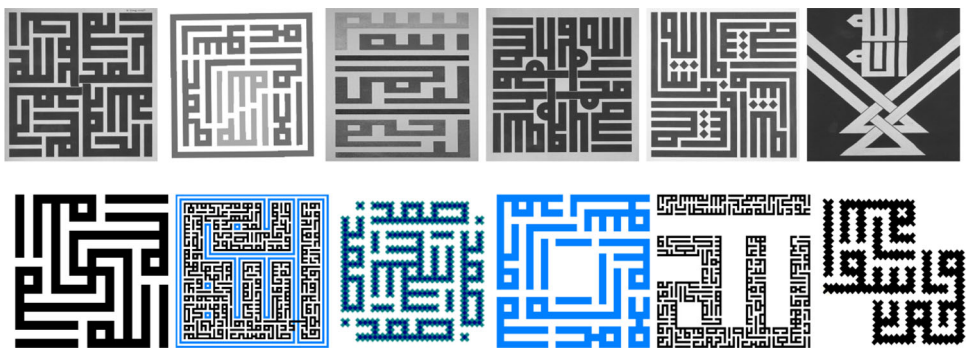


Fig. 8 Some example square Kufic images. On the *first row*, the 4th image from the *left* has four *Allah* and *Muhammed* patterns, while the 5th image has four *Mashaallah* patterns. Note that in the *second row*, the 2nd and 5th images have very small sub-patterns, and their outer contours also form *Allah* patterns. The 3rd and 6th images in the *second row* have patterns which have some zigzags on the contours, which make line extraction process difficult. On the *first row*, the 1st, 4th, 5th, and 6th images are from [39], the 2nd image is from [40], and the 3rd image is from [41]. The *second row* images are from [5]

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. The assumption 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 [42] for binarization without further processing.

Extraction and labeling of sub-patterns 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 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 10 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 Fig. 12). *Resul* pattern is formed by three sub-patterns as it can be seen in Figs. 11 and 12. The longest pattern type in dataset is *La ilaha illa Allah* pattern (see Fig. 12), which is composed of seven sub-patterns: one *lillah*, three *Alifs*, one *leh*, two *la*.

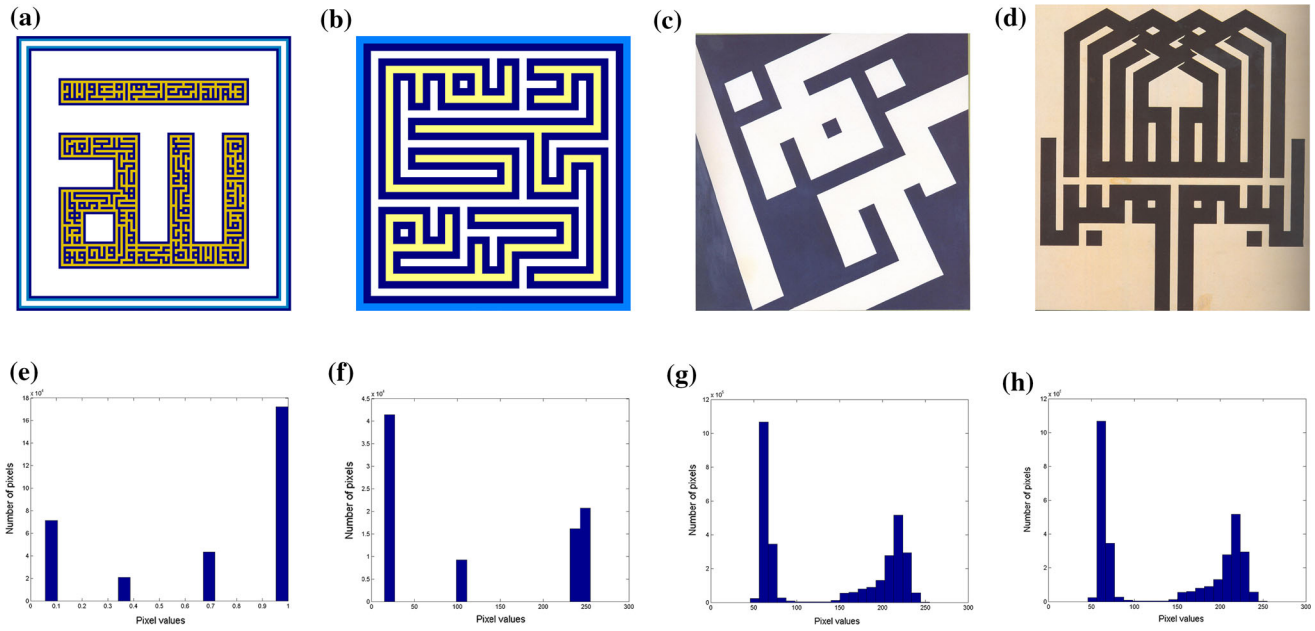


Fig. 9 *Top* **a–d** Example Kufic images, *Bottom* **e–h** their corresponding color histograms. As it can be seen from the color histograms (**a**) and (**b**) have a few distinct colors, while the degradations in (**c**) and **(d)** result in multiple colors

In this study, we consider the sub-patterns as basic units and extracted connected components to be used as sub-patterns. First, all the connected components (CCs) from the binarized images are extracted [43] using OpenCV Library [44]. In total, there are 8,941 extracted CCs. We will refer to the CCs that are parts of the patterns 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 two methods for representing and matching sub-patterns. The first method exploits sequence matching techniques to match the contours of sub-patterns represented in the form of chain codes [45]. In the second 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.

We performed experiments with four different common pattern types: *Allah, Muhammed, LIIA, Resul*, and these patterns consist of different sub-patterns. For example, *Allah* pattern has two sub-patterns: (i) *Alif* (ii) *lillah*, and shape of sub-pattern *Alif* does not change over the dataset, while *lillah* sub-pattern has big variations. Additionally, letter/sub-pattern *Alif* is a common sub-pattern that many of patterns/words in Arabic may have this letter. Thus, we ignore these common, uninformative sub-patterns during our matching process and only focus on sub-patterns that are more important to identify patterns, and we name these sub-patterns as discriminative. Thus, rather than labeling all sub-patterns of these four patterns, we label only the discriminative sub-patterns in each pattern, which are *lillah* in *Allah* pattern, *Mohammed* pattern itself (since it has only one sub-pattern), *su* sub-pattern in *resul* and *lillah* and *leh*. And we perform experiments based on these 4 discriminative sub-pattern groups.

Table 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.

As shown in Fig. 13 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 varies between 14 pixels and 1,918 pixels; median height is 50 pixels. Width of the components varies between 8 and 1,840 pixels, median width is 47 pixels. Median aspect ratio is 1, 25 % of the components that are square, 35 % are landscape, and 40 % are portrait.

4.2 Sub-pattern matching

As described earlier, all the sub-patterns are extracted automatically both for query and for 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. 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.

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 two methods for representing and matching sub-patterns. The first method exploits sequence matching techniques to match the contours of sub-patterns represented in the form of chain codes [45]. In the second method, each sub-pattern is represented with a graph and then graph and sub-graph isomorphisms are applied to match the patterns.

Yalniz et al. [46, 47] also exploited graph matching and chain code matching methods in their study for retrieval of Ottoman documents, but there are some differences between our method and theirs. In [46], they said that after thinning, a refinement could also be performed to reduce the number of nodes, but this was not necessary in their case since they did not use a graph-matching algorithm, and in [47], character graphs are compared by extracting a

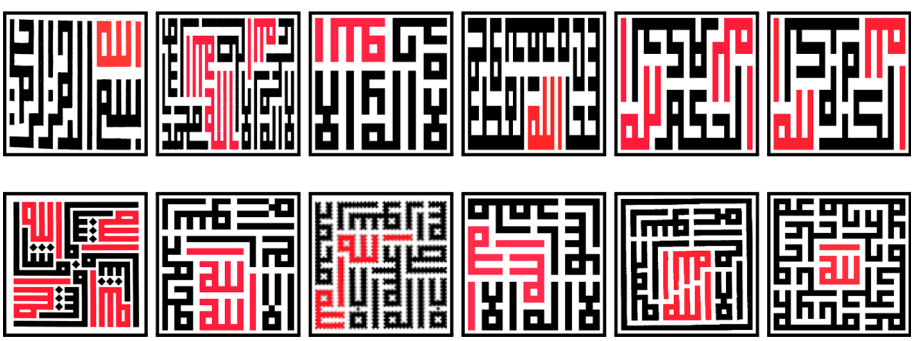


Fig. 10 Some square Kufic images with *Allah* patterns are shown in *red* (*gray*) . Note that this word has different shapes in different designs (Images 1–3 are taken from [13], 4–10 and 12 from [22]) (color figure online)

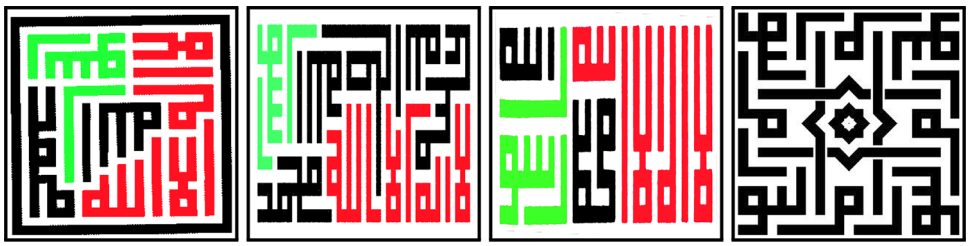


Fig. 11 The patterns in *green* (*light gray*) are *Resul* patterns, which are formed by three sub-patterns. The *red* (gray) sub-patterns are from the *La ilaha illa Allah* pattern. The last image contains four *Resul* patterns at each corner (First image is taken from [40] and second one is from [5] and third-fourth ones are from [39]) (color figure online)

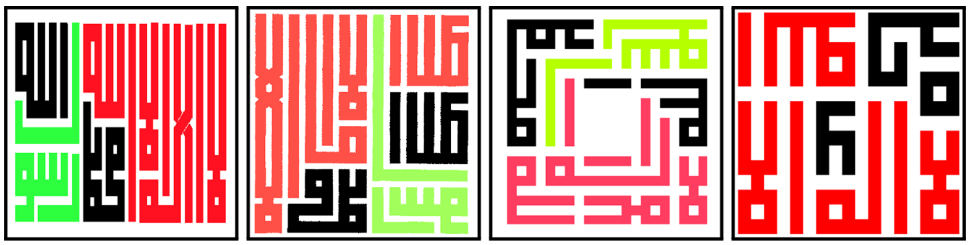


Fig. 12 The patterns in *red* (*gray*) are *La ilaha illa Allah*. The *green* (*light gray*) patterns are *Resul* patterns and note that the last one's two sub-patterns are connected. In the first image, first black pattern is *Allah*, while second black one is *Muhammed,* and in the second image, first 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 [39] and the third from [5]) (color figure online)

Table 1 Number of components per class and number of images where these patterns are found

Label	Number of components	Number of images	Example
la	80	26	
leh	170	45	
lillah	938	129	
muh	184	37	
su	26	15	
unlabeled	6,684	203	

Note that an image may contain more than a single labeled pattern

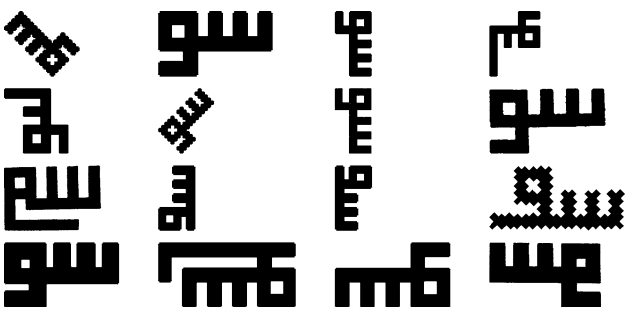


Fig. 13 16 sample images from *su* sub-pattern. This is the middle component of the word *Resul*

number of features from the graphs and calculating an overall similarity score by weighting the similarity scores between corresponding features of the graphs. On the other hand in our study, we match extracted graphs by isomorphism. Moreover, they compute distance and angular spans for all characters in the library, and for each extracted segment, the angular span vector is computed by obtaining the number of black pixels in theta-degree slices centered at the character's center of mass with respect to the x-axis. But in our case, we firstly extract lines from the contours on the boundaries, and then starting and end points of these lines are used to extract chain codes instead of image pixels.

We performed experiments with 4 different common pattern types: *Allah, Muhammed, LIIA, Resul* and these patterns consist of different sub-patterns. For example, *Allah* pattern has two sub-patterns: (i) *Alif* and (ii) *lillah*, and shape of sub-pattern *Alif* does not change over the dataset, while *lillah* sub-pattern has big variations. Additionally, letter/sub-pattern *Alif* is a common sub-pattern that many of patterns/words in Arabic may have this letter. Thus, we ignore these common, uninformative sub-patterns during our matching process and only focus on sub-patterns that are more important to identify patterns, and we name these sub-patterns as discriminative. Thus, rather than labeling all sub-patterns of these four patterns, we label

only the discriminative sub-patterns in each pattern, which are *lillah* in *Allah* pattern, *Mohammed* pattern itself (since it has only one sub-pattern), *su* sub-pattern in *resul* and *lillah* and *leh*. And we perform experiments based on these 4 discriminative sub-pattern groups.

In the following, we describe the details of each method.

4.2.1 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 [48]. Since the two problems resemble to each other, we use profile-based features as a baseline for comparison.

We utilize the features used in [49]. Namely, we used 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).

Profile features are compared using Dynamic Time Warping (DTW) [49]. Without normalization, DTW algorithm may favor the shorter signals. In the literature, a post-normalization is performed after calculating the distance. In our work, we normalized signals before inputting them into the algorithm.

Matching of patterns in different forms has been studied heavily in the literature, and several fantastic works are available for different problems. In this paper, our goal is to motivate the readers with an interesting application of pattern matching that could be very beneficial for cultural heritage. We propose and compare two methods that suit to the problem. Although we have experimented some other state-of-the-art methods, we have observed that they are not suitable for the problem at hand. Kufic styles have the characteristics of a word being written in many different orientations, with the letters, or components could be placed separately in different forms. The proposed methods are able to capture most of the challenging forms. We believe that, other methods or features may also help in improving the performance but may require hand tuning. Our goal was to keep the methods and features as simple as possible but as well to be generalizable to cope with the large variability of the styles. As a future work, we would like to exploit other representations as well as clustering of feature points, such as using k-means, normalized cuts, non-negative matrix factorization, etc. [50–52].

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

4.2.2 Approximate contour line matching

Contours are an important feature for shape matching and image retrieval as discussed in [53]. In [54], an application of approximate contour lines to text recognition is presented. In this section, we discuss contour line approximation as another baseline approach for Kufic recognition problem.

As it is vital to have robust features to compare handwriting images, contours undergo an approximation process to have length and location information. We employ Douglas–Peucker (DP) line approximation algorithm to obtain approximate contour lines from component contours.

DP algorithm is a polygonal approximation method which is used for the description of the boundaries as a sequence of straight lines [55, 56]. The 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 [57]. 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. [54] 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 [53]. 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 is 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 [54] is not feasible for matching Kufic patterns.

We made the tests for contour line approximations in two versions. In the first version, we used the comparison method in [54] as is. Each line segment is described by its length (l), angle (a), and position of midpoint relative to upper left corner of the component (r). In the first version, dissimilarity of two line segments is calculated using $d = d_l + 2d_a + 4d_r$ where $d_l = \log(|l_1 - l_2|)$, $d_a = |a_1 - a_2|$, and d_r is the Euclidean distance between r_1 and r_2.

In the second version, we omit the relative locational information (r) in these comparisons and use $d = d_l + 2d_a$ in order to avoid dissimilarity arising from location variation for line segments.

In both of these versions, components A and B are compared for their dissimilarity by total minimum dissimilarity score between each line segments. Let D be the total dissimilarity score between A and B. Since arg min is not symmetric, $D = D_{(A,B)} + D_{(B,A)}$ and $D_{(A,B)} = \sum_{a \in A} \arg\min(d(a,b))$ and $D_{(B,A)} = \sum_{binB} \arg\min(d(b,a))$. In other words, a comparison between A and B matches all contour line segments of A with the most similar line segment in B, then matches all contour line segments of B with the most similar line segment in A. After matching, it calculates the total dissimilarity score between these matches.

As the results Sect. 5 reflect, neither of these approaches were fruitful, as the comparisons ignore the order of the line segments. In Kufic texts, line segments are repeating patterns, and their context is important to determine their value in comparison. This context and neighborhood information are discarded in contour segment comparisons; therefore, contour line comparison is not a feasible strategy for Kufic recognition.

4.2.3 Sequence matching based on contour representation

First described by Freeman [58], chain codes are used to represent binary images efficiently. The representation stores the adjacency information of pixels by listing their relative positions in between each other.

In its standard form, chain codes are not size or rotation invariant. Therefore, they are not suitable for our purposes to match rotating Kufic components in the images. Instead of using the chain codes for pixels, we adapted a novel approach to extract chain codes from contour lines. Instead of pixel by pixel chain codes, we propose a representation based on lines to describe sub-patterns in square Kufic images. Since each line is represented with a single item in the code, this representation is size invariant.

Firstly, contours of a component are extracted, and contour points on these contours are approximated to lines using the Douglas–Peucker line approximation algorithm [56] similar to [54] (see Fig. 14).

We propose a method for matching lines in sub-patterns based on chain code representation [59], 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 [60] is constructed. In Fig. 14, two sub-patterns and their chain code representations are given.

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 [60]. In our case, this method is not feasible 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. Alternatively, we start extracting chain codes at the upper left corner of each sub-pattern, but we rotate the sub-pattern by 45○ and take the match with the best score.

Chain code matching is performed utilizing a sequence matching algorithm [45]. Matching score of two chain code representations is calculated as follows:

$$D(I,J) = max \begin{Bmatrix} D(I(i),J(j-1)) \\ D(I(i-1),J(j)) \\ D(I(i-1),J(j-1)) \end{Bmatrix} + d(I(i),J(j)) \tag{1}$$

Here, I and J are two chain code representations, and D is the score matrix; *I(i)* is the ith element of chain code representation textitI (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, we have a matching score for each pair of sub-patterns, and these scores will be used in experiments. In our study, we follow a similar scoring approach as in [45], where aminoacid sequences are tried to be matched. Here, we give a higher score when a chain code is matched, since a match is valuable, while we penalize a gap less than a mismatch, since there may be some additional lines in different instances of a same pattern because of drawbacks of dataset as we described in the previous section.

For example, in Fig. 14, sub-patterns A and C are matched with score 20. At the end of this step, we have a matching score for each pair of sub-patterns, and these scores will be used in experiments.

4.2.4 Graph matching-based method

As a second approach for matching Kufic sub-patterns, we propose a new method based on graph matching. In the following, first we describe how we represent sub-patterns as graphs, then we present the details of graph matching method.

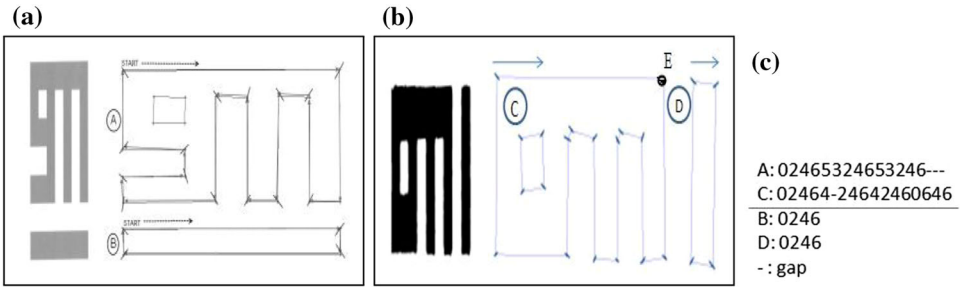


Fig. 14 **a**, **b** Two *Allah* patterns in different shapes and the outputs of the line simplification process. Start-end points of lines are shown with *small slashes*. The chain code representation of sub-pattern A is 0246424642460646, B is 0246, C is 02465324653246, and D is 0246. **c** Output of the string matching algorithm for sub-patterns *A–C* and *B–D*

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.[3]

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 that the distances between each junction point pairs and only the ones that exceeds a predefined threshold are kept. This threshold is set relative to the minimum of width and height of the connected component. As seen in Fig. 15, even for the complicated cases, junction and end points are extracted correctly.

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 16 shows the graph of a sub-pattern with the corresponding matrix representation.

For graph matching, we first apply graph-isomorphism [61]. 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

$$\forall a, b \in V_1, (a, b) \in E_1 \Leftrightarrow (f(a), f(b))) \in E_2 \qquad (2)$$

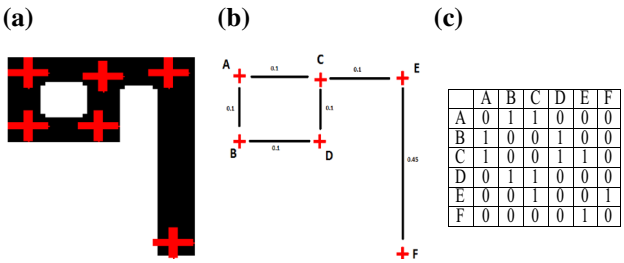
V_1 is the vertex set of G1, and E_1 and E_2 are the edge sets of G1 and G2 [61].

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 17 shows some different graph representations which are isomorphic. As it can be observed from the examples (such as Fig. 17c), graph matching is rotation and scale invariant.

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

Fig. 15 Junction and end points of some example sub-patterns



	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

Fig. 16 **a** An example sub-pattern **b** the sub-pattern's graph **c** matrix that represents the undirected, non-weighted graph

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 Fig. 18).

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 [62] as in our case.

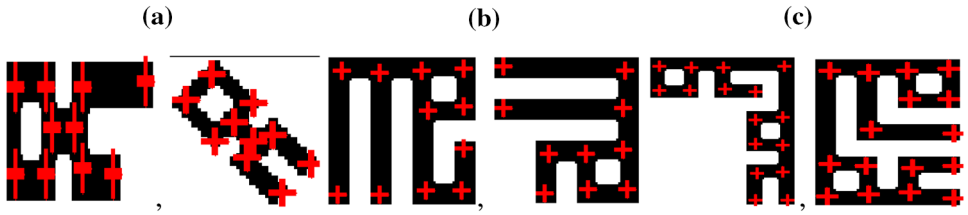


Fig. 17 Example sub-patterns with their graph representations, the graph pairs are isomorphic. **a** La sub-pattern **b** Lillah sub-pattern **c** Muhammed sub-pattern

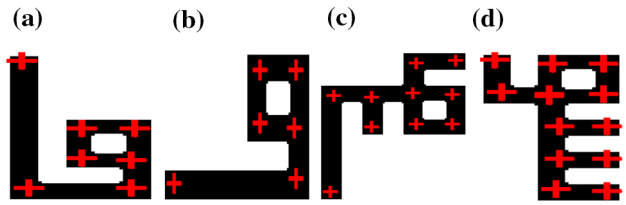


Fig. 18 Example sub-patterns with their graph representations. Although the pairs are same sub-patterns, their graphs are not isomorphic. **a**, **b** Leh sub-pattern **c**, **d** Su sub-pattern

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 $O(n^m)$ [62].

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, Fig. 18a, c is 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 predefined 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, Fig. 18c, d is going to be a true match, and Figure (a) and (c) is not going to be matched.

5 Experimental results

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. Throughout this section, we will refer to the first baseline method which matches profile-based features, described in Sect. 4.2.1 as *profile*, the second baseline method in which we compare approximate line segments, described in Sect. 4.2.2 as *contour line*, our first method which represents the lines extracted from contours as chain codes and utilizes sequence matching with penalties for gaps, described in Sect. 4.2.3 as *sequence matching*, and our second method which represents the sub-patterns as graphs and exploits graph isomorphism for matching, described in Sect. 4.2.4 as *graph matching*.

5.1 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 1486-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 (Fig. 19).

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 is due to the difficulty of labeling.

In Table 2, we compare the proposed methods with the baseline method 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, both of the proposed methods outperform the baseline method.

In Fig. 20, the two proposed methods, i.e., *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 method. In Table 3, TPR and FPR values are given for each of the four patterns separately.

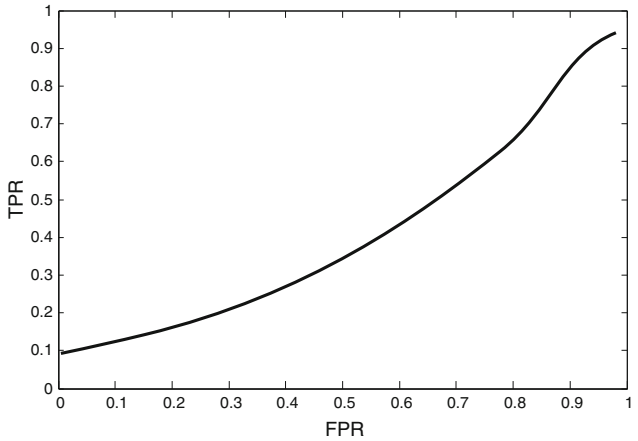


Fig. 19 This figure shows average TPR versus FPR results for all types of query patterns in dataset for approximate contour line feature. The feature is used as a baseline and has considerably lower performance

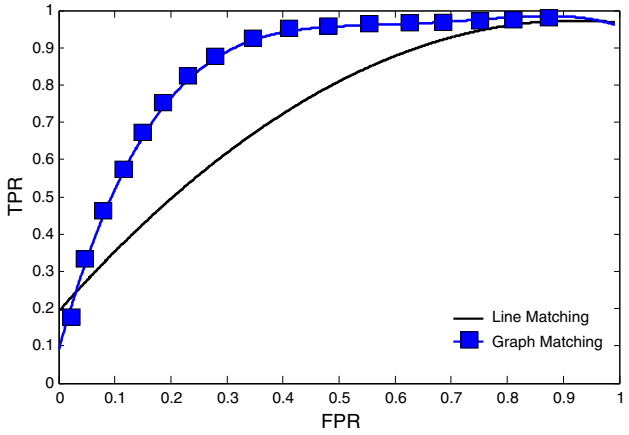


Fig. 20 This figure shows average TPR versus FPR results for all types of query patterns in dataset. Results show that sequence matching is good at finding instances of a pattern but it cannot easily eliminate false matchings, while graph matching can discriminate false matchings

Table 2 Comparison of three methods on query retrieval based on Area Under ROC curve (AUC) and F1 values

Feature	AUC	F1
Graph matching	0.83	0.77
Sequence matching	0.72	0.36
Profile	0.63	0.25
Contour line	0.35	0.13

Table 3 True positive rate (TPR) and false positive rate (FPR) values of query retrieval task performed by two different approaches: sequence matching and graph matching

	Sequence matching		Graph matching	
	TPR	FPR	TPR	FPR
Allah	0.5478	0.2133	0.9088	0.3072
Muhammed	0.4923	0.0945	0.4708	0.0978
LIIA	0.2843	0.0201	0.8563	0.5013
Resul	0.3016	0.52005	0.9833	0.3217

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 occurs when more than one instance of a sub-pattern is connected to each other and they are extracted as only one sub-patterns (see Fig. 25). 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 matchings would be much higher.

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 Sect. 4.2.4. The results are shown in Fig. 21. 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.

Similarly, the ROC curve in Fig. 20 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 increases, 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.

In Fig. 22, we can see that graph matching method is good at detecting patterns even if their shapes have some differences among different images.

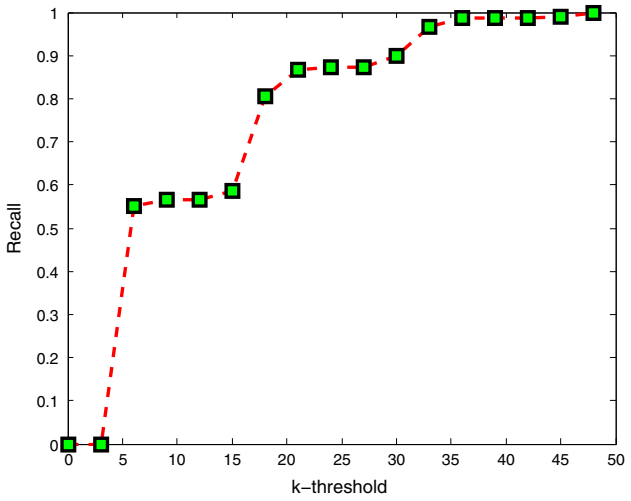


Fig. 21 Connected pattern detection experiment results done by graph matching

5.2 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. These experiments are performed to show that the proposed approach could be used in indexing the images without localizing the patterns. Table 4 shows ROC curve for the graph matching method.

5.3 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 square Kufic image is accepted as a repeating sub-pattern. For example, in Fig. 23, the image on the left has two repeating patterns and the others have more, because they are symmetrical.

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, True Positive Rates (TPR) and False Positive Rates (FPR) are given for each category.

Repeating sub-patterns with different shapes in the same image cannot be retrieved. For example, returning to Fig. 10, in the second image from the left in the first row contains three *Allah* patterns (in gray), but as their shapes

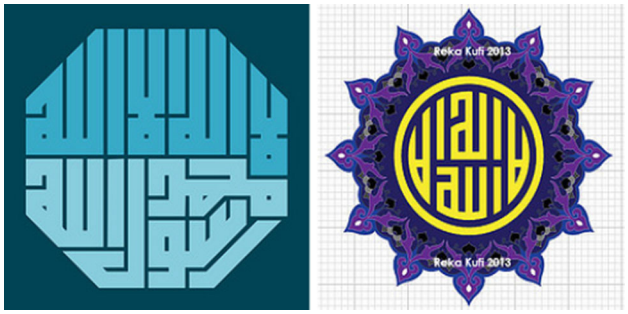


Fig. 22 Sequence matching method cannot match *Allah* patterns in none of the images since lines are not horizontal and their shapes are very different than most of other instances of this pattern. Because of their curved lines, their chain codes representations differ very much and thus, in sequence matching step, their mismatch penalties will be higher. Graph matching method can detect the patterns in the first image, while it cannot detect them in the second one. Because of the inclined lines, the detection of the junction points is going to be more challenging. The incomplete junction points might cause an error in the graph representation; thus, results will be badly affected. And profile-based method or contour line method cannot detect patterns in both images. (Images are taken from [63, 64])

Table 4 Image categorization success rates with line and graph matching methods

	Sequence matching		Graph matching	
	TPR	FPR	TPR	FPR
Allah	0.5012	0.0912	0.9745	0.3945
Muhammed	0.8449	0.2120	0.4782	0.2210
LIIA	0.5698	0.2619	0.8970	0.4765
Resul	0.3728	0.3650	0.9734	0.2965

Graph matching method again outperforms sequence matching method

are different from each other, they cannot be detected as repeating patterns.

The advantage of detecting repeating sub-patterns is that we can automatically find possible words in a given square 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 of results

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

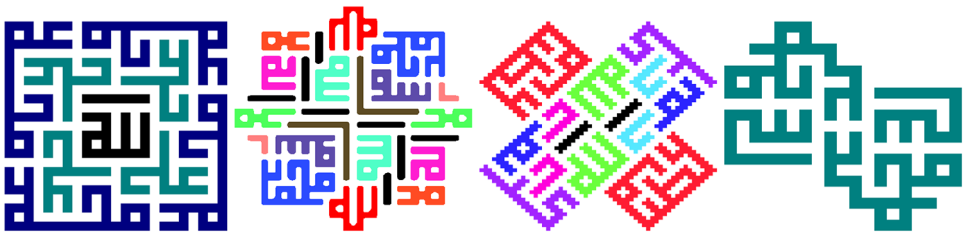


Fig. 23 Repeating pattern examples (The images are taken from [5])

Table 5 Repeating pattern detection by sequence matching and graph matching methods

	Sequence matching		Graph matching	
	TPR	FPR	TPR	FPR
Allah	0.8011	0.33786	0.9002	0.1264
Muhammed	0.9022	0.2598	0.7973	0.0627
Resul	0.57143	0.02381	0.9813	0.0336

We did not provide results for *La ilaha illa Allah*, because at most only one instance of that pattern in images, which makes it non-repeating pattern

representation and graph matching algorithms give promising results with matching Islamic patterns in square 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 square Kufic design, precision rates in the experiments are low. Our method cannot retrieve instances of a query pattern when the patterns are created in different shapes as in Fig. 24.

Connected patterns pose another problem because they have sub-patterns within them (see Fig. 25). The left most image has two *Allah* scripts that are connected at their upper parts for the purpose of decoration. The second image has 4 *Ali* scripts at each corner, and these patterns are connected with each other. The last image has four *Allah* scripts, each facing a different direction. The *Alif* letters are connected and forms the boundary of the image. Our line representation algorithm cannot detect the patterns in these images.

5.5 Application Scenario

As an important application of our work, we plan to recognize patterns in images taken from mobile phones or cameras possibly with a person in the image in front of a Kufic design. In this case, detection of areas for Kufic scripts would be necessary. We plan to use gradient histograms since the rapid and regular change in black/white pixels in images is a characteristic of Kufic designs. Since images in real conditions are taken in different perspectives, patterns will not be in perpendicular angles in such a case. Therefore, rectification is another important issue that we pan to handle as a future work.

An example to the mobile application can be seen in Fig. 26. In this case, it is assumed that a tourist takes a photo, and the software identifies the region of Kufic inscriptions via their highly descriptive gradients. After localizing the inscription, the image is binarized and components are recognized with the proposed method. The classification of components results in identification of the inscription.

Note that, detection of Kufic pattern areas is not the focus of this study, and therefore, we do not go further.

6 Conclusion and future work

Motivated by the importance of calligraphy in Islamic decorative art throughout the history and increasing interest in designing new patterns as observed through web search results, in this study, we attack the problem of automatic analysis of Kufic images. Alternative to character recognition-based approaches that are very difficult if not impossible to work on the Kufic scripts, we proposed two methods based on sequence matching and graph matching to identify Islamic patterns in Kufic image collections. Our experiments on identification of four common patterns show the effectiveness of the proposed methods on various type of Kufic images, and a relatively limited number of experiments on finding repeated patterns without the specification of the query pattern are promising in discovering the hidden patterns.

Inherent in the design of Kufic calligraphy, connected sub-patterns were the most important challenge and the limitation for the proposed method. As a future work, we are planning to solve this problem by a sliding window approach that would detach patterns so that matching them

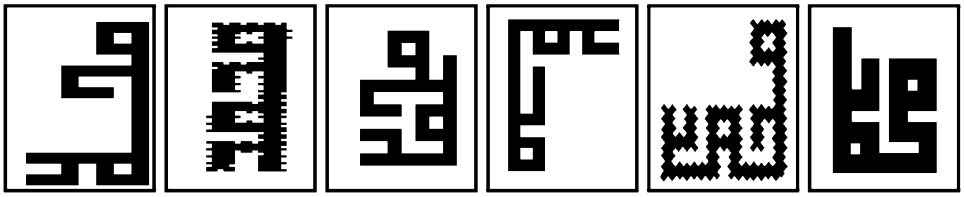


Fig. 24 *Muhammed* patterns in different formats that our proposed methods cannot match

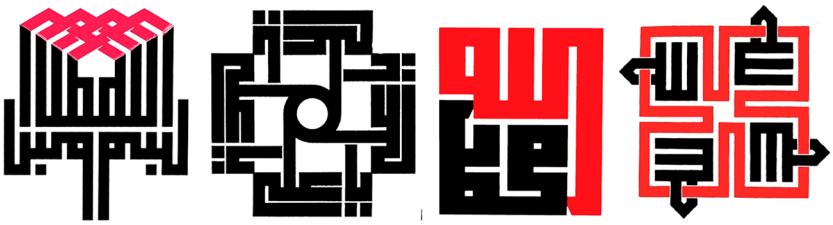


Fig. 25 Connected pattern examples that our sequence matching method cannot detect (The images are taken from [39])

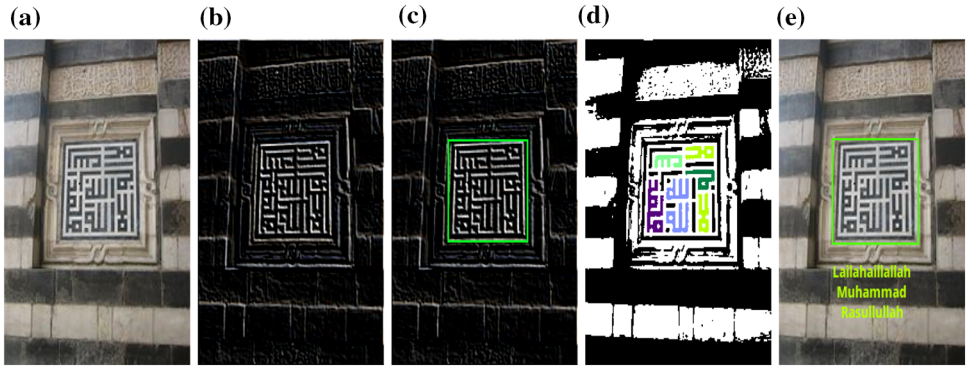


Fig. 26 Localization of Kufic inscription through edge detection and component recognition. **a** Original image **b** Edge detection **c** Localization **d** Binarization and component detection **e** Identification of complete inscription (Image is from [6])

will be easier. We are also planning to extend our dataset in order to study different and longer words. Moreover, we are planning to extract feature points and then do clustering before matching Kufic patterns.

References

1. Kuficpedia (2009). www.kuficpedia.com
2. Maghribi kufic (2009). www.calligraphyqalam.com/styles/kufic-maghribi.html
3. Abas SJ (2001) Islamic geometrical patterns for the teaching of mathematics of symmetry. Symmetry Ethnomath 12(1–2):53–65
4. Etikan S (2008) The use of the Kufic script, an element of Islamic ornament in Turkish Rug Art. Social Sci 3(2):104–112
5. Kufic info (2009). www.kufic.info/
6. Cairo mosque (2014). https://www.flickr.com/photos/helenromberg/352643197/
7. Sehzade mosque (2014). https://www.flickr.com/photos/gandara/785871440/
8. Royal mosque (2014). http://www.smashingmagazine.com/2014/03/20/taking-a-closer-look-at-arabic-calligraphy/
9. Kufic example in a photograph taken in a historical site (2014). http://gal2.piclab.us/key/kufi%20writing
10. Salam kaligrafi (2014). http://salamkaligrafi.blogspot.com.tr/
11. Square kufic patterns on bags (2014). http://www.popinjay.co/kufic-quilted-foldover-black
12. Square kufic patterns on rings (2014). https://www.etsy.com/listing/77020914/allah-ring-in-stainless-steel-islamic
13. Sakkal, M.: The art of arabic calligraphy (1993). www.sakkal.com/ArtArabicCalligraphy.html
14. Sakkal M (2003) Mysteries of square Kufi. in Future Vision 12
15. Sakkal M (2003) Square Kufic in islamic architecture. Arab Culture Center, Aleppo, Syria
16. Sakkal M (2004) Principles of square kufic design. Hrouf Arabiyya (Arabic Letters Journal) (13)
17. Sakkal M (2006) Square Kufic calligraphy in identity design. Identity magazine (No. 8)
18. Amin A (1997) Off line arabic character recognition—a survey. In: Proceedings of the 4th international conference on document analysis and recognition, pp. 596–599. Washington, DC
19. Chan J, Ziftci C, Forsyth D (2006) Searching off-line arabic documents. In: Proceedings of the 2006 IEEE computer society conference on computer vision and pattern recognition, pp. 1455–1462. Washington, DC
20. Khorsheed MS (2002) Off-line arabic character recognition—a review. Pattern Anal Appl 5(1):31–45
21. Amin A (2001) Segmentation of printed arabic text. In: Proceedings of the second international conference on advances in pattern recognition, pp. 115–126. London
22. Kufic (2009). www.en.wikipedia.org/wiki/Kufic
23. Alibeighi R, Charee A (2009) The evolution of design structuralism, position and evolution of kufic script in the greay quarans of first to fifth centruies and survey of its structure. Negareh 4(12)
24. Moustapha H, Krishnamurti R (2001) Arabic calligraphy: A computational exploration. Math Design pp. 294–306 (2001)

25. The Topkapi scroll: Geometry and ornament in Islamic architecture (2014). http://www.ee.bilkent.edu.tr/history/geometry.html
26. Grana C, Borghesani D, Cucchiara R (2009) Picture extraction from digitized historical manuscripts. In: Proceeding of the ACM international conference on image and video retrieval, pp. 1–8. New York
27. Landre J, Morain-Nicolier F, Ruan S (2009) Ornamental letters image classification using local dissimilarity maps. In: Proceedings of the 2009 10th international conference on document analysis and recognition, pp. 186–190. Washington, DC
28. Zitova B, Flusser J, Sroubek F (2004) An application of image processing in the medieval mosaic conservation. Pattern Anal Appl 7(1):18–25
29. Roman-Rangel E, Pallan C, Odobez JM, Gatica-Perez D (2011) Analyzing ancient maya glyph collections with contextual shape descriptors. Int J Comput Vision 94:101–117
30. Albert F, Gomis JM, Valor M (2005) 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, pp. 127–130. New York
31. Aljamali AM, Banissi E (2004) Grid method classification of Islamic geometric patterns. In: Sarfaz M (ed) Geometric modeling: techniques, applications, systems and tools. Springer, Netherlands, pp 234–254
32. Djibril M, Thami R (2008) Islamic geometrical patterns indexing and classification using discrete symmetry groups. Comput Cult Herit
33. Dunham D (2007) An algorithm to generate repeating hyperbolic patterns. Proc ISAMA 2007:111–118
34. Kaplan CS (2000) Computer generated islamic star patterns. In: Proceedings Bridges 2000: mathematical connections in art, music and science, p. 4 (2000).
35. Kaplan CS (2002) Computer graphics and geometric ornamental design. Ph.D. thesis
36. Ostromoukhov V (1998) Mathematical tools for computer-generated ornamental patterns. In: Electronic publishing, artistic imaging and digital typography. Lecture notes in computer science, pp. 193–223. Springer (1998).
37. Valor M, Albert F, Gomis JM, Contero M (2003) Textile and tile pattern design automatic cataloguing using detection of the plane symmetry group. Computer graphics international conference, p. 112
38. Minoofam SAH, Bastanfard A (2008) 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)
39. Ozpalabiyiklar S (2002) Bir Yazi Sevdalisi: Emin Barin. Yapi Kredi
40. Kufic example 2 (2009). www.waterholes.com/ dennette/1995/islam/shahada.htm
41. Kufic example 1 (2009). www.farm4.static.flickr.com/3377/3318123762_ea07344f17.jpg?v=0
42. Otsu N (1979) A threshold grey scale histogram. IEEE Trans Syst Man Cyber. pp. 62–66
43. Suzuki S, Abe K (1985) Topological structural analysis of digitized binary images by border following. CVGIP 30(1):32–46
44. Intel opencv library. http://opencvlibrary.sourceforge.net/ (2008)
45. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
46. Yalniz I, Altingovde I, Gudukbay U, Ulusoy O (2009) Integrated segmentation and recognition of connected ottoman script. Optic Eng 48(11):1–12
47. Yalniz IZ, Altingovde IS, Gudukbay U, Ulusoy O (2009) Ottoman archives explorer: a retrieval system for digital ottoman archives. JOCCH 2(3):8
48. Rath TM, Manmatha R (2003) Features for word spotting in historical manuscripts. In: Proceedings of the 7th international conference on document analysis and recognition, pp. 218–223
49. Rath TM, Manmatha R (2003) Word image matching using dynamic time warping. In: Proceedings of the conference on computer vision and pattern recognition. 2, 521
50. Guan N, Tao D, Luo Z, Yuan B (2012) Nenmf: an optimal gradient method for nonnegative matrix factorization. IEEE Trans Signal Process 60(6):2882–2898
51. Guan N, Tao D, Luo Z, Yuan B (2012) Online nonnegative matrix factorization with robust stochastic approximation. IEEE Trans Neural Netw Learn Syst 23(7):1087–1099
52. Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401(6755):788–791
53. Ferrari V, Fevrier L, Jurie F, Schmid C (2008) Groups of adjacent contour segments for object detection. IEEE Trans Pattern Anal Mach Intell 30(1):36–51
54. Can E, Duygulu P (2011) A line-based representation for matching words in historical manuscripts. Pattern Recognit Lett 32(8):1126–1138
55. Agarwal PK, Varadarajan KR (2000) Efficient algorithms for approximating polygonal chains. Discret. Comput. Geom. 23:273–291
56. Douglas D, Peucker T (1973) Algorithms for reduction of the number of points required to represent a digitized line or its caricature. Can Cartogr 10(2):112–122
57. Heckbert PS, Garland M (1997) Survey of polygonal surface simplification algorithms. School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, Technical report
58. Freeman H (1961) On the encoding of arbitrary geometric configurations. IRE Trans Electron Comput 2:260–268
59. Lu G (1997) Visual information systems., Chain code-based shape representation and similarity measure. Springer, London, pp 135–150
60. Freeman H (1974) Computer processing of line-drawing images. Comput Survey 6(1):57–97
61. Fortin S (1996) The graph isomorphism problem. Technical report, MIT
62. Eppstein D (1995) Subgraph isomorphism in planar graphs and related problems. In: Proceedings of the sixth annual ACM-SIAM symposium on discrete algorithms, pp. 632–640. Society for Industrial and Applied Mathematics
63. Kufic example (2014). http://islamic-cs.blogspot.com.tr/2011/09/free-simple-cross-stitch-chart.html
64. Reka Kufi (2014). http://reka-kufi.blogspot.com.tr/