Tel Aviv University

The Raymond and Beverly Sackler Faculty of Exact Sciences

The Blavatnik School of Computer Science

Computerized Paleography
Exploration of Historical Manuscripts

A Thesis submitted for the degree

Master of Computer Science

by

Liza Potikha

Supervised by

Professor Lior Wolf

April 2011

Contents

Acknowledgments

This research project would not have been possible without support of many people.

Foremost, I would like to express my sincere gratitude to my advisor Prof. Lior Wolf for his continuous support during the research and for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Deepest gratitude are also due to Prof. Nachum Dershowitz for his help, extremely valuable ideas and comments. "Evidence charts" are the best evidence of this.

It is a pleasure to thank Ms. Ruth Fridberg, for her kindness and help with all the administrative issues.

I would like to thank my husband for suffering my bad temper when something went wrong and for entertaining our daughter alone during most of the weekends.

I am very grateful to my mother for innumerable hours of babysitting (and of course for giving birth to myself).

I would like to show my gratitude to my mother in law, who also helped a great deal with the babysitting.

Another person whom I want to mention here is my wonderful daughter for whom "mommy is working" was one of the first phrases she learned to say.

Last, but not least I would like to say a special thank you to my father, for his help, and for not letting me give up my M.Sc. degree so many times I was about to. If not him, I would have never finished this research.

Thank you all.

Abstract

The modern scholar of history or of other disciplines is often faced today with hundreds of thousands of readily-available and potentially-relevant full or fragmentary documents, but without computer aids it is a very hard and usually even impossible task to find the sought-after needles in the proverbial haystack of online images. The Cairo Genizah is a collection containing approximately 250,000 fragments of mainly Jewish texts discovered in the late 19th century. The fragments are today spread out in some 75 libraries and private collections worldwide, and there is an ongoing effort to document and catalogue all extant fragments.

Paleographic information plays a key role in the study of the Genizah collection. Script style, and – more specifically – handwriting, can be used to identify fragments that might originate from the same original work ("joins"), same author or (looking more broadly) same time and/or place. Joins, are currently identified manually by experts, and presumably only a small fraction of existing joins have been discovered to date. In their fundamental work "Automatically Identifying Join Candidates in the Cairo Genizah" Wolf et al [43] showed that automatic handwriting matching functions, obtained from non-specific features using a corpus of writing samples, can perform this task quite reliably.

Following their approach we proposed a modified algorithm which is carefully designed not only to provide a high level of accuracy, but also to provide a clean and concise justification of the inference results. This last requirement enforces challenges such as sparsity of the representation in which the existing solutions are not appropriate for document analysis. We therefore suggest a new method called sparse document coding, in which the diversity of a given document is estimated automatically and used to control the sparsity of the resulting representation. Taking an advantage of the sparse coding scheme we present a tool called "Evidence Charts" which goal is to present human expert with concise evidence justifying the matching score of every two document vectors.

Next we explore the problem of grouping various Genizah documents by script style, without being provided any prior information about the relevant styles. The results show that the automatically obtained grouping agrees, for the most part, with the paleographic taxonomy. In cases where the system fails, it is due to the apparent similarities between related scripts. For the better grouping we present semi-automatic framework that allows, using reasonable human effort, to classify documents that were not classified by the automatic algorithm.

Finally we study a paleographic classification tool that matches a given document to a large set of paleographic samples. Here additional challenges arise from the need to design dictionaries of visual prototypes that either deal with non-uniform and noisy input or are trained on idealized data that does not match the actual documents.

All of the above require careful preprocessing stages in order to deal with the input some of which is extremely noisy and challenging.

This thesis is based on paper "Automatic Paleographic Exploration of Genizah Manuscripts" by L.Wolf, N.Dershowitz, L.Potikha, T.German, R.Shweka, Y.Choueka [41] and on the paper "Computerized Paleography: Semi Automatic Tools for Historical Manuscripts" by L.Wolf, L.Potikha, N.Dershowitz, R.Shweka, Y.Choueka [44]

Chapter 1

Introduction

1.1 A Background and Problems Description

Wikipedia says that *"The Information Age, also commonly known as the Computer Age or Information Era, is an idea that the current age will be characterized by the ability of individuals to transfer information freely, and to have instant access to knowledge that would have been difficult or impossible to find previously."* Nowadays hundreds of thousands of readily-available and potentially-relevant full or fragmentary historical documents are available online to the scholars of history and other disciplines. However without computer aids valuable pieces of potential knowledge into it may stay just the *data* - "symbols, ... which simply exists and has no significance beyond its existence" (Ackoff [1]) without ever getting chance to become *information*, which according to Ackoff is "data that are processed to be useful; provides answers to "who", "what", "where", and "when" questions". And these are exactly the questions that paleographers are most often asked regarding the medieval documents: "what was written", "where was it written", "when was this written" and "by whom".

Written text is one of the best sources for understanding historical life. Community documents, religious works, personal letters, and commercial records can all contribute to a fuller understanding of a given place and time. In this respect, the Cairo Genizah is a unique treasure trove of preserved middle-eastern texts, comprising some 250,000 manuscripts fragments, written mainly in the 10th to 15th centuries. Discovered in the 1890s in the attic of a synagogue in Fostat, an old quarter of Cairo, the Genizah is a large collection of discarded codices, scrolls, and documents. It contains a mix of religious Jewish documents with a smaller proportion of secular texts. With few exceptions, these documents are made of paper or vellum, and the texts are written mainly in Hebrew, Aramaic, and Judeo-Arabic (Arabic language in Hebrew characters), but also in many other languages (including Arabic, Judeo-Spanish, Coptic, Ethiopic, and even one in Chinese).

After its discovery, the Genizah attic was emptied in several stages. The bulk of the material was obtained by Solomon Schechter for Cambridge University, but there were various acquisitions by others, too. By now, the contents have found their way to over 75 libraries and collections around the world. Most of the items recovered from the Cairo Genizah have been microfilmed and cataloged in the intervening years, but the photographs are of mediocre quality and the data incomplete, with thousands of fragments still not listed in published catalogues.

Genizah documents have had an enormous impact on 20th century scholarship in a multitude of fields, including Bible, rabbinics, liturgy, history, and philology. The major finds include fragments of lost works (such as the Hebrew original of the apocryphal Book of Ecclesiasticus), fragments of hitherto unknown works (such as the Damascas Document, later found among the Qumran scrolls), and autographs by

famous personages, including the Andalusians Yehuda Halevi (1075–1141) and Maimonides (1138–1204). (See Reif [36] for the history of the Genizah and of Genizah research.)

Consider that, unfortunately, most of the manuscript leaves that were found were not found in their original bound state. Worse, many are fragmentary, whether torn or otherwise mutilated. Pages and fragments from the same work (book, collection, letter, etc.) may have found their way to disparate collections around the world. Some fragments are very difficult to read, as the ink has faded or the page discolored. Scholars have therefore expended a great deal of time and effort on manually rejoining leaves of the same original book or pamphlet, and on piecing together smaller fragments, usually as part of their research in a particular topic or literary work. Throughout the years, scholars have devoted a great deal of time to manually identify such groups of fragments, referred to as "joins", often visiting numerous libraries for this purpose. Despite the several thousands of such joins that have already been identified by researchers, much more remains to be done [28]. Accordingly, to make the future study of the Genizah more efficient, there is an acute need to group the fragments together and to try to reconstruct the original codices as well as is possible.

In their fundamental work "Automatically Identifying Join Candidates in the Cairo Genizah" Wolf et al. [43] showed that *automatic handwriting matching* functions, obtained from non-specific features using a corpus of writing samples, can perform this task quite reliably.

Following their approach we propose a modified algorithm which is carefully designed not only to provide a high level of accuracy, but also to provide a clean and concise justification of the inference results. This last requirement enforces challenges such as sparsity of the representation in which the existing solutions are not appropriate for document analysis. We therefore suggest a new method called *sparse document coding*, in which the *diversity* of a given document is estimated automatically and used to control the sparsity of the resulting representation.

In addition to the automated join-finding effort, we also study the problem of *automatically deriving the script style* of Genizah documents. We choose to do it in an unsupervised manner, in which a clustering algorithm groups the various documents, thereby separating the image sets according to the script style of each image,with no a priori bias towards a particular classification scheme. Nevertheless, the resulting division is a close match to the standard taxonomy. For even better grouping we present a semi-automatic framework that allows, using reasonable human effort, to classify documents that were not classified by the automatic algorithm.

The third problem we study is a *paleographic classification* tool that matches a given document to a large set of paleographic samples. Here additional challenges arise from the need to design dictionaries of visual prototypes that either deal with non-uniform and noisy input or are trained on idealized data that does not match the actual documents.

1.2 Thesis structure

The rest of this paper is organized as follows: Chapter 2 describes preliminary Image Processing steps done for each of the images in order to prepare it for "letters" (local descriptors) extraction step. It involves coarse alignment, foreground segmentation, detection and removal of non-relevant components, binarization, auto alignment and physical measurements. All but the last step follow the baseline work [43]. Physical measurements stage is refined to give more accurate results for the line height estimation which sometimes is crucial in a later stage of letters extraction.

Chapter 3 shows a method to obtain compact representation given a handwriting image. First it describes a baseline method for detection of interest points that will represent the image and presents an extension that handles problematic interest points

more accurately. Next it refers to the "descriptors" chosen for those interesting points and a baseline method for a dictionary creation and document coding — representing a handwriting image as a vector given its interesting points and a dictionary. Finally we describe our method for dictionary creation (*multilevel*) and for *sparse* document coding. Last section of this chapter presents *Evidence Charts* tool which takes advantage of our sparse representation and presents a human expert with concise evidence justifying the matching score of the two document vectors.

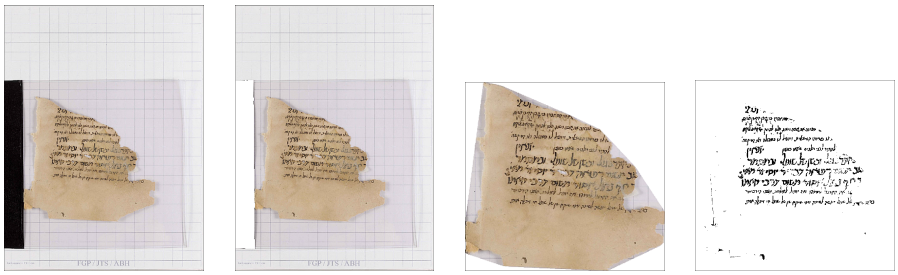
Chapters 4, 5 and 6 study the applications of the proposed sparse coding scheme to the 3 different problems respectively: handwriting matching, automatic grouping by script style and paleographic classification. Each chapter contains relevant details of the dictionary constructions method and experimental results.

The remaining chapters (7,8) discuss related work, provide conclusions and introduce the future work.

Chapter 2

Image Processing and Physical Measurements

The images supplied by the Friedberg Genizah Project were in the format of 300–600 dpi JPEGs, with arbitrarily aligned fragments placed on varying backgrounds. While uncompressed images of higher resolution are available, it was chosen not to use these since the type of methods that are used do not require higher resolution, and since the compression artifacts are negligible in comparison to the deformations created to the original fragment over the centuries. An example, which is relatively clean, is shown in Figure 2.1(a). Many of the images, however, contain superfluous parts, such as paper tags, rulers, color tables, etc. Therefore, a necessary step in the pipeline is preprocessing of the images to separate fragments from the background and to align fragments so the rows of text are horizontal. Then the physical properties of the fragments and of the text lines are measured. Both stages are described in detail in a baseline work [43].



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

Figure 2.1: Example of a document from the Cairo Genizah (ENA collection). (a) The original image. (b) After removing the black folder. (c) After segmentation (using the convex hull). (d) After binarization and alignment.

2.1 Processing

The goal of the preprocessing stage is to eliminate parts of the images that are irrelevant or may bias the image comparison process, and to prepare the images for the representation stage.

2.1.1 Coarse Manual Alignment

In the first, manual stage, the written sides of each fragment were identified. All the images were then manually rotated as necessary in multiples of 90°, resulting in alignment in the range of $[-45°, 45°]$ from upright. This initial rotation prevents the auto-alignment step, explained below, from rotating documents upside-down.

2.1.2 Foreground Segmentation

The process of separating fragments from the background in the photographs depends on the way the image was captured. At first, a machine classifier was used to identify foreground pixels based on RGB color values or HSV values. To create a region-based segmentation of the fragments, the connected components of the detected foreground pixels are marked, and the convex hull of each component is calculated. Those steps retain almost all of the relevant parts of the images, while excluding most of the background.

2.1.3 Detection and Removal of Non-relevant Components

Labels, ruler, color swatches and any other non-relevant components that fall in separated regions were manually removed. In some images, especially of large documents, a ruler appears adjacent to the actual fragments and is not separated by the region-segmentation process. The ruler used in the images is of a known type, so it is located by an automated detector based on correspondence to a reference image of this ruler. The correspondence is done by employing a randomized algorithm called RANSAC [22] in combination with scale-invariant feature transform (SIFT) [29] keypoint matching. The region of the detected ruler is segmented by color and removed.

2.1.4 Binarization

The regions detected in the foreground segmentation process are then binarized, that is, every ink pixel is assigned a value of 1 (representing black), and all other pixels are assigned a value of 0 (for white). This is done using the autobinarization tool of the ImageXpress 9.0 package by Accusoft Pegasus. To cope with failures of the Pegasus binarization, the images are also binarized using the local threshold set at 0.9 of the local average of a 50×50 patch around each pixel. The final binarization is the pixel-wise AND of those two binarization techniques. Pixels nearby the fragment boundary are set to 0. A sample result is shown in Figure 2.1(d).

2.1.5 Auto-alignment

Each region is automatically rotated so that the rows (lines of text) are in the horizontal direction. This is done by applying Hough Transform on the image [21]. See explanations in section 2.2 below.

2.2 Physical Measurements

The measurements that are being used in the later stages of our workflow are characteristics of the text rows. The height of the rows and the spaces between the rows are calculated automatically using the projection profile of the fragment (the proportion of black in each row of pixels). The text rows themselves are localized at the maxima points of these projections. Next we describe a refined algorithm that differs from the one used in the baseline.

Line height estimation algorithm We assume that the range of the valid line heights is $[h_m, h_M]$. We start with the binary image and find the estimation for the line heights using the following procedure:

1. Apply a Hough transform on the central part of the image (50% of the image area) and denote the resulting matrix $H(\rho, \theta)$.
2. Find the rotation angle $\Theta = argmax(Var_\rho H(\rho, \theta))$ and let $I(\rho) = H(\rho, \Theta)$.
3. Using initial threshold for text line: $T_0 = min(I) + \alpha_0 \cdot (max(I) - min(I))$, where $\alpha_0 = 1/4$, calculate height estimation and error:
   (a) Apply threshold T_0 on $I(\rho)$ and get a binary vector B_V.
   (b) Calculate derivative $B_V^{'}$ in order to find transitions and heights of text segments $\{h_i\}$.
   (c) Calculate $m = median(\{h_i : h_i \geq h_m\})$.
   (d) The segments, that are close to the median are considered "good". Error is a percentage of "not good" segments. We increase error on each big segment in order to penalty the situation where several lines are treated as one because of the too small threshold:
   $Err_0 = \frac{n}{N} + \beta \cdot \sum_{h_i > h_M} (\frac{h_i}{h_M} - 1)$,
   where $n = |h_i : abs(h_i - m) > \gamma \cdot m|$,
   $N = |h_i|$, and β and γ are some predefined parameters.
4. If the error is smaller than < 0.5 (at least 50% of lines were close to the median) – the line height estimation is found, otherwise we use several thresholds
   $T_j = min(I) + \alpha_j \cdot (max(I) - min(I))))$,
   where $0 < \alpha_j < 1$ and for each of them do the procedure described above.
   The threshold with the smallest error wins, and the height that was calculated using this threshold is reported as the line height estimation for this document.

In the same manner it is possible to find estimation for the spaces between the lines. We chose $\beta = 0.2$, $\gamma = 0.2$ and numeric experiments showed that the results are tolerant to these parameters.

We examined large variety of examples and compared this method to the one used in the baseline. The results are shown in Figure 2.2(a). We saw that in most cases where the two methods disagree our method outperforms the basic method, or both methods give wrong result, due to the problematic image (no text, vertical text etc.). Figure 2.2(b) presents an example image - one from the group of images, corresponding to the points in the marked region. It can be seen that a slight offset/gain exists even where both methods are generally agree on the estimation. This is explained by the variation of the definition of "line height" between the two methods.

Figure 2.3 illustrates application of the algorithm to image in Figure 2.2(c). The blue graph is $I(\rho)$; horizontal blue lines represent the thresholds for all test values of α. The table contains the results obtained for each of the thresholds. As can be seen T_2 minimizes the error for this example and so the value of 29 is chosen for this height estimation.

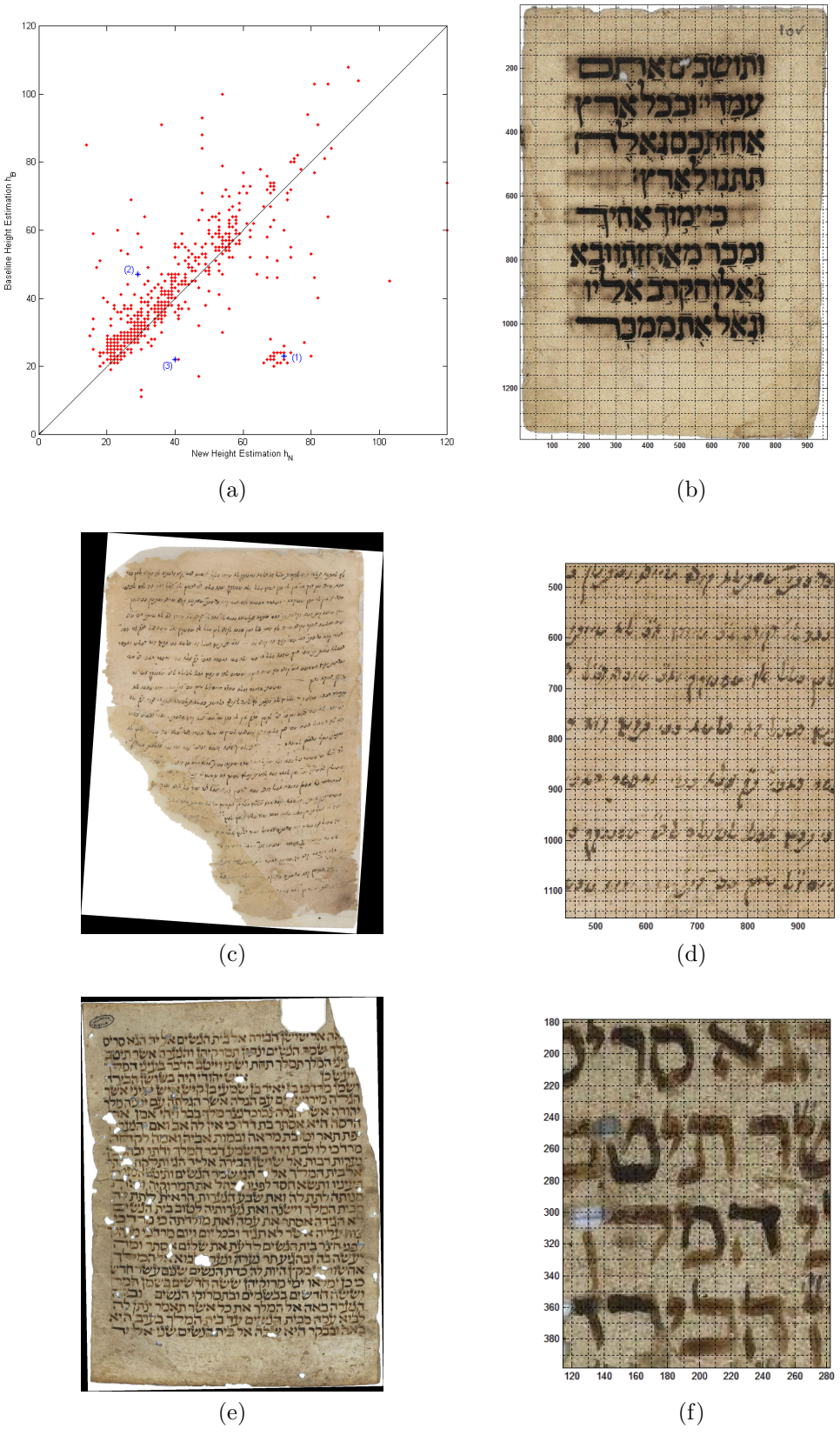
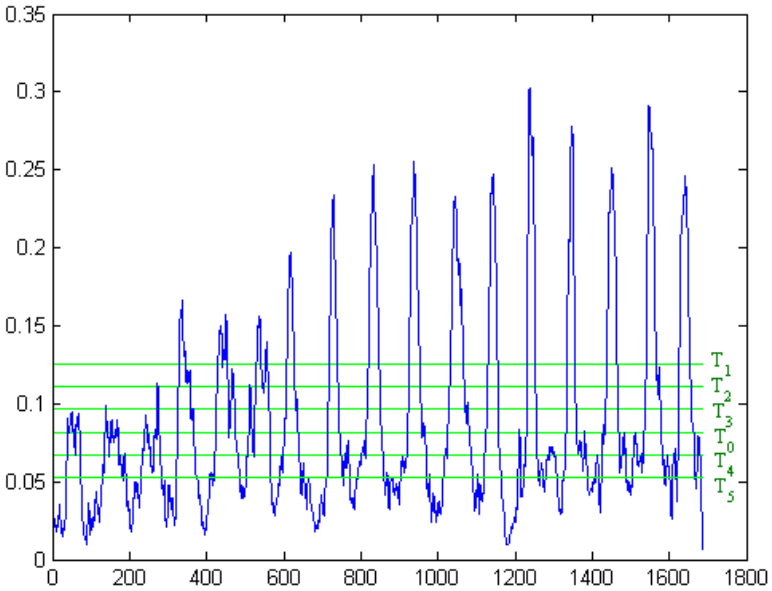


Figure 2.2: Line Height estimation (a) Our Method vs. Baseline method comparison for 600 random documents from the Genizah Database. Each point represents a document.Points, corresponding to the following example images are marked with blue asterisk and number 1-3. (b) An example image (1) - one from the group of images, corresponding to the points in the region marked with ellipse in (a) . Baseline Height estimation for this image $h_B = 23$. Our algorithm height estimation $h_N = 72$. (c) Example image (2): $h_B = 47$, $h_N = 29$ and (d) the same image zoomed in. (e) Example image (3): $h_B = 22$, $h_N = 40$ and (f) the same image zoomed in.



	T_1	T_2	T_3	T_0	T_4	T_5
α	0.40	**0.35**	0.30	0.25	0.20	0.15
Error	0.88	**0.52**	0.61	0.69	0.72	0.81
Line height est.	22	**29**	33	37	42	53 8
"Inter line" est.	76	**56**	50	20	13	8

Figure 2.3: Example: application of the algorithm to image in Figure 2.2(c). The blue graph is $I(\rho)$; horizontal blue lines represent the thresholds for all test values of α. The table contains the results obtained for each of the thresholds. As can be seen T_2 minimizes the error for this example and so the value of 29 is chosen for this height estimation.

Chapter 3

Handwriting Image Representation and Evidence Charts

The baseline method follows previous work [43] and employs a general framework for image representation that has been shown to excel in domains far removed from document processing, a method based on a bag of visual keywords [19, 26].

The "signature" of a document is based on descriptors collected from local patches in its fragments, centered around key visual locations, called "keypoints". Such methods are based on the following pipeline. First, keypoints around the image are localized by examining the image locations that contain most visual information. Next, the local appearance at each such location is encoded as a vector. In our case, the pixels of the letters themselves are good candidates for keypoints, while the background pixels are less informative. The entire image is represented by the obtained set of vectors, which, in turn, is represented as a single vector. This last encoding is based on obtaining a "dictionary" containing representative prototypes of visual keywords, and counting, for each image, the frequency of visual keywords that resemble each prototype appearing in the dictionary.

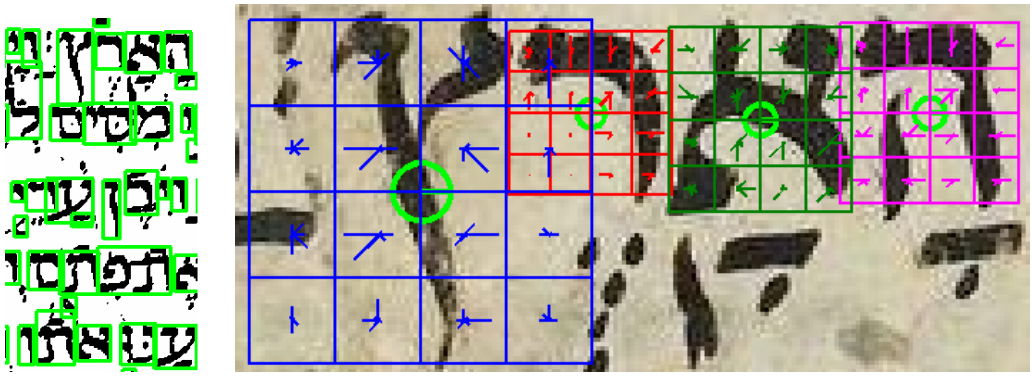
3.1 Keypoint Detection

We experimented with several methods that are often effective in general object recognition task. Among them Harris-Affine and Hessian Affine [31] but the results were not good enough. In [43], it was suggested to detect the image keypoints using the fact that, in Hebrew writing, letters are usually separated.

This method starts by calculating the connected components (CCs) of the binarized images. To filter out broken letter parts and dark patches arising from stains and border artifacts, the size of the CC is compared to the *line height*, which is estimated in the 2.2 physical measurements stage of the previous section. The CCs which are not of the *legal* size are filtered out.

For encoding, each keypoint requires a scale, which is taken as the maximum dimension of the associated CC. Figure 3.1(a) shows the keypoints found using the SIFT and CC detectors.

We saw that relaxing to some extent the assumption of separated letters improves the results in some problematic cases. We try to separate between the connected letters whenever fewer than 4 letters are connected in a row and most of the letters in the same image are disconnected. This situation is very common and may happen both due to the peculiarities of some letters writing or as an effect of digitalization



(a) (b)

Figure 3.1: (a) Keypoint detection methods using the proposed CC method. (b) SIFT descriptors of four neighboring detected keypoints

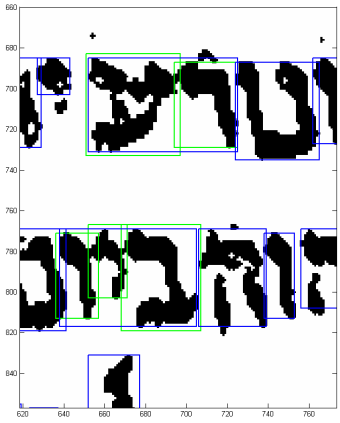


Figure 3.2: Keypoint detection example. Blue rectangles show initial detection using the CC-s as is; green additional lines show the refined detection after the proposed separation process is applied. Two occurrences of separation may be seen here - in the first row there is a *wide* box that is separated into 2 independent letters and in the second row there is a box separated into 3 independent letters.

of the documents (resizing, binarization etc.). The separation is done in a following very preservable way. After the calculation of CCs we estimate the *main orientation* of the letters in the image (by analyzing the histogram of all CC orientation). Next we concentrate on the components that are wide (relatively to the line height). We erode such components with the line angled in the main orientation. This line is of a very small width which is proportional to the pre-estimate lines height. If the number of the *legal* CC in each such eroded wide patch is exactly 2 or 3 we separate this patch, otherwise we take the whole component if it is still legal and discard if it is too wide. This improvement is especially important in the images with small amount of letters where each letter becomes very valuable. Figure 3.2 represents an example separation of letters using our method.

3.2 Local Descriptors

Each keypoint is described by a descriptor vector. In [43], the SIFT descriptor [29], which encodes histograms of gradients was demonstrated to outperform many other

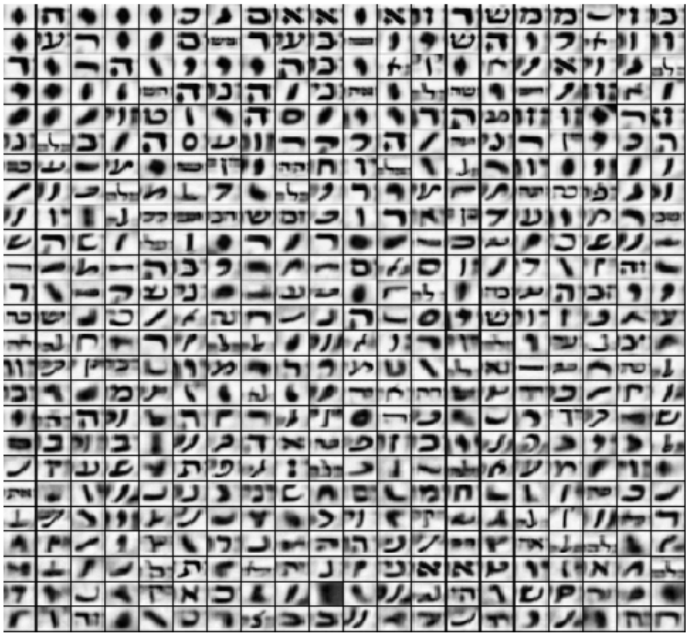


Figure 3.3: Cluster "centers" obtained by taking the average image of all images in each cluster. By clustering (grouping into homogeneous groups) a large collection of visual descriptors obtained from random images, a set of distinctive visual keywords or prototypes, referred to as a "dictionary", is formed. Note that the cluster centers look blurry, since they stem from averaging multiple images.

descriptors. We therefore use the SIFT descriptor to encode the image keypoints. Figure 3.1(b) illustrates the application of SIFT to one fragment.

3.3 Baseline Dictionary Creation and Vectorization

Bag-of-visual-keyword techniques [19] rely on a dictionary that contains a representative selection of descriptors obtained on various interest points. A representative subset of documents is set aside for this purpose. To construct a dictionary, keypoints are detected in those documents and all the descriptors are then clustered by the k-means algorithm to obtain a dictionary of varying sizes. The result is a set of prominent prototypes or "visual keywords"; see Figure 3.3.

In the baseline method, given a dictionary, a histogram-based method [19] is employed to encode each manuscript leaf as a vector: for each cluster-center in the dictionary, the number of leaf descriptors (in the encoded image) closest to it are counted. The result is a histogram of the descriptors in the encoded leaf with as many bins as the size of the dictionary. To account for the variability in fragment sizes, the histogram vector is normalized so that its Euclidean norm is 1.

In previous work [43] several variations have been presented, for example, distance-based representation techniques [38, 11] sometimes replace the histograms. In addition, metric learning techniques such as weighting the feature vectors by weights provided by a linear SVM classifier or the so-called One-Shot-Similarity [42] were shown to be very effective. In this work, which is designed to provide human interpretable results, we refrain from metric learning since it may result in counterintuitive similarity functions that mix together distinct histogram bins or employ negative weights in the similarity functions.

Another aspect of previous work that we do not reproduce is the combination of

different similarities into a more distinctive similarity function. For example, in [41], a moderate increase in performance was obtained by employing multiple dictionaries, each arising from one script type. A more significant improvement was observed in [43, 41] when adding information regarding the physical measurements of the manuscript leaves. In this work we forgo these improvements in order to obtain an easily interpretable similarity function.

3.4 Sparse Coding and Multilevel Dictionaries

We wish to present evidence to the user regarding the similarity of two documents in a concise and intuitive way. This evidence should therefore be much more compact than the size of the dictionary, which typically contains hundreds if not thousands of prototypes. This motivates the use of a sparse image representation.

Recently, several advancements have been obtained for the bag-of-visual keywords approach. The sparse coding method [46] tries to represent each keypoint's descriptor by a linear combination of a handful of dictionary vectors. The coefficients of this combination are used instead of affinities and are typically positive. An entire image is represented by a vector as large as the number of prototypes in the dictionary, where each coefficient is computed as the maximal value of this coefficient over all of the image's keypoints.

The sparse coding optimization problem needs to be solved once for each local keypoint, and the size of the computational problem depends on the size of the dictionary. Therefore, the sparse coding procedure is infeasible for large collections of images.

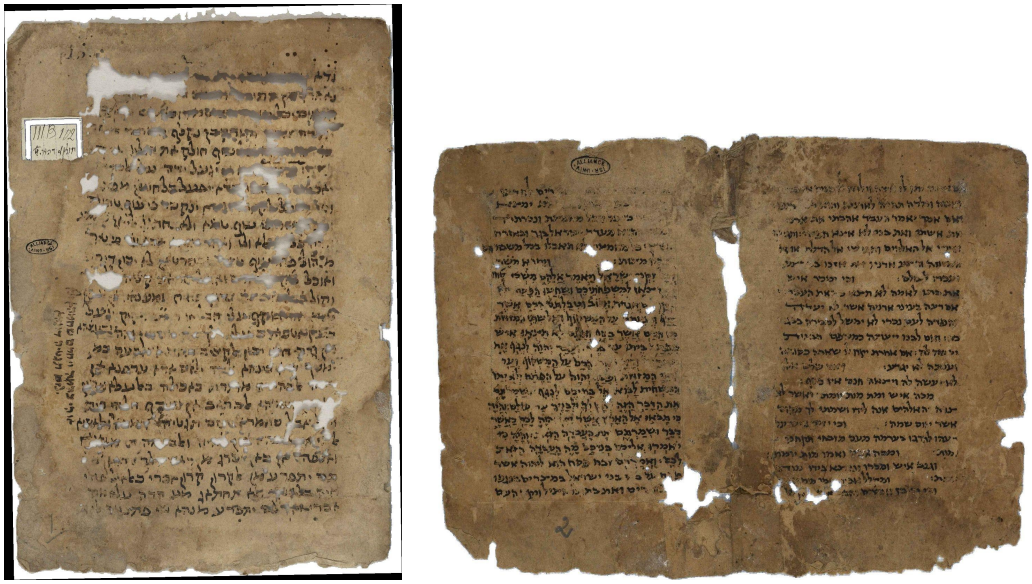
An alternate solution [40] is obtained by requiring that the solutions be local, i.e. that coefficients are nearly 0 for all dictionary items that are not in the vicinity of the descriptor. This is enforced by adding weights in an L2 minimization settings. Another, even more efficient approach considers—for each local descriptor—only the set of k closest dictionary items, and solves for these coefficients only (setting all others to 0).

While these methods (sparse coding, local coding) outperform the standard bag-of-feature techniques for object recognition tasks, they do not seem to perform well on Genizah fragments, neither for join finding nor for paleographic classification. We hypothesize that the reason is that these methods were designed as hashing schemes, where recognition is obtained by detecting the footprint of unique and distinctive descriptors in the vector representing the image. In manuscripts images, however, the distinguishing descriptors repeat multiple times with small, yet significant, variations between documents.

Moreover, our interest in sparseness arises from the need to have a succinct representation for the entire document (or pair of documents). In both sparse and locality coding individual keypoints are represented by sparse vectors; however, the entire image is represented by a combination of these vectors, in which the zero-elements are only those elements that are zero in all of the image keypoints.

The last advantage of our sparse method to mention (but definitely not the least one) is the fact that such "clean" representation of documents as will be shown later, gives better results when measuring the similarity scores between documents of certain categories. In such case the similarity measurement benefits from absence of noisy "tail" of the infrequent occurrences that may mask the real similarities as well as to contribute to the false similarities.

Consider for an example a situation when two compared documents are noisy and/or have similar texture of the paper or vellum but in fact are very different and belong to different script styles. The baseline (non-sparse) comparison will tend to overestimate the similarity between such two documents, adding up all those rare occurrences of the artifacts, that will be cut during the sparse document coding procedure.



(a) (b)

Figure 3.4: Example documents where sparse comparison greatly outperforms the baseline comparison (a) Example document of "Semi-cursive Spanish" script type (D_a) (b) Example document of "Semi-cursive Oriental" script type (D_b). Using the baseline measure, D_b is ranked among the first 10 closest documents to document D_a and $Sim(D_a, D_b) = 0.64$. Nevertheless, using the sparse representation: $Sim_{sparse}(D_a, D_b) = 0.14$ and D_b appears only in the second half of the ranked list of documents closest to D_a, as desired.

Figure 3.4(a) shows an example of such noisy document (D_a) which we compared to all other documents in the database which we used for script style clusterization problem in Chapter 5. The similarity value of D_a with the document D_b presented in Figure 3.4(b) (using the baseline representation) equals to 0.64. Thus, when the baseline measure is used D_b is ranked among the first 10 closest documents to document D_a. Nevertheless, when we compare these documents using our sparse representation: $Sim_{sparse}(D_a, D_b) = 0.14$ and D_b appears only in the second half of the ranked list of documents closest to D_a, as desired.

On the contrary, the same effect of increasing the similarity based on the paper pattern/noisiness properties in the baseline representation becomes desirable when two similar documents (joins) with similar paper qualities and of the similar script type are compared. In other words similarity measured by baseline representation takes into account both paper properties, such as texture and the conditions of document's storage, and the letters themselves, whilst the sparse representation similarity gives cleaner comparison mostly by letters and demolishes the effect of other properties.

3.4.1 Sparse Document Coding

We propose a novel per-document compacting process that is more suitable for the manuscripts images. In this process, that we use both for handwriting matching and for paleographic classification, each document is represented by a limited set of descriptors. The size of this set depends on the diversity of the document, which is estimated by the number of prototypes required to represent the documents.

Given a document, we extract the set of keypoints and cluster them in the following manner.

First we apply a procedure described later in this section for choosing the k initial values for the k-means clustering algorithm. We choose k to be approximately three

times the number of different letters in the alphabet. Then, k-means is applied and upon convergence, up to k clusters of varying sizes are obtained for a given document D. We filter the clusters such that all clusters whose cardinality is smaller than half of the size of the largest cluster are discarded. The resulting clusters serve two purposes. First, the number of remaining clusters C_D is taken as the diversity measure of the document, and second, the cluster centers themselves serve as training examples for constructing the prototype dictionary (see below). Typical values of C_D are between 10 and 40 for non-empty documents.

The sparsity of each fragment representation is enforced by a postprocessing step that nullifies every coefficient that is not among the highest C_D coefficients. In other words, the unnormalized representation is a dictionary-based (possibly weighted) histogram representation in which all coefficients other than the highest C_D are zeroed. The representation of a manuscript leaf may be a normalized sum of the individual representations of the fragment images (both recto and verso), or just a set(usually of size 2) of such representations (each normalized separately). The last representation preserves information (for example when one of the fragments is empty, we may want to treat it differently) while the former is more compact. Also the normalization should be done.

Note that for documents that contain multiple fragments, the sparsity is not heavily affected by the combination of multiple sparse vectors: the prototypes in each fragment are typically similar and small fragments typically have a lower C_D value. In comparison to other sparse methods [46, 40]:

1. The summation of sparse vectors in our method is limited by the number of fragments and not by the number of keypoints.

2. In methods that are based on the sparse coding minimization problem such as [46], the more unusual the descriptor is, the more coefficients are needed to represent it. In our method, small clusters of descriptors in the image are discarded.

3. In sparse coding, two similar descriptors might share some of the non-zero coefficients in their representation, however, even small differences could lead to the use of different basis elements. In our method, similar descriptors that are grouped together do not increase C_D unless the cluster's size passes the prescribed threshold.

When comparing a pair of documents, we employ the dot-product operator. The contributing coefficients are only those coefficients that are non-zero in both histogram vectors. Therefore the score is composed out of a handful of contributing factors. Each contributing factor is easily interpreted as the product of frequencies of a specific prototype in both documents.

3.4.2 Multilevel Dictionary

The dictionary in contrary to the baseline dictionary, where all descriptors from all the training examples are clustered to the final dictionary in a single k-means procedure, is done in two phases.

In the first phase each document from some training documents set goes through the process detailed in Section 3.4.1 in which the keypoint are clustered on the document's level and the smaller clusters are discarded. Then one more iteration of k-means is applied: all the descriptors of a document are assigned to the nearest of the remained centers (outliers that are not close enough to their closest center are excluded) ("assignment" k-means step) and the centers are recalculated once again ("update" k-means step). The centers of the resulting clusters for a document are referred to as *"averaged*

(a) (b)

Figure 3.5: (a) An example image from the Genisah database. (b) Set of corresponding "averaged" descriptors sorted descendingly by number of its occurrences in the document and visualized in the same manner as in Figure3.3, but on the single document level. For this document $C_D = 34$

descriptors". An example visualization of such averaged descriptors for a document is depicted in Figure 3.5.

Then, in the second phase the centers of the relatively large "averaged" clusters from all these documents are clustered into N prototypes using the k-means algorithm. Finally, similarly to the first phase, the "assignment" and "update" k-means steps are applied to all of the original descriptors with these centers.

This process ensures that only prominent templates are considered while constructing the dictionary, which is important in order to avoid dictionary prototypes that are based on visual patterns that arise as a result of binarization errors due to stains and other artifacts.

Additional advantage of such a "multilevel" procedure may be explained by the following intuition: Averaging and clustering at a single document level make use of the available extra information that the "letters" belong to the same document.

Performance

It is worth noting that the described "multilevel" procedure of constructing a dictionary works much faster than the baseline method of dictionary construction. For example the dictionary construction from 500 test documents as used for the Genizah Benchmark (see Chapter 4) will result in a $\approx 500 \times 1000 = 500000$ elements for a k-means procedure for the baseline. Even though we accelerate the k-means with the kd-tree [32] data structure, the new "multilevel" procedure runs more than 10 times faster. This procedure also allows simple parallelization of the first phase, since for each document, the averaged descriptors may be calculated independently. Running several processes in parallel speeds it up even further.

Cluster centers initialization

We assumed (and saw it in our experiments) that the number of final cluster centers N should be chosen to reflect the number of various script styles S. Thus N is calculated as $N \approx S \cdot A$, where A is the number of letters in the alphabet and additional graphical symbols as non-standard letters, connected letters and stains. In our experiments N was chosen as 600 ($S \sim 15, A \sim 40$).

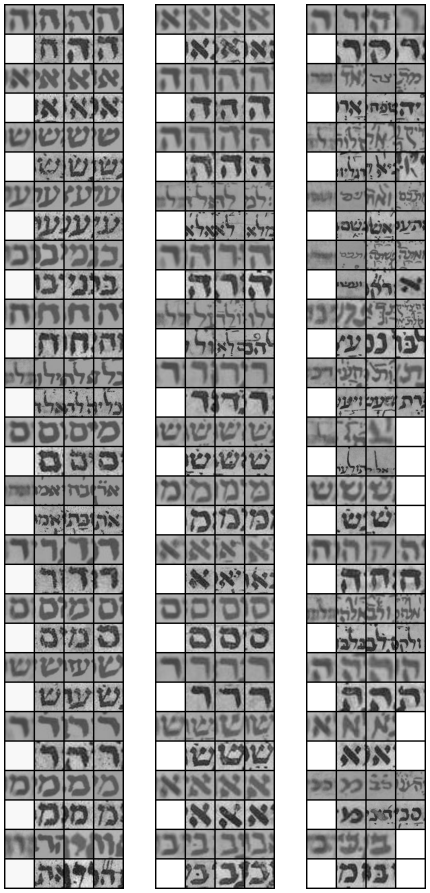
As was described earlier in this section, we apply k-means algorithm in the two levels of our workflow. Firstly, for the file level clustering which is done for each of the documents. Secondly for the global dictionary level, where N clusters are created from the averaged descriptors (from the files in the training set), which were calculated at the file level. Next we outline our method for choosing k initial values for the clustering which is similar for both levels. First, the correlation between every pair of all the available descriptors (whether regular or averaged) is computed. If the number of the available descriptors is big (as in the global level case) instead of taking all the descriptors, we choose $3k$ descriptors at random. Next a graph is constructed in which the nodes correspond to the descriptor and edges exist between nodes for which the correlation between the descriptors is above some big threshold T. The connected components of this graph are then computed. If there are more than k connected components, the k random components are selected. Otherwise all the available connected components are selected. The resulting k (or possibly less) centers are again randomly chosen from each of the selected connected components. We experimentally found that $T = 0.85$ serves well to combine really close components in the global level and $T = 0.95$ suits well for the file level. The intuition for the difference is that the similarity between different occurrences of the same letter in the same document is usually bigger than the similarity between different occurrences of the same letter between different documents (even for joins, but even more for the documents of the same script type).

3.5 Evidence Charts

Our goal is to present the human expert with concise evidence justifying the matching score of every two document vectors. For this purpose we make use of the sparsity property and provide the user with a limited number of blocks, each containing sample keypoints from both documents that were assigned to the same dictionary prototype. There is one such block for each dictionary prototype whose associated coefficients are positive for both vectors, and the blocks are sorted by the contribution of the associated coefficient to the similarity score. Examples of such charts are presented in Figure 3.6.

In each block we present examples from the first document on top of matching examples from the second document. The leftmost pair is the best matching pair. The next pair is the next best matching pair after the "influence" of the first pair is "subtracted". More formally, we consider the set of keypoints from the first document that were assigned to a particular dictionary prototype, and the analogues set from the second document. All pairwise similarities between these two sets are computed. The most similar pair is selected, and the descriptors of all other keypoints are projected to the subspace perpendicular to the average of the two selected descriptors. The process then repeats, a second pair is selected and so on.

It should be noted that in addition to its main purpose, the Evidence Charts tool is very useful for the understanding of the strong and weak sides of a method used for documents matching. For example Figure 3.6 emphasize the problem of connected letters and letters with big aspect ratio. Connected letters problem was addressed in section 3 and a problem of letters with big aspect ratio will be addressed in the Conclusion and Future work section 8.



(a) (b) (c)

Figure 3.6: Example of charts produced for 3 pairs of matching documents for the Paleographic Classification task (a,b) and a non-matching "false-positive" pair (c). Every two rows constitute a block that corresponds to one specific dictionary prototypes. Each block segment depicts the cluster center (top-left corner), one row of keypoint examples from one document of the pair, and one row of examples from the second document. Pairs of matching keypoints (one from each document) are placed one on top of the other; most matching pair on the left. Shown are the top 15 contributing blocks for each document, sorted from top to bottom. The total number of blocks is 25, 22, and 18 for the examples shown in (a),(b), and (c) respectively.

Chapter 4

Handwriting Matching

4.1 Dictionary Construction

The dictionary for handwriting matching is constructed from the 500 documents set aside in the "Geniza Benchmark" [43] for this purpose. Then a multilevel dictionary construction method which is described in section 3.4.2 is applied.

4.2 Weighting

Following the clustering process, each prototype is weighed by the so called idf term (the logarithm of the quotient obtained by dividing the total number of documents by the number of documents containing the prototype) computed over the training set at each run. The weighed histogram count of each document therefore becomes a tf-idf vector [25] (which in turn will be made sparse as described in Section 3.4.1)

4.3 The Genizah Benchmark Experiments

To evaluate the quality of our join-finding efforts, we use the comprehensive benchmark (modeled after the LFW face recognition benchmark [24]) consisting of 31,315 manuscript leaves, all from the New York (ENA), Paris (AIU), and Jerusalem (JNUL) collections [43]. The benchmark comprises 10 equally sized sets, each containing 1000 positive pairs of images taken from the same joins and 2000 negative (non-join) pairs. Care is taken so that no known join appears in more than one set, and so that the number of positive pairs taken from one join does not exceed 20.

To report results, one repeats the classification process 10 times. In each iteration, 9 sets are taken as training, and the results are evaluated on the 10th set. Results are reported by constructing an ROC curve for all splits together by computing statistics of the ROC curve (area under curve, equal error rate, and true positive rate at a low false positive rate of 0.001) and by recording average recognition rates for the 10 splits.

Table 4.1 summarizes the obtained benchmark results. As can be seen our sparse document coding method that is aimed at providing transparent (justifiable) scores performs similarly to the best baseline method, which employs the OSS metric learning technique. It also considerably outperforms the LLC [40] sparse coding approach, which uses the same dictionary and was optimized for best performance over a large range of its parameter k. Moreover, the resulting LLC vectors are not sparse (see Section 3.4), and even for small values of k over 60% of the coefficients in each vector are non-zeros.

Our method, that is based solely on handwriting is, as can be expected, not as good as the state-of-the-art method, which combines multiple dictionaries and physical mea-

Method	Area Under ROC	Equal Error Rate	Accuracy	TP rate at FP rate of 0.001
Baseline method w/ OSS learning [43]	95.6	9.2	93.7	76.0
LLC [40]	86.8	17.3	86.7	40.9
Sparse document coding (ours)	96.2	8.4	93.3	64.7
Multiple + physical + subject [41]	98.9	4.3	96.8	84.5

Table 4.1: Results obtained (percent) for the single dictionary baseline method, the LLC method, our sparse document coding method, and the state-of-the art method that combines non-handwriting data.

surements together with catalogical subject classification information. Note, however, that these auxiliary sources of information could be combined with our method as well.

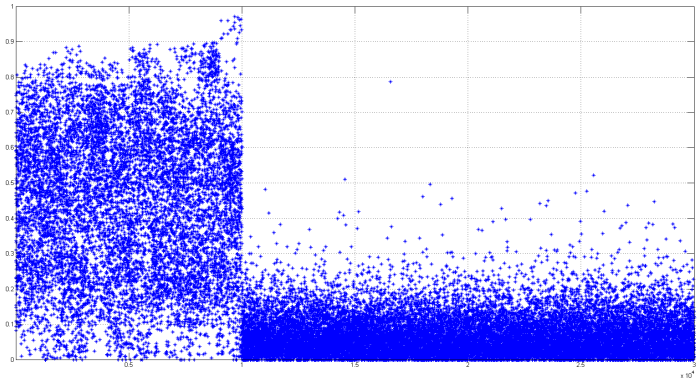
Our results for the 30,000 benchmark pairs of documents are depicted at Figure 4.1. We manually went through a big part of false positive and false negative results (given with a big confidence) and saw that several main categories may be distinguished:

False negatives

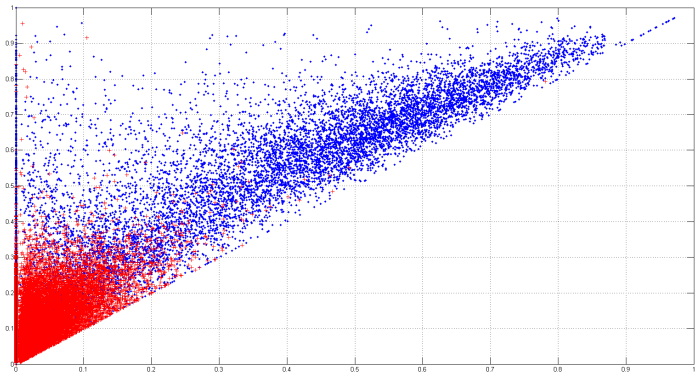
- Wrong classification in the database., i.e. non-joins that are mistakenly defined as joins in the database, and after human check (following our algorithm results) appeared not to be.
- Documents with very small number of descriptors/empty documents
- Non-standard documents (non-text documents, documents with several text directions, etc.). See for example Figure 4.2.

False positives

- Newly discovered joins (joins that are mistakenly defined as non-joins in the database)
- Documents with very small number of descriptors
- Very noisy images of a very bad quality
- Documents with apparent similarities. See for example Figure 4.2.

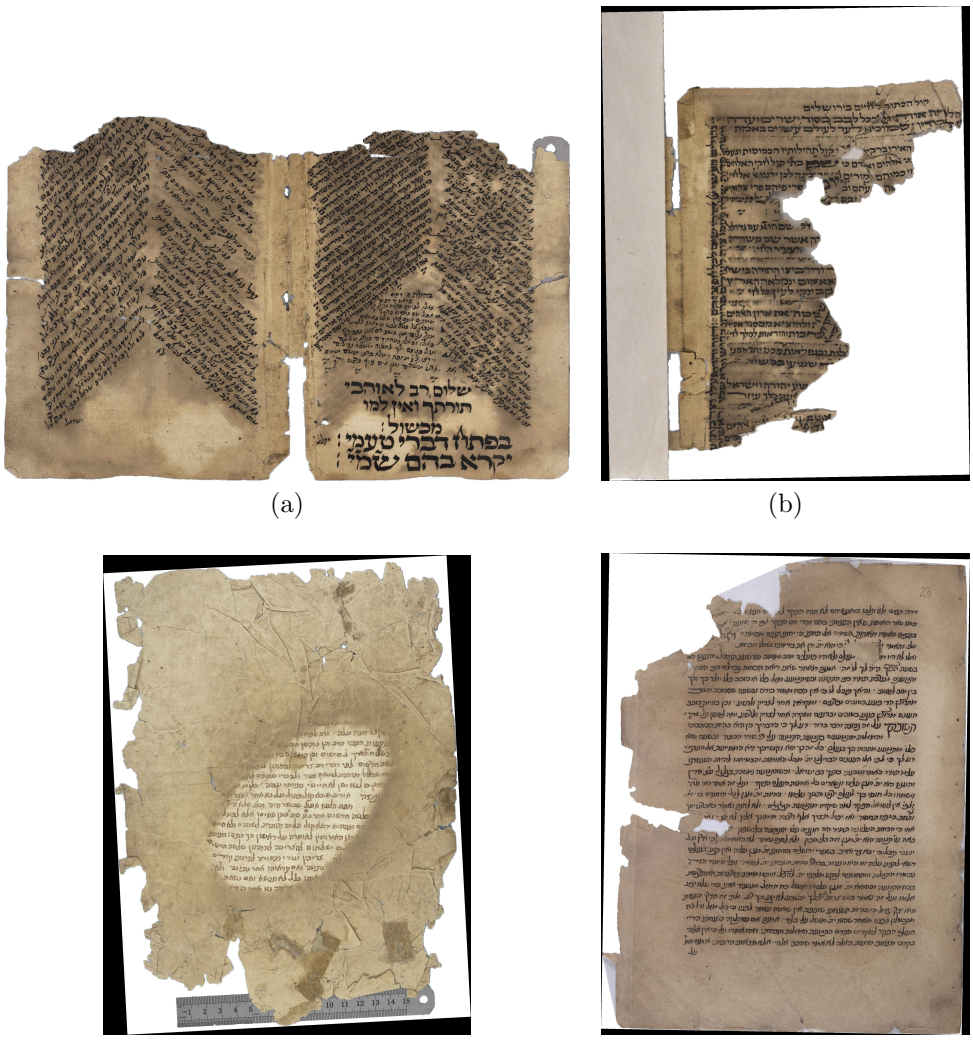


(a)



(b)

Figure 4.1: Similarities for the benchmarks pairs of documents. Here by document we refer a set of (usually) 2 pages. Given $D1(i)$ and $D2(i)$ for a pair i, $SimMin(i) = min(sim(D1 \otimes D2))$ and $SimMax(i) = max(sim(D1 \otimes D2))$ (a) Pair order number i vs. summarized similarity $SimMin(i) + SimMax(i)$ of the documents. For better visualization, pairs sorted in such a way, that first 10,000 pairs are labeled as joins and the last 20,000 pairs as non-joins. (b) $SimMin(i)$ vs. $SimMax(i)$ similarity for the same 30,000 pairs. Each point represents a pair (blue point for a pair labeled as a join, red cross for a pair labeled as a non-join)



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

Figure 4.2: Example pairs from the benchmark (a) and (b) False Negative example. (c) and (d) False Positive example (possibly newly discovered join)

Chapter 5

Unsupervised Grouping by Script Style

The search for joins focuses on minute differences that exist between various scribes. We now turn our attention to grouping the documents by a much coarser distinction: the one between script styles.

We sample 300 documents from the Genizah collection that have been classified into one of 12 script styles: "Square Ashkenazi", "Square Italian", "Semi-cursive Oriental", "Square Oriental", "Cursive Oriental", "Semi-cursive Spanish", "Square Spanish", "Cursive Spanish", "Semi-cursive Yemenite", "Square Yemenite", "Square North-African", "Cursive North-African". We then attempt to cluster the documents automatically.

We found that conventional clustering algorithms such as k-means or single-linkage algorithm are not fully appropriate for the problem of separating the documents into script-styles. Indeed, k-means focuses on clusters of similar sizes, and might produce unintuitive results for data that is not distributed homogeneously in the parameter space.

In addition k-means requires to set number of clusters in advance which is problematic since each script-style may (or may not) have several sub-styles which look very different and so the number of actual clusters is unknown beforehand.

Finally, k-means "insists" on clustering all of the data, which is not desired behavior for this problem, because of the existence of documents that are very different from all the others and documents that are very noisy. Such documents are to be examined by the professional paleographer, and we don't want them to be hidden in the general clustering. Also clustering such outliers may influence other documents clustering, since this will affect the centers update step of the k-means algorithm.

The single-linkage algorithm suffers from the same problem as well. An additional drawback of the single-linkage algorithm is the so-called chaining phenomenon: clusters may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant from each other. Because of the nature of the data (there are in-between documents for different script types) this drawback makes the use of single-linkage algorithm problematic for our case.

We therefore employed the following method that was developed in order to deal with an unknown number of clusters, variability in cluster size, and inhomogeneous data.

A single global dictionary is built using the multilevel method described in Chapter 4. Multiple dictionaries would not be appropriate here, since we would like to obtain the script styles from the data, and not impose it on the representation. First, each document is represented as a sparse vector as describe in section 3.4.

Next we apply a clustering procedure which may be seen as some mixture of top-

down and bottom-up clustering approaches. First we build a graph in which every document is a node, and an edge exists between two documents iff the correlation between their modified vectors is above 0.5. The connected components of this graph are taken as the initial clusters (*seeds*). Connected components that contain single points are referred to below as "singletons" and are considered unclustered.

We then refine these clusters by iterating, until convergence, over two alternating steps. In the first (merge) step, pairs of clusters for which the distances between each cluster's points and the cluster center are similar to the distances between the two clusters are merged. In the second (assign) step, singletons are assigned to clusters if their distance to the closest cluster center is not larger than three times the standard deviation of distances within that cluster.

Cluster center is calculated as the mean of *non-sparse* representation of all the documents in cluster A, which is made sparse using the same procedure described in 3.4 using $C_D = mean_{i \in A} C_{D_i}$

After convergence, this procedure yields 18 clusters and 34 singletons. The clusters are pretty homogeneous with regard to script style: 93% of the documents are clustered within clusters in which their script-style is the most frequent script-style; 7% are clustered in clusters in which they belong to the minority. The accuracy is measured as in [45] and is equivalent to the probability that for two documents D_i, D_j drawn uniformly at random, but from the same cluster (as determined by our clustering) with probability 0.5, and from different clusters with probability 0.5 from the dataset, our clustering agrees with the "true" clustering on whether D_i, D_j belong to same or different clusters.

Table 5.1 shows the comparative results of k-means and single-linkage algorithm with different number of clusters with the results of our method. It can be seen that single-linkage with big k gives a very good true rate, while its accuracy decreases because of the growing number of clusters. It should be noted that our method automatically chooses the number of clusters while both k-means and single-linkage need it to be given externally. For the single-linkage it is possible to supply the threshold instead, but we use the version that gets the number of clusters as a parameter for easier comparison between the algorithms.

The distribution of documents of various script styles among the 18 clusters is shown in the confusion matrix presented in Figure 5.1. Each row of this matrix corresponds to one script style, and each column to one cluster.

Figure 5.2 shows samples from representative clusters. As can be seen, confusion is often a result of script styles that are superficially similar. Naturally, a more detailed analysis of individual letters would lead to more accurate results; however, this requires accurate optical character recognition, which is beyond current state of the art for the vast majority of Genizah images.

5.1 Semi-Automatic Clustering

For the better grouping we propose semi-automatic framework that allows, using reasonable human effort, to classify documents that were not classified by the automatic algorithm.

At the first stage we apply the regular clustering algorithm until convergence as described in the previous section. All the resulted clusters are represented by their centers calculated as described earlier. Singletons are also denoted as clusters and the center of each singleton is its sparse vector.

Next we alternatively present a user (paleographer) with pairs of documents that most probably belong to the same script style (and there is not enough confidence for the automatic algorithm to decide), and then make automatic decisions where possible, based on the new information obtained from the user. Our aim is to minimize the

Method	Accuracy	True Rate	False Rate	%Singletons	NumClusters
k-means (k=12)	0.71	0.63	0.37	0	12
k-means (k=18)	0.66	0.74	0.26	0	18
single-linkage(k=12)	0.80	0.63	0.37	0	12
single-linkage(k=18)	0.80	0.78	0.22	0	18
single-linkage(k=24)	0.80	0.85	0.15	1	21
single-linkage(k=60)	0.76	0.96	0.04	6	48
our method	0.82	0.93	0.07	11	18

Table 5.1: Results obtained for the k-means method with k=12, for the k-means method with k=18, single-linkage method with k=12, k=18, k = 60, and our method. The accuracy, true and false rates are calculated as described in [45]. It can be seen that single-linkage with big k gives very good true rate, while its accuracy decreases because of the growing number of clusters. Note that our method automatically chooses the number of clusters while both k-means and single-linkage need it to be supplied.

	cluster 1	cluster 2	cluster 3	cluster 4	cluster 5	cluster 6	cluster 7	cluster 8	cluster 9	cluster 10	cluster 11	cluster 12	cluster 13	cluster 14	cluster 15	cluster 16	cluster 17	cluster 18	unclustered
Square Ashkenazi	0.00	0.00	0.00	0.33	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
Square Italian	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Semi-cursive Oriental	0.00	1.00	1.00	0.67	0.00	0.00	0.20	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15
Square Oriental	0.00	0.00	0.00	0.00	0.64	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.18
Cursive Oriental	0.00	0.00	0.00	0.00	0.04	0.00	0.80	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.03
Semi-cursive Spanish	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.44	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12
Square Spanish	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15
Cursive Spanish	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.15
Semi-cursive Yemenit	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Square Yemenite	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.06
Square North-African	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.00	0.00	0.00	0.09
Cursive North-Africa	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.71	1.00	0.00

Figure 5.1: A confusion matrix that shows that frequency within each obtained cluster of each script style. For example, the first cluster is composed entirely out of the documents of square Italian script style, while the forth cluster is one-third square Ashkenazi and two-thirds Semi-cursive Oriental.

questions to the user, while maximizing the accuracy of clustering. For this sake we maintain 2 matrices:

Graph of size $N \times N$ where N is the overall number of documents, where $Graph(i, j) =$ 1 iff documents $D_i, D_j \in$ to the same cluster

SeenGraph also of size $N \times N$. Elements of this matrix may represent one of the four possibilities:

1. ± 1 - the user was presented with this pair of documents and decided (whether positively or negatively) regarding their belonging to the same script type; or both files are in the same cluster.
2. $Don't\_Know\_Val$ - the user was presented with this pair of documents and did not make a decision
3. 0 - there is not enough information to decide automatically on this pair and input from the user is needed
4. 2 - second order decision, i.e. based on the input of the user on an other pair, this pair may be decided automatically,

where both 0 and Don'tKnowVal represent the absence of edge in the graph.

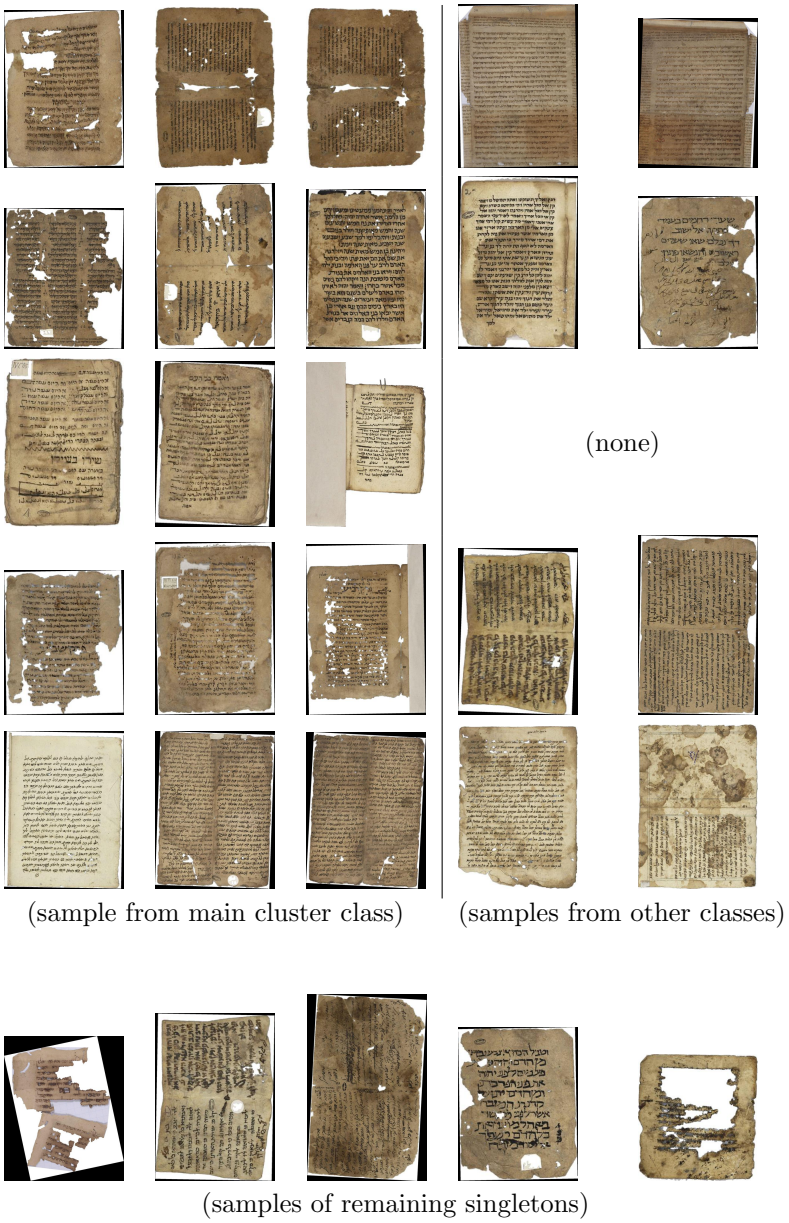


Figure 5.2: Each row shows samples from a single cluster. The left three samples are from the main script style of this cluster, while the two samples to the right are samples of the same cluster that belong to other script styles. Shown, from top to bottom are clusters 4, 5, 6, 8, 17. Also shown (last row) are samples of singletons, i.e., documents that were not assigned to any of the clusters.

After the first stage *Graph*'s elements are initialized according to the clusterization obtained and *SeenGraph* is set equal to *Graph*.

The algorithm runs as follows:
All distances between all of the clusters (their centers) are calculated and a similarity matrix $S_{M\times M}$, where M is the number of clusters at a given step, is produced. All of the elements of such matrix (excluding the diagonal) are sorted in the descending order. We go through these sorted elements and for each of the elements S_{ij} and check the corresponding *SeenGraph* value. If it is different from 0 we skip this element and go to the next one in the sorted list, otherwise decide whether the corresponding

clusters (C_i and C_j) may be merged automatically, i.e. if the similarity value is above some threshold (the same threshold that was used in the first stage when obtaining the seeds). If this is the case, both *Graph* and *SeenGraph* values for all the files in these clusters are updated with value of 1.
If similarity is not big enough, the user is presented with the C_i, C_j *connection* – pair of documents $D_1 \in C_i$ and $D_2 \in C_j$ such that the correlation between them is maximal over all pairs of documents $C_i \otimes C_j$. Note that such a question may be seen as an edge in the *SeenGraph* and the user is to set the value of this edge, If the user chooses the "Don't Know" option the only update needed is: $SeenGraph(C_i \otimes C_j) = Don't\_Know\_Val$. Otherwise, depending on the user's answer both Graph and SeenGraph are updated in the following way:
(1) $SeenGraph(C_i \otimes C_j) = Graph(C_i \otimes C_j) = \pm 1$
(2) $SeenGraph(C_i \otimes N_{SeenGraph}(C_j)) = 2$ and $SeenGraph(N_{SeenGraph}(C_i) \otimes C_j) = 2$, where $N_{SeenGraph}(C_j)$ is the neighborhood of all nodes (documents) of C_j.

The second update means putting edges between all nodes in C_i and all neighbors of the C_j (and vice versa). This update of the *SeenGraph* is important as it will avoid us presenting the user with the unnecessary questions, because this way we always maintain a set of cliques disconnected from each other and this guarantee that a new question edge can never close a cycle in the SeenGraph. For example consider the following subgraph $A \xrightarrow{1} B \xrightarrow{1} C \xrightarrow{-1} D \xrightarrow{?} A$. The question on the edge between D and A is obsolete, since it is clear that once A, B and C are known to be in the same cluster and the user has already marked C and D as not belonging to the same cluster, D and A can not be in the same cluster. Using our method we will never get to this question as the edge between D and A will be created during one of the previous steps.
Note though that Don'tKnowVal is not considered an edge and used just to avoid asking the same question on the same pair of clusters.

After the update the clusters are redefined as the connected components of the updated *Graph*, similarity matrix M is recalculated and the same procedure is iteratively repeated. The process stops upon convergence (reach the end of the similarity list without a single update) or when the user decides to quit.

We saw that this semi-automatic procedure allows to improve clustering with relatively small amount of user interactions. In our clustering example with fewer than 100 user questions, percent of singletons reduced almost twice (from 11% after the automatic clustering to 6%) while the false positive and false negative rates were not affected.

We suggest (and address this issue in the future work section) that further improvement of the semi-automatic framework may be achieved in particular using the Evidence Charts tool. Queries may be asked on a specific letters, rather than whole documents and it will help the algorithm to learn the right thresholds for letters similarities, which is essentially one of the main issues in the document comparison.

Chapter 6

Paleographic Classification

An additional task we consider is paleographic classification. The aim is given a fragment, to provide suitable candidates for matching writing styles and dates within a corpus of gold standard paleography samples. Our methodology makes it possible in addition to the nominal candidates listing, to provide visible evidence for each proposed match thus reducing the amount of effort needed from the human expert in order to validate the matches or select among the best matches.

6.1 Dictionary Construction

For the paleographic classification task, we use sample documents for each of 365 script types. These samples are extracted from the pages of the medieval Hebrew script specimen volumes [8, 9], which contain many example manuscripts whose provenance are known, and serve as an important tool in Hebrew paleography. They are structured such that high-quality sample pages of manuscripts are printed side-by-side with professionally-drawn sample letters of the alphabet, based on exemplars from the manuscript.

This task differs from the previously described Handwriting Matching (Chapter 4) and Grouping by Script Style (Chapter 5) by the fact that there, the two compared documents were taken from the same database, whenever here the document are taken from different sources (Genizah collection and Hebrew script volumes). We assume that this specifics explains the fact that a regular dictionary creation approach used for previous two tasks does not work good enough here and propose a modified dictionary creation method tailored to this specific application.

The dictionary for the paleographic classification tool are constructed out of the extracted sample letters. As a result, the prototypes are constructed from letters that are much cleaner than the connected components that are extracted from the documents. In addition, the letter frequencies in the dictionary training set do not reflect the frequencies encountered in real documents.

The dictionary construction in this case follows two steps. First, a multilevel dictionary construction method described in section 3.4 is applied to all professionally-drawn sample letters from the specimen volumes resulting in 600 clusters. The centers of each cluster are selected as prototypes and referred to as initial dictionary D_0. Second, the keypoints of each manuscript page from the specimen volumes are localized by extracting connected components of the binary images and the keypoints assigned to the prototypes by means of descriptor similarity. Lastly, prototypes for which the assigned keypoints (from the manuscripts) greatly differ from the clustered samples of the professionally-drawn letters are discarded. The remained prototypes form the new D_1 dictionary which is used in the experiments.

This filtering is done in a spatial sense and the numerical criterion is as follows: first

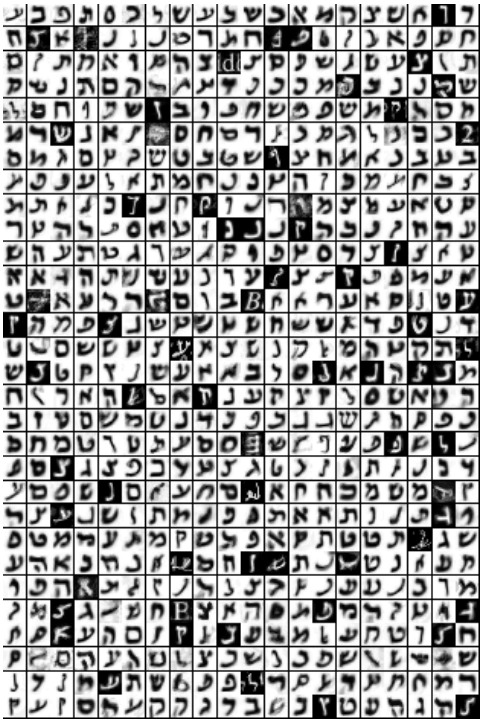


Figure 6.1: Dictionaries constructed for the paleographic classification tool. The dictionary prototypes that were removed since they were found to attract keypoints that are different than the ones clustered during training, are shown as negatives.

we calculate a "projection" of the manuscript keypoints on the D_0 dictionary. This is done by taking the mean rectangle of all the keypoints associated with a certain cluster. Second in the same manner we calculate a "projection" of the documents with professionally-drawn sample letters on D_0. Finally a cross-correlation (with registration) between these two rectangles is calculated. Clusters for which a cross-correlation of these two projections falls below some threshold are discarded.

Figure 6.1 shows the initial dictionary obtained and the results of the filtering step. As can be seen ambiguous clusters that do not correspond to actual letters are being discarded by this process.

6.2 The Experiments

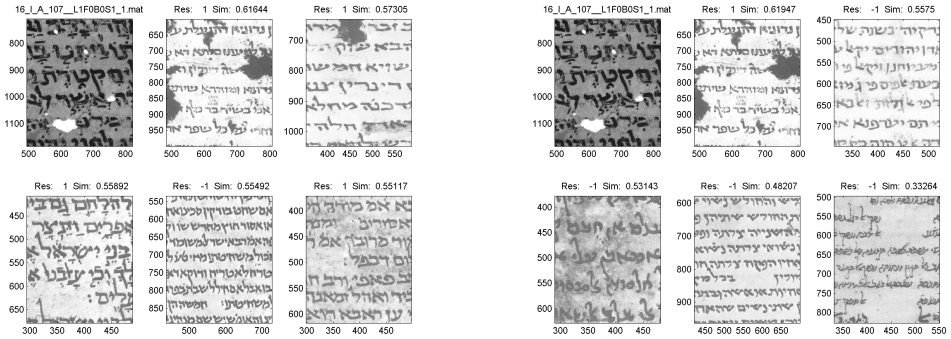
To evaluate the performance of our paleographic classification tools, we collect a sample of 500 Genizah documents of varying script types and apply the tool to them. We showed earlier (Chapter 5) that unsupervised clustering is able to group such documents into (18) clusters that are relatively pure with regard to a coarse paleographic classification. Here, the focus is on the supervised task of matching documents to an annotated gallery of examples.

For each Genizah document out of the above-mentioned sample, we retrieve the most similar documents from those in the sample volumes [8, 9]. We use both the baseline representation from [41] and our new sparse coding method, which relies on a dictionary extracted from these volumes, as described in Section 6.1. The results are then verified by a human who relies on both the evidence charts of Section 3.5 and on the original documents themselves. We mark whether the correct match is the first, second, third, fourth or fifth highest-scoring match, or none of those.

Table 6.1 summarizes the results, comparing the two methods. As can be seen, the

method/ ranking	**baseline method**	**sparse document coding**
1st match	68%	79%
2nd match	27%	16%
3rd match	3%	3%
4th match	1%	0%
5th match	0%	1%
other	1%	1%

Table 6.1: Results obtained for the baseline method and for the proposed sparse document coding method. The five highest ranking results are obtained by each method, and the average percent of correct matches per rank is recorded.



(a) (b)

Figure 6.2: Example with query image (zoomed and cropped) from a Genizah database and 5 best candidates (also zoomed and cropped) selected by our method. The query ("Square Oriental" script type) is at the top left corner. Next from left to right, top to bottom presented 5 best candidates sorted by the similarity score order. "Res" value of 1 or -1 above each snapshot indicates if the query and the candidate appeared to be of the same script type. The "Sim" is the value of similarity of a specific candidate and the query. (a) 5 first candidates with best score. (b) 5 first candidates with unique script styles.

proposed method does better than the baseline for the top-ranked document; however, the baseline method "catches-up" with the document that is ranked second. The LLC method was tried as well; however, given the human effort needed, the experiment was terminated once it became clear that this method underperforms.

Figure 6.2 shows an example with query image from a Genizah database and 5 best candidates selected by our method.

Chapter 7

Related Work

7.1 Binarization

An important paleographic task is to identify the writer of a manuscript by considering morphological characteristics of the handwriting. Since historical documents are often incomplete and noisy, in most of the cases it is required to perform separation of background as well as noise reduction. Much research has been done on this issue. A survey of the related works (prior to 2000) can be found in [33]. As examples of more recent research we would like to mention [27] and [12]. In [27] the decomposition algorithm uses local feature vectors in order to analyze an image patch and to find the best approach to threshold a local area. In [12] denoising of images is done based on the Hermite transform techniques. These algorithms are universal and can be applied on a multi-language and multi-alphabet corpus.

A method for denoising and binarization of Hebrew letters is proposed in [5]. An input sensitive system works in several stages: first a global thresholding is applied; second, the quality of the produced components are evaluated by an automatic evaluation procedure, and finally only noisy components are refined with an accurate local method.

In our work we mostly follow methods from [43] with enhancements described in Chapter 2.

7.2 Letters Segmentation

Latin letters are typically connected, unlike Hebrew ones which are usually connected only sporadically. Therefore efforts were thus expended on designing segmentation algorithms to disconnect Latin letters and facilitate identification. A survey of the segmentation methods is provided in [17].

As can be seen from the survey there is a specialization to individual languages, employing language-specific letter structure and morphological characteristics [14, 34, 20]. In our work, we mostly take advantage of the separation of Hebrew characters by employing a keypoint detection method that relies on connected components in the thresholded images. We still try to disconnect between some of the connected letters, but use the assumption that "*most* of the letters are disconnected".

From the other hand Hebrew letters sometimes appear to be "too disconnected" - i.e. composed of more than one connected component. In [6] the authors try to cope with this problem and propose a segmentation-free approach for extraction of pre-specified letters. The described method is based on erosion operator, while several stages are applied during the letter extraction process: structuring element generation, character extraction, character validation and structuring element adaptation.

Because of the specifics of further processing used in our method (a descriptor is calculated for the corresponding rectangle, rather than for the connected component itself) in most of the cases, when all of the connected components of a certain letter lay within such rectangle bounds (e.g. *aleph*, *hey*) it is possible not to take into consideration the property of a letter of being "too disconnected", which is undesired in this case.

However for certain "non-convex" letters this is not the case (like *kuf* and *zadik* in some script styles) and disconnectedness may be a problem. A possibility of an automatic combining procedure should be investigated for these letters. Following the terminology used in [17], the method in our work belongs to the classical approach that partitions the input image into sub-images which are then classified.

7.3 Writer Identification

For the identification itself two classes of methods are usually distinguished: those based on local features (text-dependent) and those based on global statistics (text-independent). Recent approaches tend to employ local features, such as using letter- or grapheme-based methods, and employ textual feature matching [34, 10]. In paper [6] a preselected set of several letters is used and for each letter the most discriminative features are extracted employing dimension reduction techniques during the training process. Text-independent statistical features, were used in [12]. Another text-independent approach was presented in [37] and is based on texture analysis, where each writer's handwriting is regarded as a different texture. In this paper the Standard texture recognition algorithm (multichannel Gabor filtering and gray-scale co-occurrence matrices) was used for handwriting-based writer identification.

Other efforts combine both local and global statistics. For example, in [20], for style identification of ancient Hebrew handwriting features based on run-length histogram were used. In [14] it was proposed to combine textural features (joint directional probability distributions) with allographic features (grapheme-emission distributions).

Most of the literature is dedicated to identifying the writer of the document from a list of known authors. In contrast, in this paper, main tasks are to search and identify text fragments written by the same hand, called *joins* and to identify/cluster fragments written at the same time and in the same place (i.e. of the same script type). We do not assume a labeled training set, especially since, in the absence of a colophon or signature, the writers of manuscripts are often unknown. Note that the handwriting techniques we use are not entirely suitable for distinguishing between different works of the same writer. However, additional data, such as text or topic identification, page size and number of lines, as used in [43], can help solving this problem.

7.4 Digital paleography

Paleographers traditionally use a mix of qualitative and quantitative features to distinguish hands. In his canonic work Jean Mallon ([30]) describes seven aspects of handwriting that have to be studied when trying to distinguish between several hands. These include: form (the morphology of the letters); angle of writing in relation to the base line; ductus (the sequence and direction of a letter's different traces); modulus (the dimensions of the letters); contrast (the difference in thickness between the hair lines and the shadow lines); writing support; internal characteristics (the nature of the text). The method used in our research puts emphasis on the first aspect (form), takes into consideration some of the remained aspects (ductus, contrast and writing support) and completely disregard others (angle, modulus and internal characteristics).

Early uses of image analysis and processing for paleographic research include [23, 39, 20]; see [35] for a survey. Quantitative aspects can be measured by automated

(or semi-automatic) means and the results can be subjected to computer analysis and to automated clustering techniques [18, 3, 2]. Features amenable to automation, including texture [37, 15], angularities [16], and others [13, 4], have been suggested. Concavity, moments and other features have been used to correctly classify selected Hebrew letters by the writer [6, 7]. What distinguishes our work is that we use generic image features for this purpose.

Chapter 8

Conclusions and Future Work

The term "exploration" used in the title of this thesis hides behind it different questions regarding the Genizah manuscripts that one would wish to be answered automatically. The main focus in this work was put on the next three questions:

- Handwriting matching: Given two documents - do they belong to the same join (i.e. both are drawn from the same original book or pamphlet) ?
- Grouping by script style: Given two documents - do they have the same script style ?
- Paleographic classification: Given an unclassified document, what is its closest standard paleography sample ?

Beyond a terse answer to all of these questions a Genizah researcher would normally also want to know "why" the machine gave the answer it gave.

In our work we introduced a scheme for data representation that produces results that are not only accurate but also allow justification of the inference. For this purpose we developed a new method for sparse coding of documents and multilevel techniques for dictionary construction that produce meaningful prototypes. We suggested (Section 3.4) that while similarity measured by baseline representation takes into account both paper properties (such as texture and the conditions of document's storage) and the letters themselves, the sparse representation similarity tends to give a cleaner comparison (mostly by letters) and demolishes the effect of other properties.

The three above-mentioned questions were thoroughly studied while applying the described scheme. The next paragraphs go into more detailed conclusions for each of them.

The first question - handwriting matching - was dealt with recently in the effort of digitizing the Genizah and for it a novel system was proposed ([43]). While very successful in finding new joins, it is essentially a ranking system that provides the human expert with nothing more than a simple numeric matching score for every pair of documents. The human expert is left with the task of validating the join, without being handed any insight as to the origin of the score. In our work we suggested that by using sparse representations and by avoiding metric learning, the results obtained by our method are easily interpretable. Some variants of sparse representation for the Bag of Features method was used for other tasks ([46, 40]). However we found these approaches to be ineffective for the Genizah exploration, while a new scheme is shown to be effective.

Investigating the second question, we explored the grouping of Genizah documents using the newly proposed Multilevel Dictionaries method (3.4), and have shown that, when the heterogeneous nature of the data set is accounted for, the paleographic information emerges as the most visually prominent characteristic. Here we applied

a clustering procedure which may be seen as some mixture of top-down and bottom-up clustering approaches, that was found to be more appropriate than the standard clustering algorithms which need to know the predefined number of clusters in advance and insist on clustering all of the data. In addition we suggested that taking into account the very heterogeneous input data, a natural extension to the clustering will be semi-automatic clustering that queries the user on selected data connections in an interactive way. We presented such semi-automatic procedure and showed that it allows us to improve clustering with a relatively small amount of user interactions.

Regarding the third question we developed a paleographic classification tool that matches a given document to a large set of paleographic samples. Here additional challenges arise from the need to design dictionaries of visual prototypes that either deal with non-uniform and noisy input or are trained on idealized data that does not match the actual documents. For this aim we introduced a notion of "projection" of a dataset on a dictionary which made it possible to filter irrelevant dictionary entries. We showed, that the proposed method, does better than the baseline for the top-ranked document; however, the baseline method "catches-up" with the document that is ranked second. Here, similarly to the first two tasks, our methodology makes it possible in addition to the nominal candidates listing, to provide visible evidence for each proposed match thus reducing the amount of effort needed from the human expert in order to validate the matches or select among the best matches.

We saw that for all of these tasks a visualization Evidence Charts tool appears to be useful for the researchers (for the understanding of the strong and weak sides of a method used) and we hope it will be used by paleographers. This tool may also be helpful in further developments of the semi-automatic framework, when queries may be asked on specific letters, rather than whole documents and it will help the algorithm to learn the right thresholds for letter similarities, which is essentially one of the main issues in documents comparison.

One of the main difficulties involved in all of the tasks described earlier is the absence of the clear ground truth for a part of the data - which makes it extremely difficult to optimize algorithms and methodology during the development and to correctly evaluate the results - this should find its partial solution in collaboration with professional paleographers. Another difficulty lies in the fact that the input is very non-homogeneous and some documents are damaged and extremely noisy. Preprocessing stages (sections 2 and 3.1) allowed us to reduce this problem to some extent.

The methods presented in this work are applicable to other corpora as well. Many archives hold large unstructured sets of hand-written forms, letters, or other documents. The same technology could provide meta-data, and enable queries based on similarity, and automatic grouping of the documents. Note that although we did not focus on Latin scripts, the method is suitable for such scripts as well with relatively straightforward adaptations to the keypoint mechanisms.

The information employed is complementary to that obtained by Optical Character Recognition (OCR) systems, and would remain so even were the accuracy of the OCR systems to increase substantially. For example, currently we are collaborating with another group, that works on the development of an OCR system for Genizah. The idea is to identify the script type of the document first which makes it possible to select the best OCR algorithm given the identified script type. Besides using our methodology to cluster patches that represent individual letters within a document allows to apply the OCR algorithms not on every such individual patch but rather on the identified clusters.

The long term goal of our work is twofold: First, we would like to enable standardization of the work of paleographers, while expediting it. Second, we would like to allow non-paleographers, with limited training, to achieve—aided by our tools—a level of accuracy that approaches that of trained paleographers.

The future work directions for this research, that were not mentioned earlier in this section, go along the following dimensions:

- **Width**
  Application of the methodology presented here to a data corpus, different from the Genizah collection such as The Dead Sea Scrolls, may lead to valuable historic findings.

- **Low Level**
  One of the most important researches in this dimension should concern local descriptors that will describe handwriting letters in an even better way. For example it may be an extension of the SIFT descriptor that we currently use, to contain information of the interrelation of the current letter to the other letters in the document it belongs to such as its relative size or its place in line etc.

  One of the problems with the SIFT descriptor we encountered, is that when calculating the descriptor for the connected component of a letter with big aspect ratio, we have to give a circle region for the SIFT (in order to preserve the aspect ratio - using ellipses works bad) and this circle region includes other neighbor letters. The solution that should be tried in order to solve most of this problem - is to cut the rectangles for the CC and to put it back to an empty document with enough place in between.

  During our experiments in addition to the SIFT descriptor, we tried to use a regular cross-correlation (with registration) between two interest points. After two interest points were considered alike (in the SIFT sense) we applied cross-correlation filtering both during the k-means of the creation and vectorization later (see Chapter 3.3). Interestingly it appeared that such filtering makes the results worse. Such an effect may point at the importance of general style related features rather than each particular letter similarities and requires further investigation.

- **Height**
  Combining OCR systems with the methodology such as, for example, per-document letters clustering, should produce added value for both OCR and the clustering itself all in one.

  While our research aimed at a "letter-based" comparison, future application could combine a pure "letter-based" comparison from one side with a pure "physical properties" comparison from the other, in order to produce a complex classifier, as it was done in [43].

- Finally, collaboration with paleographers in order to formalize their knowledge in order to enhance automatic decision making mechanisms, opens a new **"forth"** dimension and perspectives.

Bibliography

[1] R. L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16:3–9, 1989.

[2] F. Aiolli and A. Ciula. A case study on the system for paleographic inspections (SPI): Challenges and new developments. In *Proceeding of the 2009 conference on Computational Intelligence and Bioengineering*, pages 53–66, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press.

[3] J. F. A. Aussems. Christine de Pizan and the scribal fingerprint – a quantitative approach to manuscript studies. Master's thesis, Utrecht, 2006.

[4] M. Aussems and A. Brink. Digital palaeography. In *Malte Rehbein, Patrick Sahle & Torsten Schassan (eds), Palaeography and Codicology in the Digital Age*, pages 293–308, 2009.

[5] I. Bar-Yosef. Input sensitive thresholding for ancient hebrew manuscript. *Pattern Recogn. Lett.*, 26:1168–1173, June 2005.

[6] I. Bar-Yosef, I. Beckman, K. Kedem, and I. Dinstein. Binarization, character extraction, and writer identification of historical Hebrew calligraphy documents. *Int. J. Doc. Anal. Recognit.*, 9(2):89–99, 2007.

[7] I. Bar-Yosef, K. Kedem, I. Dinstein, M. Beit-Arie, and E. Engel. Classification of Hebrew calligraphic handwriting styles: Preliminary results. In *DIAL '04: Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, page 299, Washington, DC, USA, 2004. IEEE Computer Society.

[8] M. Beit-Arie, E. Engel, and A. Yardeni. *Specimens of Mediaeval Hebrew Scripts, Volume 1: Oriental and Yemenite Scripts (in Hebrew)*. The Israel Academy of Sciences and Humanities, Jerusalem, 1987.

[9] M. Beit-Arie, E. Engel, and A. Yardeni. *Specimens of Mediaeval Hebrew Scripts, Volume 2: Sefardic Script (in Hebrew)*. The Israel Academy of Sciences and Humanities, Jerusalem, 2002.

[10] A. Bensefia, T. Paquet, and L. Heutte. Information retrieval based writer identification. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 946–950, 3-6 2003.

[11] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[12] S. Bres, V. Eglin, and C. Volpilhac Auger. Evaluation of handwriting similarities using hermite transform. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France) France, 2006. Suvisoft.

[13] A. A. Brink, J. Smit, M. L. Bulacu, and L. R. B. Schomaker. Quill dynamics feature for writer identification in historical documents. Forthcoming.

[14] M. Bulacu and L. Schomaker. Automatic handwriting identification on medieval documents. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 279 –284, 10-14 2007.

[15] M. Bulacu and L. Schomaker. Text-independent writer identification and verification using textural and allographic features. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(4):701–717, 2007.

[16] M. Bulacu, L. Schomaker, and L. Vuurpijl. Writer identification using edge-based directional features. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, pages 937–941, Washington, DC, USA, 2003. IEEE Computer Society.

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

[18] A. Ciula. Digital palaeography: using the digital representation of medieval script to support palaeographic analysis. In *Digital Medievalist*, 2005.

[19] C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[20] I. Dinstein and Y. Shapira. Ancient Hebraic handwriting identification with run-length histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 12:405–409, 1982.

[21] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, January 1972.

[22] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6), 1981.

[23] J.-M. Fournier and J. Vienot. Fourier transform holograms used as matched filters in Hebraic paleography. *Isr. J. Technol.*, pages 281–287, 1971.

[24] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. 2007.

[25] K. Jones et al. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[26] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[27] G. Leedham, S. Varma, A. Patankar, and V. Govindarayu. Separating text and background in degraded document images; a comparison of global threshholding techniques for multi-stage threshholding. *Frontiers in Handwriting Recognition, International Workshop on*, 0:244, 2002.

[28] H. G. Lerner and S. Jerchower. The Penn/Cambridge Genizah fragment project: Issues in description, access, and reunification. *Cataloging & Classification Quarterly*, 42(1), 2006.

[29] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[30] J. Mallon. *Paléographie Romaine*. Consejo Superior de Investigaciones Científicas, Instituto Antonio de Nebrija, de Filología, Madrid, 1952.

[31] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[32] A. Moore. A tutorial on kd-trees. 1991.

[33] G. Nagy. Twenty years of document image analysis in pami. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:38–62, January 2000.

[34] M. Panagopoulos, C. Papaodysseus, P. Rousopoulos, D. Dafi, and S. Tracy. Automatic writer identification of ancient greek inscriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(8):1404 –1414, aug. 2009.

[35] R. Plamondon and G. Lorette. Automatic signature verification and writer identification – the state of the art. *Pattern Recognition*, 22(2):107–131, 1989.

[36] S. C. Reif. *A Jewish Archive from Old Cairo: The History of Cambridge University's Genizah Collection*. Curzon Press, Richmond, England, 2000.

[37] H. E. S. Said, T. N. Tan, and K. D. Baker. Personal identification based on handwriting. *Pattern Recognition*, 33(1):149–160, 2000.

[38] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994–1000, 20-25 2005.

[39] C. Sirat. *L'examen desécritures: l'oeil et la machine. Essai de méthodologie*. PhD thesis, Editions du Centre National de la Recherche Scientifique, Paris, 1981.

[40] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[41] L. Wolf, N. Dershowitz, L. Potikha, T. German, R. Shweka, and Y. Choueka. Automatic paleographic exploration of genizah manuscripts. *Codicology and Palaeography in the Digital Age II*, 2011.

[42] L. Wolf, T. Hassner, and Y. Taigman. The one-shot similarity kernel. In *ICCV*, 2009.

[43] L. Wolf, R. Littman, N. Mayer, T. German, N. Dershowitz, R. Shweka, and Y. Choueka. Identifying join candidates in the cairo genizah. *IJCV*, 2010.

[44] L. Wolf, L. Potikha, N. Dershowitz, R. Shweka, and Y. Choueka. Computerized paleography: Semi automatic tools for historical manuscripts. To appear.

[45] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell. Distance metric learning with application to clustering with side-information. In *NIPS'02*, pages 505–512, 2002.

[46] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.