Υπέρθεση περιγραμμάτων για εντοπισμό λέξεων σε εικόνες χειρόγραφων κειμένων

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Άγγελο Γιώτη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ – ΕΦΑΡΜΟΓΕΣ

Οκτώβριος 2012

Thanks

I would like to thank my advisor Cristophoros Nikou, Assistant Professor of the Department of Computer Science, University of Ioannina, for our very good cooperation during my thesis implementation. Of great importance was also the assistance of Demetris Gerogiannis, Phd student of the Department of Computer Science, University of Ioannina, for his guidance into the right direction. Finally, I would like to deeply thank my family who supported me till the very end of my thesis in their own way.

Contents

Index Of Figures

Index Of Tables

Index Of Algorithms

Abstract

Giotis, Angelos. MSc, Computer Science Department, University of Ioannina, Greece. October, 2012. Word spotting in handwritten text using contour-based models. Thesis Supervisor: Christophoros Nikou.

The research topic of this dissertation addresses the problem of word spotting in images of handwritten text. Relying on an object detection system in real images, local contour features are extracted from segmented images of handwritten words and they are incorporated into a learning framework which allows obtaining word-specific shapes. The representative shape model of a word class is trained using a random subset of the images belonging to that class. To accommodate both intra and inter-writer variability, a statistical model of intra-class deformations is learnt using principal component analysis (PCA). To identify a word instance, the model is firstly matched to a presegmented test image by a Hough-style voting scheme which determines its approximate position and scale in the test image. Then, a non-rigid point set registration algorithm deforms the model according to its learnt modes of variation to capture the shape of the unknown word. Initially, the consistency of the modeling process is strengthened by training five randomly selected models for each word class. Moreover, a principled approach for word model creation is proposed, where the training set is no longer randomly or manually selected, but similar words are grouped by applying the normalized cut clustering algorithm on the training images of a particular word class. The resulting clusters determine both the number of models to be used for a word class and which images are responsible for model creation. An extensive experimental evaluation of the application of these models to word spotting is finally presented.

Εκτενης Περιληψη στα Ελληνικα

Γιώτης Άγγελος του Παύλου και της Ελευθερίας. Msc. Τμήμα Πληροφορικής Πανεπιστημίου Ιωαννίνων, Οκτώβριος, 2012. Υπέρθεση περιγραμμάτων για εντοπισμό λέξεων σε εικόνες χειρόγραφων κειμένων. Επιβλέπων: Χριστόφορος Νίκου.

Η παρούσα εργασία πραγματεύεται το πρόβλημα του εντοπισμού λέξεων σε εικόνες χειρόγραφων κειμένων. Στηριζόμενοι σε ένα σύστημα εντοπισμού αντικειμένων σε φυσικές εικόνες, εξάγουμε τοπικά χαρακτηριστικά από περιγράμματα κατατετμημένων εικόνων που αναπαριστούν μεμονωμένες λέξεις. Στη συνέχεια, τα χαρακτηριστικά αυτά ενσωματώνονται σε ένα σύστημα εκπαίδευσης με στόχο τη δημιουργία συνόλων σχημάτων που περιγράφουν λέξεις. Αρχικά, για κάθε κατηγορία λέξεων, ένα αντιπροσωπευτικό μοντέλο εκπαιδεύεται χρησιμοποιώντας ένα τυχαίο υποδύνολο των εικόνων του συνόλου των δεδομένων της κατηγορίας. Προκείμενου να αντιμετωπιστεί η μεταβλητότητα των γραφικών χαρακτήρων ανάμεσα σε διαφορετικούς γραφείς, καθώς και οι διαφοροποιήσεις στο γραφικό χαρακτήρα για τον ίδιο γραφέα, δημιουργείται ένα στατιστικό μοντέλο με ανάλυση σε πρωτεύουσες συνιστώσες. Η ανίχνευση μιας νέας λέξης σε μια εικόνα επιτυγχάνεται με έναν αλγόριθμο δύο βημάτων. Στο πρώτο βήμα, η ευθυγράμμιση του μοντέλου με τα δεδομένα της νέας εικόνας, ως προς τη θέση και την κλίμακα, γίνεται με έναν αλγόριθμο που βάσίζεται στο μετασχηματισμό Hough. Ακολούθως το ευθυγραμμισμένο μοντέλο παραμορφώνεται με έναν αλγόριθμο μη γραμμικής υπέρθεσης συνόλων σημείων που βελτιστοποιεί μια συνάρτηση κόστους ως προς τις παραμέτρους του μετασχηματισμού, οι οποίες περιορίζονται από τις πρωτεύουσες συνιστώσες του συνόλου εκπαίδευσης. Η συνέπεια του μοντέλου ενισχύεται σε πρώτη φάση με την εκπαίδευση πέντε διαφορετικών, τυχαία επιλεγμένων μοντέλων για κάθε κατηγορία λέξεων. Στη συνέχεια, ο καθορισμός του αριθμού των μοντέλων που θα αποτελέσουν το σύνολο εκπαίδευσης για κάθε κατηγορία, αλλά και το ποιες ακριβώς εικόνες είναι πιο αντιπροσωπευτικές για να συμμετέχουν σε αυτό το σύνολο εκπαίδευσης, πραγματοποιείται από τον αλγόριθμο φασματικής ομαδοποίησης. Τέλος, όλες οι εκδοχές των μοντέλων ελέγχονται σε εκτενή πειραματική αξιολόγηση για τον εντοπισμό λέξεων.

Chapter 1

Introduction

1.1 Word spotting

During the last two decades, the explosion of information has led to a need for indexing it. If the required data is in machine readable form (ASCII), text retrieval engines are able to deal with this matter. However, most of today's information is on paper or on videos and not in machine readable format. Traditional Optical Character Recognition (OCR) techniques which usually recognize words by processing letters indipendently [6], work well with machine printed fonts against clean backgrounds. These methods are rather inefficient when applied to handwritten text, such as letters, manuscripts, or entire books and sometimes even fail, in the case of historical documents where the quality is often significantly degraded due to faded ink, stained paper and other adverse factors.

The automatic recognition of handwritten text is a problem that still remains unsolved [58], especially in the field of unconstrained handwriting recognition (HWR) where high intra-writer and inter-writer variability exists. Given a page of handwritten text, a common query for a user to ask is whether or not a word or words of interest appear on that page. This allows the user to search through a set of documents for a subset that is of most interest. This is the motivation behind the *word spotting idea.* Given a scanned image of a handwritten document and a query that consists of either an actual example from a collection of interest or it is artificially generated from an ASCII keyword, the *word spotting problem* asks if the image contains a handwritten image of the query word and if so, at which particular location in the image that keyword exists.

There is a variety of applications on word spotting that require handwritten texts to be available for searching and browsing:

- Detection of words in handwritten mails for document routing purposes.
  - Documents containing the word "urgent" might be given a higher priority.
  - Documents containing the word "cancelation" may be re-routed to the customer service department.
  - Finding all occurrences of the word "complain" in the letters sent to a company is crucial.
- Metadata extraction from document images for document categorization.
- Searching and indexing historical handwritten collections written by a single or few authors.
  - Significant factor for preserving the world's cultural heritage.
  - Great assistance is provided in searching and browsing such documents for researchers and the public alike.
- Segmentation of images of historical documents into meaningful regions, which can be improved with keyword spotting.
- Identification of figures and their corresponding captions.
  - Spotting the keyword "Fig.", like in [36].
- Accessibility of handwritten books through the Internet.
  - Google and Yahoo have already made an effort to incorporate this application into their search engines [42].

The implementation of the word spotting procedure in general is not only cumbersome, regarding computational costs, but it may also result in poor performance in detecting keywords against irrelevant data, as a high separability between "interesting" and not "interesting" words is essential. We refer the reader to [62] for a clear view about clusters that contain "interesting" words.

Early work in information retrieval by Luhn [47] provides us a general idea of such clusters. A plot of term frequencies, where terms are ordered by decreasing frequency of occurrence, exhibits a distribution that is known as Zipf's law [77]. Specifically the frequency of the kth most frequent term has a frequency that is f_0/k, where f_0 is the frequency of the most frequent term. Luhn argued that index terms should be taken from the middle of that distribution. Figure (1.1) shows an example of the actual distribution of term frequencies and the distribution predicted by Zipf. It seems that the large amount of mass is concentrated in high-frequency terms and the long tail of the distribution to the right, which continues beyond the shown range.

The reason is that terms with frequencies that are high (left side of the plot) are often *stop words*, such as as *"and", "the", etc.*, which do not carry any meaning. Terms with very low frequencies are often sporadic, and are not descriptive of the content in the collection. Terms that are descriptive of the content can often be found in the middle of the plot. Their repeated, but not excessive use suggests that they are essential to describe the content of the collection and should consequently be part of the index. In the following section we review related work from this area.

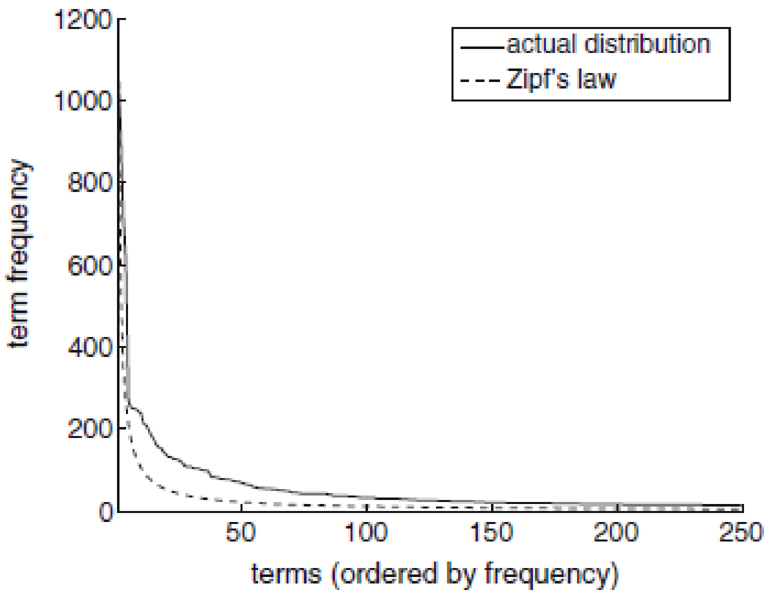


Figure 1.1: Zipf's law. The plots show the actual distribution of term frequencies and the prediction made with Zipf's law based on the actual frequency of the most frequent term. The collection size is 21,324 words; only the left-hand portion of the graph is shown. Figure reproduced from [62].

1.2 Related work

Either as a pattern classification task or as a content-based image retrieval (CBIR) procedure, the word spotting problem can be categorized into three main fields:

- Word-based word spotting
- Line-based word spotting
- Document-based word spotting

1.2.1 Word-based word spotting

In *word-based* keyword spotting, a first attempt to detect a word or a phrase in an image was proposed in [11], [35], [39] for printed text, and later, in [50] for handwritten text. However, the idea of spotting keywords has been proposed earlier in speech analysis [33]. These first attempts deemed single-word images and deployed techniques similar to Optical Character Recognition (OCR). Some methods make comparisons between individual pixels of the query and the test image (or selected parts of it, called zones of interest (ZOI)), while others calculate a global distance value between the two pixel sets. This consists in matching an input image with one or multiple query images to determine a similarity (distance) that might indicate a correspondence.

More specifically, two main types of approaches have been proposed in this context. The first type uses holistic techniques where an image is described with a single feature vector and a distance between vectors is defined. For example, Manmatha et al. [50] use directly the image pixels as features and apply the Scott and Longuet-Higgins (SLH) distance [69], which is invariant to affine transformations. Khoubyari and Hull showed in [35] that Euclidean distance mapping (EDM) could be used to match printed words but as it was later shown by Manmatha et al. [51] the handwriting field is much more challenging and EDM is insufficient for it. Interesting works in this approach include XOR comparison [49] as well as the use of Hausdorff distance between connected components [46]. More complex holistic features are the moments of the black pixels [5]. Zhang et al. [75] employ a set of binary features called GSC (Gradient-Structural-Convexity) and match them using a correlation like measure. Several binary GSC features are also explored in [71] and [76]. Another interesting approach to holistic word recognition is presented in [2] for historical handwritten manuscripts. It is based on matching word contours instead of whole images or word profiles. In contrast with our approach, which will be presented in a following section, this method does not involve training. It rather consists of robust extraction of closed word contours and the application of an elastic contour matching technique proposed originally for general shapes [1]. Finally, holistic word features in conjunction with a probabilistic annotation model are proposed in [63] for historical handwritten documents. This first automatic retrieval system allows one to spot arbitrary words. However, keywords that do not occur in the training set can't be dealt with properly.

The second type of approaches describes a word image as a set of local features. For instance, Leydier et al. [43] use the gradient angles as features and a cohesive elastic distance. Similarly, an elastic matching procedure is used in [44] where different pixelwise gradient matchings are compared. Rothfeder et al. [67] use a corner detector and the sum of Euclidian distances of corresponding keypoints (corner features) as an elastic distance between corner positions. Furthermore, the state-of-the-art technique for computing the distance is dynamic time warping (DTW) [61]. This is the most common local approach where a word is represented as a sequence of features, extracted via a sliding window. Comparing such sequences using DTW is one of the most commonly used word spotting methods. For this reason, a detailed reference is made to Rath et al. [62] where their word matching approach with DTW is compared with a number of different matching techniques. In addition to their matching method, they present an integrated word spotting system for indexing historical documents which provides us a clear view of the word spotting problem in general.

Due to the great amount of variability in handwriting and the high noise levels in historical documents, handwritten historical documents are currently transcribed by hand. In essence, this means that each occurrence of a word in a corpus must be annotated. The goal of the word spotting idea applied to handwritten documents, which is presented in figure 1.2, is to greatly reduce the amount of annotation work that has to be performed, by grouping all words into clusters. First, a document is segmented into word images. The idea of word spotting is to use image matching for calculating pairwise "distances" between word images, which can be used to cluster all words occurring in a collection of handwritten documents. Ideally, each cluster would contain all the words with a particular annotation. Clusters that contain terms which are "interesting" for an index for the document collection are selected and labeled manually. By assigning the cluster labels to all word images contained in a cluster, a partial transcription of the document collection is provided and in turn allows the creation of a partial index for the collection. Consequently, a retrieval of text portions that contain only the manually assigned labels is feasible.

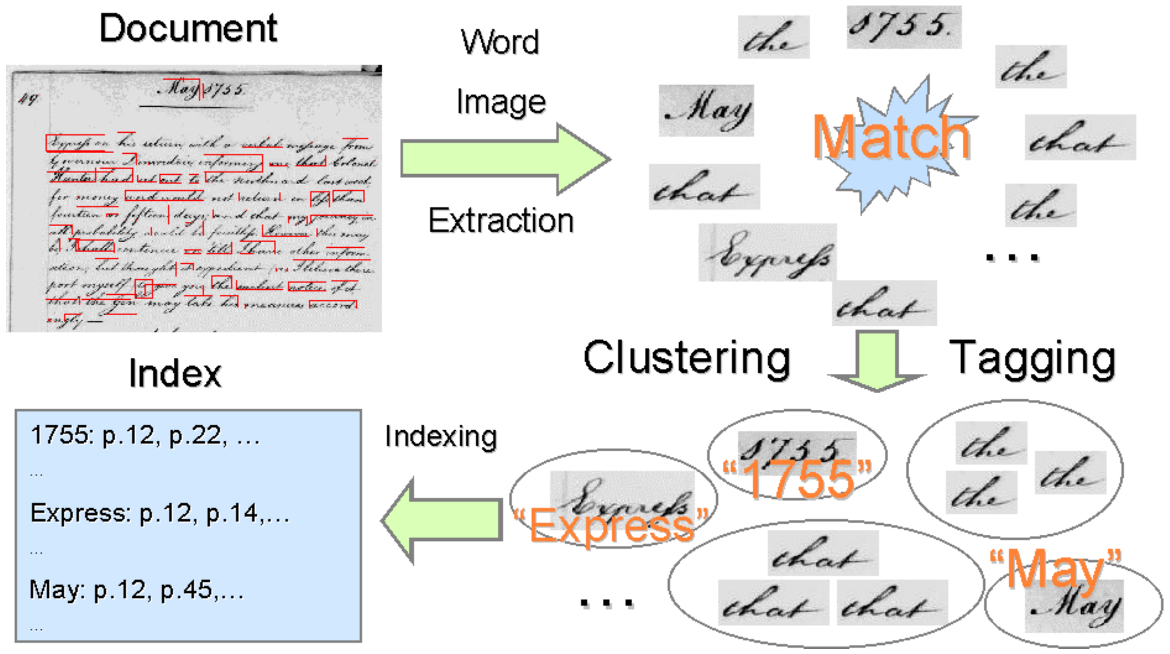


Figure 1.2: An illustration of the word spotting process. Documents are segmented and distances between word images are calculated. After clustering the word images, some clusters are manually labeled and can be used as index terms. Figure reproduced from [62].

At the heart of the word spotting idea is the word image matching algorithm. Its accuracy and efficiency determine the quality of the clustering and the size of the collection that can be processed using the word spotting process. The approach to matching images using dynamic time warping of profiles of the word images is discussed by Rath et al. in [62]. DTW has been widely used to match 1-D signals in the speech processing, bio-informatics and also the online handwriting communities. It can handle local distortions in word images and is not restricted to a single global transform. Their method is compared with a number of techniques referred already, including XOR, affine-corrected Euclidean Distance Matching (EDM) [35], shape context [4], intensity correlation using sum of squared distances, an affine matching point matching algorithm due to Scott and Longuet-Higgins (SLH) [69] and a point correlation voting algorithm [67]. Moreover, a comparison between GSC-based spotting and DTW is given in [72] and [76].

Apart from word profiles used in [62], a variety of other features has been proposed for DTW matching such as "eigenslits" (PCA projections of the image pixels) [73] or contours [2]. These methods show relatively good accuracy and speed in applications such as historical document retrieval. However, it is worth mentioning that these works, except of [75] report results on test data by a single or few writers. Thus the generalization capability of these approaches to more complex detection tasks is not clear. In [64], Rodriguez-Serrano et al. propose a different set of local gradient histogram features for spotting handwritten text in realistic, unrestricted conditions where high variability among writing styles and writers is present. This feature set is inspired by the SIFT keypoint descriptor [45], which is basically a histogram of oriented gradients at localized portions of an image. Their application of this idea for describing words consists in generating a sequence of such descriptors by moving a sliding window from left to right over the word image. In holistic approaches, similar features (GSC) have been proposed in [76], although these are binary and applied globally to the image. Their method instead, shows significant improvements in unconstrained handwriting after comparing these features with other state-of-the-art methods, both in a DTW-driven approach and in a hidden Markov model-based word spotting system [7].

1.2.2 Line-based word spotting

In the methods presented in the previous section, word spotting is mostly applied after segmenting the text into individual words. In the case though, where a document is segmented into lines only, a number of different approaches take place. A DTW-based system that automatically spots occurrences of a known template word in each line of several pages is examined in [38]. Unlike the sequence matching problem of DTW word-based approaches mentioned so far, this algorithm solves a very expensive subsequence matching problem. Since it does not perform segmentation, the word templates are hand generated. In addition, the technique requires multiple handpicked training samples for each word and thus makes it impractical for automation. For general automatic segmentation-dependent systems, a method is proposed in [9] that also takes the probability of a correct segmentation into account.

A way to alleviate the segmentation problem for word spotting is to use a handwritten text line recognition system. Methods based on handwriting recognition have become fairly popular recently, especially those using Hidden Markov Models (HMM) [60]. For instance, in [64], the local gradient histogram features are tested under such techniques. In addition, a first systematic holistic word recognition approach is applied in [40] for handwritten historical documents. Therein, a document is described using a HMM, where words to be recognized represent hidden states. The state transition probabilities are estimated from word bigram frequencies. A bigram or digram is defined as every sequence of two adjacent elements in a string of tokens, which are typically letters, syllables, or words. They are n-grams for $n = 2$. The frequency distribution of bigrams in a string is commonly used for simple statistical analysis of text in many applications, including computational linguistics, cryptography and speech recognition. Other HMM implementations are examined in [10] for Arabic documents and in [25] for spotting arbitrary words in handwritten text that do not necessarily belong to the training set. In [39], pseudo-2D HMMs were investigated and Edwards et al. [17] proposed generalized HMMs where more than one emission per each hidden state is allowed. Unsupervised adaptation of whole word HMMs to a specific writer was proposed in [66] and in [57] the usage of the Fisher Kernel of the HMM to estimate a good confidence measure was discussed.

Neural Networks (NN) have also been used for keyword spotting in speech with the form of bidirectional long short-term memory (BLSTM) NN [19], [74]. In the later, a discriminative learning procedure that non-linearly maps speech features into an abstract vector space is applied. By incorporating the outputs of a BLSTM network into the speech features, the system is able to make use of past and future context for phoneme predictions which overcomes the drawbacks of generative HMM modeling in speech decoding. Nevertheless, one node in the output layer of the neural network symbolizes one keyword and is triggered when the word occurs in the input data. This makes it inefficient for word spotting in handwritten text as the number of keywords to be spotted are limited, the word has to be known beforehand and the keyword must occur in the training set. Another novel word spotting method is proposed in [28] for handwritten text based on BLSTM Neural Networks. A former implementation using these networks integrated with the so-called CTC Token Passing algorithm for transcribing a portion of handwritten text was initially given in [31]. In [28] a new version of the CTC Token Passing algorithm is proposed and applied for keyword spotting. Unlike the preliminary versions of this system, presented in [26] and [27], the current one provides significant extensions regarding the underlying methodology as well as the experimental evaluation. The system's applicability to both historical data and modern handwriting is demonstrated and an extensive comparison with several reference systems is presented. A common DTW algorithm and a modern HMM-based algorithm seem to be inferior to this approach in terms of performance. Finally, a handwriting recognition system is used for testing the performance of this method by producing an ASCII output on which the keyword search is done.

1.2.3 Document-based word spotting

In the case of completely unsegmented pages of handwritten text, a system can either include a segmentation step as in [5], or follow a segmentation-free approach. In [68], a codebook of shapes is used to create a compressed version of each document. This way a keyword search is done using the stored shape codebook entries. Another segmentation-free algorithm which performs spotting directly on lines of unsegmented text is presented in [3] and it is compared with other segmentation-based techniques. This method performs spotting and segmentation concurrently using a sliding window. Rather than a candidate word image, an entire line image acts as input. The line is split into segments based on an algorithm similar to the ligature-based segmentation algorithm which is used in [37]. All realistic combinations of adjacent connected components are considered as potential areas where the desired word may appear. This approach searches a line thoroughly, looking for a given word image, while keeping the number of evaluations manageable by considering only a small subset of potential regions in the image. The spotting method used to judge the performance of the algorithms presented in [3] is character-based, but the results shown are independent of the actual spotting method used.

A common approach to segmentation-free word spotting is to consider the task as a content-based image retrieval procedure where an input shape represents the word image. This task usually results in a ranked list of word images that are similar to a query word image. The query word image is fitted to the corresponding word images in the document without any segmentation being involved, mostly treating the underlying problem as a template matching. As mentioned in the word-based methods, Leydier et al. [44] use differential features that are compared with a cohesive elastic matching technique, based on zones of interest in order to match only the informative parts of the words. In the same spirit with [44], a segmentation-free word spotting methodology is proposed in [30] which permits a fast and effective retrieval relied on block-based document image descriptors. These descriptors are used at a template matching process satisfying invariance in terms of translation, rotation and scaling. Time expense improvement is also obtained by applying the matching process only in salient regions of the image. Similar type of approaches to [30] are presented in [43] and [53].

1.3 Word query

Before we introduce our word spotting approach in general, we would like to identify two main types of word spotting approaches, depending on how the input is specified:

- Query-by-string approaches
- Query-by-example approaches

Query-by-string approaches like in [10] and [17] are very similar to OCR-based Handwriting Recognition (HWR) systems. Character models are trained in advance and at query time the models of the characters forming the string are concatenated into a word model and the probability of each word image is evaluated. Once trained, these methods allow searching for every possible keyword. However, they present similar drawbacks to HWR systems. Another query-by-string but not OCR-dependent approach is presented in [16], for printed text. It does not require training, rather than determining an alphabet.

In query-by-example approaches as in [2], [38], [50], [61], [73] and [76], the input is an image of the word and the output is a ranked list of word images according to similarity in appearance between the query and the list's images. The search for a probe keyword is subject to having a template image of this keyword available. While the result is based on a distance measure between the query and all candidate word images, no training is involved. Therefore, the performance is limited. In most of the related works that perform well, the datasets contain words from a single or few writers which reduces the variability in handwriting styles. Thus, it remains uncertain how these methods would work in multi-writer conditions. In some cases, the efficiency can be increased by querying multiple times with different images and combining the results. This inspired Rodriguez-Serrano and Perronnin [65] to develop a statistical model that could boost the performance by combining the different queries into a single model. Their idea to overcome the drawbacks of both approaches is to query by "word-class", instead of using string and example queries. After selecting one or multiple examples from the desired keyword and training a probabilistic model for this word, word candidates are detected by evaluating the posterior probability of the candidate given the model.

1.4 Contribution

In the same spirit with [65], we propose a method for learning a prototypical shape of a word class by using only a subset of images belonging to that class. Unlike the method in [65], who made use of hidden Markov models (HMM) as a statistical tool to represent words, we propose a way to build an explicit shape model directly from images, which represents the whole word class. Our approach to the word spotting problem derives from a technique for object detection in real images using contour segment networks. This methodology was first developed in [24] where Ferrari et al. detect an object in real cluttered images. However, instead of training an average shape, they use a hand-drawn example as model. Their idea is to partition test image edges into contour segments and incorporate them into a contour segment network (CSN), which allows in turn an object to be detected by finding paths through the network that resemble the hand-drawn model's outlines. An improvement of the method is presented in [21], regarding the introduction of deformable shape models, learnt from images. Also, an optimization of the CSN, concerning feature extraction, is given in [20]. Finally, the whole object detection system is integrated in [22], where class models are learnt directly from images and novel object instances are localized up to their boundaries in the presence of intra-class variations, clutter and scale changes.

At the heart of our word spotting process lies the word matching algorithm. In order to achieve a matching of high accuracy in unconstrained handwriting, we make use of the technique in [22] and detect a keyword up to its boundaries. To accomplish this, we first extract the boundaries of all training and testing words by applying a thinning morphological operation. This implies that the word images have already been segmented from a handwritten document and therefore renders our approach segmentation-based. Subsequently, we train an average word formed by continuous connected curves (mean shape), from a subset of word images that belong to the same class as well as a statistical model of intra-class deformations. The only prerequisite for this learning stage is for the word to be annotated by a bounding-box. In our case of pre-segmented word images, this is unnecessary as the bounding box is set to be the whole image, thus covering the significant parts of the word. This learning procedure avoids the pairwise matching used in previous approaches and it is computationally cheaper and more robust to clutter edge pixels (edgels) due to the global view gained by considering all training images at once. For real images, clutter is a kind of confusion in determining an object's presence in the image or not, due to the fact that a portion of an object of the real world may be covered by another object or missing. In our case of handwritten words, we consider clutter either as information not relevant to the specific word, for instance segmentation errors, or as parts of the word that do not reoccur among training samples, such as semicolons, full stops and accents which are not deemed useful. Eventually, as intra-class deformations are modeled and enforced at test time, the system is capable of accurately localizing the complete boundaries of previously unseen keyword instances.

Motivated by the work of Ferrari et al. [22], we employ a similar framework to their object detection system, so as to recognize handwritten words, in the presence of both inter-writer and intra-writer variability for a particular word class. Our main contribution is a technique for learning a representative shape of a word class using a random train subset. Moreover, improved feature extraction is achieved by a word image preprocessing step, which allows to capture high distinctiveness among writers and their writing styles. Finally, an integrated learning method is presented where the training subset is no longer randomly or manually selected. This is achieved by applying spectral clustering on each training set which results into groups that specify both the number of models to be used for a word class and which images are responsible for model creation. The models produced by this learning process are also used for word spotting and an extensive experimental evaluation is presented.

The rest of our work is structured as follows: In chapter 2, the theoretical background is analyzed, with respect to the feature extraction from word images and their use for training class-specific shapes. In chapter 3, we show in detail the learning algorithm of an average shape that represents a word class. Chapter 4 includes the word image matching algorithm. Finally, in chapter 5 we evaluate the performance of the word matching technique and we present an extensive experimental evaluation on word spotting as a pattern classification task.

Chapter 2

Word description using local contour features

2.1 Introduction

In the present chapter, we present the theoretical background needed to describe the data preprocessing stage for feature extraction in paragraph 2.2. Both training an average shape (chapter 3) and matching it upon word images (chapter 4) rely on these features. Apart from the local contour features, their properties, which make them attractive for detecting novel word instances in the presence of intra-class variability and scale changes, are also analyzed in paragraph 2.3. Finally, in paragraph 2.4 we show how these features are organized into groups, composing a codebook of feature types useful for efficient matching.

2.2 Word image preprocessing

The keypoint to support boundary-level localization of a word instance in a test image is to build an explicit shape model formed by continuous connected curves, completely covering the word outlines. Therefore, the challenge is to determine which contour points belong to the word-class boundaries, while discarding background and details specific to individual instances, such as the extended parts of calligraphy letters or accents, as it is depicted in figure 2.1. These typically form the majority of points, yielding a poor signal-to-noise ratio.

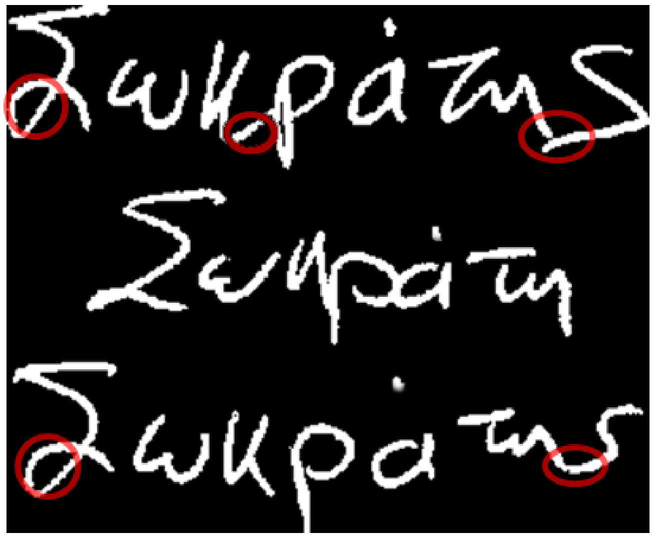


Figure 2.1: Three instances of the Greek word "Σωκράτης" (Socrates in English) written by the same writer. The red areas indicate parts of the word specific to this particular writer which are rarely repeated among instances.

2.2.1 Early processing

The first step to construct the contour features of Ferrari et al. [22] is to extract edgels from a word image using the excellent Berkeley natural boundary detector [52] and to chain them. The resulting edgel-chains are linked at their discontinuities and approximately straight segments are fit to them, using the technique described in [24]. Segments are fit over individual egdel-chains and bridged across their links to form the image representation on which our method relies, the countour segment network (CSN). This brings robustness to the unavoidable broken edgel-chains, as we will present in the following section according to Ferrari et al. [24].

However, to alleviate the errors of the underlying Berkeley edge detector, we extract the skeleton of a word by applying a thinning morphological operation to the word images. Since the data set used in our experiments consists of relatively clean, presegmented words, the skeleton of foreground pixels results into edge-maps which can be used efficiently for further processing. In the following, we describe in detail how the thinning mechanism works.

2.2.2 Thinning

Thinning is a morphological operation that is used to remove selected foreground pixels from binary images, somewhat like erosion or opening. It can be used for several applications, but we particularly use it for skeletonization. In this mode, it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning, is normally only applied to binary images and produces another binary image as output. Like other morphological operators, the behavior of the thinning operation is determined by a structuring element. The binary structuring elements used for thinning are of the extended type described under the hit-or-miss transform (i.e. they can contain both ones and zeros). The thinning of an image I by a structuring element J in terms of the hit-or-miss transform is:

$$\text{thin}(I, J) = I - \text{hit-or-miss}(I, J),$$

where the subtraction is a logical operation defined by

$$X - Y = X \cap \text{NOT } Y.$$

In mathematical morphology, hit-or-miss transform is an operation that detects a given configuration (or pattern) in a binary image, using the morphological erosion operator and a pair of disjoint structuring elements. The result of the hit-or-miss transform is the set of positions, where the first structuring element fits in the foreground of the input image, and the second structuring element misses it completely.

More formally, in binary morphology, an image is viewed as a subset of an Euclidean space $\mathbb{R}^d$ or the integer grid $\mathbb{Z}^d$, for some dimension d. Let us denote this space or grid by E. A structuring element is a simple, pre-defined shape, represented as a binary image, used to probe another binary image, in morphological operations such as erosion, dilation, opening, and closing.

Definition 2.1. Let C and D be two structuring elements satisfying $C \cap D = \varnothing$. The pair (C, D) is sometimes called composite structuring element. The hit-or-miss transform of a given image A by $B = (C, D)$ is given by:

$$A \odot B = (A \ominus C) \cap (A^C \ominus D),$$

where $\ominus$ denotes the morphological erosion operator and A^C is the set complement of A.

That is, a point x in E belongs to the hit-or-miss transform output if C translated to x fits in A, and D translated to x misses A (fits the background of A).

The thinning operation is calculated by translating the origin of the structuring element to each possible pixel position in the image, and at each such position comparing it with the underlying image pixels. If the foreground and background pixels in the structuring element exactly match foreground and background pixels in the image, then the image pixel underneath the origin of the structuring element is set to background (zero). Otherwise, it is left unchanged. Note that the structuring element must always have a one or a blank at its origin if it is to have any effect. So far, the effects of a single pass of a thinning operation over the image are described. In fact, the operator is normally applied repeatedly until it causes no further changes to the image, or else, until convergence. Alternatively, in some applications such as pruning, the operations may only be applied for a limited number of iterations.

The choice of structuring element determines under what situations a foreground pixel will be set to background, and hence it determines the application for the thinning operation. In our case, we would like to reduce the thresholded output of an edge detector, for instance our presegmented words, to lines of a single pixel thickness, while preserving the full length of those lines (i.e. pixels at the extreme ends of lines should not be affected). This is done by the procedure shown in algorithm 1.

Algorithm 1 Skeletonization by morphological thinning

```
1 Consider all pixels on the boundaries of foreground regions.
2 Delete any such point that has more than one foreground neighbor, as long as doing so
      does not locally disconnect (i.e. split into two) the region containing that pixel.
3 Iterate until convergence.
```

This procedure erodes away the boundaries of foreground objects as much as possible, but does not affect pixels at the ends of lines. In other words, pixels from a binary image are removed, so that an object without holes shrinks to a minimally connected stroke and an object with holes shrinks to a ring, halfway between the hold and outer boundary. Two out of the eight structuring elements we use to achieve this effect are illustrated in figure 2.2. The rest elements result by applying the remaining six 90° rotations to each one of them. The extracted connected skeleton from a binary image, as a result of this thinning operation, is shown in figure 2.3.

0	0	0
	1	
1	1	1

	0	0
1	1	0
	1	

Figure 2.2: Structuring elements for skeletonization by morphological thinning. At each iteration, the image is first thinned by the structuring element on the left, and then by the structuring element on the right, and then with the remaining six 90° rotated versions of the two elements. The process is repeated in cyclic way until thinning produces no further change. As usual, the origin of the structuring element is at the center.

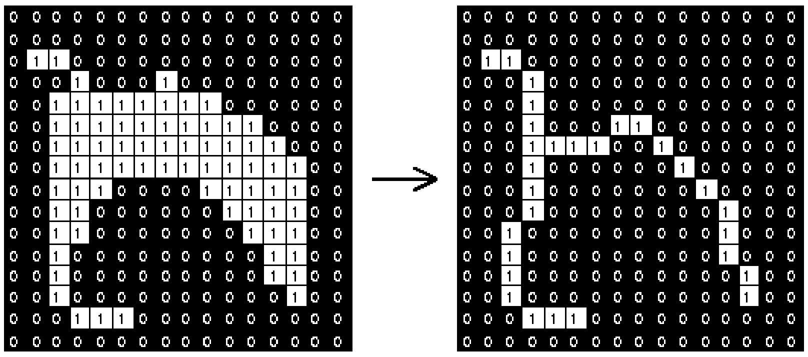


Figure 2.3: Skeletonization example by morphological thinning of a simple binary shape, using the structuring elements of figure 2.2.

2.3 Feature extraction

The idea of relying on the object detection system of Ferrari et al. [22], in order to spot handwritten words, is influenced by the fact that their system addresses several challenges:

- The image edges are not reliably extracted from word images in the segmentation step.
- The contour of a desired word may be fragmented over several pieces and sometimes parts are missing.
- Locally, edges lack specificity and can be recognized only when put in the wider context of the whole shape (word) [4].
- A deformable template word (mean shape) is required to handle variations among instances within a word-class.

Before introducing the local features used in our approach, we analyze the contour segment network [24] on which they are detected.

2.3.1 Linking edgel chains

After extracting the skeleton from a binary presegmented image using the aforementioned thinning operation, the edgels comprising the skeleton are chained and a smoothing spline curve is fit to each edgel-chain, providing estimates of the edgel's tangent orientations. Since a contour may be broken into several edgel-chains, or it might have branchings which are not captured by simple edgel-chaining, we link edgel-chains to counter these issues with the following criterion:

Criterion 2.1. *An edgel-chain c_1 is linked to an edgel-chain c_2 if any edgel of c_2 lies within a search area near an endpoint of c_1 as it is illustrated in figure 2.4. The search area is an isosceles trapezium. The minor base rests on the endpoint of c_1 and is perpendicular to the curve's tangent orientation, while the height points away from c_1.*

This criterion links c_1 to edgel-chains lying in front of one of its endpoints, thereby indicating that it could continue over c_2. The trapezium shape expresses that the uncertainty about the continuation of c_1's location grows with the distance from the breakpoint. Note how c_1 can link either to an endpoint of c_2, or to an interior edgel. The latter allows to properly deal with T-junctions, as it records that the curve could continue in two directions (figure 2.4b). Besides, it is pointed out that it is not necessary for the end of c_1 to be oriented like the bit of c_2 it links to, as in figure 2.4(b). Tangent-discontinuous links are also possible (figure 2.4(c)).

These edgel-chain links are the backbone structure on which the contour segment network will be built. To obtain the elements composing the network, namely, the *contour segments*, each edgel chain is partitioned into roughly straight segments. In addition to these regular segments, we also construct segments bridging over tangent-continuous links between edgel-chains, as it is shown in figure 2.4(d). The idea is to bridge the breaks in the edges, thus recovering useful segments missed due to the breaks.

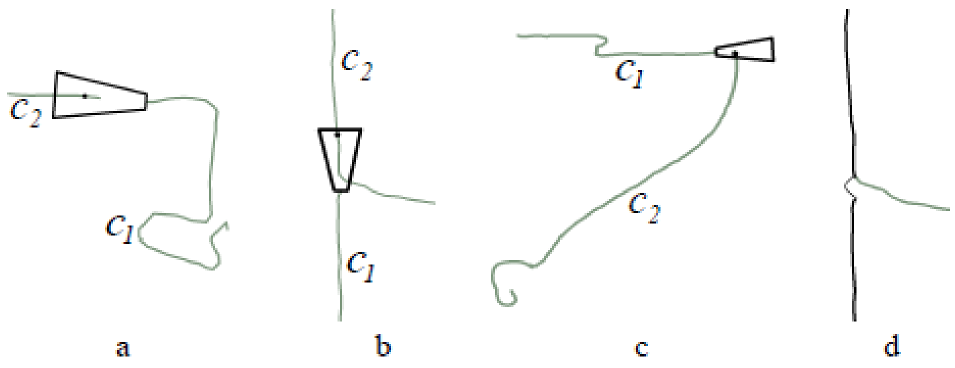


Figure 2.4: (a-c) Example links between edgel-chains. (a) Endpoint-to-endpoint link. (b) Tangent-continuous T-junction link. (c) Tangent-discontinuous link. (d) A segment (marked with an arc) bridging over link b). Figure reproduced from [24].

2.3.2 Contour segment network

Before explaining how to build the CSN, a few definitions are provided in line with Ferrari et al. [24].

- Every segment is directed, in that it has a back and a front. This only serves to differentiate the two endpoints, they have no semantic difference. As a convention, the front of a segment is followed by the back of the next segment on the edgel-chain.

- every edgel-chain link is directed as well, meaning that the edgel-chain c_1, on which the trapezium search-area rests, is at the back, while the other edgel-chain c_2 is at the front. This also defines the front and back endpoints of a segment bridging between two edgel-chains.

- For clarity, we use the word links between edgel-chains, and connections between segments.

The network is built by applying the following six rules, as it is presented in figure 2.5. These rules connect the front of each segment to a set of segments, and its back to another set of segments. Therefore, the network structure is unconstrained and its complexity adapts to the image content.

1. The front of a segment is connected to the back of the next segment on the same edgel-chain.

2. When two edgel-chains c_1, c_2 are linked at endpoints, the segment of c_1 before the link is connected to the segment of c_2 after the link.

3. Consider a T-junction link (i. e. from an endpoint of c_1 to the interior of c_2). The segment of c_1 before the link is connected to the two segments of c_2 with the closest endpoints. As can be seen in figure 2.53, this records that the contour continues in both directions.

4. Let s be a segment bridging over a link from c_1 to c_2. The segment s is connected to the segment of c_2 coming after its front endpoint, and to the segment of c_1 coming before its back endpoint.

5. Two bridging segments which have consecutive endpoints on the same edgel-chain are connected. Here, "consecutive" means that no other segment lies in between.

6. Consider a bridging segment s without front connection, because it covers the front edgel-chain c_2 until its end. If c_2 is linked to another edgel-chain c_3, then s is connected to the segment of c_3 coming after its front endpoint. A respective rule applies if s lacks the back connection.

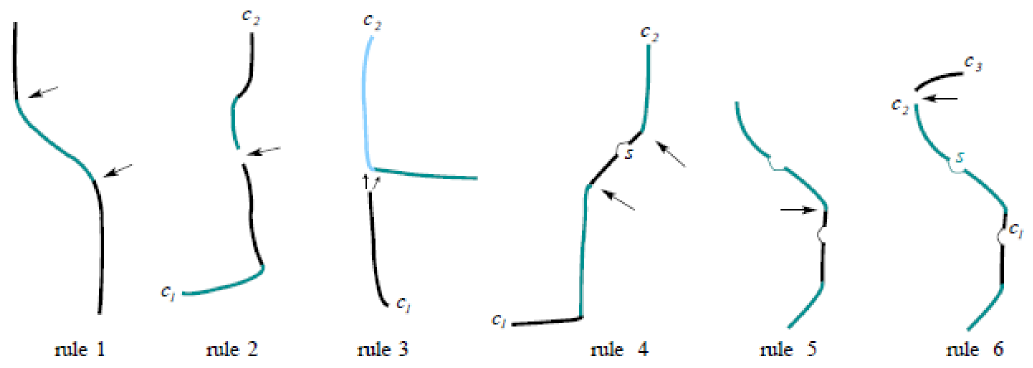


Figure 2.5: The six rules used in order to build the Contour Segment Network. They connect (arrows) regular segments and bridging segments (marked with an arc). Rules 2-6 connect segments over different edgel-chains c_i. Figure reproduced from [24].

The above rules naturally connect two segments if the edges provide evidence that they could be connected on an ideal edge-map, where all edges would be detected and perfectly chained. Moreover, it is interesting to notice that the last three rules, dedicated to bridging segments, create connections analog to those made by the first three rules for regular segments. As a consequence, both types are treated consistently.

Since each edgel-chain is typically linked to several others, these rules generate a complex branching structure, a network of connected segments. The systematic connections across different edgel-chains, together with the proper integration of bridging segments, make the network robust to incomplete or broken edgel-chains. This is unnecessary for the data set used in our experiments, as edge-maps resulting by applying thinning are sufficiently connected. However, the connectivity between segments provided by the CSN, allows an efficient searching for paths through the network that resemble the model outlines, even in poorly segmented word images, such as handwritten historical document images.

In the following, we summarize several advantages provided by operating on the contour segment network, which motivate the local features proposed:

- Even when most of the image does not contain frequently repeated segments of the word, among instances, only a limited number of segments is connected to a path corresponding to a model outline. This greatly limits the choices to be made by the matching step, thus making the computational complexity linear in the number of test image segments.

- By connecting segments over edge discontinuities, the system is robust to interruptions along the word contours and to short missing parts.

- The CSN includes paths going along the contour of the desired word.

2.3.3 Feature Description

The features that are used in our approach, belong to a family of local shape features formed by chains of k connected roughly straight contour segments (kAS), or else paths of length k through the CSN [20]. For several values of k, kAS can form various local shape structures:

- Individual segments for $k = 1$,

- L, S shapes, 2-segment T, S, "η" and other shapes for $k = 2$, as it illustrated in figure 2.7.

- C, Y, F, Z shapes, 3-segment T shapes and triangles for $k = 3$.

As k increases, features increase in complexity. On one hand, they become more and more informative, whereas on the other hand, they gradually get less and less repeatable across word instances. This means that the number of features covering partly boundary and partly clutter (i.e. parts of a word rarely repeated among instances) also grows with k, actually faster than pure boundary ones, leaving a lower signal-to-noise ratio. Therefore, for rather low values of k, kAS have an attractive intermediate complexity, offering a convenient compromise:

- Simple enough to be detected repeatedly,
- yet complex enough to capture informative local word structures.

Depending on these intuitions, the local features we use are pairs of connected segments (2AS). Each pair of connected segments forms one feature, called a PAS, for *pair of adjacent segments*. A PAS feature $P = (x, y, s, e, d)$ has:

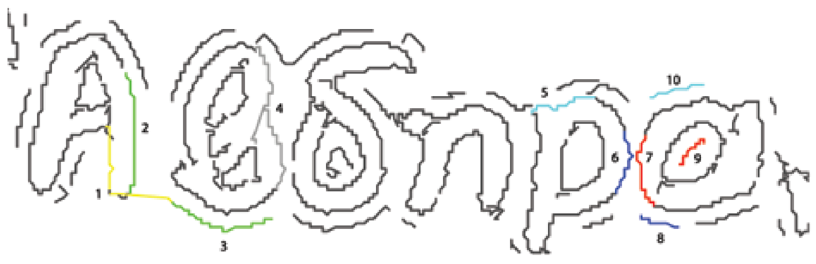
1. a location (x, y) which consists of the mean over the two segment centers,
2. a scale s which is the distance between the segment centers,
3. a strength e as the average edge detector confidence over the edgels, with values in [0, 1] (in our case of thinned binary word images, $e = 1$) and
4. a descriptor $d = (\theta_1, \theta_2, l_1, l_2, r)$, invariant to translation and scale changes.

The descriptor encodes the shape of the PAS, by the segments' orientations θ_1, θ_2 and lengths l_1, l_2 and the relative location vector r, going from the center of the first segment to the center of the second (a stable way to derive the order of the segments in a PAS is given in [20]. It is also interesting to notice that the PAS descriptor is of different nature than conventional local textured feature descriptors. Although this is irrelevant for handwritten words, where texture cues do not provide significant information, the PAS descriptor encodes the geometric properties of the segments (orientation and length) and of their spatial arrangement, due to the location vector r.

A number of example PAS features are illustrated in figure 2.6, both for an edge-map extracted from Berkeley's boundary detection algorithm and in the case of thinned word images. First, these examples show that skeletonization by thinning has a positive impact on a word image before detecting PAS features. The image noise levels decrease, as the number of clutter PAS becomes smaller and PAS stemming from thinned images cover mainly informative parts. This reduces the computational complexity of detecting them.

Moreover, as we can see in figure 2.6, some PAS features, either derive from segments which are adjacent on the same edgel-chain (PAS 2, 3, 5-10 in figure 2.6(a) and PAS 2-8 in figure 2.6(b)), or they consist of one segment at the end of an edgel-chain directed towards another (PAS 1, 4 in figure 2.6(a) and PAS 1 in figure 2.6(b)). As two segments from a pair are not limited to come from a single edgel-chain, but may come from adjacent edgel-chains, the extraction of pairs is robust to the typical errors of the underlying edge detector.

Below we summarize some attractive properties of PAS features:


(a)

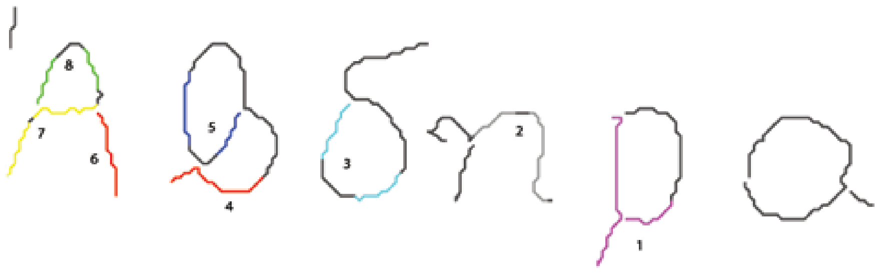

(b)

Figure 2.6: Examples of PAS features. (a) Ten PAS for the edge-map resulted after Berkeley's edge detection [52], (b) Eight PAS detected on the same word image after thinning.

- Both lengths and relative location are normalized by the scale of the PAS, making the descriptor and successively, the PAS feature, scale invariant.
- PAS can overlap, meaning that two different PAS can share a common segment. This way, the computational complexity of detecting them diminishes.
- They are robustly detected, as they connect segments even across gaps between edgel chains.
- Since both PAS and their descriptors cover solely the two segments, the can cover pure portion of an object boundary, without including clutter edges which often lie in the vicinity.
- Their descriptors respect the nature of boundary fragments, to be one-dimensional elements embedded in a 2D image, as opposed to local appearance features, whose extent is a 2D patch.
- A correspondence between two PAS induces a translation and scale change. Therefore, they can re readily used within a Hough-style voting scheme, not only for object detection but also for word recognition.

2.4 Feauture similarities - codebook

A flexible measure to accommodate intra-class variability, initially proposed in [20] for kAS, is the PAS dissimilarity measure. The dissimilarity $D(P, Q)$ between the descriptors d^p, d^q of two PAS P, Q is defined by:

$$D(d^p, d^q) = w_r \|r^p - r^q\| + w_\theta \sum_{i=1}^{2} (D_\theta(\theta_i^p, \theta_i^q)) + \sum_{i=1}^{2} (|\log(l_i^p/l_i^q)|) \tag{2.1}$$

where the first term is the difference in the relative locations of the segments, $D_\theta \in [0, \pi/2]$ measures the difference between segment orientations, and the last term accounts for the difference in lengths. As segment lengths are often inaccurate, higher weight is given to the two other terms of the dissimilarity measure. In all our experiments, the weights w_r, w_θ are fixed to the same values in accordance with Ferrari et al. [20] ($w_r = 4$, $w_\theta = 2$).

Finally, following the bag of features paradigm [14], [34], we construct a codebook of PAS types, or a "visual vocabulary", each capturing a different kind of local shape structure, such as the L structures or small T-junctions mentioned before. The codebook is created by clustering the PAS inside the training bounding boxes, in our case, the whole word images, according to their descriptors. Apart from revealing the frequency at which feature types occur, the codebook is convenient because it allows to avoid explicitly comparing every test image features to every feature from the training images. Instead, comparison to much fewer feature types suffice.

For clustering, we use the following clique-partitioning (CP) approach:

- Let G be a complete graph whose nodes are the training PAS and arcs are weighted by $d - D(d^p, d^q)$. We partition G into cliques in order to maximize the sum of intra-clique weights, using the CP approximation algorithm [23].
- Each resulting clique is a cluster of similar PAS.

The choice of CP instead of K-means, commonly used for building visual codebooks, is appropriate in our context where the dissimilarity measure D makes the descriptor space circular (D_θ terms). Moreover, the parameter d is relatively easy to set, because it represents a rough indication of the acceptable intra-cluster dissimilarity (akin to the kernel-width in mean-shift clustering [34]). K-means instead requires the number of clusters as input, which is unknown a priori and varies from data set to data set. Experimental results in [20] indicate that the exact choice of d has little impact on the overall system performance.

For each cluster, the centermost PAS, minimizing the sum of dissimilarities to all the others is selected as a representative. The codebook $C = t_i$ is the collection of the descriptors of these centermost PAS, the PAS types t_i, a number of which are illustrated in figure 2.7. A codebook is useful for efficient matching, since all features similar to a type are considered in correspondence. The codebook is class-specific and built form the same images used later to train the average word.

The whole procedure for describing a word by a set of PAS features is summarized in algorithm 2:

Algorithm 2 Word description using PAS

```
1 Extract the skeleton of a presegmented word by applying thinning.
2 Chain the skeleton's edgels into edgel-chains and link these edgel-chains at their discontinuities.
3 Fit roughly straight segments to the edgel-chains and connect them along the edges and across their links so as to form the contour segment network (CSN).
4 Derive PAS features by detecting paths of length 2 (PAS = $k$AS, for $k = 2$) through the CSN.
5 Define a descriptor $d^p$ for a PAS $P$, essential for its reconstruction, as well as a dissimilarity $D(P, Q)$ between the descriptors $d^p$, $d^q$ of two PAS $P, Q$.
6 Cluster a set of training PAS according to their descriptors and determine a representative PAS for each cluster, as the one minimizing the sum of dissimilarities to all other PAS in the cluster.
7 Build a class-specific codebook of PAS types, composed of the descriptors of each cluster's representative PAS.
```

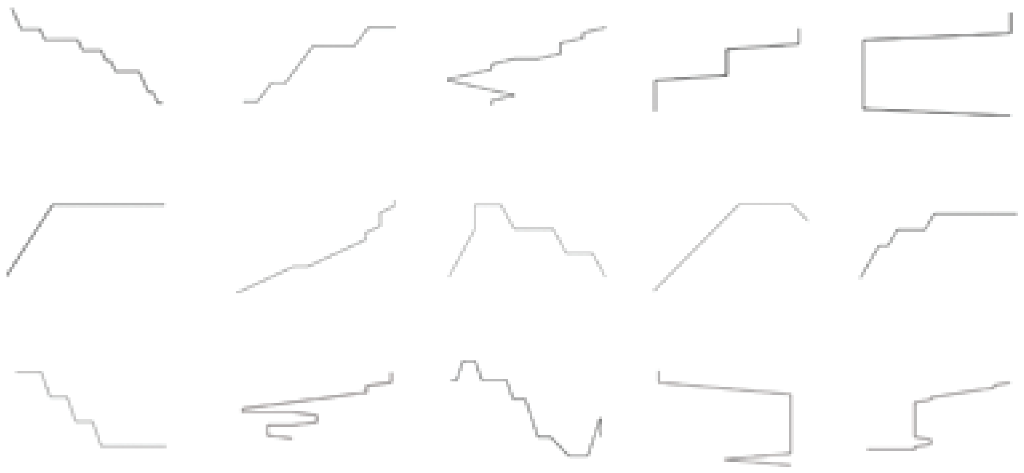


Figure 2.7: The 15 most frequent PAS types from 38 thinned instances of the word "Σωκράτης" (Socrates in English) used to train the average word.

Chapter 3

Constructing the Word Model

3.1 Introduction

In the present chapter, the learning method of a prototype shape for a word class is analyzed in paragraph 3.2. Its principal intra-class deformation modes are also presented, given image windows W containing example word instances. Since the data set used in our experiments consists of segmented words, W is set to be the whole word image.

3.2 Learning Algorithm

The challenge in training a mean shape is to discover which contour points belong to the common class boundaries and to put them in full point-to-point correspondence across the training examples. For instance, the basic outline of the word "Σωκράτης" (Socrates in English) is more desirable than edges that differentiate characters and words in general, among instances (figure 2.1). The technique for building such a shape is composed of four stages, as it is illustrated in figure 3.1(b-e):

1. Determine model parts as PAS frequently reoccurring with similar locations, scales and shapes (3.2.1).

2. Assemble an initial shape by selecting a particular PAS for each model part from the training examples (3.2.2).

3. Refine the initial shape by iteratively matching it back to the training images (3.2.3).

4. Learn a statistical model of intra-class deformations from the corresponded shape instances produced by stage 3 (3.2.4).

The shape model output at the end of this procedure is composed of a prototype shape S, which is a set of points in the image plane and a small number of n intra-class deformation modes $E_{1:n}$, so that new class members can be described as $S + E_{1:n}$.

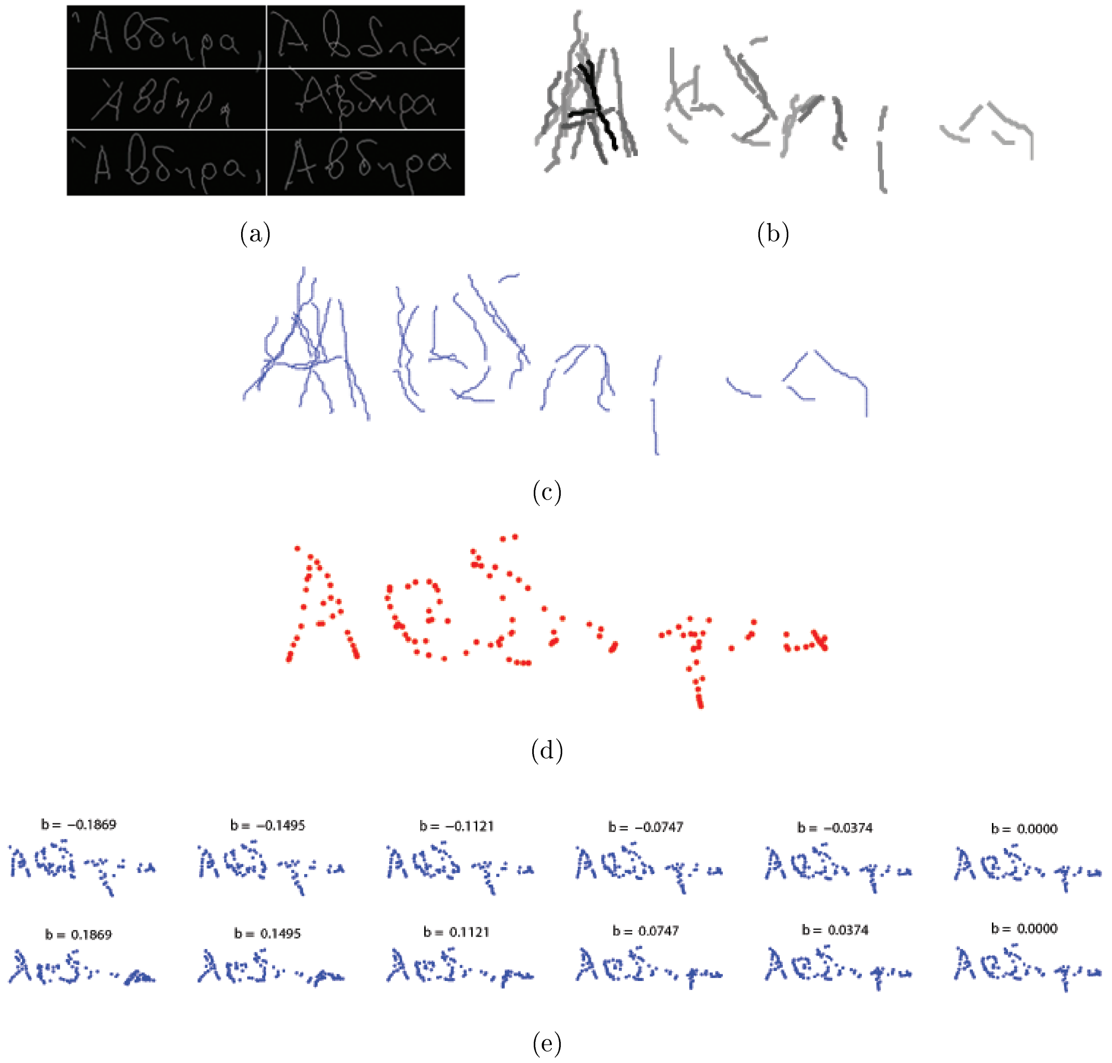


Figure 3.1: Learning the shape model. (a) Six training examples (out of a total 38). (b) Collection of parts (COP) model. (c) Occurrences selected to form the initial shape. (d) Refined shape. (e) First two modes of variation (mean shape for $b = 0$).

3.2.1 Finding model parts

The first stage towards learning the model shape is to determine which PAS lie on boundaries, common across the word class, as opposed to those that lie on the background clutter (i.e. segmentation errors) and those on details specific to individual training instances. The basic idea is that a PAS belonging to the class boundaries will recur consistently across several training instances with a similar location, size, and shape. Although they are numerous, PAS not belonging to the class boundaries are not correlated across different examples. In the following, we refer to any PAS or edgel not lying on the class boundaries as clutter.

The procedure for finding which PAS belong to the model's outlines consists of three steps:

1. *Windows alignment.* Let a be the geometric mean of the aspect-ratios of the training windows W (width over height). Each window is transformed to a canonical zero-centered rectangle of height 1 and width a. This removes translation and scale differences and cancels out word variations due to different aspect-ratios. Therefore, the learning task is reinforced, as PAS on the class boundaries are now better aligned.

2. *Voting for model parts.* Let V_i be a voting space associated with a PAS type t_i. There are $|C|$ such voting spaces, all initially empty. Each voting space has three dimensions:

   - two for location (x, y) and
   - one for size s.

   Every PAS $P = (x, y, s, d)$ from every training window casts votes as follows:

   (a) P is soft-assigned to all types T within a dissimilarity threshold γ:

   $$T = \{t_j | D(d, t_j) < \gamma\},$$

   where d is the shape descriptor of P.

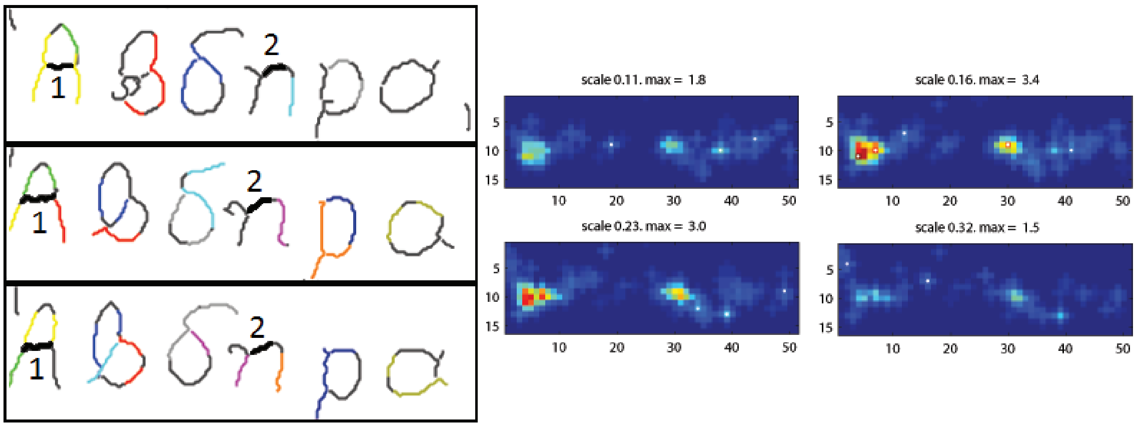
   (b) For each assigned type $t_j \in T$, a vote is casted in V_j at (x, y, s), namely, at the location and size of P. The vote is weighted by $e \cdot (1 - D(d, t_j)/\gamma$, where e is the edge strength of P.

   Assigning P to multiple types T and weighting votes according to the similarity $1 - D(d, t_j)/\gamma$ reduce the sensitivity to the exact shape of P and the exact codebook types. Also, weighting by edge strength allows to take into account the relevance of the PAS, by means of how important it may be as a model part. Although this might lead to better results over treating edgels as binary features as noticed in [15] and [24], the thinned word images we use in our experiments are binary images and thus $e = 1$. Essentially, each PAS votes for the existence of a part of the class boundary with shape, location and size like its own. This is the best it can do from its limited local perspective.

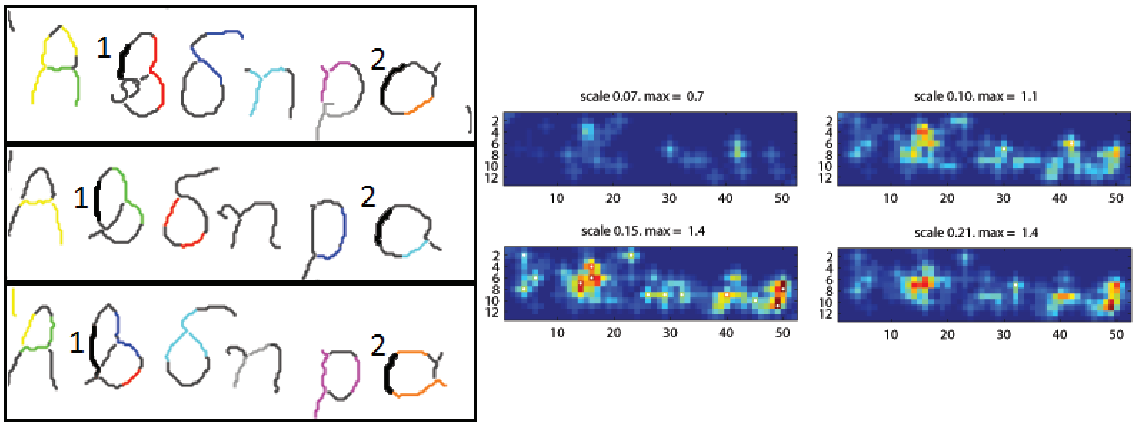
3. *Detecting local maxima.* All voting spaces are searched for local maxima. Each local maximum yields a *model part* $M = (x, y, s, v, d)$, with a specific location (x, y), size s and shape $d = t_i$ (the PAS type corresponding to the voting space where M was found). The value v of the local maximum measures the confidence that the

part belongs to the class boundaries. The (x, y, s) coordinates are relative to the canonical window.

As it illustrated in figure 3.2, recurring PAS among training instances of the same PAS type, tend to form peaks in the voting space at different locations. This occurs either independently of size variations (3.2(a)), or due to scale changes (3.2(b)). Consequently, the method allows for different models parts with the same PAS type.



(a)



(b)

Figure 3.2: Finding model parts. Left: three training instances with two recurring PAS of the same PAS type (parts of 'A' and 'η' in (a) and parts of 'β' and 'α' in (b)). Right: four slices of the accumulator space for this PAS type (each slice corresponds to a different size). (a) The two recurring PAS form peaks at different locations regardless of the scale. (b) The two recurring PAS form peaks at different locations and sizes.

The success of this procedure is partly attributed to adopting PAS as basic shape elements. A simpler alternative would be to use individual edgels. In that case, there would be just one voting space, with two location dimensions and one orientation dimension. In contrast, PAS bring two additional degrees of separation:

- the shape of the PAS, expressed as the assignments to codebook types and
- its size (relative to the window).

Individual edgels have no size and the shape of a PAS is more distinctive than the orientation of an edgel. As a consequence, it is very unlikely that a significant number of clutter PAS will accidentally have similar locations, sizes and shapes at the same time. Hence, recurring PAS stemming from the desired class boundaries tend to form peaks in the voting spaces, whereas clutter PAS do not.

Intra-class shape variability is addressed by two significant factors:

- the soft-assign of PAS to types and
- a substantial spatial smoothing to the voting spaces before detecting local maxima.

This creates wide basins of attraction for PAS from different training examples to accumulate evidence for the same part. We can afford this flexibility while keeping a low risk of accumulating clutter, because of the high separability provided by separate voting spaces for different codebook types. This yields the distinctiveness necessary to overcome the poor signal-to-noise ratio, while allowing the flexibility essential to accommodate for intra-class shape variations.

The Hough-style voting procedure is similar in spirit to recent works on finding frequently recurring spatial configurations of local appearance features in unannotated images [29], [59], but it is specialized for the case when bounding-box annotation is available.

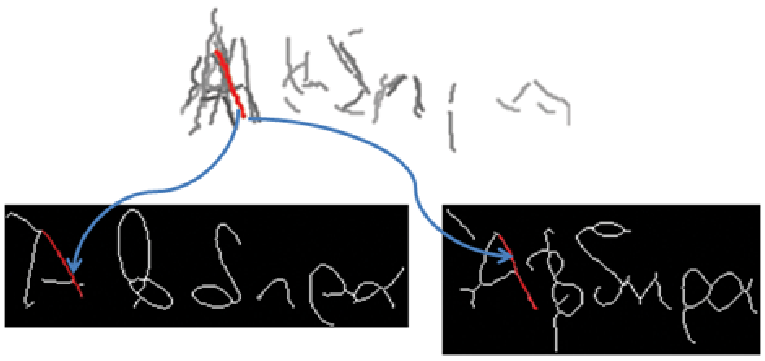
Finally, as the proposed methodology sees all training data at once, it reliably selects parts and robustly estimates their locations, sizes and shapes. As another benefit, the complexity of the whole voting process is linear in the total number of PAS in the training windows and therefore can learn from large sets efficiently.

3.2.2 Assembling the initial model shape

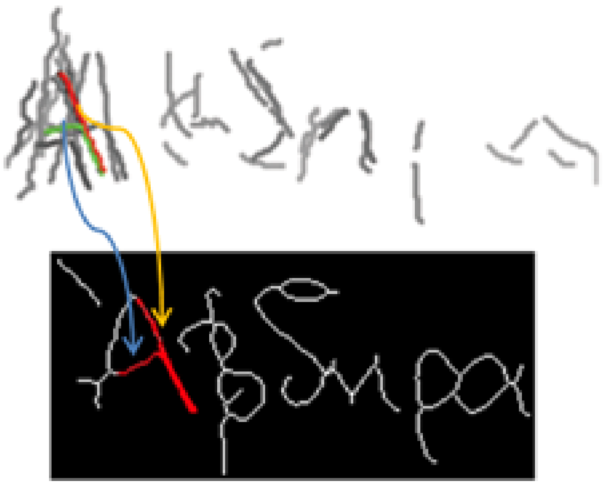
The collection of parts learned so far captures class boundaries well and delivers a sense of the general shape of the word class (figure 3.1(b)). Most of the basic structure of a word is included while details and background clutter is excluded. Relying on this collection-of-parts (COP) model, one could attempt to detect a word instance in a test image, by matching parts based on their descriptor and enforcing their spatial relationship. Earlier approaches based on appearance features [18], [41] and on contour features [55], [70] are able to localize such instances up to a bounding box which contains the word of interest.

However, this would be impractical for a word spotting task as the accuracy of the matching outcome is far from sufficient, so as to classify a word into classes. The COP model has no notion of shape at the global scale. It is a loose collection of parts learnt rather independently, each focusing on its own local scale. In order to support localizing word instances up to their boundaries, accurately and completely on novel test images, a more globally consistent shape is needed. Ideally, its parts would be connected into a whole shape featuring smooth, continuous lines.

The key idea for constructing such a shape lies on the fact that a model part occurs several times on different images, as it is depicted in figure 3.3(a). These occurrences offer slightly different alternatives for the part's location, size and shape. We can assemble variants of the model shape by selecting different occurrences for each part. The basic concept for obtaining a globally consistent shape is to pick one occurrence for each part so as to form larger aggregates of connected occurrences (figure 3.1(b)). The shape assembly task is casted as a search for the assignment of parts to occurrences leading to the best connected shape and the process is explained in detail in three steps:



(a)



(b)

Figure 3.3: Occurrences and connectedness. (a) A model part (above) and two of its occurrences (below). (b) Two model parts with high connectedness (above) and two of their occurrences which share a common segment (below).

1. *Computing occurrences.* A PAS $P = (x^p, y^p, s^p, d^p)$ is an occurrence of model part $M = (x^m, y^m, s^m, v^m, d^m)$ if they have similar location, scale and shape (figure 3.3(a)). The following function measures the confidence that P is an occurrence of M (denoted $M \to P$):

$$\text{conf}(M \to P) = e^p \cdot D(d^m, d^p) \cdot \min\left(\frac{s^m}{s^p}, \frac{s^p}{s^m}\right) \times \exp^{\left(-\frac{1}{2\sigma^2}\left((x^p - x^m)^2 + (y^p - y^m)^2\right)\right)} \quad (3.1)$$

It takes into account P's edge strength (first factor) and how close it is to M in terms of shape, scale and location (second to last factors). The confidence ranges in [0,1] and P is deemed an occurrence of M if $\text{conf}(M \rightarrow P) > \delta$, with δ being a threshold. By analogy $M_i \rightarrow P_i$ denotes the occurrence of model segment M_i in image segment P_i (with $i \in \{1, 2\}$).

2. *Computing connectedness.* As a PAS P is formed by two segments P_1, P_2, two occurrences P, Q of different model parts M, N might share a segment, as it shown in figure 3.3(b). This suggests that M, N explain connected portions of the class boundaries and thus they should be connected in the model. As model parts occur in several images, we estimate how likely it is for two parts to be connected in the model, by how frequently their occurrences share segments.

   Let the *equivalence* of segments M_i, N_j be

   $$\text{eq}(M_i, N_j) = \sum_{\{P,Q|\, s\in P,\, s\in Q,\, M_i\rightarrow s,\, N_j\rightarrow s\}} (\text{conf}(M \rightarrow P) + \text{conf}(N \rightarrow Q)) \qquad (3.2)$$

   The summation runs over all pairs of PAS P, Q sharing a segment s, where s is an occurrence of both M_i and N_j (figure 3.3(b)). Let the *connectedness* of M,N be the combined equivalence of their segments (for the best of the two possible segment matchings):

   $$\text{conn}(M, N) = \max(\text{eq}(M_1, N_1) + \text{eq}(M_2, N_2),\ \text{eq}(M_1, N_2) + \text{eq}(M_2, N_1)) \qquad (3.3)$$

   Two parts have high connectedness if their occurrences frequently share a segment. Two parts sharing both segments have even higher connectedness, suggesting that they explain the same portion of the class boundaries.

3. *Assigning parts to occurrences.* Let $\mathcal{A}(M) = P$ be a function assigning a PAS P to each model part M. The problem is to find the mapping $\mathcal{A}$ that maximizes the objective function:

   $$\sum_M \text{conf}(M \rightarrow \mathcal{A}(M)) + \alpha \sum_{M,N} \text{conn}(M, N) \cdot \mathbf{1}(\mathcal{A}(M), \mathcal{A}(N)) - \beta K \qquad (3.4)$$

   where $\mathbf{1}(a, b) = 1$ if occurrences a, b come from the same image, and 0 otherwise; K is the number of images contributing occurrences to $\mathcal{A}$; α, β are predefined weights. The first term prefers high confidence occurrences. The second favors assigning connected parts to connected occurrences, because occurrences of parts with high connectedness are likely to be connected when they come from the same image (by construction of function (3.3). The last term encourages selecting occurrences form a few images, as occurrences from the same image fit together naturally. Overall, function (3.4) encourages the formation of aggregates of good confidence and properly connected occurrences.

Algorithm 3 Assignment of model parts to occurrences

```
1 Assign the model part with the single most confident occurrence.
2 Consider the part most connected to those assigned so far and assign it to the occurrence
      maximizing (3.4).
3 Iterate until all parts are assigned to an occurrence.
```

Optimizing (3.4) exactly is expensive, as the space of all assignments is huge. In practice, an approximation which brings satisfactory results is described in algorithm 3. This way a well connected shape is built, where most segments fit together and form continuous lines. The remaining discontinuities are smoothed out by the refinement procedure in the next section.

3.2.3 Model shape refinement

The key idea to refine the initial model shape learnt in section 3.2.2 is to match it back onto the training image windows W, by applying the deformable matching algorithm of Chui and Rangarajan [12] (figure 3.4(b)). This results in a backmatched shape for each window, as it is presented in figure 3.4(c)-top for several training images. An improved shape is obtained by averaging these backmatched shapes as it is shown in figure 3.4(c)-bottom. The process is then iterated by alternating backmatching and averaging (figure 3.4(d-e)). The whole procedure is described in the following four steps.

1. *Sampling.* Sample 100 equally spaced points from the initial model shape, generating the point set S (figure 3.4(a)).
2. *Backmatching.* Match S back to each training window $w \in W$ by:
   (a) *Alignment.* Translate, scale and stretch S so that its bounding-box (whole image) aligns with w (figure 3.4(b)-left). This provides the initialization for the shape matcher.
   (b) *Shape matching.* Let E be the point set consisting of the edgels inside w. Put S and E in point-to-point correspondences using the non-rigid robust point matcher TPS-RPM (Thin-Plate Spline Robust Point Matcher) described in [12]. This estimates a TPS transformation from S to E, while rejecting edgels not corresponding to any point of S. This is important, as only some edgels lie on the object boundaries. The TPS-RPM method is analyzed in detail in chapter 4 where it is reused to localize word instances up to their boundaries.
3. *Averaging.* Extract an average shape model at each backmatching step by applying the following:

   - Align the backmatched shapes $\mathcal{B} = \{B_i\}_{i=1,\ldots,|W|}$ using Cootes' variant of Procrustes analysis [13], by translating, scaling and rotating each shape so that the total sum of distances to the mean shape $\bar{B}$ is minimized: $\sum_{B \in \mathcal{B}} |B_i - \bar{B}|^2$.
   - Update S by setting it to the mean shape: $S \leftarrow \bar{B}$ (figure 3.4(c)-bottom).

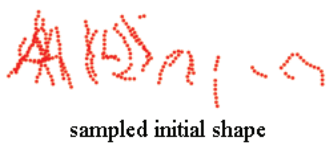
4. Iterate to step 2, using the updated model shape S (in our experiments, steps 2 and 3 are repeated two to three times).

Step 3 is possible because the backmatched shapes $\mathcal{B}$ are in point-to-point correspondence, as they are different TPS transformations of the same set S (figure 3.4(c)-top). This enables to define $\bar{B}$ as the coordinates of corresponding points averaged over all $B_i \in \mathcal{B}$. It also enables to analyze the variations in the point locations. The differences remaining after alignment are due to non-rigid shape variations, which will be learnt in section (3.2.4).

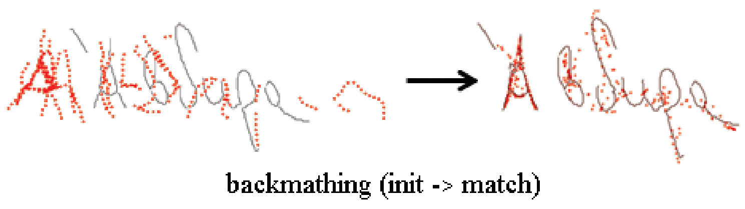
The alternation of backmatching and averaging results in a succession of better models and better matches to the data, as the point correspondence cover more and more of the class boundaries of the training instances (figure 3.4(d-e)). Segments of the model shape are moved, bent and stretched so as to form smooth, connected lines, thus recovering the shape of the word-class well on a global scale. This is due to backmatching, which deforms the initial shape onto the class boundaries of the training images, delivering natural, well formed shapes. The averaging step then integrates them into a generic-looking shape and smoothes out occasional inaccuracies of the individual backmatches (e.g. accents, commas and calligraphy parts of letters).

The proposed technique can be seen as searching for the model shape that best explains the training data, under the general assumption that TPS deformations account for the difference between the model and the class boundaries of the training words. As it is illustrated in figure 3.4(e)-bottom, the running example improves further during the third (and last) iteration, where the average shape is less noisy and more specific to the class than that in previous iterations. The backmatched shapes also improve in the third iteration, because matching is easier given a better model, providing in turn a better average shape. This mutual help between backmatching and updating the model is the key for the success of the procedure.

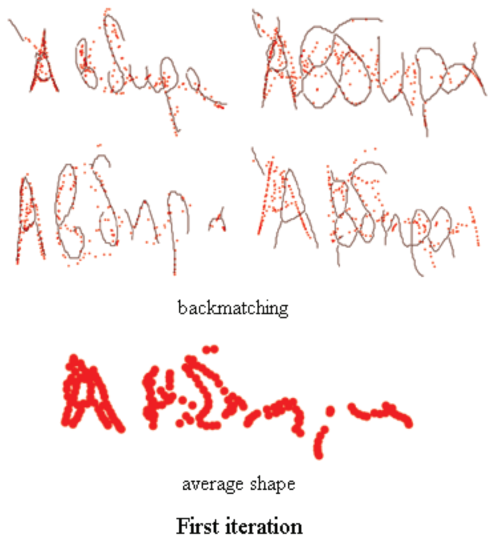
Finally, examples of other models evolving over the three stages of the learning process are depicted in figure 3.5. As it seems, model shape refinement has a large positive impact. Also, the number of different instances per writer for a specific word class is of great importance. This is explained by the refined shape shown in the last row of figure 3.5 for the word "αρετή" (virtue in English), which is more colorful (the figure is better seen in color) than the other two words. More specifically, the word class "αρετή" contains only one instance per writer, as opposed to the other two classes which include three instances per writer. Hence, the variability to be captured for this word is smaller, thus yielding more accurate model parts and occurrences, which result into a better colored shape.



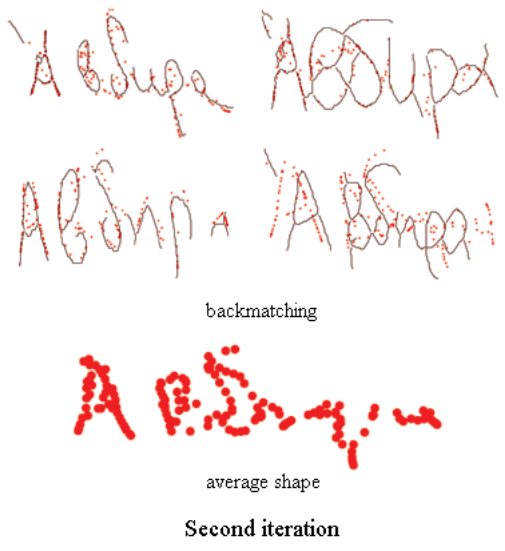
(a)



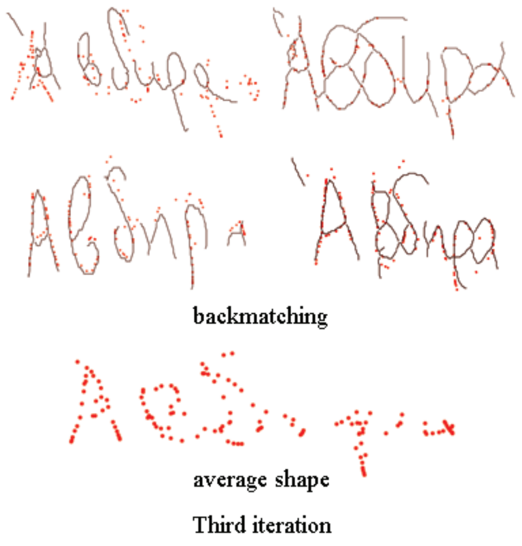
(b)



(c)



(d)



(e)

Figure 3.4: Model shape refinement. (a) Sampled points from the initial shape. (b) After initializing backmatching by aligning the model with the image bounding-box (left), the model deforms so as to match the image edgels (right). (c) The first, (d) second and (e) third iteration of shape refinement along with the corresponding average shape.

3.2.4 Learning shape deformations

Due to the backmatching of the model shape to each training image in the previous section, examples of the variations within the desired word-class are provided. Since

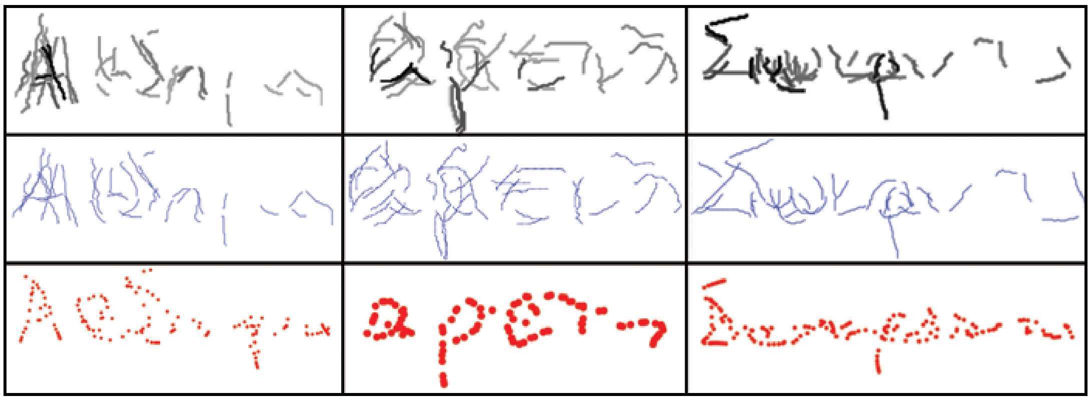


Figure 3.5: Evolution of shape models over the three stages of learning. Top row: model parts (Section 3.2.1). Second row: initial shape (Section 3.2.2). Bottom row: refined shape (Section 3.2.3). The mean shape for the word "αρετή" (virtue in English) is more intense than the other two, due to lower intra-class variability.

these examples are in full point-to-point correspondence, we can learn a compact model of the intra-class variations using the statistical shape analysis technique by Cootes et al. [13]. The idea is to consider each example shape as a point in a $2p - D$ space (with p the number of points on each shape) and model their distribution with Principal Component Analysis (PCA). The eigenvectors returned by PCA represent modes of variation and the associated eigenvalues λ_i their importance, namely, how much the example shapes deform along them, as it is depicted in figure 3.1(e).

By keeping only the n largest eigenvectors $E_{1:n}$ representing 95% of the total variance, it is feasible to approximate the region in which the training examples lie by:

$$S + E_{1:n}\, b, \quad \text{where:}$$

- S is the mean shape,
- b is a vector representing shapes in the subspace spanned by $E_{1:n}$ and
- the i^{th} component of b is bound by $\pm 3\sqrt{\lambda_i}$.

This defines the valid region of the shape space, containing shapes similar to the example ones. Typically, $n < 15$ eigenvectors suffice. The first two deformation modes for our running example word "Άβδηρα" (Abdera in English) are shown in figure 3.1(e). In chapter 4, we take advantage of this deformation model to constrain the shape matcher to search only inside the valid region.

It is interesting to mention that deformation is defined in terms of geometric transformation from the shape of an instance of the word class to another instance. This implies that a non-rigid transformation is essential to map the shape of a word instance to another one. Lastly, earlier works on these deformation models require either the example shapes as input [32], or they need the point-to-point correspondences [13]. In contrast, our method automatically learns shapes, correspondences and deformations given just images of words.

Chapter 4

Word Image Matching

4.1 Introduction

In this chapter, we describe the method for localizing the boundaries of previously unseen word instances that belong to a specific word class. In paragraph 4.2, the way this problem is addressed is introduced. An initialization for the shape matcher, by means of candidate locations and scales of the desired word instance, inside its bounding box, is presented in paragraph 4.3. The non-rigid point matching approach between word images is described in detail in paragraph 4.4 and an evaluation of the matching outcome by scoring detections is described in paragraph 4.5.

4.2 Integrated approach

The task of matching the shape model, learnt in chapter 3, to the test image edges, presents several challenges:

- Segmentation errors may result in a cluttered word image where only a percentage of total edges is deemed valid for further processing.

- To handle both inter-writer and intra-writer variability for a word class, the shape model must be deformed into the shape of the particular instance shown in the test image.

In order to achieve this objective, the problem is decomposed into two stages:

1. Rough estimates for location and scale of the word instance are obtained using a Hough-style voting scheme (section 4.3).
2. These estimates are then used to initialize the non-rigid shape matcher of Chui and Rangarajan [12] (section 4.4).

The first stage greatly simplifies the subsequent shape matching, as it lifts three degrees of freedom, namely, translation and scale. The combination of the two stages enables the matcher to operate in cluttered images, thus allowing to localize word boundaries. Furthermore, in section 4.4, the matcher is constrained to explore only the region of the shape space spanned by the training examples, thereby ensuring that output shapes are similar to class members.

Finally, the data set used in our experiments consists of foreground edgels that prevail against clutter. Hence, the bounding box is set to be the whole image as it covers mainly informative parts of the word of interest. This implies that the first stage of estimating the candidate location and scale of the word's center, inside the bounding box, is rather redundant and therefore, the matching stage could already commence without initialization. Nevertheless, as it will be explained in section 4.5, possible locations and scales of the desired word result into separate detections, from which we retain the one with the highest score. This provides better results, contrary to those accrued by applying the matcher without an initial location and scale of the word instance inside the test image.

4.3 Initialization by Hough Voting

In section 3.2.1 the shape of a word class is represented as a set of PAS parts, each with a specific shape, location, size and confidence. In the present section, these parts are matched to PAS from a test word image, based on their shape descriptors. In particular, a model part is deemed matched to an image PAS if their dissimilarity (2.1) is below a threshold γ (this is the same as used in section 3.2.1).

Since a pair of matched PAS induces a translation and scale transformation, each match votes for the presence of a word instance at a particular location (word's center) and scale. This is done through a Hough-style voting process, similar to the one used in the learning stage, which is widely exploited in object detection tasks [41], [55], [70]. Each vote is weighed by:

- the shape similarity between the model part and test PAS,
- the edge strength of the PAS and

- the confidence of the part.

Local maxima in the voting space define rough estimates of the location and scale of candidate word instances. As it is shown in figure 4.1(a) (the figure is better seen in color), the local maxima (red areas) denote the center of the word "Άβδηρα", which is used as an initialization for the shape matcher (figure 4.1(b)).

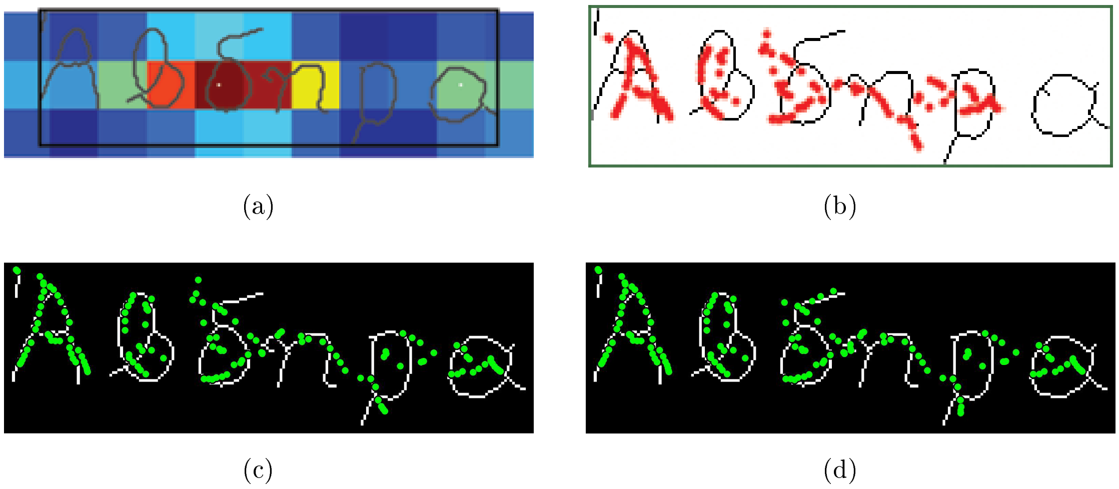


Figure 4.1: Word detection. (a) A local maximum in Hough space defines the word's center. (b) Initialization of TPS-RPM by centering the model to the word's center. (c) The output shape with unconstrained TPS-RPM. It captures the word relatively well, except for the letters 'δ' and 'ρ', where it is strongly attracted by the edgel orientations. (d) Output of the shape-constrained TPS-RPM. Now the word is more properly recovered.

This voting procedure generates 2 to 10 local maxima in a typical word image used in our experiments, as the local features are not very distinctive on their own. Regarding the problem complexity, this process does not make significant improvements for the data set used in our experiments. Nevertheless, for word images which derive from poor segmentation, or images that contain unsegmented words (only bounding-box annotation), the number of possible locations and scales a word instance could take place might vary considerably. Hence, Hough voting acts as a focus of attention mechanism, drastically reducing the complexity involved. This is more preferable than running the matcher directly, without initialization.

4.4 Word Matching by TPS-RPM

For each initial location l and scale s found by Hough voting:

- A point set V is obtained by centering the model shape on l and rescaling it to s and
- the point set X contains all image edge points within a larger rectangle of scale $1.8s$ (figure 4.1(b)).

This larger rectangle is designed to contain the whole word, even when s is underestimated. Any point outside this rectangle is ignored by the shape matcher. Given the initialization, V is put in correspondence with the subset of X that lies on the word boundary. The associated non-rigid transformation is estimated with the Thin-Plate Spline Robust Point Matching (TPS-RPM) algorithm, which also rejects image points that do not correspond to any model point. A brief summary of TPS-RPM is provided in this section and we refer the reader to [12] for more details.

4.4.1 Soft-assign and deterministic annealing

TPS-RPM matches two point sets $V = \{v_\alpha\}_{\alpha=1,\ldots,K}$ and $X = \{x_i\}_{i=1,\ldots,N}$, by applying a non-rigid TPS mapping parameterized by $\{d, w\}$ to V. The thin plate spline is chosen because it is the only spline that can be decomposed into affine and non-affine subspaces:

$$f(v) = v_\alpha \cdot d + \phi(v_\alpha) \cdot w \tag{4.1}$$

where d is the affine component and w is a non-affine warping coefficient, which is combined with the TPS kernel $\phi(v_\alpha)$ to form the non-rigid warp.

In the meanwhile, TPS estimates both the correspondence matrix $M = \{m_{\alpha i}\}$ between V and X and the mapping $\{d, w\}$ that minimize an objective function including:

1. the distance between points of X and their corresponding points of V after mapping them by the TPS and
2. the regularization terms for the affine and warp components of the TPS.

This energy function is defined as follows:

$$E(M, d, w) = \sum_{\alpha=1}^{K} \sum_{i=1}^{N} m_{\alpha i} ||x_i - v_\alpha d - \phi(v_\alpha) w||^2 + \lambda \operatorname{trace}(w^T \Phi w) \tag{4.2}$$

where Φ is a $K \times K$ matrix formed by the kernels $\phi(v_\alpha)$ and M always satisfies:

$$\sum_{\alpha=1}^{K+1} m_{\alpha i} = 1,\ \forall i \in \{1, 2, \ldots, N\},\ \sum_{i=1}^{N+1} m_{\alpha i} = 1,\ \forall \alpha \in \{1, 2, \ldots, K\},\ m_{\alpha i} \in [0, 1].$$

In addition to the inner $K \times N$ part, M has an extra row and an extra column to reject points as unmatched (outliers).

Since neither the correspondence M nor the TPS mapping $\{d, w\}$ are known beforehand, TPS-RPM iteratively alternates between updating M, while keeping $\{d, w\}$ fixed and updating the mapping with M fixed. M is a continuous soft-assign matrix, allowing the energy function to improve gradually during the optimization, without jumping around in the space of binary (hard correspondence) permutation matrices (and outliers). It is updated by setting $m_{\alpha i}$ as a function of the distance between x_i and v_α, after mapping by the TPS:

$$m_{\alpha i} = \frac{1}{T} \exp\left(\frac{(x_i - f(v_\alpha, d, w))^T (x_i - f(v_\alpha, d, w))}{2T}\right) \tag{4.3}$$

where $f(v_\alpha, d, w)$ is the mapping of point v_α by the TPS $\{d, w\}$ and T is a temperature parameter which will be explained later on.

The update of the mapping fits a TPS between V and the current estimate $Y = \{y_\alpha\}_{\alpha=1,\dots,K}$ of the corresponding points by minimizing the following energy function:

$$E_{TPS}(f) = \sum_{\alpha=1}^{K} ||y_\alpha - f(v_\alpha)||^2 + \lambda \int\int \left[\left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 \right] dxdy \quad (4.4)$$

where λ is a regularization parameter. For a fixed value of λ there exists a unique minimizer f which comprises of the matrices d and w, thus yielding equation (4.1). Each point y_α in y is a linear combination of all image points $\{x_i\}_{i=1,\dots,N}$ weighted by the soft-assign values $m_{\alpha i}$:

$$y_\alpha = \sum_{i=1}^{N} m_{\alpha i} x_i \quad (4.5)$$

The TPS fitting maximizes the proximity between the points Y and the model points V after TPS mapping, under the influence of the regularization terms, which penalize local warpings w and deviations of d from the identity. The latter accounts for alleviating the problem of unphysical reflection mappings, which can flip the entire plane. Fitting the TPS to $V \leftrightarrow Y$ rather than to $V \leftrightarrow X$, allows to harvest the benefits of maintaining a full soft-correspondence matrix M. Combining equations (4.2) and (4.5) and the newly inserted regularization terms the final energy function to be minimized by the TPS is:

$$E_{TPS}(d, w) = ||Y - Vd - \Phi w||^2 + \lambda_1 \operatorname{trace}(w^T \Phi w) + \lambda_2 \operatorname{trace}([d - I]^T [d - I]) \quad (4.6)$$

where Y and V are just concatenated versions of point coordinates y_α and v_α and λ_1, λ_2 are the control parameters of the last two terms.

The optimization procedure of TPS-RPM is embedded in a deterministic annealing framework by introducing a temperature parameter T, which decreases at each iteration. The entries of M are updated as it is denoted in equation (4.3) and then they are normalized to ensure row and column summation to 1 [12]. Since the temperature T is the bandwidth of the Gaussian kernel in (4.3), as it decreases M becomes less fuzzy, progressively approaching a hard correspondence matrix. At the same time, the regularization terms of the TPS are given less weight. This is done in the same spirit with the annealing schedule on the correspondence, using a linear annealing schedule for the parameters λ_1 and λ_2 in (4.6) where:

- $\lambda_i = \lambda_i^{init}\, T$, for $i = 1, 2$, instead of a fixed value and
- to provide more freedom for the affine transformation, λ_2^{init} is set to be much smaller than λ_1^{init}.

Consequently, the TPS is rigid in the beggining and gets more and more deformable as the iterations continue. These two phenomena enable TPS-RPM to find a good solution even when given a rather poor initialization. At first, when the correspondence uncertainty is high, each y_α essentially averages over a wide area of X around the TPS-mapped point and the TPS is constrained to near-rigid transformations. This can be seen as a large value of T in (4.3) generates similar valued $m_{\alpha i}$, which are then averaged by (4.5). As the iterations continue and the temperature decreases, M looks less and less far and pays increasing attention to the differences between matching options from X. Since the uncertainty diminishes, it is safe to let the TPS looser, freeer to fit the details of X more accurately. The whole process is also described in algorithm 4:

Algorithm 4 The TPS-RPM algorithm

```
1 Initialize parameters $T$, $\lambda_1$ and $\lambda_2$ of the deterministic annealing procedure.
2 Initialize parameters $M$, $d$ and $W$ of the alternating update step.
3 Deterministic Annealing
    3.1 Alternating Update
        (a) Update the correspondence matrix $M$ using (4.3).
        (b) Update the transformation parameters $(d, w)$ by minimizing (4.4).
    3.2 Iterate to step 3.1 until convergence.
4 Decrease $T$, $\lambda_1$ and $\lambda_2$.
5 Iterate to step 3 until T decreases no more.
```

Finally, the TPS-RPM technique is extended [22], by adding two terms to the objective function:

1. the orientation difference between corresponding points and
2. the edge strength of matched image points.

These two terms improve the accuracy and stability of the method even when initialized farther away form the best location and scale.

4.4.2 Constrained word matching

TPS-RPM treats all shapes according to the same generic TPS deformation model, simply prefering smoother transformations such as low $2D$ curvature in w and low affine skew in d. Two shapes with the same deformation energy are considered equivalent. This might result in output shapes unlike any of the training examples. In this section, the TPS-RPM is extended with the class-specific deformation model learned in section 3.2.4. In line with Ferrari et al. [22], we constrain the optimization to explore only the valid region of the shape space, containing shapes plausible for the class (defined by S, $E_{1:n}$, λ_i from section 3.2.4).

At each iteration of the TPS-RPM, the current shape estimate Y (equation (4.5)) is projected inside the valid region, just before fitting the TPS. This amounts to:

1. Align Y and S with regard to translation, rotation and scale.
2. Project Y onto the subspace spanned by $E_{1:n}$: $b = E^{-1}(Y - S)$, $b_{(n+1):2p} = 0$.
3. Bound the first n components of b by $\pm 3\sqrt{\lambda_i}$.
4. Transform b back into the original space: $Y^c = S + E \cdot b$.
5. Apply the inverse of the transformation used in 1 to Y^c.

The assignment $Y \leftarrow Y^c$ imposes hard constraints on the shape space. While this guarantees output shapes similar to class members, it might sometimes be too restrictive. To match a novel instance accurately, it could be necessary to move a little along some dimensions of the shape space not recorded in the deformation model. The training data cannot be assumed to present all possible intra-class variations. To overcome this issue, a soft-constrained variant is proposed, where Y is attracted by the valid region, with a force that decreases with temperature:

$$Y \leftarrow Y + \frac{T}{T_{init}}(Y^c - Y).$$

This causes TPS-RPM to start fully constrained and then, as temperature decreases and M looks for correspondences closer to the current estimates, later iterations are allowed to apply small deformations beyond the valid region (typically along dimensions not in $E_{1:n}$). As a result, output shapes fit the image data more accurately, while still resembling class members. This behaviour is inspired by the TPS-RPM, which also lets the TPS more and more free as T decreases.

The proposed extension to TPS-RPM has a deep impact, in that it alters the search through the transformation and correspondence spaces. Apart from improving accuracy, it can help TPS-RPM to avoid local minima far from the correct solution, thus avoiding significant failures. Figure 4.1(d) shows the improvement provided by the constrained shape matching, against the TPS-RPM with just the generic TPS model, as is illustrated in figure 4.1(c).

4.5 Scoring detections

Every local maximum in Hough space constitutes an initialization for the shape matching, and results in different shapes (detections) localized in the test image. In this section we score the detections, making it possible to reject detections and to evaluate the detection rate and false-positive rate of the word matching.

Each detection is scored by a weighted sum of four terms:

1. The number of matched model points, i.e. for which a corresponding image point has been found with good confidence. According to Chui and Rangarajan [12], these are all points v_α with $\max_{i=1,\ldots,N}(m_{\alpha i}) > 1/N$.

2. The sum of squared distances from the TPS-mapped model points to their corresponding image points. This measure is made scale-invariant by normalizing by the squared range r^2 of the image point coordinates (width or height, whichever is larger). Only matched model points are considered.

3. The deviation $\sum_{i,j\in\{1,2\}} (I(i,j) - d(i,j)/\sqrt{|d|})^2$ of the affine component d of the TPS from the identity I . The normalization by the determinant of d factors out deviations due to scale changes.

4. The amount of the non-rigid warp w of the TPS $\text{trace}(w^T \Phi w)/r^2$, where $\Phi(a,b) \propto ||v_\alpha - v_b||^2 \log||v_\alpha - v_b||$ is the TPS kernel matrix.

This score integrates the information a matched shape provides. It is high when the TPS fits many (term 1) points well (term 2), without having to distort much (terms 3 and 4). In our current implementation, the relative weights between these terms have been selected manually, they are the same for all classes, and remain fixed in all experiments.

As a final refinement, if two detections overlap substantially, we remove the lower scored one. Notice that the method can detect multiple instances of the same class in an image. Since they appear as different peaks in the Hough voting space, they result in separate detections. This indicates that we should only keep the highest scored detection for the presegmented words used in our experiments, as the nubmer of instances to be detected is at most 1. Nevertheless, we retain the afforementioned criterion, so as to account for future choices of unsegmented images containing handwritten words.

Chapter 5

Experimental Evaluation

5.1 Datasets and protocol

Before evaluating the performance of the proposed word matching technique and its application for word spotting, we first introduce the image data used in our experiments. The datasets originate from the ICDAR'07 Handwriting Segmentation Contest and were particularly used by Papavassiliou et al. [56] in both ICDAR'07 and ICDAR'09 contests. The document images in the datasets cover a wide range of cases which occur in unconstrained handwriting. One type of such cases comes from 25 writers, who were asked to copy a given text of approximately 150 words. The segmentation output of the algorithm described in [56], on these document images, results into clean word images comprising our training and testing datasets.

In order to train a class-specific model, we have manually annotated the words belonging to a particular word class. This was carried out for 10 word classes, as it is shown in Table 5.1. Each word class contains one to four instances per writer and thus the number of words for a class varies from 25 to 100. The dataset comprising each class is split into training and validation data. For each such split, we learn a model from the training data and test it upon the validation data (cross validation). Initially, the training data consist of 50% of the images belonging to each class. Hence, each class-specific model is trained from a random sample containing half of the available images (Table 5.1). We iterate this process five times, yielding five models per class, from different training sets, in order to prove the stability of the learning process through a *repeated random sub-sampling validation*. We refer to learning and testing on a particular split of the images as a *trial*.

Table 5.1 also illustrates a number of images that do not contain any instance of the respective word class. These negative test sets are generated from a random sample of words written by a specific writer and they are used to evaluate the efficiency of our method, in terms of detection rate (on validation data) against false positives, or else, it's classification capability. Both validation and negative test sets are equally distributed. Finally, all experiments are run with the same parameters (no class-specific tuning is applied).

Table 5.1: Number of images comprising training and validation data sets and number of negative images used to evaluate the performance of word recognition.

word class	training set	validation set	negative set
Σωκράτης	38	37	37
Δημόκριτος	50	50	50
Άβδηρα	38	37	37
αρετή	13	12	12
αγαθό	25	25	25
δικαστήριο	25	25	25
σοφία	13	12	12
Θράκη	38	37	37
φιλοσοφία	25	25	25
πατέρας	25	25	25

A number of example images from the dataset used in our experiments after applying thinning (section 2.2.2), is depicted in figure 5.1 for our vocabulary, which consists of 10 word classes. Apart from the variability among writing styles, there also exist variations with regard to scale, word ending characters, commas, etc.

5.2 Intra-class word recognition

The performance of the proposed word matching approach is initially estimated in terms of how accurately it recovers the true class boundaries of previously unseen word instances. For this estimation, the true boundaries of all word instances used in our experiments have to be manually annotated. However, the thinning operation applied in the image preprocessing step (section 2.2.2) alleviates this issue, as the words have already been transformed to their outlines, thus yielding the desired ground-truth boundaries.

Assuming that B_{gt} expresses the ground-truth boundaries and that B_{model} consists of the matched output points to a test image, the accuracy of the proposed method can be quantified by two measures:

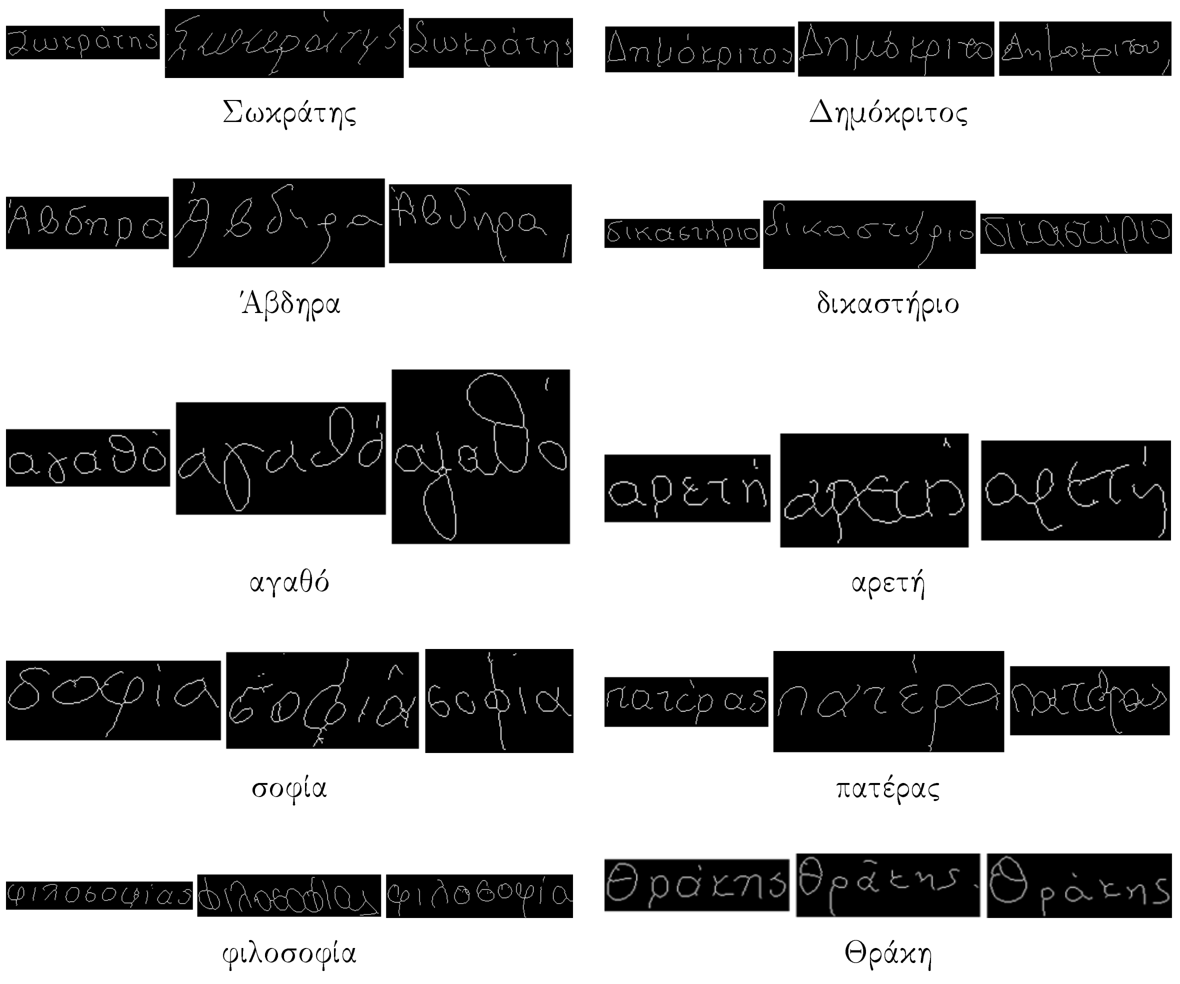


Figure 5.1: Example word images from the datasets used in our experiments.

1. *Coverage* is the percentage of points from B_{gt} closer than a threshold t from any point of B_{model}.
2. *Precision* is the percentage of points from B_{model} closer than t from any point of B_{gt}.

The measures are complementary and t is set to be 4% of the diagonal of the bounding-box of B_{gt}. In other words, coverage captures how much of the word's boundaries has been recovered by the algorithm, whereas precision denotes how much of the algorithm's output lies on the word's boundaries (instead of background).

First, we match the models learnt for each class to the images of the validation dataset and calculate the detection rate. Subsequently, we compute the detection rate against the number of false positives, averaged on images contained in both validation and negative sets (i.e. $37 + 37 = 74$ test images for word class 'Σωκράτης') of each class, as well as over the five trials. This averaging process serves us to set a common reference point for our comparisons, defined by the number of false positives per image (FPPI). Such a reference point can be obtained by the process described by the following steps:

1. Consider the scores $S = \{s_i\}$ obtained after matching a model to both positive (correct scores) and negative (false scores) images.

2. Sort the scores s_i in ascending order and assign to each score the corresponding threshold t_i, for $i = 1, \ldots, |S|$.

3. For each threshold t_i:

    - Count the number of correct and false scores which are larger than t_i, thus yielding the detection rate and false positive rate at the specific threshold.

4. Determine the detection rate at a particular number of false positives per image, by approximately finding the threshold t_i which is closest to that value and corresponds to the desired detection rate.

The first row of Table 5.2 shows the mean value (accuracy percentage) of the coverage and precision values for several word classes from our vocabulary, which consists of 10 word classes. This value is also averaged over trials and correct detections at 0.1 FPPI (this is the reference point for all comparisons). These classes were selected on purpose, as they represent most of the vocabulary's variability in terms of scale, letters of the alphabet involved and more importantly, in terms of number of training instances per-writer. Specifically, there are four instances per writer for the word 'Δημόκριτος' (Democritus in English), three instances for the word ''Αβδηρα', two for the word 'αγαθό' (good in English) and one for the word 'σοφία' (wisdom in English).

Table 5.2: Accuracy of localized word boundaries for some word classes using 50% and 80% of the dataset for training, respectively. Each entry is the average value of coverage and precision over all trials and correct detections at 0.1 FPPI.

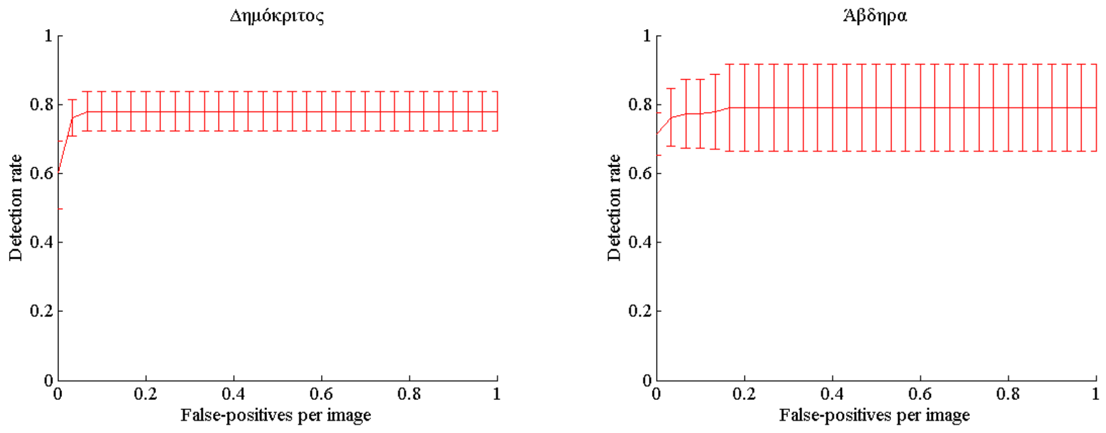
training set	**Δημόκριτος**	**Άβδηρα**	**αγαθό**	**σοφία**
50%	92.8%	91.06%	87.75%	92.21%
80%	94.32%	92.11%	88.96%	90.19%

To evaluate a detection up to a bounding-box, we adopt the standards of the PASCAL challenge criterion, which is widely used in object detection tasks [22]:

Criterion 5.1. *A detection upon a positive word image is counted as correct only if the intersection-over-union (IoU) ratio between the detection's bounding-box and the ground-truth's one overlap more than* 50%*. All other detections are counted as false positives.*

The detection rate against the number of false positives, which was computed in the previous step, is illustrated in figure 5.2, for the same word classes used above to evaluate the algorithm's accuracy. The vertical bars in each plot represent two standard deviations of the detection rate from the mean value, calculated for the different models.

Note that the system performs relatively well but for the word 'σοφία'. Although it has a high accuracy (Table 5.2, top row), its detection rate is much lower compared to the other words and the deviation from the mean value varies considerably. We assume that this happens due to the limited number of instances written by each writer for word 'σοφία' (only one instance per-writer). Using only half of the images to train a representative mean shape for this particular class does not allow to capture various writing styles in novel word instances, as the information stored in the deformation model is rather inadequate. Furthermore, we are unable to properly learn a single writer if no instance is recorded for him in the model.



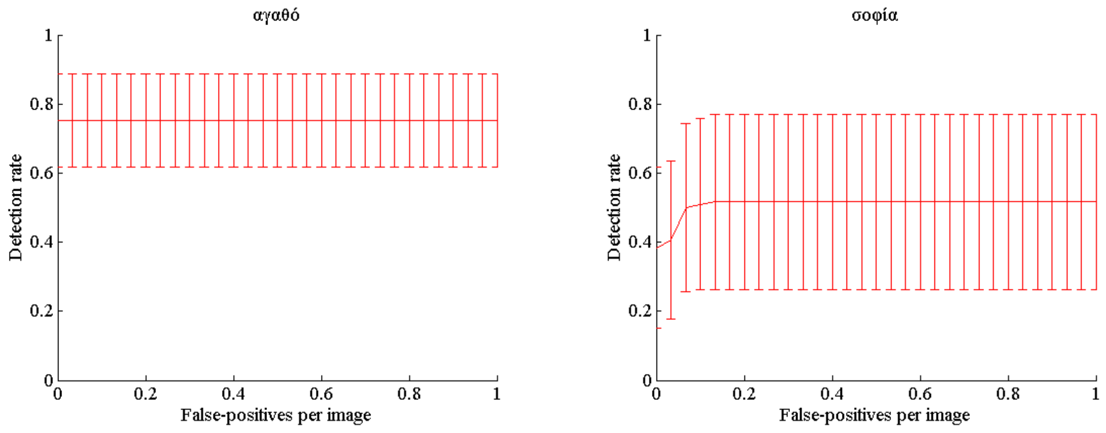


Figure 5.2: Word recognition performance using 50% of the dataset for training.

To account for a higher degree of variability among writers and their writing styles, we train new models using a random sample containing 80% of the available images, for each word class. This does not necessarily suppress our system in terms of how many images should be used for training in a word spotting application, as in realistic handwriting conditions the number of instances per writer is usually abundant. The current configuration is related to the dataset used in the following experiments, which is described in Table 5.3. No additional tuning is applied on the parameters.

For consistency reasons, we evaluate the accuracy of the proposed methodology with regard to coverage and precision, for the previously used example word classes and we use the same reference point of 0.1 FPPI. The bottom row of Table 5.2 shows the values obtained by the new models using 80% of the dataset for training. The performance gets

Table 5.3: Number of images comprising training and validation data sets and number of negative images used to evaluate the performance of word recognition.

word class	training set	validation set	negative set
Σωκράτης	60	15	15
Δημόκριτος	80	20	20
Άβδηρα	60	15	15
αρετή	20	5	5
αγαθό	40	10	10
δικαστήριο	40	10	10
σοφία	20	5	5
Θράκη	60	15	15
φιλοσοφία	40	10	10
πατέρας	40	10	10

only 0.44% better on average over the four classes. A clearer view of the improvement gained by the current fix for our trials, is seen in figure 5.3, where the word 'σοφία' is now properly detected, thus verifying our prior assumption. In addition, the detection capability is substantially improved for the other three word classes.

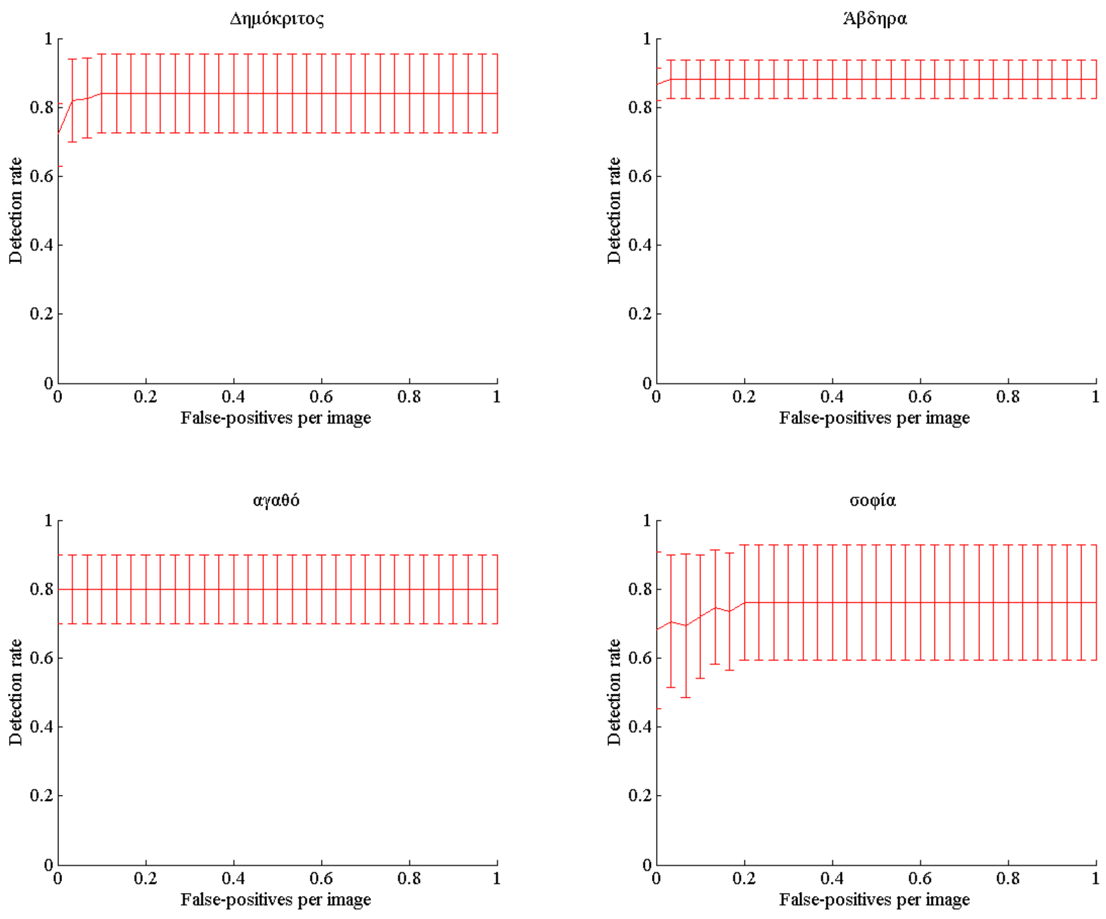


Figure 5.3: Word recognition performance using 80% of the dataset for training.

These results can also be identified more precisely in Table 5.4, through indices such as the total detection rate (TDR) of the cross validation process, the detection rate at 0.1 FPPI (DR at 0.1 FPPI) and the false positive rate (FPR), all averaged over the five trials. Table 5.4 also shows the results accrued from the previous trials (using 50% of available images for training) on the same word classes. It is interesting to notice that regardless of the improvement gained in detection rate for all classes, the false positive rate for word 'σοφία' still remains high, even though the number of negative images to be tested is reduced (five for class 'σοφία', as it is shown in Table 5.3). Since local maxima are found in the Hough space, they lead to detections either on positive (validation set), or on negative test images. Yet, as it will be explained in the next section, the scores of false detections on negative images, are low enough, so as to allow discriminating them among other classes and classify them properly in a word spotting task.

Table 5.4: Statistical comparison between the experimental setups using 50% and 80% of the dataset for training.

Training images (%)	Index	Δημόκριτος	Άβδηρα	αγαθό	σοφία
50	TDR	0.780	0.789	0.752	0.516
	FPR	0.116	0.081	0.096	0.366
	DR at 0.1 FPPI	0.780	0.773	0.752	0.433
80	TDR	0.840	0.853	0.800	0.760
	FPR	0.190	0.093	0.080	0.340
	DR at 0.1 FPPI	0.840	0.853	0.80	0.640

Finally, the negative test set, from which false positives may arise, consists of images written only by a single writer. In order to strengthen the system's stability on recognizing unknown word instances which do belong to a known word class, independently of negative images, we extend the negative test sets by including all writers who contributed for learning class-specific models. More specifically, we repeat the five trials which use random subsets of 80% of the training images, 25 times, each time using a randomly selected writer-specific negative test set. Then, we average the statistics produced by each set of trials (5 trials per set) over all 5×25 trials and present the results for all classes of the vocabulary in Table 5.5.

Table 5.5 also illustrates the weighted mean value of each index and its deviation, averaged over all classes. Note that the weights used for TDR and FPR correspond to the number of images in validation and negative sets, respectively. In contrast, the weights used for DR at 0.1 FPPI correspond to the number of images in both data sets. Notice again that the word 'αρετή' does not succeed a high detection rate due to the limited number of training instances.

Table 5.5: Total statistics for all word classes averaged on all trials and models using 80% of the dataset for training and negative examples to account for false positives.

word class	TDR	FPR	DR at 0.1 FPPI
Σωκράτης	0.987	0.403	0.960
Δημόκριτος	0.860	0.246	0.848
Άβδηρα	0.880	0.192	0.861
αρετή	0.640	0.318	0.574
αγαθό	0.820	0.240	0.808
δικαστήριο	0.920	0.324	0.899
σοφία	0.800	0.365	0.755
Θράκη	0.880	0.440	0.855
φιλοσοφία	0.920	0.384	0.896
πατέρας	0.840	0.320	0.804
mean	0.874	0.317	0.850
std	0.090	0.074	0.103

5.3 Word spotting using a vocabulary

In the previous section, we investigated the conditions under which our system behaves properly, with regard to it's ability of detecting novel keywords correctly. More precisely, the number of images to be used for training a class-specific model is determined after statistical comparisons between configurations using the 50% and 80% of the available images. A repeated random sub-sample cross validation was used in order to evaluate the system's word recognition performance. In addition, it's ability of discriminating correct word instances against false ones, was examined thoroughly.

In this section, we assess the system's performance in a word spotting task by combining the information provided by all models. Particularly, given an unknown word that belongs to an already known (in our vocabulary) word class, the system matches the word to all the class-specific models learnt and classifies it to a particular word class according to the following criterion:

- The class-specific model achieving the highest matching score with the keyword, is the one specifying the keyword's original class.

The process is iterated five times (trials) to evaluate the system's consistency. Each word is annotated by a ground-truth bounding-box, which denotes the class it belongs to, so as to compare the retrieved data against the real data, for performance statistical extraction. Note that the score of a model's detection inside a test images is defined by:

$$S_m = w_1 D_s + w_2 O_s, \text{ where} \tag{5.1}$$

- D_s is the detection's weighted sum of the four terms, described in section 4.5,
- O_s denotes the overlap percentage between the detected and the ground-truth bounding boxes and

- w_1, w_2 are normalized predefined weights with values 0.6 and 0.4 respectively.

The second term of equation (5.1) was added to tackle false detections caused by models representing small words, such as 'αρετή' and 'σοφία', which were located inside larger ones, such as 'Δικαστήριο', 'Δημόκριτος' and 'φιλοσοφία'. Although this may seem visually peculiar, as these words differ substantially, it was experimentally proven that the second term suppresses the number of false positives caused by words of different sizes. After all, the matching process treats all words as shapes in the $2D$ space, without involving any contextual information. Hence, the second term acts as a smoothness measure of the scoring function. It's given less weight though, so as to let the first term loose enough to decide more accurately the correct detections.

Before estimating the efficiency of the proposed word spotting approach, we first introduce some evaluation indices, which are widely used to measure the performance of pattern classification tasks. For such tasks, the terms *true positives*, *true negatives*, *false positives*, and *false negatives* compare the results of the classifier under testing them against a ground-truth. The terms *positive* and *negative* refer to the classifier's prediction (sometimes known as the observation) and the terms *true* and *false* refer to whether that prediction corresponds to the ground-truth (sometimes known as the expectation).

Following these definitions and the indices tp, fp for true and false positives respectively, we define:

$$\text{Precision} = \frac{tp}{tp + fp} \, ,$$
$$\text{Recall} = \frac{tp}{tp + fn} \, ,$$

where $tp + fn$ is actually the total number of images comprising the incoming flow of words. A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \, .$$

The F-measure measures the effectiveness of retrieval with respect to a user who attaches as much importance to recall as to precision.

To add some visual sense of the system's ability to discriminate words among word classes we present the confusion matrices produced at each of the five trials. A confusion matrix is a specific table layout that allows visualization of the performance of a supervised learning algorithm, such as the proposed mean shape learning method. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the system is confusing two or more classes (i.e. commonly mislabeling one as another).

5.3.1 Word spotting using one model per class

Following the configuration defined in section 5.2, we make use of the same models learnt for each word class, using 80% of the available images. The test set consists of the words that were not used to train each class-specific model. As mentioned before, we match the 10 models (one per word class) to a new word from the test set and predict the word's original class by favoring the model which achieved the maximum score (among 10 scores). The whole process is iterated five times using the corresponding models of each trial (section 5.2).

The confusion matrices for this first experimental setup are illustrated in figure 5.4. Each column's entry denotes the percentage of words retrieved by each class. In the best case scenario the diagonal elements of the confusion matrix should be close to 100%. Along with the confusion matrices figure 5.4 shows the F-measure obtained by the system, both for each single trial and on average over all trials.

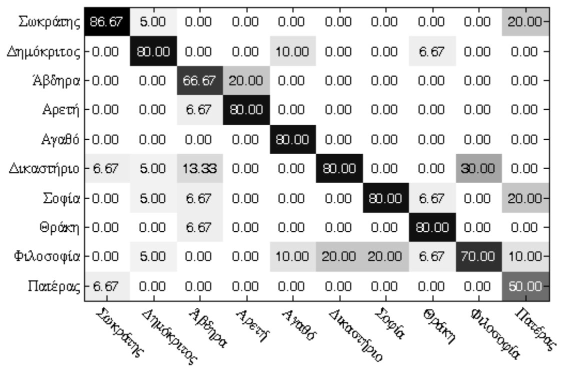
Notice that in figure 5.4(e) the model representing the word 'Σωκράτης' overwhelms the other models in terms of word retrieval. Not only it captures 100% of all instances belonging to its word-class but it also confuses other words for 'Σωκράτης'. This model is not desirable for our system as we would like to capture as much valid information as possible.

Word spotting using the best model

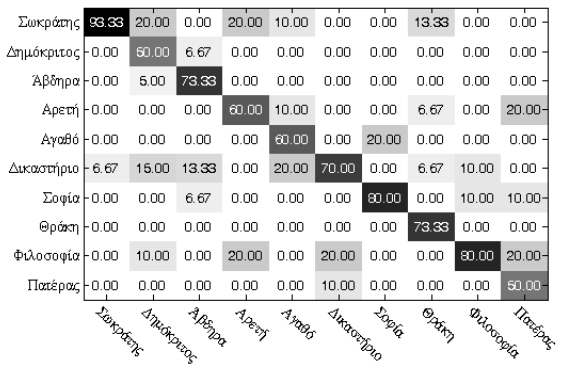
A way to achieve higher performance and counter issues such as the large number of false positives accrued by overtrained models (model for word 'Σωκράτης' of the fifth trial in the previous section) is to pick the best model of the five models learnt so far for each word class. To determine such a model, we take into account its overall behavior when matching it upon a wide range of negative images. As it is shown in Table 5.5 in the previous section, the total statistics accrued for all word classes are averaged on all trials and models and all 25 writers are considered. By extracting the same statistics for all word classes and writers before averaging them over the five models, we are able to trace each model's performance and select the one with the highest value, for a particular word class.

Table 5.6 shows the performance statistics with regard to the three indices used in section 5.2 for the word ''Αβδηρα'. The criterion we use for selecting the best model lies at the detection rate at 0.1 FPPI, which for the case of the word ''Αβδηρα' favors the second model (trial 2). Although this model does not correspond to the lowest false positive rate among the other models, it has a higher detection potential even when the circumstances get worse, namely, a larger number of negative images arises, resulting to more false positives. As mentioned in section 5.2, the detection rate at a specific threshold is computed by counting the number of correct detections whose scores exceed the score corresponding to that threshold. This implies that a high detection rate at this threshold is equivalent to large scores, which allow us to properly distinguish a new word instance from false scores in the decision step of the classification task.

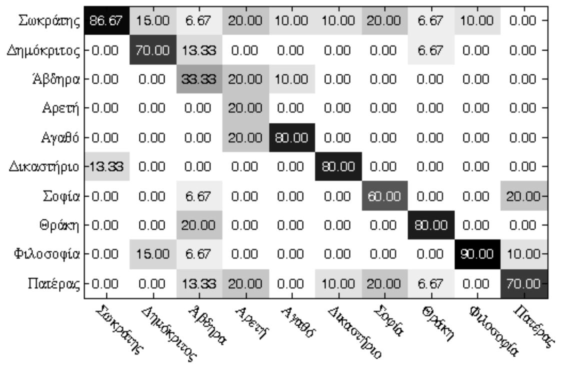
Based on this observation we apply word spotting in the same spirit with the configuration defined in the first experimental setup. Instead of iterating the process between trials, we run the same experiment only once, using the previously defined best model in



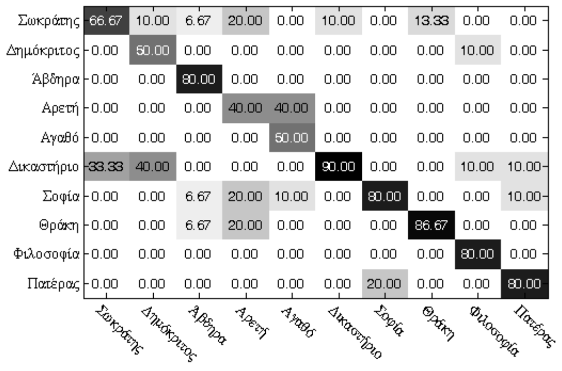
(a) F = 0.763



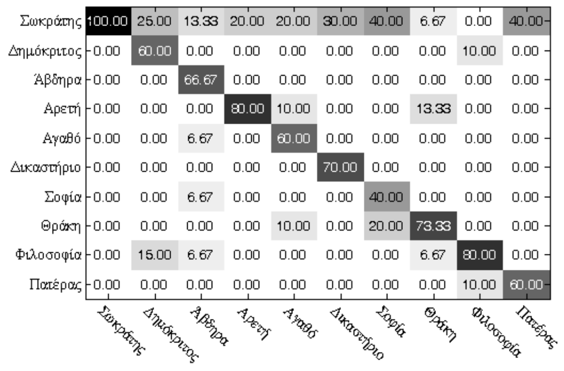
(b) F = 0.693



(c) F = 0.705



(d) F = 0.704



(e) F = 0.714

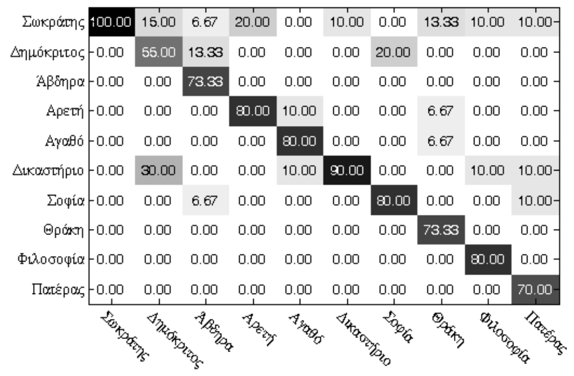
Figure 5.4: Confusion matrices for the five trials (a)-(e) of the first experimental setup and corresponding F-measures. The average F-measure is 0.716.

a single trial. Again, the test set contains all class-specific words that were not used for training the best models of each class. We define this trial as the second experimental setup.

In line with the previous experiments, we present the confusion matrix obtained by the current run in figure 5.5. Finally, the F-measure of this experiment is 0.765, which is almost 5% better than the average F-measure gained in the previous set of experiments and only 0.2% better than the best F-measure obtained by the first trial. Assuming a larger variety of models trained for each word class (i.e. 10 models per class), this configuration might bring in more satisfactory results. However, the number of possible writing styles to be captured by a single model is enormous, especially in unconstrained handwriting. Hence, its efficiency for word spotting is limited. Moreover, most of the computational time has to be occupied by the cross-validation and statistical extraction presented in section 5.2, which renders the current configuration inappropriate for automation.

Table 5.6: Performance statistics for determining the best model for the word ''Αβδηρα'. Each model corresponds to a different trial.

Άβδηρα	TDR	FPR	DR at 0.1FPPI
model 1	0.866	0.125	0.848
model 2	0.933	0.248	0.906
model 3	0.933	0.218	0.903
model 4	0.801	0.208	0.784
model 5	0.866	0.160	0.861



	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	100.00	15.00	6.67	20.00	0.00	10.00	0.00	13.33	10.00	10.00
Δημόκριτος	0.00	55.00	13.33	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Άβδηρα	0.00	0.00	73.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	80.00	10.00	0.00	0.00	6.67	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	80.00	0.00	0.00	6.67	0.00	0.00
Δικαστήριο	0.00	30.00	0.00	0.00	10.00	90.00	0.00	0.00	10.00	10.00
Σοφία	0.00	0.00	6.67	0.00	0.00	0.00	80.00	0.00	0.00	10.00
Θράκη	0.00	0.00	0.00	0.00	0.00	0.00	0.00	73.33	0.00	0.00
Φιλοσοφία	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	80.00	0.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	70.00

F = 0.765

Figure 5.5: Confusion matrix of the single trial using the best model for each class (2^{nd} experimental setup).

5.3.2 Word spotting using more models concurrently

So far, we have trained a number of models for each word class and matched them independently (one class-model per trial) to unknown words in order to classify them according to the matching score. The idea is to combine the scores of separate models learnt for the same word class and incorporate them in the decision step of the classification task. More specifically, we train again five models for each word class by using random samples of a subset containing 80% of the training images. The rest 20% of the images is left out for testing. Each sample is made up using 80% of the respective training subset, namely, each sample consists of 80% $\times$ 80% = 64% of the initial training images, but derives from the specific randomly chosen subset.

Table 5.7: Number of images used to train each class-specific model (samples) from a subset of 80% of the total images.

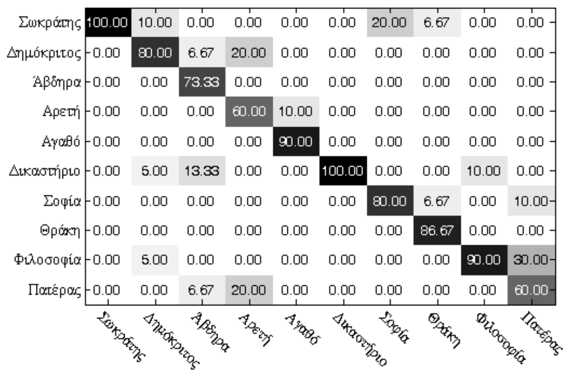
word class	training subset	training sample	test set
Σωκράτης	60	48	15
Δημόκριτος	80	64	20
Άβδηρα	60	48	15
αρετή	20	16	5
αγαθό	40	32	10
δικαστήριο	40	32	10
σοφία	20	16	5
Θράκη	60	48	15
φιλοσοφία	40	32	25
πατέρας	40	32	25

Table 5.7 shows the current configuration for a single trial of our experiment. The reason we learn different models from a subset of training images and not directly from the initial set is to iterate the whole process five times for consistency reasons with the aforementioned first experimental setup. Each iteration (trial) produces a different training subset of 80% of the training images from which a new group of five models is generated. The rest 20% of the initial set is left out again for testing. The contribution of this configuration is that a new word of the testing set is matched concurrently to all five models for each class ($5 \times 10 = 50$ total matches for every word) and therefore it has higher chances to be classified correctly.

In the same spirit with the first experimental setup, the performance of the classification task is illustrated by the confusion matrices in figure 5.6. The system performs substantially better both in each separate trial and on average, as it can be seen from the new F-measure values.

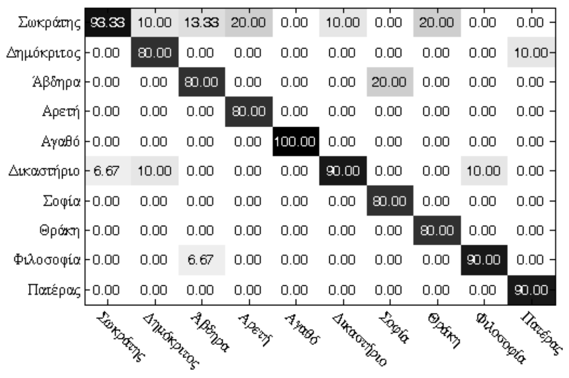
5.4 Principled training

The application of the proposed system for word spotting described in the previous section, showed that the system's classification capability improves considerably when querying with multiple models per word class. However, the process analyzed in the third experimental setup (section 5.3.2), requires a large amount of computational time, both for training models representing each word class (5 models $\times$ 10 words = 50 models per trial, or, 250 models for all trials) and more significantly, for matching these models upon new instances. Therefore, it imposes limits to the system's potential of extending to a practical word spotting application, such as the off-line handwritten word recognition [8].



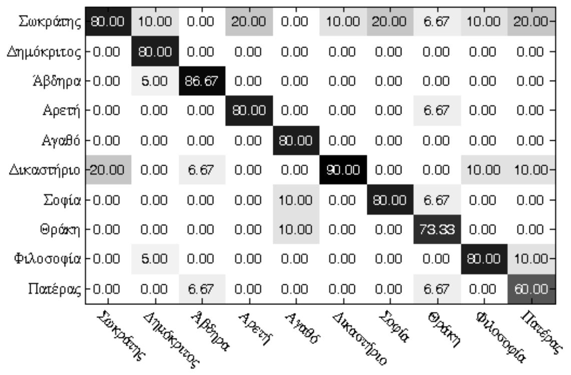
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	100.00	10.00	0.00	0.00	0.00	0.00	20.00	6.67	0.00	0.00
Δημόκριτος	0.00	80.00	6.67	20.00	0.00	0.00	0.00	0.00	0.00	0.00
Άβδηρα	0.00	0.00	73.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	60.00	10.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	90.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	0.00	5.00	13.33	0.00	0.00	100.00	0.00	0.00	10.00	0.00
Σοφία	0.00	0.00	0.00	0.00	0.00	0.00	80.00	6.67	0.00	10.00
Θράκη	0.00	0.00	0.00	0.00	0.00	0.00	0.00	86.67	0.00	0.00
Φιλοσοφία	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00	30.00
Πατέρας	0.00	0.00	6.67	20.00	0.00	0.00	0.00	0.00	0.00	60.00

(a) F = 0.842



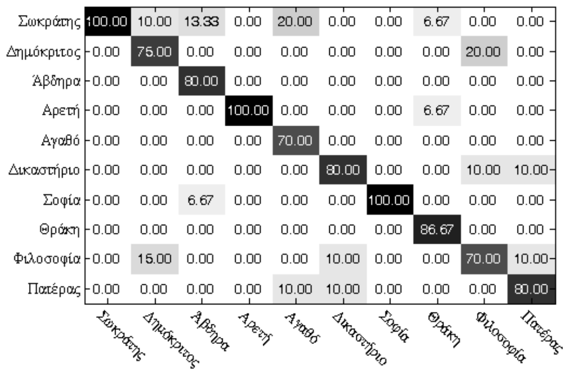
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	93.33	10.00	13.33	20.00	0.00	10.00	0.00	20.00	0.00	0.00
Δημόκριτος	0.00	80.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00
Άβδηρα	0.00	0.00	80.00	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	6.67	10.00	0.00	0.00	0.00	90.00	0.00	0.00	10.00	0.00
Σοφία	0.00	0.00	0.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00
Θράκη	0.00	0.00	0.00	0.00	0.00	0.00	0.00	80.00	0.00	0.00
Φιλοσοφία	0.00	0.00	6.67	0.00	0.00	0.00	0.00	0.00	90.00	0.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00

(b) F = 0.865



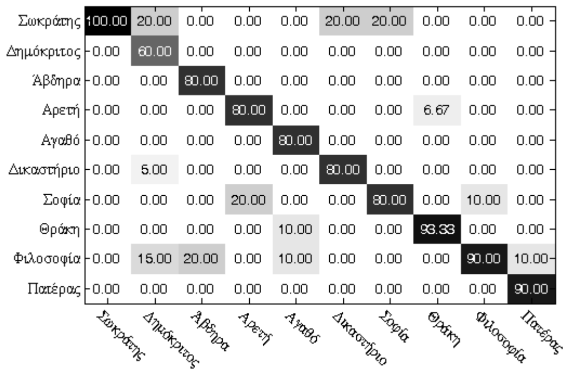
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	80.00	10.00	0.00	20.00	0.00	10.00	20.00	6.67	10.00	20.00
Δημόκριτος	0.00	80.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Άβδηρα	0.00	5.00	86.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	80.00	0.00	0.00	0.00	6.67	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	20.00	0.00	6.67	0.00	0.00	90.00	0.00	0.00	10.00	10.00
Σοφία	0.00	0.00	0.00	0.00	10.00	0.00	80.00	6.67	0.00	0.00
Θράκη	0.00	0.00	0.00	0.00	10.00	0.00	0.00	73.33	0.00	0.00
Φιλοσοφία	0.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	80.00	10.00
Πατέρας	0.00	0.00	6.67	0.00	0.00	0.00	0.00	6.67	0.00	60.00

(c) F = 0.798



	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	100.00	10.00	13.33	0.00	20.00	0.00	0.00	6.67	0.00	0.00
Δημόκριτος	0.00	75.00	0.00	0.00	0.00	0.00	0.00	0.00	20.00	0.00
Άβδηρα	0.00	0.00	80.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	100.00	0.00	0.00	0.00	6.67	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	0.00	0.00	0.00	0.00	0.00	80.00	0.00	0.00	10.00	10.00
Σοφία	0.00	0.00	6.67	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Θράκη	0.00	0.00	0.00	0.00	0.00	0.00	0.00	86.67	0.00	0.00
Φιλοσοφία	0.00	15.00	0.00	0.00	0.00	10.00	0.00	0.00	70.00	10.00
Πατέρας	0.00	0.00	0.00	0.00	10.00	10.00	0.00	0.00	0.00	80.00

(d) F = 0.833



	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	100.00	20.00	0.00	0.00	0.00	20.00	20.00	0.00	0.00	0.00
Δημόκριτος	0.00	60.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Άβδηρα	0.00	0.00	80.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	80.00	0.00	0.00	0.00	6.67	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	0.00	5.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00
Σοφία	0.00	0.00	0.00	20.00	0.00	0.00	80.00	0.00	10.00	0.00
Θράκη	0.00	0.00	0.00	0.00	10.00	0.00	0.00	93.33	0.00	0.00
Φιλοσοφία	0.00	15.00	20.00	0.00	10.00	0.00	0.00	0.00	90.00	10.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00

(e) F = 0.830

Figure 5.6: Confusion matrices for the five trials (a)-(e) of the third experimental setup. The average F-measure is 0.834.

Moreover, the images comprising each training set are randomly selected from the initial datasets and no additional information is used from the collection of images itself.

In the present section, we propose a structured approach for training a specific number of models for each word class, so as to alleviate the computational time spent for model creation and matching. The key point of our approach is to make use of the similarities among images used for training and group them into clusters which define the training sets for our models. Such similarities can be obtained by pairwise matchings between all images used for training. The process is described in detail in algorithm 5 for a particular word class.

Algorithm 5 Spectral clustering of the training set for model determination.

```
1 Match all pairs $(i, j)$ of training images using the unconstrained version of TPS-RPM,
    which was employed in the model shape refinement stage of the learning step (section
    3.2.3).
2 Construct a similarity matrix $W = \{w_{ij}\}$, as follows:
    • For each matched pair $(i, j)$, assign to $w_{ij}$ the matching score returned by
      TPS-RPM.
3 Apply the normalized cut clustering algorithm [54] to $W$ with the number of clusters
    being defined by the user.
4 The resulting clusters correspond to the sets used for training the specific models.
```

A common choice for the weights of the similarity graph to be partitioned by the normalized cut clustering algorithm is a Gaussian kernel with a standard deviation σ which determines the neighborhood of each node of the graph under a specific threshold ϵ, which allows to assign zero weight to distant points. In our case, the nodes correspond to the scores returned by TPS-RPM. The appropriate values for σ and ϵ may be determined by the following procedure:

- Select a particular value for σ and ϵ based on the parameters used in the learning process. For instance, ϵ could be chosen to be equal to the dissimilarity threshold γ used in section 3.2.1 for finding models parts.
- For the values selected in the previous step, apply the normalized cut clustering algorithm to the affinity matrix which consists of the weighted elements of W by the Gaussian kernel (σ, ϵ) and determine the class-specific models to be trained from the resulting clusters.
- For these models apply a cross-validation scheme on the validation dataset which was initially left out for testing.
- Iterate the whole process until the configuration between σ and ϵ lead to the best results of the cross-validation procedure, and determine their values from this configuration.

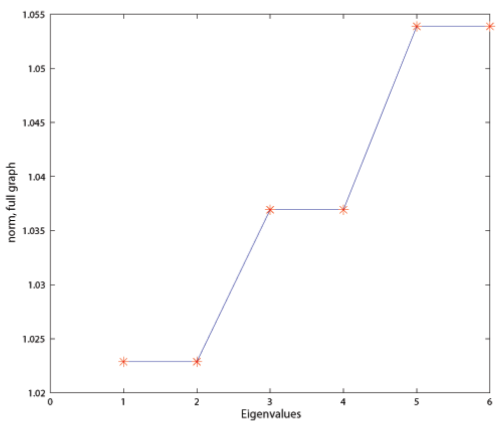
Nevertheless, such a procedure drains most of the proposed algorithm's (algorithm 5) computational time on training and testing for cross-validation. Hence, we omit this smoothness step of the similarity matrix W and use it directly for spectral clustering, relying on the promising scores of the pairwise matching outcome. Note that the matrix W should be symmetric. Due to the fact that an element w_{ij} might be slightly different from w_{ji}, as a result of different non-rigid registrations of image I into an image J and vice versa, we select their average value to represent the similarity of the corresponding entries in W, meaning $w_{ij} := \frac{w_{ij}+w_{ji}}{2}$.

Step 3 of algorithm 5 also requires the number of clusters k to be defined by the user. Following the eigengap heuristic from the tutorial on spectral clustering of Luxburg et al. [48], we can determine the number of clusters as follows:

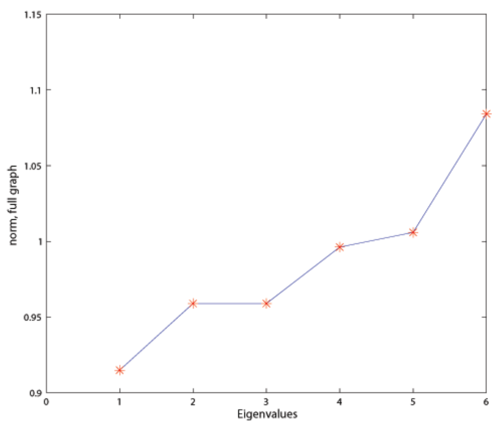
- Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues corresponding to the eigenvectors of the Laplacian matrix.

- Select the number k such that all eigenvalues $\lambda_1, \ldots, \lambda_k$ are very small, but λ_{k+1} is relatively large. In other words, there exists a gap (eigengap) between λ_{k+1} and the previous eigenvalues.

An example of the way the number of clusters is decided is illustrated in figure 5.7 for the words 'αγαθό' and 'αρετή'. The initial training sets using 80% of the available images for each class, contain 40 images for the word 'αγαθό' and 20 for the word 'αρετή'. The eigangaps shown in figure 5.7 denote that the number of clusters should be two (25 and 15 images) and one (all 20 images) for the words 'αγαθό' and 'αρετή', respectively. This meets our expectations for these particular words and especially for the word 'αρετή', as the number of instances per writer is two and one instance for the words 'αγαθό' and 'αρετή', respectively.



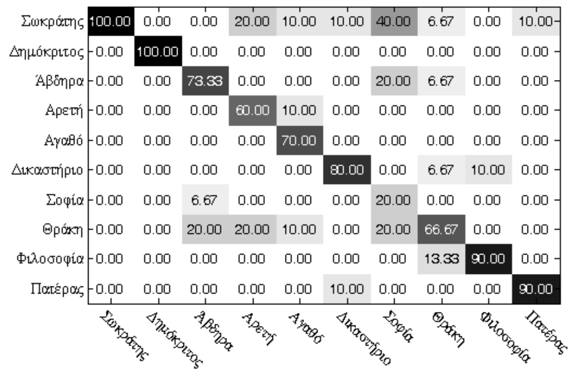
αγαθό



αρετή

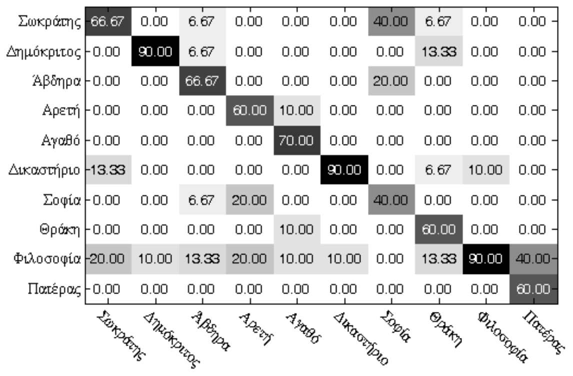
Figure 5.7: Eigenvalues of the Laplacian matrix (normalized cut) based on the fully connected graph for the words 'αρετή' and 'αγαθό'.

In the same spirit with the experimental setups defined in the previous sections, we implement the proposed clustering methodology on a random subset using 80% of the available images for training. This results into a number of training sets (clusters) from which models are created. Again, the process is iterated five times for consistency, thus yielding a different number of models per class for each trial. Now, the number of models to be used for word spotting and the images used for their creation is known beforehand. To compare the current approach with the methods presented in section 5.4, we illustrate the performance of the current experimental setup in figure 5.8 though the confusion matrices and the F-measure indices of each trial.



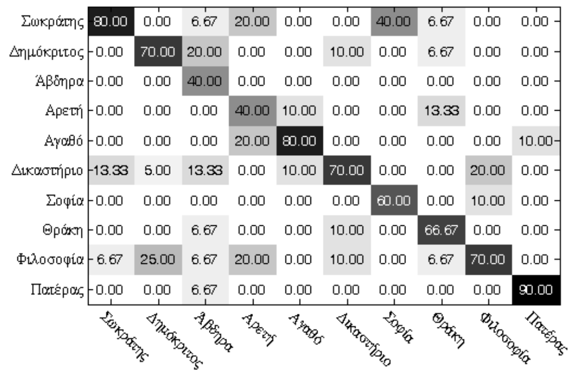
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	100.00	0.00	0.00	20.00	10.00	10.00	40.00	6.67	0.00	10.00
Δημόκριτος	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Άβδηρα	0.00	0.00	73.33	0.00	0.00	0.00	20.00	6.67	0.00	0.00
Αρετή	0.00	0.00	0.00	60.00	10.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	0.00	0.00	0.00	0.00	0.00	80.00	0.00	6.67	10.00	0.00
Σοφία	0.00	0.00	6.67	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Θράκη	0.00	0.00	20.00	20.00	10.00	0.00	20.00	66.67	0.00	0.00
Φιλοσοφία	0.00	0.00	0.00	0.00	0.00	0.00	0.00	13.33	90.00	0.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	10.00	0.00	0.00	0.00	90.00

(a) F = 0.809



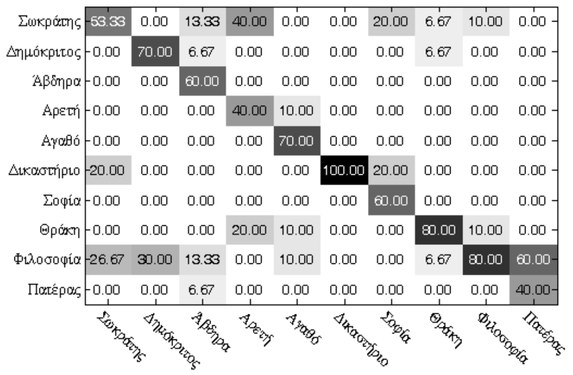
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	66.67	0.00	6.67	0.00	0.00	0.00	40.00	6.67	0.00	0.00
Δημόκριτος	0.00	90.00	6.67	0.00	0.00	0.00	0.00	13.33	0.00	0.00
Άβδηρα	0.00	0.00	66.67	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	60.00	10.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	13.33	0.00	0.00	0.00	0.00	90.00	0.00	6.67	10.00	0.00
Σοφία	0.00	0.00	6.67	20.00	0.00	0.00	40.00	0.00	0.00	0.00
Θράκη	0.00	0.00	0.00	0.00	10.00	0.00	0.00	60.00	0.00	0.00
Φιλοσοφία	20.00	10.00	13.33	20.00	10.00	10.00	0.00	13.33	90.00	40.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	60.00

(b) F = 0.722



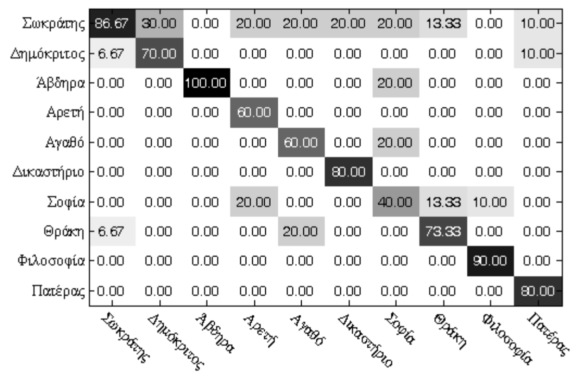
	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	80.00	0.00	6.67	20.00	0.00	0.00	40.00	6.67	0.00	0.00
Δημόκριτος	0.00	70.00	20.00	0.00	0.00	10.00	0.00	6.67	0.00	0.00
Άβδηρα	0.00	0.00	40.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	40.00	10.00	0.00	0.00	13.33	0.00	0.00
Αγαθό	0.00	0.00	0.00	20.00	80.00	0.00	0.00	0.00	0.00	10.00
Δικαστήριο	13.33	5.00	13.33	0.00	10.00	70.00	0.00	0.00	20.00	0.00
Σοφία	0.00	0.00	0.00	0.00	0.00	0.00	60.00	0.00	10.00	0.00
Θράκη	0.00	0.00	6.67	0.00	0.00	10.00	0.00	66.67	0.00	0.00
Φιλοσοφία	6.67	25.00	6.67	20.00	0.00	10.00	0.00	6.67	70.00	0.00
Πατέρας	0.00	0.00	6.67	0.00	0.00	0.00	0.00	0.00	0.00	90.00

(c) F = 0.680



	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	53.33	0.00	13.33	40.00	0.00	0.00	20.00	6.67	10.00	0.00
Δημόκριτος	0.00	70.00	6.67	0.00	0.00	0.00	0.00	6.67	0.00	0.00
Άβδηρα	0.00	0.00	60.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	40.00	10.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	70.00	0.00	0.00	0.00	0.00	0.00
Δικαστήριο	20.00	0.00	0.00	0.00	0.00	100.00	20.00	0.00	0.00	0.00
Σοφία	0.00	0.00	0.00	0.00	0.00	0.00	60.00	0.00	0.00	0.00
Θράκη	0.00	0.00	0.00	20.00	10.00	0.00	0.00	80.00	10.00	0.00
Φιλοσοφία	26.67	30.00	13.33	0.00	10.00	0.00	0.00	6.67	80.00	60.00
Πατέρας	0.00	0.00	6.67	0.00	0.00	0.00	0.00	0.00	0.00	40.00

(d) F = 0.670



	Σωκράτης	Δημόκριτος	Άβδηρα	Αρετή	Αγαθό	Δικαστήριο	Σοφία	Θράκη	Φιλοσοφία	Πατέρας
Σωκράτης	86.67	30.00	0.00	20.00	20.00	20.00	20.00	13.33	0.00	10.00
Δημόκριτος	6.67	70.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.00
Άβδηρα	0.00	0.00	100.00	0.00	0.00	0.00	20.00	0.00	0.00	0.00
Αρετή	0.00	0.00	0.00	60.00	0.00	0.00	0.00	0.00	0.00	0.00
Αγαθό	0.00	0.00	0.00	0.00	60.00	0.00	20.00	0.00	0.00	0.00
Δικαστήριο	0.00	0.00	0.00	0.00	0.00	80.00	0.00	0.00	0.00	0.00
Σοφία	0.00	0.00	0.00	20.00	0.00	0.00	40.00	13.33	10.00	0.00
Θράκη	6.67	0.00	0.00	0.00	20.00	0.00	0.00	73.33	0.00	0.00
Φιλοσοφία	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	90.00	0.00
Πατέρας	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	80.00

(e) F = 0.774

Figure 5.8: Confusion matrices for the five trials (a)-(e) of the 4th experimental setup. The average F-measure is 0.731.

It seems that the system performs almost 2% better than the first experimental setup, almost 3% worse than the second and about 10% worse than the third experimental setup, on average. This may be due to the nature of the problem of training a representative model for a word class, in terms of capturing as much intra-class variability as possible. The models generated from training sets containing similar images, record a lower degree of variability in the deformation model than models which originate from randomly selected images.

However, the variability to be obtained in the deformation model depends on the clustering outcome, namely, on the number of clusters and the images contained in each cluster. Regarding the spectral clustering algorithm, there are aspects that still remain unexplored, such as smoothing the similarities to a certain degree in order to account for higher inter and intra-writer variability.

Finally, according to the repeated random sub-sample validation task (section 5.2), models are created using random samples of subsets of the available images for a word class. This means that some images may never be selected in the validation sub-sample, whereas others may be selected more than once. In other words, validation subsets may overlap. By applying spectral clustering on all images belonging to a class, we can split the dataset to a fixed number training and validation sets. Then, we can alternate between training and validation sets in order to apply a more stratified k-fold cross validation procedure, where the value of k corresponds to the number of folds, namely, the number of clusters determined for spectral clustering. By these means, all images are considered in a structured way.

Chapter 6

Conclusion - Future Work

In the present dissertation, we have proposed a method for spotting keywords in segmented images of handwritten text, using shapes as word class queries, built directly from images. After determining the local contour features used to describe a word, we integrated them into a leaning framework which generated class representative models from a random subset of images belonging to that class. The performance of the proposed system in categorizing unknown words to known word classes was evaluated using a vocabulary of class-specific models. Moreover, a principled training approach was presented, in order to determine the number of models to be used for word spotting and the images comprising these models.

The presented methodology can be extended in several ways. In a segmentation-free approach, the system is able to recognize word instances of an already trained class model in a document image containing various words of different sizes and writing styles. The only prerequisite is that the training images should be annotated by their bounding boxes. Moreover, instead of using PAS features [22] to describe words, one could use 3AS (kAS for $k = 3$) [20] so as to capture more complex shapes of a word's characters. Hence, the variability recorded in the deformation model could be raised significantly.

Finally, the whole system was tested in a word spotting task using words inside a known vocabulary. One possibility to classify a word that does not exist in the vocabulary (no model is created for the class it belongs to), is to specify a threshold in the decision step of the classification process defined in sections 5.3 and 5.4. Scores that do not exceed such a threshold represent words out of the vocabulary and can be used for training a new model (increase the vocabulary's size).

BIBLIOGRAPHY

[1] T. Adamek and N. E. O'Connor. A multi-scale representation method for non-rigid shapes with a single closed contour. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(5):742–753, 2004.

[2] T. Adamek, N. E. O'Connor, and A. F. Smeaton. Word matching using single closed contours for indexing handwritten historical documents. *International Journal on Document Analalysis and Recognition*, 9(2):153–165, Apr. 2007.

[3] G. R. Ball, S. N. Srihari, and H. Srinivasan. Segmentation-based and segmentation-free methods for spotting handwritten arabic words. In *Proceedings of the* 10^{th} *International Workshop on Frontiers in Handwriting Recognition*, pages 53–58. Suvisoft, Oct. 2006.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, Apr. 2002.

[5] A. Bhardwaj, D. Jose, and V. Govindaraju. Script independent word spotting in multilingual documents. In *Proceedings of the* 2^{nd} *workshop on Cross Lingual Information Access (CLIA)*, pages 48–54, 2008.

[6] A. Bhardwaj, S. Kompalli, S. Setlur, and V. Govindaraju. An OCR based approach for word spotting in Devanagari documents. In *Proceedings of the* 15^{th} *SPIE Conference on Document Recognition and Retrieval*, volume 6815, pages 0–9, 2008.

[7] H. Bunke, S. Bengio, and A. Vinciarelli. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):709–720, June 2004.

[8] J. Cai and Z. LIU. Off-line unconstrained handwritten word recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(03):259–280, 2000.

[9] H. Cao, A. Bhardwaj, and V. Govindaraju. A probabilistic method for keyword retrieval in handwritten document images. *Pattern Recognition*, 42(12):3374–3382, Dec. 2009.

[10] J. Chan, C. Ziftci, and D. Forsyth. Searching off-line arabic documents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1455–1462, 2006.

[11] F. R. Chen, D. S. Bloomberg, and L. D. Wilcox. Spotting phrases in lines of imaged text. In *Proceedings of the SPIE Conference on Document Recognition II*, volume 2422, pages 256–269, 1995.

[12] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, Feb. 2003.

[13] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models: their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, Jan. 1995.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 886–893, June 2005.

[16] N. Doulgeri and E. Kavallieratou. Retrieval of historical documents by word spotting. In *Proceedings of the* 16th *SPIE Conference on Document Recognition and Retrieval (DRR)*, volume 7247, pages 1–10, Jan. 2009.

[17] J. Edwards, Y. W. Teh, D. Forsyth, R. Bock, M. Maire, and G. Vesom. Making latin manuscripts searchable using gHMM's. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 385–392. MIT Press, Cambridge, MA, 2005.

[18] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271, june 2003.

[19] S. Fernandez, A. Graves, and J. Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the* 17th *International Conference on Artificial Neural Networks ICANN*, pages 220–229, 2007.

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

[21] V. Ferrari, F. Jurie, and C. Schmid. Accurate object detection with deformable shape models learnt from images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[22] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *International Journal of Computer Vision*, 87(3):284–303, May 2010.

[23] V. Ferrari, T. Tuytelaars, and L. V. Gool. Real-time affine region tracking and coplanar grouping. In *Proceedings of the IEEE Computer Society Conference on*

Computer Vision and Pattern Recognition (CVPR), volume 2, pages 226–233. IEEE Computer Society, Dec. 2001.

[24] V. Ferrari, T. Tuytelaars, and L. V. Gool. Object detection by contour segment networks. In *Proceedings of the 9th European Conference on Computer Vision (ECCV) - Volume Part III*, pages 14–28, Springer-Verlag, Berlin, Heidelberg, 2006.

[25] A. Fischer, A. Keller, V. Frinken, and H. Bunke. HMM-based word spotting in handwritten documents using subword models. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)*, pages 3416–3419, Aug. 2010.

[26] V. Frinken, A. Fischer, and H. Bunke. A novel word spotting algorithm using bidirectional long short-term memory neural networks. In *Proceedings of the 4th Workshop on Artificial Neural Networks in Pattern Recognition*, volume 5998, pages 185–196, 2010.

[27] V. Frinken, A. Fischer, H. Bunke, and R. Manmatha. Adapting BLSTM neural network based keyword spotting trained on modern data to historical documents. In *Proceedings of the 12th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 352–357, IEEE Computer Society, Washington, DC, USA, 2010.

[28] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 34(2):211–224, 2012.

[29] M. Fritz and B. Schiele. Towards unsupervised discovery of visual categories. In *Proceedings of the 28th Conference on Pattern Recognition (DAGM)*, pages 232–241, Berlin, Heidelberg, 2006. Springer-Verlag.

[30] B. Gatos and I. Pratikakis. Segmentation-free word spotting in historical printed documents. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR)*, pages 271–275, July 2009.

[31] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 31(5):855–868, May 2009.

[32] A. Hill and C. J. Taylor. A method of non-rigid correspondence for automatic landmark identification. In *Proceedings of the 7th British Machine Vision Conference (BMVC)*, pages 323–332. BMVA Press, 1996.

[33] G. Jones, J. Foote, K. Sparck, and S. Young. Video mail retrieval: the effect of word spotting accuracy on precision. In *Proceedings of the 20th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 309–312, May 1995.

[34] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 604–610, oct. 2005.

[35] S. Khoubyari and J. J. Hull. Keyword location in noisy document images. In *Proceedings of the* 2nd *Annual Symposium on Document Analysis and Information Retrieval*, pages 217–231, Apr. 1993.

[36] K. Khurshid, C. Faure, and N. Vincent. Fusion of word spotting and spatial information for figure caption retrieval in historical document images. In *Proceedings of the* 10th *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 266–270, 2009.

[37] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 19(4):366–379, Apr. 1997.

[38] A. Kolcz, J. Alspector, and M. Augusteijn. A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis and Applications*, 3(2):153–168, 2000.

[39] S. S. Kuo and O. E. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):842–848, Aug. 1994.

[40] V. Lavrenko, T. M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proceedings of the* 1st *International Workshop on Document Image Analysis for Libraries*, pages 278–287, 2004.

[41] B. Leibe and B. Schiele. Scale-invariant object categorization using a scale-adaptive mean-shift search. In C. E. Rasmussen, H. Bólthoff, B. Schφlkopf, and M. Giese, editors, *Pattern Recognition*, volume 3175 of *Lecture Notes in Computer Science*, pages 145–153. Springer Berlin Heidelberg, 2004.

[42] S. Levy. Google's Two Revolutions. *Newsweek*, Dec./Jan. 2004. Also available as http://goo.gl/uB1ur.

[43] Y. Leydier, F. L. Bourgeois, and H. Emptoz. Omnilingual segmentation-free word spotting for ancient manuscripts indexation. In *Proceedings of the* 8th *International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 533–537, 2005.

[44] Y. Leydier, F. L. Bourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recognition*, 40(12):3552– 3567, 2007.

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

[46] Y. Lu and C. L. Tan. Word spotting in chinese document images without layout analysis. In *Proceedings of the* 16th *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 57–60, 2002.

[47] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958.

[48] U. V. Luxburg, M. Belkin, O. Bousquet, and Pertinence. A tutorial on spectral clustering. *Statistics and Computing*, 17(4), 2007.

[49] R. Manmatha and W. Croft. *Word spotting: indexing handwritten archives*, chapter 3, pages 43–64. MIT Press, 1997.

[50] R. Manmatha, C. Han, and E. Riseman. Word spotting: a new approach to indexing handwriting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 631–637, 1996.

[51] R. Manmatha and T. Rath. Indexing of handwritten historical documents - recent progress. In *Proceedings of the Symposium on Document Image Understanding Technology (SDUIT)*, pages 77–85, Apr. 2003.

[52] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machince Intelligence*, 26(5):530–549, May 2004.

[53] R. F. Moghaddam and M. Cheriet. Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In *Proceedings of the* 10th *International Conference on Document Analysis and Recognition (ICDAR)*, pages 511–515, July 2009.

[54] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001.

[55] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the* 9th *European Conference on Computer Vision (ECCV) - Volume Part II*, pages 575–588, Berlin, Heidelberg, 2006. Springer-Verlag.

[56] V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis. Handwritten document image segmentation into text lines and words. *Pattern Recognition*, 43(1):369 – 377, Jan. 2010.

[57] F. Perronnin and J. A. Rodríguez-Serrano. Fisher kernels for handwritten wordspotting. In *Proceedings of the* 10th *International Conference on Document Analysis and Recognition (ICDAR)*, pages 106–110, Washington, DC, USA, 2009.

[58] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, Jan. 2000.

[59] T. Quack, V. Ferrari, B. Leibe, and L. V. Gool. Efficient mining of frequent and distinctive feature configurations. In *International Conference on Computer Vision (ICCV)*, Oct. 2007.

[60] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE Computer Society*, 77(2):257–286, Feb. 1989.

[61] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 521–527, June 2003.

[62] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal on Document Analysis and Recognition*, 9(2-4):139–152, 2007.

[63] T. M. Rath, R. Manmatha, and V. Lavrenko. A search engine for historical manuscript images. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 369–376. ACM, 2004.

[64] J. A. Rodríguez and F. Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *Proceedings of the 1st International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Aug. 2008.

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

[66] J. A. Rodríguez-Serrano, F. Perronnin, G. Sanchez, and J. Llados. Unsupervised writer style adaptation for handwritten word spotting. In *Proceedings of the 19th International Conference on Pattern Recognition (ICPR)*, pages 1–4, Dec. 2008.

[67] J. L. Rothfeder, S. Feng, and T. M. Rath. Using corner feature correspondences to rank word images by similarity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, volume 3, page 30, June 2003.

[68] E. Saykol, A. K. Sinop, U. Gudukbay, O. Ulusoy, and A. E. Cetin. Content-based retrieval of historical ottoman documents stored as textual images. *IEEE Transactions on Image Processing*, 13(3):314–325, Mar. 2004.

[69] G. Scott and H. C. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society B on Biological Sciences*, volume 244, pages 21–26, Apr. 1991.

[70] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 503–510, Oct. 2005.

[71] S. Srihari, H. Srinivasan, P. Babu, and C. Bhole. Spotting words in handwritten Arabic documents. In *Proceedings of the 13th SPIE Conference on Document Recognition and Retrieval*, volume 606702, pages 1–12, 2006.

[72] S. Srihari, H. Srinivasan, C. Huang, and S. Shetty. Spotting words in latin, devanagari and arabic scripts. *Indian Journal of Artificial Intelligence*, 16(3):2–9, 2006.

[73] K. Terasawa, T. Nagasaki, and T. Kawashima. Eigenspace method for text retrieval in historical document images. In *Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 437–441, Sept. 2005.

[74] M. Wollmer, F. Eyben, J. Keshet, A. Graves, B. Schuller, and G. Rigoll. Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks. In *Proceedings of the* 34th *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3949–3952, Washington, DC, USA, 2009.

[75] B. Zhang and S. N. Srihari. Binary vector dissimilarity measures for handwriting identification. In *Proceedings of the* 10th *SPIE Conference on Document Recognition and Retrieval*, volume 5010, pages 28–38, 2003.

[76] B. Zhang, S. N. Srihari, and C. Huang. Word image retrieval using binary features. In *Proceedings of the* 11th *SPIE Conference on Document Recognition and Retrival*, volume 5296, pages 45–53, 2004.

[77] G. Zipf. Human behaviour and the principle of least-effort. Addison-Wesley, Cambridge, MA, 1949.

Curriculum Vitae

Giotis Angelos was born in Ioannina in 1986. He was admitted at the Computer Science Department of the University of Ioannina in 2004. He received his BSc degree in computer science in 2010 with degree 6.69 “very good”. That year he became a postgraduate student at the same institution from where he graduated in October 2010. His main research interests lie in the field of Computer Vision and Pattern Recognition.