Handwritten Document Image Retrieval

Xi Zhang

School of Computing

National University of Singapore

Supervisor: Prof. Chew Lim Tan

A thesis submitted for the degree of

Philosophy of Doctor (PhD)

November 2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Zhang Xi

Xi Zhang

13 August 2014

I would like to dedicate this thesis to my beloved parents and Su Bolan for their endless support and encouragement.

Acknowledgements

I would like to express my deep and sincere appreciation to my PhD supervisor Professor Chew Lim Tan, in School of Computing, National University of Singapore. He is very kind and provides a lot of support to my research work. Moreover, he always makes my research environment full of freedom, so that I can really focus on the works what I am interested in. With his wide knowledge and constructive advice, I am inspired with various ideas in order to solve the challenges and open my eyes to different new directions. Without his generous help, this thesis would not have been possible.

I also would like to thank all my lab fellows, who always have great ideas and work very hard. I can discuss difficult problems with them and obtain exciting solutions. They are Dr. Chen Qi, Situ Liangji, Tian Shangxuan, Dr. Sunjun, Dr. Li Shimiao, Dr. Gong Tianxia, Dr. Wang Jie, Dr. Liu Ruizhe, Dr. Mohtarami Mitra, Ding Yang, who help me a lot in my research work or non-academic aspects, especially give me a very happy research environment. Furthermore, I wish to extend my warm thanks to all my friends who came across my life during my four-year PhD study in Singapore, I would not be able to overcome difficulties and have so many happy and memorable moments without them. I am so sorry that I can only list some of them: Xu Haifeng, Yu Xiaomeng, Li Hui, Dr. Shen Zhijie, Dr. Wang Guangsen, Dr. Wang Chudong, Fang Shunkai, Dr. Li Xiaohui, Dr. Cheng Yuan, Dr. Zheng Yuxin, and etc.

Last but not least, I would like to give my most sincere gratitude to my parents, who love me endlessly and selflessly. They always provide their support to anything I would like to do, and understand any my bad mood unconditionally. I also wish to express my special appreciation to my husband Dr. Su Bolan, who accompanies me every day, no matter happy or sad hours and gives me a colourful life, full of love.

Contents

List of Figures

List of Tables

Abstract

A vast amount of information is stored as text format in large databases or digital libraries. Users can easily access them by traditional text retrieval methods which many researchers have worked on for decades. However, paper-less life is impossible nowadays and many important and valuable documents are available only as imaged format. Therefore, it is now an important and urgent issue to let users access these imaged documents effectively and efficiently, similar to retrieving text format documents produced by computer software. Information retrieval for handwritten document images is more challenging due to the difficulties in complex layout analysis, large variations of writing styles, and degradation or low quality of historical manuscripts. Optical Character Recognition (OCR) can convert word or text line images directly to their transcriptions and traditional text retrieval methods can be used to retrieve user specified information. However, OCR needs large segmented and labelled training data, and recognizing the entire documents is a waste of time if the objective is to just to retrieve an imaged document without having the process the recognized text. Furthermore, OCR may provide poor recognition results due to unconstrained writing styles. In order to overcome the limitations of OCR, keyword spotting becomes an alternative way to retrieve handwritten documents. It only needs the features extracted from the imaged documents, and has no use of the ASCII content. In view of large variations in handwriting styles, this thesis will first present a method for extracting text lines from multilingual handwritten documents. Secondly, a combination of two well-trained networks is used to increase the recognition performance for word images. Thirdly, Heat Kernel Signature (HKS), which can better tolerate non-rigid deformations than gradient information, is used to represent the key points detected on the documents, and to achieve word image matching and segmentation-free

keyword spotting. Finally, we will present our proposed retrieval method which can achieve writer identification and content relevance retrieval based on the same set of extracted features.

Chapter 1

Introduction

1.1 Background and history

With the development of computer and network, a large amount of documents are stored as text format, and users can easily access useful information by traditional text retrieval techniques which many researchers are interested in and working on. However, in recent years, people are devoting themselves to protect important and valuable documents, most of which are printed or written by single hand, such as historical books or manuscripts, business contracts or letters, published books or magazines, bank cheques, handwritten notes or records, and so on. In order to preserve and archive precious information in these documents, and also let more people access them conveniently, the documents are always scanned into large digital databases as image format. How to deal with these imaged documents, how to let users access and retrieve them efficiently and effectively similar to the text format documents become an important issue. Firstly, predominant document image retrieval is achieved by applying traditional information retrieval methods to the OCR'ed (Optical Character Recognition) transcriptions of document images. In other words, in order to retrieve imaged documents, document images are converted into text format which is machine readable using OCR, and then conventional text retrieval techniques are applied to achieve retrieval tasks, such as the methods used in The Heinz Electronic Library Interactive On-Line System (HELIOS) (5), Excalibur EFS and PageKeeper from Caere. OCR can do well with machine printed documents in which character font and size can be predefined, and text and background can be distinguished easily, but OCR cannot do a good job if the original documents are of low quality containing noise, the font and language are rare, or the content is handwritten which includes different kinds of variations of characters or words among different writers or even for the same writer. Consequently, we cannot preserve the imaged documents as full-text format by applying OCR on the whole documents, especially when containing non-text content that cannot be converted with sufficient accuracy completely. Either, we cannot directly index converted documents which may contain some kinds of errors because of the weakness of OCR discussed above. Therefore, we cannot provide reliable retrieval systems for users. Motivated by these observations, some efforts are concentrated on tolerating OCR errors or improving OCR results by using OCR candidates (6) (7) (8) (9) (10) (11) (12) (13) (14) (15) (16) (17) (18). Besides these methods, an alternative, another from a different perspective way is available, namely keyword spotting with no need for correct and complete character recognition, but directly characterizes imaged document features at the character-level, word-level or even document-level, and manipulate retrieval tasks efficiently even for imaged documents containing both text and non-text content, such as graphs, forms or natural images. The essentially idea inside keyword spotting is representing characters or words shape features extracted directly from imaged documents instead of complete recognition by OCR. Spitz presented some research on keyword spotting using character shape code (19) (20) (21) (22). In the proposed methods, deciding the proper number of bits used for indexing is an important issue. These methods are simple and efficient, but with drawback of ambiguity. Many other efforts are made to avoid directly character recognition. However, in order to obtain the character codes, character segmentation must be implemented correctly which cannot be applicable in some cases, such as when the characters are interconnected or overlapped resulting in segmentation errors, so that, word-level methods were proposed which treat a single word as a basic unit for recognition in (23) (24) (25) (26) (27) (28). (29) introduces many strategies for character level or word level recognition. But, similar to character level methods, word level methods also suffer from word segmentation errors. In order to overcome segmentation problems, segmentation free approaches were proposed in (30) (31).

Nowadays, more research focus is on handwritten document retrieval because a large amount of valuable historical manuscripts written by hand are scanned into databases as digital format for public access, and there are also other kinds of important handwritten documents which need to be preserved for a long enough time and printed versions of which are not available. Due to the characteristics of handwritten documents, more efforts are needed. Besides, we can see that most of the methods did the experiments on relative small size of data, but large scale collection of documents are becoming a focus (32) (33), so that, how to index and retrieve imaged documents on large scale such as millions of pages, with low computation cost and high speed, is an imperative problem.

Always, small, but important aspects for document image retrieval may have much effect on performance. For instance, different fonts have their distinguished, intrinsic patterns, so features which are perfect for one kind of fonts may not be adequate for another. (34) proposed a font adaptive word indexing method. However, font is one of many characteristics the imaged document has, and many other features should be considered carefully to adapt for different situations, such as different languages, degraded documents with noise or handwritten documents with variant writing styles, etc. (35) (36) (37) dealt with Chinese documents, (38) recognized Japanese documents, and (39) dealt with Urdu database. (40) (30) (41) reported language independent methods and (42) recognized documents containing two kinds of languages, Chinese and English. What is more, researchers found out that traditional algorithms cannot solve partial word problem. For example, ”develop” is the partial word for ”development”, ”developed”, ”develops”, etc. In order to retrieve the keyword ”develop”, the other formats containing ”develop” will not be included in the result set. (43) tried to solve this problem.

1.2 Motivations

A large amount of resources are created as structured content which are produced by computer software and can be indexed and retrieve easily, but even larger amount of documents are in paper format and should be scanned into digital databases as image format some of which are quite precious and important for people to preserve permanently as electronic format and widely spread around the world by internet, such as published books and magazines to avoid potential unauthorized copying and distribution, signed business contracts for legal reason, bank cheques or bills for protecting privacy, historical documents and manuscripts, etc. Imaged documents should be provided for users to access and retrieve including searching keywords though out documents, finding similar documents which contain close subject, or checking whether the document contains relevant content the user desires.

However, storing the scanned documents in the databases only as image format cannot achieve the above tasks because these documents are just images, with no information about the content. At the beginning, researchers try to convert imaged documents into traditional text format documents by OCR, so that, users can easily deal with these documents as text format documents by conventional information retrieval systems. But some problems are arising. Firstly, OCR needs good document quality, but some documents are severely degraded because of environment, long preserved time, low quality of scanning devices, etc. Secondly, OCR cannot recognition handwritten documents correctly which contain variant writing styles between different writers or even for the same writer. Thirdly, OCR can recognize separate characters quite well, but cannot manage interconnected or overlapped characters. Fourthly, completely converting imaged document by OCR to machine readable code is time consuming and inefficiently which waste much time, labour and money, because of large size of digital databases and algorithm complexity. Fifthly, OCR can manage several relative popular languages. For rare languages, OCR does not work. Therefore, nowadays, imaged documents are stored as image format without complete recognition and conversion by OCR, but with adequate index for access and retrieval.

To achieve document image retrieval, several steps are necessary, including noise removal, feature extraction, choosing matching algorithm and indexing documents. For each of these steps, many approaches are proposed to improve the recall and precision, in which many kinds of situations are taken into account during researches. For example, different languages have different kinds of distinct features and we should apply different methods respectively, so one method can get better performance for English documents, but may obtain worse results for Chinese documents. Especially, for a certain kind of language, if we can find the distinguished characteristics of it, a particular simple algorithm may get even tremendously excellent performance. But we also face some multi-language imaged documents which need language independent algorithms. Besides, different kinds of degradation for the imaged documents should be treated differently and properly, and with the size of digital databases increasing rapidly, computational speed is becoming an important consideration.

We have considered text content in imaged documents as discussed above, but there are other kinds of content on the handwritten documents, such as non-textual content, like graphs, logos, signatures, etc. How to separate them from text content, how we can retrieve text information in complex structured documents, and how to figure out whether the results of keyword retrieval are from text content or non-text content are the problems we should face. Furthermore, with the rapid development of computer and information techniques, new problems and requirements we have never considered will appear. In document image retrieval research area, variety of aspects about the document inherent characteristics should be considered, and for different applications about imaged documents should be treated and solved differently in order to achieve various aspects and needs.

1.3 Aims and Scope

In this thesis, our aim is to propose methods which can improve the performance for handwritten document image retrieval. The specific objectives are presented as following:

1. Propose a text line segmentation method for multilingual handwritten documents based on seam carving, but with the constraints to lead the energy to be passed along the main body parts of text lines;

2. Combine two trained networks to improve the handwritten word recognition results, based on our proposed method to split the training data;

3. Apply Heat Kernel signature (HKS) to represent handwritten documents. HKS has been proven to perform better than SIFT descriptor for non-rigid deformations which is always the case in handwriting scenarios. We propose different methods for word image matching and segmentation-free keyword spotting based on HKS.

4. Instead of applying Non-Sampled Contourlet Transform (NSCT) on the whole documents, NSCT is only generated on the local patches centered at each detected keypoint, so that NSCT can be used not only for writer identification, but also for document retrieval according to content relevance based on our proposed keyword spotting methods.

There are many different kinds of handwritten documents, such as envelops, forms, notes, etc. In this thesis, we focus on the documents only containing text, without non-textual content, such as graph, logo, or table, because our study only focuses on retrieving the textual content, including extracting text lines, matching or recognizing word images, or spotting query words and retrieving relevant documents.

1.4 Chapter Overview

In the rest of this thesis, a text line segmentation method is presented in Chapter 2, which applies constrained seam carving on the whole documents and extracts the text lines by calculating the energy map only once. Based on the extracted text lines, word recognition using Bidirectional Long Short-Term Memory (BLSTM) with Connectionist Temporal Classification (CTC) on word images segmented from the text lines are described in chapter 3, and instead of using one trained network, two well-trained networks are combined to improve the recognition results. Due to the limitations of the supervised learning based methods, a word image matching method and a segmentation-free keyword spotting method without training are presented in chapter 4 and 5 respectively. In Chapter 6, we will present how to retrieve relevant documents based on both writer information and content relevance. At last, chapter 6 summarizes the work done so far and discusses the future work.

Chapter 2

Text Line Segmentation

In this chapter, we will present a language-independent method to extract text lines from handwritten document images. Our proposed method is based on seam carving algorithm, which has been already used for text line segmentation. However, in order to tolerate multi-skewed text lines even in the same document image, we proposed a constrained seam carving method, which can constrain the energy to be passed along the connected components in the same text line as much as possible. Moreover, our proposed method can extract all the text lines by computing the energy map only once.

2.1 Introduction and Related Works

Text line segmentation is a very crucial step for Optical Character Recognition (OCR) (3) and keyword spotting (44) (45), both of which are used to provide reliable information retrieval throughout a large amount of document images. However, unlike printed documents, which have a finite set of constrained layouts, finite types of fonts, pre-defined sizes of characters and well-separated text lines, handwritten documents always contain unconstrained writing styles, such as long ascenders or descenders which may connect different text lines together, multi-skewed text lines even in the same document image, and small floating strokes. All these unpredictable situations can lead to much difficulties in text line segmentation tasks.

There are two main broad categories of segmenting text lines, one is top-down approaches, and the other is bottom-up approaches. As the name suggests, top-down methods try to estimate the locations of the candidate text lines first. Then the estimation is refined by assigning components to the text lines which they belong to with higher probabilities. Finally, large components, which touch multiple text lines are split into pieces, and these pieces are assigned to different text lines separately. On the other hand, bottom-up methods try to find local components first, which are always the connected components (CCs), and then group the components together into separate text lines based on different types of grouping algorithms.

For top-down methods, in (46), document images are first divided into separate column chunks, the width of which is 5% of the width of the document. Horizontal projection profiles of the foreground pixels are generated for each chunk. Based on the smoothing projection profiles, the valleys in every chunk, where the number of foreground pixels are minimum between two consecutive peaks, are located and used to indicate the positions where two text lines should be separated. The initially estimated text lines are extracted by connecting valleys in each profile with the closest ones in the previous profile. Separate lines are drawn horizontally from left to right, and for unused valleys, separate lines are drawn horizontally at the same position as in the previous profile. When a separate line encounters a component, bi-variant Gaussian densities are used to capture spatial features, and a decision is made to assign the component to the optimal text line, above or below.

Besides, (47) applied a steerable directional filter to get an Adaptive Local Connectivity Map (ALCM) of the original document. Using multiple directions of the filters, the convolution results can reflect how likely one text line appears at each position. The estimation is made using the maximum response of the convolutions. In ALCM, large values always correspond to the pixels lying in the dense text regions. Therefore, after applying a local adaptive binarization method, regions with dense text are retained, presenting the entire text lines or partial ones. Finally, components crossing multiple text lines are separated and other unassigned components are allocated to the spatially closest text lines.

For bottom-up methods, a Hough Transform based method was proposed in (48). Document images are binarized and enhanced first, and the connected components (CCs) are extracted. Based on the average height and width of all CCs, CCs are grouped into three exclusive subsets: large components, small CCs, such as accents, and the remaining normal sized CCs, which constitute the main body parts of the text lines. For each CC in the third subset, it is partitioned into equal-sized blocks.

The Hough transform is applied to the gravity center points of all blocks, and assign a CC to one text line if half of the points are assigned to this text line, according to the accumulator array. In the post processing step, the CCs in the second subset are assigned to the closest text lines, and the CCs in the first subset are either assigned to the text line they only lie on, or separated into different parts, and assigned to individual parts separately.

In (49), the distances between CCs are measured based on a special designed metric using supervised learning, which can enlarge the distance between two neighbouring CCs in different text lines, and narrow the distance if they are lying in the same text line. After removing small or large CCs, documents are represented by a graph, each node of which is a normal sized CC, and with the trained distance metric on every pair of neighbouring CCs, a Minimal Spanning Tree (MST) is built. By cutting the edges, the end nodes of which belong to different text lines, CCs are grouped into different text lines. Unassigned CCs are allocated applying similar post processing methods mentioned above.

There are always debates between top-down and bottom-up methods. Top-down methods may suffer from large curved documents, or multiple touching text lines, and bottom-up methods focus on local features, and many complicated computation and heuristics are needed. (50) presents a review of existing methods for extracting hand-written text lines.

In this chapter, we propose a method based on seam carving to capture the global characteristics of the documents, which was first used in (51) for language-independent text line extraction. We can find seams from left to right, or from top to bottom, and the seams with maximum accumulated energy indicate the possible locations of text lines. The energy we will use is the intensity values on the documents. Therefore, a seam passing through the main body part of a text line can accumulate much more energy. In order to capture more information in local regions, we constrain the scope and orientation of passing the energy, in order to transmit the energy of one point mainly to the neighbours in the same text line, and also avoid making the points accept energy passed from too far away or by the ones in the different text lines. Moreover, we extract all the text lines by computing the energy map only once, instead of recomputing the energy map after one text line is extracted, as in (51). We also smooth the generated seams by polynomial fitting, in order to correct the sharp orientation changes along seams.

2.2 Seam carving

Seam carving was first proposed for content-aware image resizing in (52). The seams with minimum gradient information will be removed to keep the important content of the image. For an image to be resized, an energy map is generated by the following energy function (52):

$$e(\mathbf{I}) = \mid \frac{\partial}{\partial x}\mathbf{I} \mid + \mid \frac{\partial}{\partial y}\mathbf{I} \mid \tag{2.1}$$

where $\mathbf{I}$ is the image with the size of $r \times c$, and $\frac{\partial}{\partial x}\mathbf{I}$ and $\frac{\partial}{\partial y}\mathbf{I}$ are the horizontal and vertical gradients respectively.

Seams in the horizontal and vertical orientations are extracted based on the energy map. Every seam is a connected path, all the pixels along which have lower energy values in its horizontal or vertical neighbours. Therefore, these extracted seams can be removed because they contain less information, if we need to shrink the image.

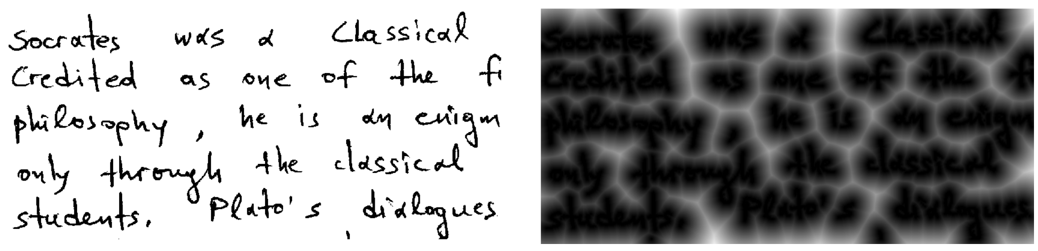
However, in order to extract text lines, we want to find the seams crossing more strokes, and these seams can indicate the locations where one text line probably appears. In (51), the energy map is calculated using Signed Distance Transform (SDT). In SDT, the pixels on the strokes have negative values, and the others in the background have positive values. As a result, horizontal seams following local minima represent the positions of the candidate text lines.

As shown in Fig. 2.1, the SDT of a binary image in Fig. 2.1(b) indicates that the nearer the points to the center axis of the strokes, the lower the values are in SDT. Therefore, in the intra-space of consecutive words or text lines, the values are very high.

Assuming $E(\mathbf{I})$ is the energy map based on SDT, seams are generated using a minimum energy accumulator M, which is constructed as following (51):

$$M(i,j) = 2 \times E(i,j) + min_{l=-1}^{1}[(M(i+l, j-1)] \tag{2.2}$$

where $i \in [1,r], j \in [1,c]$, and we only consider continuous connected seams.


Socrates was a Classical
Credited as one of the fi
philosophy , he is an enigm
only through the classical
students. Plato's dialogues

(a) The original binary image. (b) SDT of the image in (a).

Figure 2.1: An example of a binary image and its SDT. In SDT, the darker one point is, the lower value its SDT has.

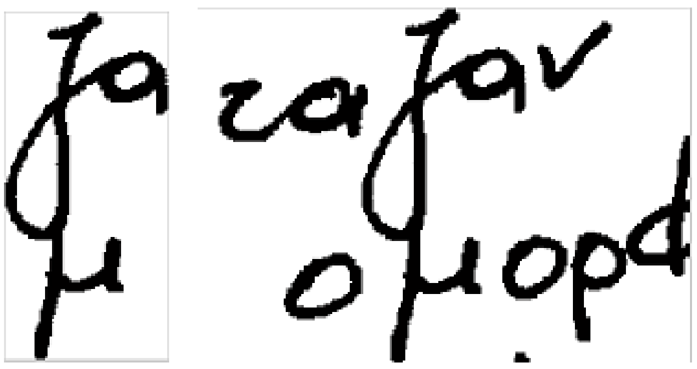
M can accumulate minimum energy for every seam, from left to right, and the seam with minimum accumulated energy is generated in an inverse direction, from right to left, by choosing the minimum value in the right-most column in M and traversing backward. In (51), after each text line is extracted, the energy map need to be recomputed, this may cause large computation effort. In the next section, we will propose a method to detect all the text lines, by computing the energy map only once.

2.3 Our proposed method

2.3.1 Preprocessing

The average height AH and the average width AW of CCs are first calculated for each document, and CCs are classified into three classes: small stokes, large components, and ordinary CCs. Small strokes are mostly located relatively far from the central axis of the text lines, and large components are the CCs with long ascenders or descenders, either only belonging to one text line or connecting multiple text lines. These two types of CCs may cause the seams jumping between different text lines. In order to avoid unwanted disturbance, small strokes are removed, and for the large components, only the parts with high density values in the horizontal histogram are kept. For example, if a large component is detected, as shown in Fig. 2.2(a), we first include all the other strokes in its $3 \times AW$ forward and backward columns, as shown in Fig. 2.2(b). The corresponding horizontal projection histogram is shown in the right part of Fig. 2.3. As shown in Fig. 2.4, we smooth the histogram by convolving the histogram with a

Gaussian kernel with mean AH, and standard deviation $AH/4$, and remove the parts of the large component with intensities lower than a threshold.



(a) (b)

Figure 2.2: A large components and its neighbouring strokes lying in the same text lines. (a) is a large component detected. In (b) are the horizontal neighbouring strokes.

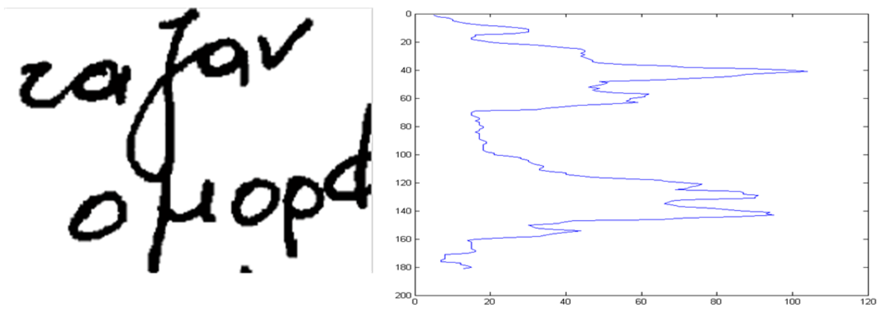


Figure 2.3: The horizontal histogram projection of the image in Fig. 2.2(b)

Unlike previous proposed methods, which always discard large components in the text line extraction process, we keep their main body parts. Because large components may be constructed by two long words in different text lines, due to their long ascenders or descenders, if we just discard them, there will be a large gap between their neighbouring CCs, so that the seams may easily jump to other text lines when they encounter these large gaps. Therefore, we keep main body parts of large components, not only avoiding large gaps, but also letting the main body parts contribute to the energy passing, positively.

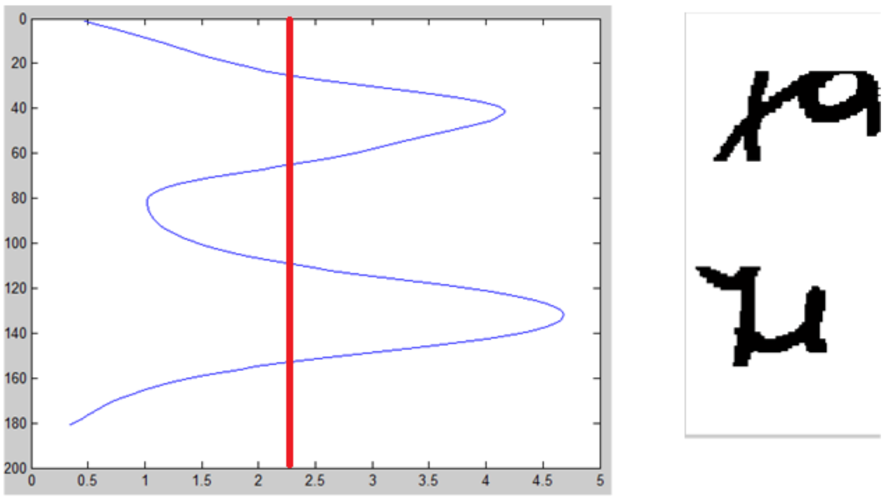


Figure 2.4: A threshold in the smoothed histogram is chosen, which is indicated by the red line. The foreground pixels lying in the rows with the values smaller than the threshold are removed.

2.3.2 Energy function

Distance maps are calculated separately for the points inside the components and others in the background. For the points inside the components, we first extract contours of all components on the documents and calculate Euclidean distance transform, denoted as C. Only the values on the components are kept. Therefore, the points along the central axis of strokes have larger values. For the points in the background, we extract skeleton of components, and calculate the Euclidean distance transform, denoted as S. Only the values in the background are kept, so that, points far from the central axis of strokes have larger values.

In order to enhance the energy along the writing orientation of the text lines, we convolve C by an ellipse-shaped Gaussian kernel, with major and minor axes of $3 \times AH$ and AH respectively. The Gaussian kernel is normalized by scaling all the values into [0,1]. We use multiple Gaussian kernels with different rotation angles for each pixel, and choose the one with the maximum energy value. By applying Gaussian kernels, the intra-space between two words in the same text line can accept energy from the components on the left and right. The energy can flow along the writing orientation in the intra-space between components. Therefore, the seams can follow the writing orientation, instead of jumping among different text lines.

Finally, we turn the sign of the positive values in C to negative and assign the other zero values to the ones in S at the same positions. The final signed distance transform is denoted as $E'(\mathbf{I})$.

2.3.3 Energy accumulation

Energy is accumulated from left to right, and the energy can be passed to the points in the following columns on the right. In some cases, the intra-space of two text lines are very narrow, such that the components in one text line may accept the energy from different text lines. If the energy accepted from the same text line is lower, the seam along the text line with lower energy will jump to other text lines.

Moreover, we do not want the energy to be passed too far away. For example, if two text lines in one document are with different lengths and they are all aligned to the right. The longer one can always accumulate more energy than the shorter one in every column, and the components in the shorter text line can be easily affected by the larger energy in the longer text line, so that the seam will also can jump between these two text lines.

In order to weaken the effect of the energy passed by a point from too far away, we accumulate the energy by weighting based on the distances, indicating from how far away the energy is passed. We also set a maximum distance, so that the energy from the distance larger than the maximum distance will be discarded. Besides, for the new energy accumulation matrix M', we generate $Hist$ for all the points in the column on the left of the column under consideration, recording all the energy accumulated so far and the distances where each energy is from. The longest length of $Hist$ is set to $\frac{1}{2}$ of the width of the document, and the elements in $Hist$ is first-in-first-out. If the size exceeds the limitation, the first added element will be discarded and the new element is added at the last position.

We initialize $M'(:,1)$ to $E'(:,1)$, and $Hist(:,1)$ to $E'(:,1)$. M' is constructed as following:

$$dist = length(Hist(i-1, j-1)) : -1 : 1 \tag{2.3}$$

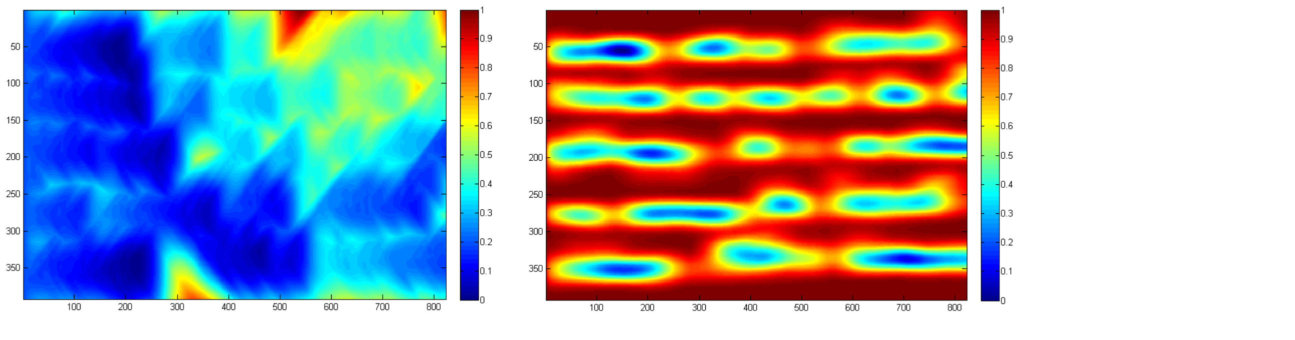
$$e1 = Hist(i-1, j-1)./dist \tag{2.4}$$

$$e2 = Hist(i, j-1)./dist \tag{2.5}$$

$$e3 = Hist(i+1, j-1)./dist \tag{2.6}$$

$$M'(i,j) = 2 \times E'(i,j) + min(e1, e2, e3) \tag{2.7}$$

where $dist$ is used to denote the distance for each energy, which has been already accumulated. For example, if the length of $Hist(i-1, j-1)$ is l, the energy accumulated from the farthest distance is l along the minimum energy accumulated path and $dist = [l, l-1, ..., 1]$. The first element in $Hist(i-1, j-1)$ is the farthest, and the last element is the nearest, namely the newest added one. So that, the energy in $Hist(i-1, j-1)$ is normalized by dividing the corresponding distances, in order to weaken the effect of the energy from far away, and enhance the influence of the neighbouring energy. When we select the minimum one among $e1$, $e2$ and $e3$, for example $e1$ is selected, then $Hist(i, j)$ $= [Hist(i-1, j-1), E'(i, j)]$. After all the points in column j are updated, the values stored in $Hist(:, j-1)$ can be discarded to save storage space.



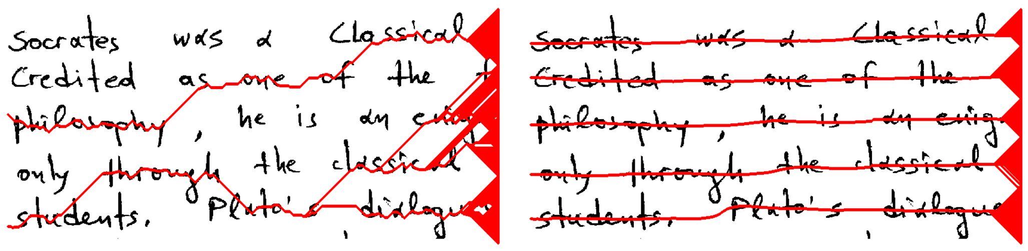
(a) M using Eq. 2.2. (b) M' using Eq. 2.7.

Figure 2.5: The energy accumulation matrix for Fig. 2.1(a). The energy values are scaled to [0,1] for visualization.

As shown in Fig. 2.5, both M and M' are calculated based on the same distance map, however, in Fig. 2.5(a), we can see that from left, the energy is propagated with a nearly 90 degree flare angle facing to the right horizontally. So that the energy can be passed across different text lines. In Fig. 2.5(b), with our proposed method, the energy is constrained to be passed along the same text lines, and interfering with the neighbouring text lines above and below can be avoided.

2.3.4 Seam extraction

At the end of constructing M', from every cell in the last column, we generate all seams from right to left and get a set of connected horizontal seams, denoted as $Seams$. If the height of the document is H, there will be H seams in $Seams$. Fig. 2.6 shows the seams we found using normal seam carving and our proposed method. We can see that, in Fig. 2.6(a), the seams which are generated based on M, jump among different text lines, if we group the components touching the same seams together, we cannot get correct text lines, and many normal size components are not located on any extracted seams. This situation is caused by large intra-space between two words in the same text line. However, in Fig. 2.6(b), the seams with our proposed method are generated correctly, all of which are along the central axis of the components, even though the intra-space between some words are large.


Socrates was a Classical
Credited as one of the
philosophy, he is an enig
only through the classical
students, Plato's dialogue

(a) Apply the method presented in Section 2.2. (b) Our proposed method.

Figure 2.6: Seams generated by M and M' in Fig. 2.5. The red lines indicate the extracted seams.

According to Fig. 2.6(b), the seams only have 5 different beginning locations in the first column (the left most column) on the document, namely, the starting positions of candidate text lines. Therefore, in $Seams$, we group the seams with the same value in the first position into one set, denoted as $Seams_i$, $i \in [1, n]$, where n is the number of candidate text lines. We only keep one seam in each set, which consists mostly smooth and similar writing orientation in any local area. Therefore, we apply the polynomial curve fitting to every seam in one set, and choose the one with minimum distance to the fitted curve. The final seams we found are shown in Fig. 2.7, denoted as s_i, $i \in [1, n]$.

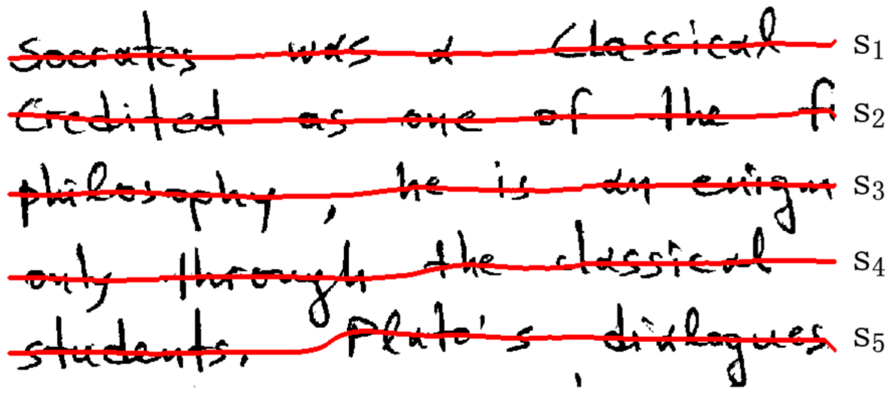


Figure 2.7: The final seams detected by our proposed method. There are total five seams, indicating the central axis positions of five text lines.

Due to the lengths of some text lines being too short, there may be no seams detected for them. Therefore, we check the regions between two detected seams. If there are many normal sized components that do not intersect with any seam, a missing text line is detected. We use the corresponding portion in M', which can cover the missing text line, and generate a new seam.

2.3.5 Postprocessing

After we generate all seams, for each s_i, we first put all the normal sized components, which only intersect with one s_i, into a component set c_i. For the remaining components, we will handle them in the following four cases separately:

Case 1: If a large component only intersects with one seam, then we just put them into the corresponding component set;

Case 2: If a large component does not intersect with any seams, we will assign them to the seam which is closest to its main body part, ignoring the long ascenders or descenders;

Case 3: If a large component intersects with multiple seams, we first thicken the intersected seams with height AH as text regions, and check the percentage of foreground

pixels of the main body part in the large component lying in each text region. If only one text region contains more than 70% of the foreground pixels, the large component should belong to this text region. If more than one text regions contain similar percentage of the foreground pixels, the large component should be split and assigned to the separate component sets. The split method we use was proposed in (48). Fig. 2.8 shows an example of splitting large components, which across two text lines;
Case 4: For small components, we assign them to the closest text lines.

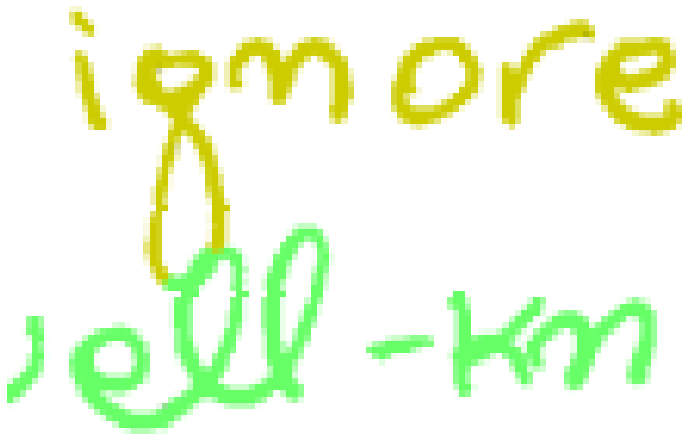


Figure 2.8: Split a large componnet into two parts, and the components belonging to the same text line are marked as the same color.

2.4 Experiments and Results

2.4.1 Evaluation method

Let I denote all the foreground pixels in one testing document, G_j the set of pixels inside the j ground truth region, and R_i the set of pixels inside the i results region. $T(s)$ is a function counting the set of pixels in s, and the matrix $MatchScore(i, j)$ is used to describe how the result region is matched to the ground truth (1):

$$MatchScore(i, j) = \frac{T(G_j \cap R_i \cap I)}{T((G_j \cup R_i) \cap I)} \tag{2.8}$$

A result region is considered as a one-to-one match to the ground truth region, if the matching score is equal or above 95%. Assume N is the number of ground truth elements, M is the number of result elements, and $o2o$ is the number of one-to-one match pairs, then the detection rate (DR) and recognition accuracy (RA) are defined as follows:

$$DR = \frac{o2o}{N} \tag{2.9}$$

$$RA = \frac{o2o}{M} \tag{2.10}$$

Combining the values of DR and RA, FM is a performance metric:

$$FM = \frac{2 \times DR \times RA}{DR + RA} \tag{2.11}$$

2.4.2 Experimental setup

We test our proposed method on ICDAR2013 Handwritten Segmentation Contest dataset (1). In the testing dataset, there are 100 English and Greek (Latin language) documents, and another 50 Indian (Bangla) documents, totally 2649 text lines. The handwritten documents contain large writing styles, multi-skewed text lines and touching connected components. For comparison, there are in total 11 different algorithms, and many techniques are used, including run-length analysis, Gaussian filtering, energy minimization, histogram projection, connected components analysis, grouping method, seam carving, and etc. For details of different algorithms, please refer to (1).

2.4.3 Results

Fig. 2.9 shows the evaluation results of 13 different algorithms based on FM, and our result has the label 'NUS' in the horizontal axis. The segmentation result of our proposed method is $FM = 98.41$, only 0.25% less than the best result, putting us in the second position. More results with details are shown in Table 2.1.

Most of our failure cases are mainly caused by small floating strokes and splitting large components. In Indian documents, some characters have different parts located vertically, so that the lower parts are sometimes misclassified to the lower text lines. In Figure 2.10, 2.11 and 2.12, three examples of segmentation results are shown, where each text line is marked as an unique color.

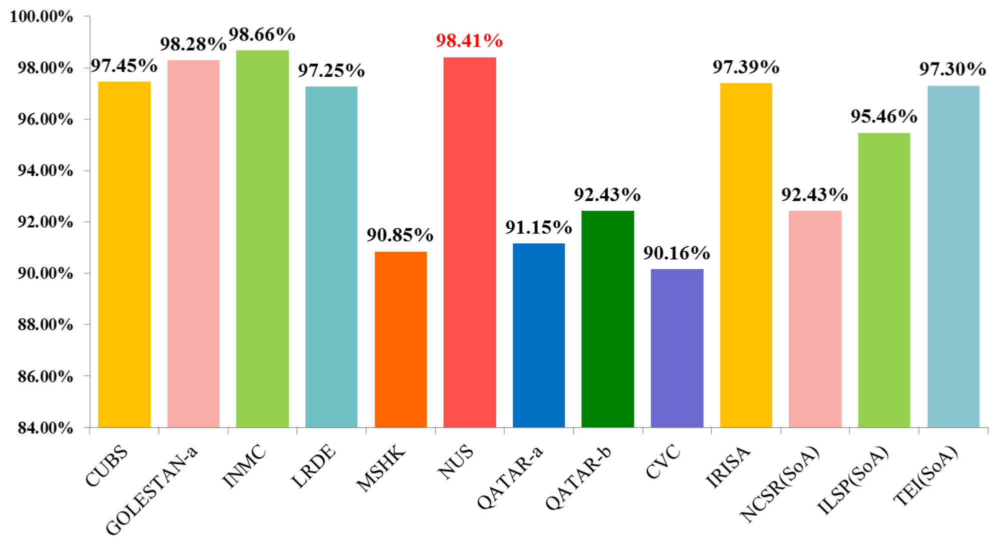


Figure 2.9: The evaluation results (1) based on FM. Our method has the label 'NUS'.

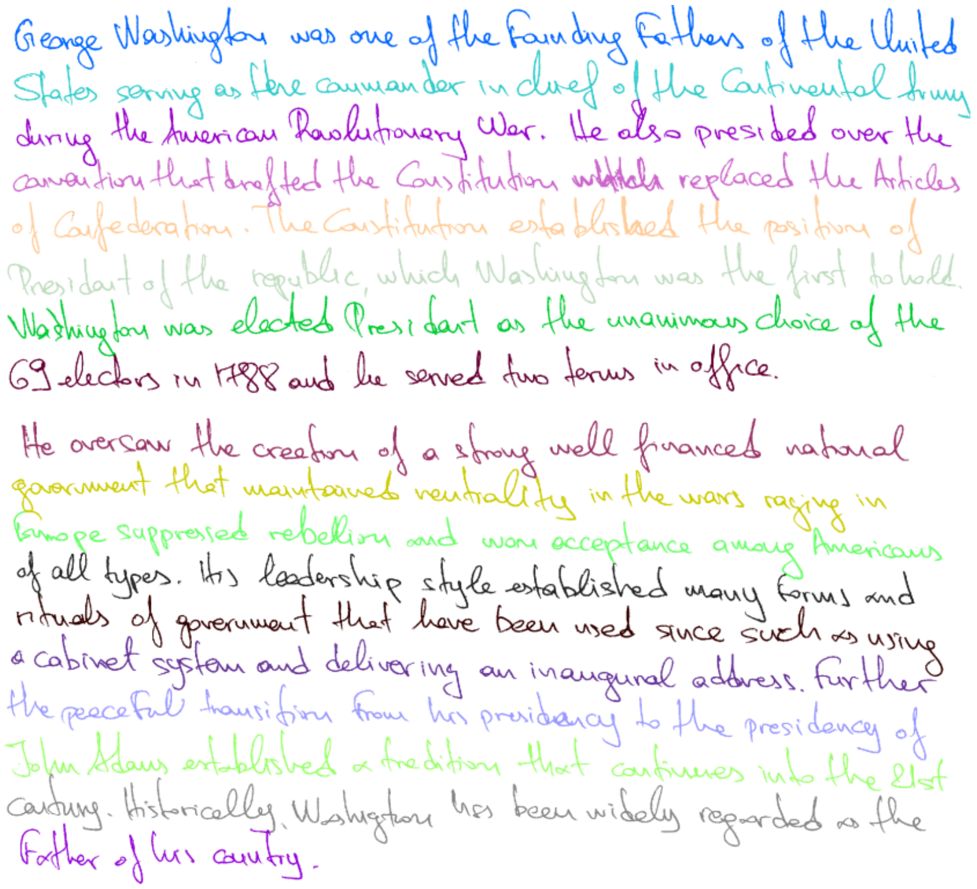
Table 2.1: Detailed Evaluation Results from (1)

Methods	M	o2o	DR(%)	RA(%)	FM(%)
CUBS	2677	2595	97.96	96.64	97.45
GOLESTAN-a	2646	2602	98.23	98.34	98.28
INMC	2650	2614	98.68	98.64	98.66
LRDE	2632	2568	96.94	97.57	97.25
MSHK	2696	2428	91.66	90.06	90.85
NUS	**2645**	**2605**	**98.34**	**98.49**	**98.41**
QATAR-a	2626	2404	90.75	91.55	91.15
QATAR-b	2609	2430	91.73	93.14	92.43
NCSR(SoA)	2646	2477	92.37	92.48	92.43
ILSP(SOa)	2685	2546	96.11	94.82	95.46
TEI(SoA)	2675	2590	97.77	96.82	97.30

2.5 Conclusion

We proposed a text line segmentation method for handwritten documents based on seam carving. Unlike the previous methods which first applied the idea of seam carving to extract text lines, we make the energy from earlier columns become less and less when accumulated to following columns. We also make a constraint that each energy can be only accumulated up to half way of the document, so that we ensure that the energy from too far way will not have great influence. Moreover, our method needs to calculate the energy map only once, and extract all the text lines together, instead of recomputing the energy map again after one text line is extracted.

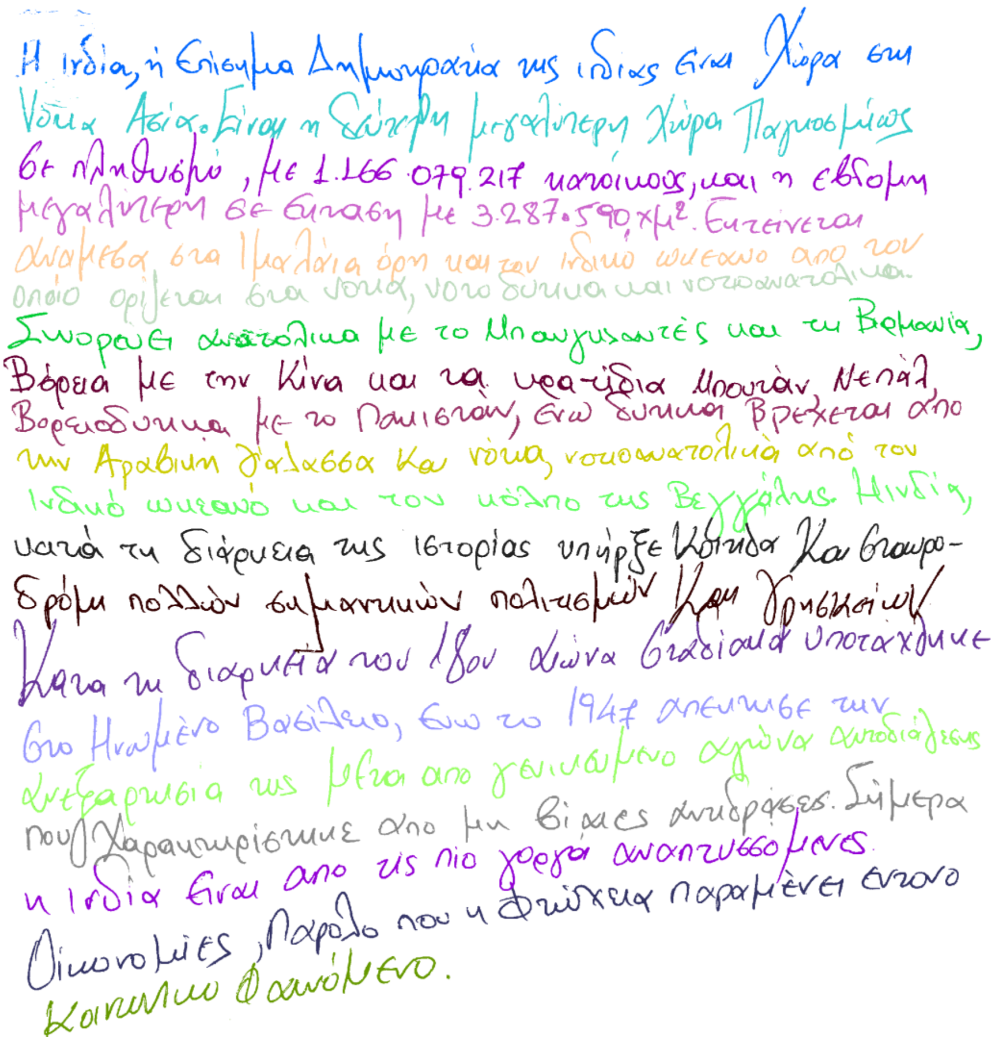
In future work, we would like to improve our energy accumulation process to reduce the computation time. Moreover, we will improve the performance of splitting large components which touch multiple text lines, and we will also work on gray level documents, which have more challenges.



George Washington was one of the Founding Fathers of the United
States serving as the commander in chief of the Continental Army
during the American Revolutionary War. He also presided over the
convention that drafted the Constitution which replaced the Articles
of Confederation. The Constitution established the position of
President of the republic, which Washington was the first to hold.
Washington was elected President as the unanimous choice of the
69 electors in 1788 and he served two terms in office.

He oversaw the creation of a strong well financed national
government that maintained neutrality in the wars raging in
Europe suppressed rebellion and won acceptance among Americans
of all types. His leadership style established many forms and
rituals of government that have been used since such as using
a cabinet system and delivering an inaugural address. Further
the peaceful transition from his presidency to the presidency of
John Adams established a tradition that continues into the 21st
century. Historically, Washington has been widely regarded as the
Father of his country.

Figure 2.10: Segmentation result of an English document.



Η Ινδία, ή Επίσημα Δημοκρατία της Ινδίας είναι χώρα στη
Νότια Ασία. Είναι η δεύτερη μεγαλύτερη χώρα παγκοσμίως
σε πληθυσμό, με 1.166.079.217 κατοίκους, και η έβδομη
μεγαλύτερη σε έκταση με 3.287.590 χμ². Εκτείνεται
ανάμεσα στα Ιμαλάια όρη και τον Ινδικό ωκεανό από τον
οποίο ορίζεται στα νότια, νοτιο δυτικά και νοτιοανατολικά.
Συνορεύει ανατολικά με το Μπαγκλαντές και τη Βιρμανία,
Βόρεια με την Κίνα και τα κρατίδια Μπουτάν, Νεπάλ
Βορειοδυτικά με το Πακιστάν, ενώ δυτικά βρέχεται από
την Αραβική θάλασσα και νότια, νοτιοανατολικά από τον
Ινδικό ωκεανό και τον κόλπο της Βεγγάλης. Η Ινδία,
κατά τη διάρκεια της ιστορίας υπήρξε κοιτίδα και σταυρο-
δρόμι πολλών σημαντικών πολιτισμών και θρησκειών
Κατά τη διάρκεια του 18ου αιώνα σταδιακά υποτάχθηκε
στο Ηνωμένο Βασίλειο, ενώ το 1947 απέκτησε την
ανεξαρτησία της μετά από γενναίο αγώνα αντιβίασης
που χαρακτηρίστηκε από μη βίαιες αντιδράσεις. Σήμερα
η Ινδία είναι από τις πιο γρήγορα αναπτυσσόμενες
οικονομίες, παρόλο που η φτώχεια παραμένει έντονο
κοινωνικό φαινόμενο.

Figure 2.11: Segmentation result of a Greek document.

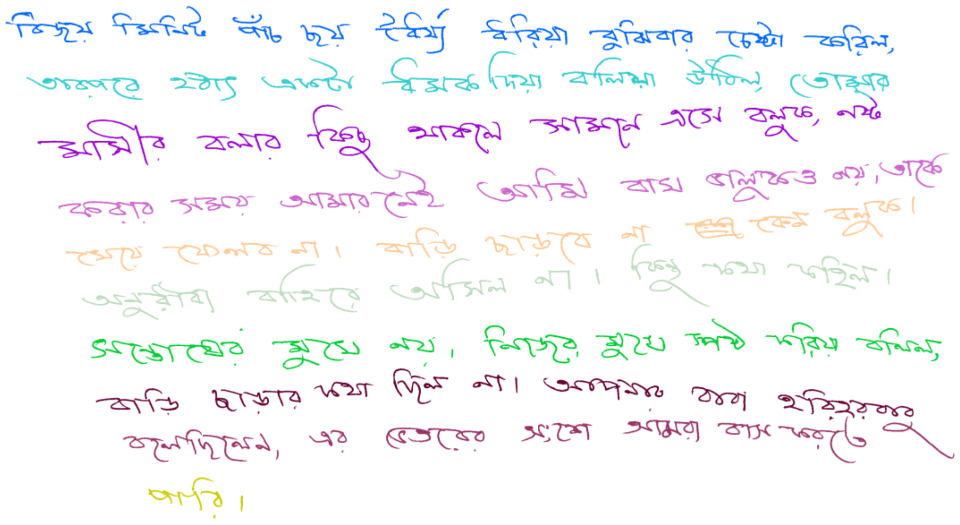


Figure 2.12: Segmentation result of a Bangla document.

Chapter 3

Handwritten Word Recognition

To get high recognition accuracy, we should train the recognizer with sufficient training data to capture enough characteristics of various handwriting styles and all possible occurring characters or words. However, in most of the cases, available training data are not satisfactory and sufficient, especially for unseen data. In this chapter, we try to improve the recognition accuracy for unseen data with randomly selected training data which is always not sufficient enough. By splitting the training data into two subsets based on trigrams, and training two recognizers separately, the recognizers can focus on different sets of trigrams. Because, each recognizer is responsible for only half cases of handwriting characteristics, the training data for each recognizer can be treated as more sufficient than training one recognizer with whole training set. We also propose a modified version of token passing algorithm, which makes use of the outputs of the two recognizers to improve the recognition accuracy.

3.1 Introduction and Related Works

Document images can be segmented into text lines as we present in the previous chapter. Then, we can extract word images from each text line, based on the distances of the inter- and intra-space between connected components. In order to provide online access, recognition of the content can allow the users to retrieve document images using traditional text retrieval methods according to their needs. However, recognition of unconstrained handwritten documents is always a challenging task and the poor results may cause unreliable and unsatisfactory retrieval service.

Handwriting recognition can be achieved at character level, or word level, even at text line level nowadays. Because of its wide usage and popularity in speech processing tasks, Hidden Markov Models (HMMs) are also applied for handwritten document analysis. In (53), one isolated handwritten word is represented by one HMM, but this word classification approach cannot be used for words which do not appear in the training data and a considerable amount of training data for each word is required. Moreover, it cannot be scaled to large vocabularies, because every distinct occurring word needs an HMM. Therefore, in order to recognize arbitrary words, HMMs are used to represent character models instead of the whole words, and one word or one text line is represented by a sequence of character HMMs, which are linearly connected (54). Based on the trained HMMs, for a given text line, we can obtain the most likely character sequence and the beginning and end positions of each character in the text line, using the Viterbi algorithm.

However, the methods based on HMMs suffer from several disadvantages. The probability of every observation only depends on the current state, and it is difficult to take the context information into account. Moreover, because handwritten document recognition is always a discriminative task, HMMs, which are generative, may not provide better performance than other discriminative models.

Combining HMMs and neural networks is a kind of hybrid method for handwriting recognition. Many kinds of architectures of neural networks are applied, such as Multilayer Perceptron (MLP) (55), time delay neural network (56), and Recurrent Neural Networks (57). Although the hybrid methods can capture context information, they also suffer from some drawbacks of HMMs.

In the recent works, Recurrent Neural Network (RNN), with Connectionist Temporal Classification (CTC) output layer is applied for unconstrained handwritten document recognition (3). Traditional RNN needs presegmented input data, namely, we should label each position of the input data. But, RNN with CTC output layer can map the whole unsegmented sequence of the input data to the output labels directly. Combined with a dictionary, the recognition results for both on-line and off-line data outperform HMMs.

In order to get satisfactory recognition results, sufficient training data is always important. However, in practice, unavailability of enough training data is always an issue. What is more, words in the training data may be distributed unevenly, namely, some words appear much more frequently than others. For unconstrained writing styles, consecutive characters are always connected and how one character is connected with other neighbouring characters are different for different writers or even the same writer. Therefore, given a set of training data, we split it into two parts based on the occurring trigrams (three consecutive characters), so that the two training sets contain exclusive words with two sets of different trigrams. We aim at training two networks separately on the two sets and make the trained networks capture reliable information about how consecutive characters appear together in the corresponding training set. If we train one network on the whole training data, the network needs to capture all the possible characteristics in the whole training set, and may be biased to the features which appear more frequently. Due to the insufficiency of the training data, the network may not be able to capture all the features very well and may not be able to tolerate unseen features in the testing set. However, if we train two networks, each of which is only responsible to capture subset of the characteristics, so that the training data for either of them can be treated more sufficient and the trained networks can be more reliable. Based on the two trained networks, in the decoding period, instead of calculating the probability of each whole word in a dictionary, we generate the weighted probabilities for all trigrams in each possible word, and combine them together to get the final probability. The word with the highest probability will be returned as the recognition result for the input word image.

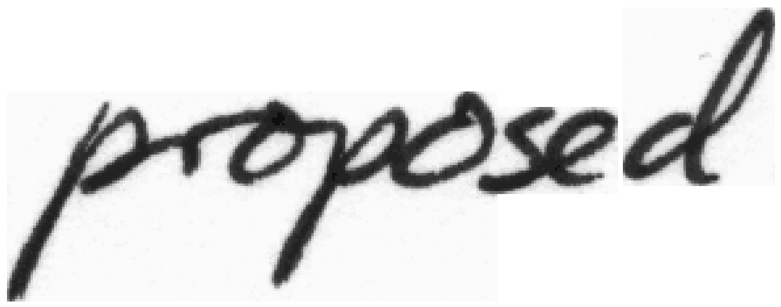
3.2 Preprocessing

The database we use in the experiments is IAM offline handwritten database (58), consisting of 657 writers. All documents are segmented into isolated and labeled word images. In order to reduce variations due to different writing styles, the word images are binarized, with skew and slant correction (59), and the heights of the ascenders, the main body parts and the descender parts are normalized to 20, 40, and 20 pixels respectively, as shown in Fig. 3.1.

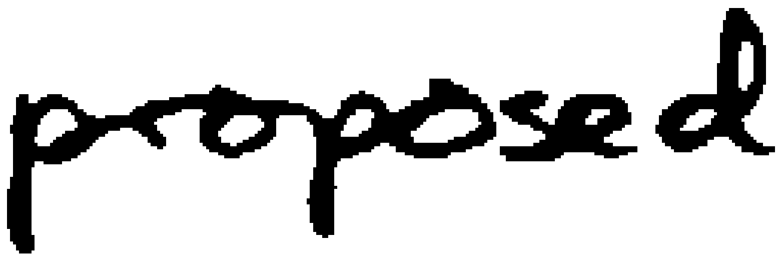
Nine geometrical features are extracted from a sliding window, with the width of 1 pixel, moving from left to right along each word image. The features are shown as follows (60) (3):

1. the number of black pixels

2. the center of gravity of the group of pixels
3. the second order moment of the window
4. the location of the upper-most pixel
5. the location of the lower-most pixel
6. the orientation of the upper-most pixel
7. the orientation of the lower-most pixel
8. the number of black-white transitions
9. the number of black pixels divided by the number of all pixels between the upper- and lower-most pixel



(a) The original word image.



(b) The normalized word image.

Figure 3.1: An example of the normalized result for an word image from IAM database.

Each dimension of the nine features in one word image is normalized by subtracting the mean and dividing by the standard deviation respectively. The mean and standard deviation are calculated on all the word images in the training set.

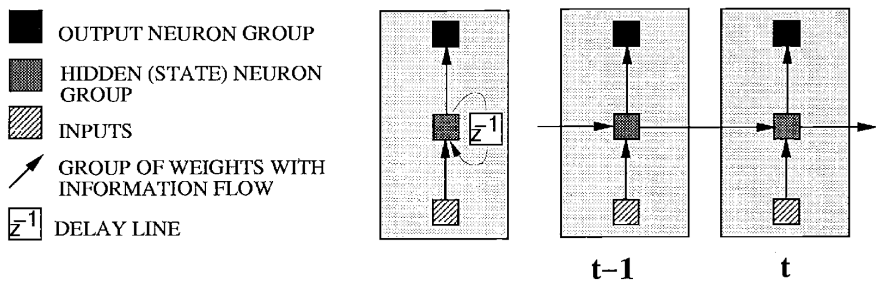
3.3 Neural Network for Recognition

The recognizer we use for off-line handwritten word images is a Recurrent Neural Network (RNN). Each hidden node of RNN is self-connected and also connected to the hidden node in the later time step, so that RNN can capture a certain range of previous input information, as shown in Fig. 3.2(a). However, only history information can be used by RNN. In order to make full use of all input data, whether in the past or in the future, an additional backward hidden layer is added, which is not connected to the original forward hidden layer, as shown in Fig. 3.2(b). The forward states are trying to capture the past information, and the backward states are making use of the future information. This structure is named as Bidirectional Recurrent Neural Network (BRNN), which allows each time step to be evaluated based on both the past and future information.

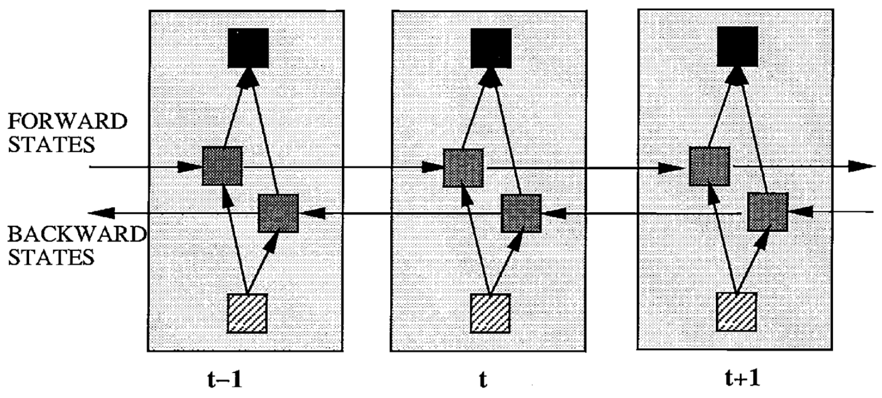
However, the authors in (61) present one drawback of the traditional RNN, named as vanishing gradient problem. There is a limitation in the range of contextual information that we can use to train RNN, because the influence of each input will be reduced exponentially over time. In order to overcome this limitation, a specially designed architecture is used instead of the traditional one, named as Long Short-Term Memory (LSTM) (62). The hidden nodes of the traditional RNN in Fig. 3.2 are replaced by the LSTM memory blocks, shown in Fig. 3.3. One LSTM memory block with a single cell has three gates, which can control the cell to access information over a long time period.

3.4 Splitting of Randomly Selected Training Data

In practice, we may have insufficient training data on hand, and the recognizer cannot recognize the words very well which do not or rarely appear in the training data. Especially, if the words in the training data are not distributed evenly, namely, some words appear hundreds of times, but others may rarely appear. As a result, the trained network fits very well the characteristics which occur much more frequently in the training data, but fail for rarely appearing features. In this section, we try to improve the recognition results, especially for the words which do not appear in the training data. Moreover, the training data is randomly selected from a collection of word images, namely, any word and any number of word images for each word can be included in


OUTPUT NEURON GROUP
HIDDEN (STATE) NEURON GROUP
INPUTS
GROUP OF WEIGHTS WITH INFORMATION FLOW
DELAY LINE
z^{-1}
t−1
t

(a)



(b)

Figure 3.2: Structure of Recurrent Neural Network from (2). (a) Unidirectional Recurrent Neural Network with 2 time steps unfolded. (b) Bidirectional Recurrent Neural Network with 3 time steps unfolded.

the training data, which is always the case in the real world. We also test our method on the writer-independence dataset, the test set of which is also a kind of unseen data with respect to the training data.

Given a set of training data, we try to split it into two subsets and train them separately. Assume $train\_data$ is a set of randomly selected training data, and $train\_data_1$, $train\_data_2$ are two exclusive training data from $train\_data$. From the ground truth of the training data, we can generate a set, $train\_dict$, which contains all distinct occurring words in the training data. We randomly select two words from $train\_dict$, which do not have the same trigrams, and assign their prefix trigrams to tri_1 and tri_2, respectively, as the initialization. For example, if 'company' and 'special' are selected, then tri_1 and tri_2 are initialized to $['com']$ and $['spe']$, respectively. Then, we will generate

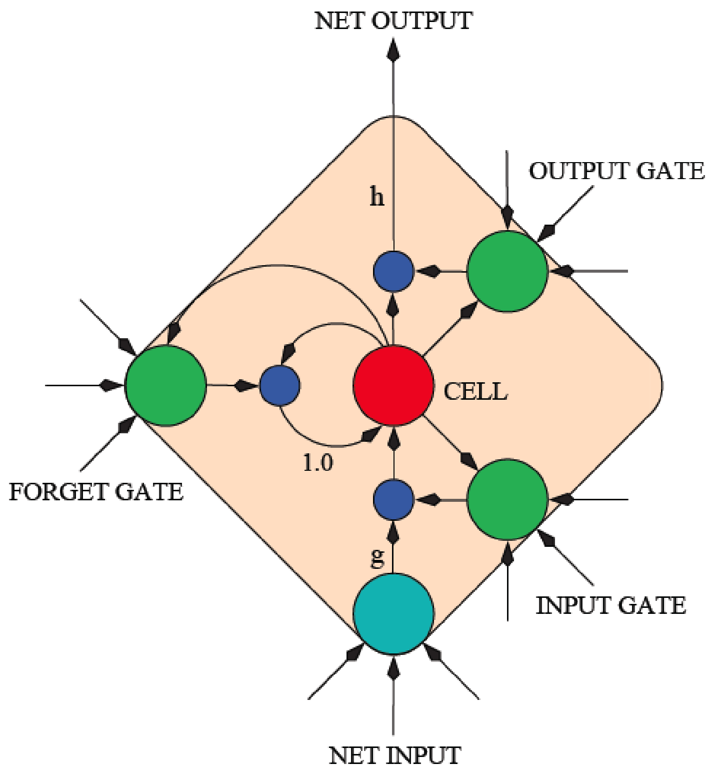


Figure 3.3: Structure of LSTM memory block with a single cell from (3). There are three gates: input gate, output gate, and forget gate. They collect the input from other parts of the network and control the information the cell can accept. The input and output of the cell are controlled by the input gate and output gate, while how the recurrent connection effects the cell is controlled by the forget gate.

two subsets set_1 and set_2 from $train\_dict$ as describe in Algorithm 1.

For each word in $train\_dict$, we first extract all its trigrams and store them into tri. Then we check whether tri_1 or tri_2 contains one of the trigrams in tri. If a trigram in tri is included in tri_1, $mark_1$ is assigned to 1, otherwise 0; if a trigram in tri is included in tri_2, $mark_2$ is assigned to 1, otherwise 0. Then, $mark$ is equal to $mark_1 \times 10 + mark_2$, and Table 3.1 shows the meanings of its different values. If both tri_1 and tri_2 contain none of the trigrams in tri ($mark = 0$), we will randomly add tri and the word to either tri_1 and set_1 or tri_2 and set_2, with equal probability. If only one of tri_1 and tri_2 contains at least one trigram in tri ($mark = 1$ or 10), we will add tri to it, and add the word to the corresponding set.

Algorithm 1 Splitting the training data into two subsets

```
$Random() \in (0, 1]$
for $i = 1 \rightarrow size(train\_dict)$ do
    $tri \leftarrow all\ trigrams\ in\ train\_dict[i]$
    $mark_1 = 0$, $mark_2 = 0$
    if $\exists t : t \in tri$ and $t \in tri_1$ then
        $mark_1 = 1$
    end if
    if $\exists t : t \in tri$ and $t \in tri_2$ then
        $mark_2 = 1$
    end if
    $mark = mark_1 * 10 + mark_2$
    switch $(mark)$
    case 0:
    if $Random() > 0.5$ then
        Add $tri$ to $tri_1$, and $train\_dict[i]$ to $set_1$
    else
        Add $tri$ to $tri_2$, and $train\_dict[i]$ to $set_2$
    end if
    case 10:
    Add $tri$ to $tri_1$, and $train\_dict[i]$ to $set_1$
    case 1:
    Add $tri$ to $tri_2$, and $train\_dict[i]$ to $set_2$
    end switch
end for

$W = train\_dict - (set_1 \cup set_2)$
for $i = 1 \rightarrow size(W)$ do
    $tri \leftarrow all\ trigrams\ in\ W[i]$
    if $size(tri \cap tri_1) > size(tri \cap tri_2)$ then
        Add $tri$ to $tri_1$, and $W[i]$ to $set_1$
    else
        Add $tri$ to $tri_2$, and $W[i]$ to $set_2$
    end if
end for
```

Table 3.1: Meanings of different values of *mark*.

$mark$	0	1	10	11
tri_1	×	×	✓	✓
tri_2	×	✓	×	✓

'✓' means containing at least one trigram in *tri*, and '×' means containing none of the trigrams in *tri*.

When we finish checking all the words in *train_dict*, tri_1 and tri_2 are two sets of trigrams, and set_1 and set_2 are two exclusive subsets of *train_dict*. However, there are some words, some of whose trigrams are both included in tri_1 and tri_2. For such cases, if tri_1 contains more trigrams of the words than tri_2, we will add the trigrams and the words to tri_1 and set_1 respectively, otherwise, put them into tri_2 and set_2.

Because of the randomness when $mark = 0$, we repeat Algorithm 1 for several times, and choose the results, in which the sizes of tri_1 and tri_2 are nearly the same and they have least common trigrams. At last, based on set_1 and set_2, we put the word images containing the words in set_1 to $train\_data_1$, and others to $train\_data_2$.

Consequently, we split the training data into two subsets, and make them contain two sets of different trigrams. For training, we construct two RNNs with LSTM hidden layers (BLSTM) (3) Net_1 and Net_2, trained on $train\_data_1$ and $train\_data_2$ respectively, in order to make the networks capture information of tri_1 and tri_2 separately. In theory, we can split the training data into three or more subsets. However, because of insufficient training data, the more sets we split into, the fewer word images each set can contain, and the more likely we cannot get well enough trained networks as we expect. Moreover, the computation time cost for combining outputs from multiple networks highly depends on the number of split training sets, therefore for the current work we split the training data into two subsets instead of three or more.

3.5 Modified CTC Token Passing Algorithm

3.5.1 CTC Token Passing Algorithm

Based on the token passing method for HMMs (63), CTC token passing algorithm is proposed in (3) for text line recognition and also used for keyword spotting in (45).

For single word recognition, we expect the probability of each possible word with a sequence of ASCII characters based on the output of the trained network. Assume we have N different characters and the input has t time steps, then the network will output the probability of each character appearing at each time step, constructing an $N \times t$ matrix. Using dynamic programming, the best path through character probabilities is calculated for each input word, and the final accumulated probability value is treated as the appearing probability. The word with the highest probability is returned as the recognition result.

As shown in Fig. 3.4, the image on the top shows the output of a trained network with the input image at the bottom, containing the word 'report'. The total number of time steps is the same as the width of the input image, and we assume that the network is trained for the 26 lower-case characters. The darker the block is, the higher the probability of the corresponding character appears with. For example, at time step 25, the 18th character 'r' has the highest probability, assuming characters are indexed in lexicographical order. However, in some cases, the trained network cannot recognize every character perfectly, such as the character 'r', which has similar probability with 'n' at time step 180. Therefore, a dictionary is used to filter out impossible character sequences.

3.5.2 Modification to spot trigrams

Because we try to spot all possible trigrams in the input image, other than the whole word, based on the word spotting algorithm proposed in (45), we propose a modified version in order to get the weighted probability for each trigram of one input word and combine them together to get the final score.

When we finish training Net_1 and Net_2, which are trained on $train\_data_1$ and $train\_data_2$ separately, we apply a modified token passing algorithm described in Algorithm 2 to calculate the probability of each word in a closed dictionary $Dict$. When the trained Net_i ($i = 1, 2$) accepts a sequence of column features extracted from a word image, it will return $Prob_i(c, t)$ with two parameters: c indicates the character (lower-case and capital characters, blank symbol $'\#'$, and any-character symbol $'*'$) and t indicates the time step or position, so the value of $Prob_i(c, t)$ is the probability of the character c appearing at time t. We assume $'*'$ can appear at any time step, so $Prob_i(*, t) = 1$ for all t.

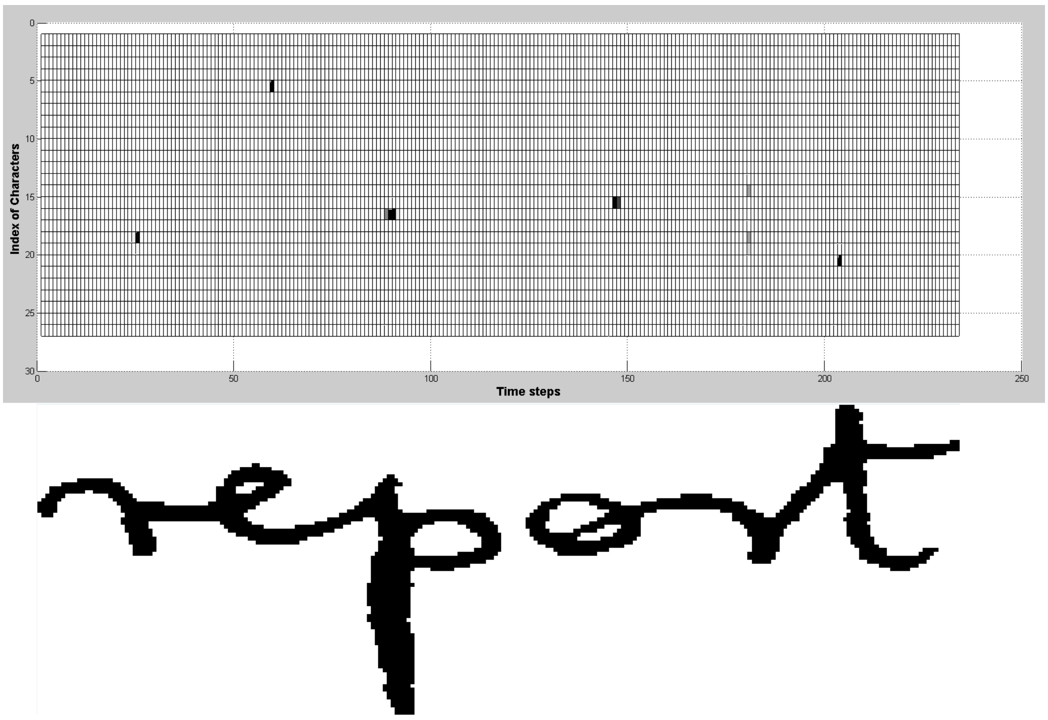


Figure 3.4: The output of a trained network for the input image 'report'. $x - axis$ indicates the time steps, with the size as same as the width of the word image, and $y - axis$ indicates the index of all lower-case characters, in the lexicographical order. At the time step 180, 't' and 'n' have similar probabilities. Using a dictionary, we can easily exclude 'n'.

For each word $w = c_1c_2...c_n$ in $Dict$, we first generate all its trigrams $\{c_1c_2c_3, c_2c_3c_4, \cdots, c_{n-2}c_{n-1}c_n\}$. We aim at obtaining the probability of each trigram appearing in the input image instead of the whole character sequence as in (45), and we also should take the order of how the trigrams appear one by one into account. So, for each trigram, we define a set w' based on the set of trigrams, by first adding $'\#'$ at the beginning and end of each character, and add $'*'$ to the beginning except for the first trigram, and at the end except for the last trigram. For example, the trigrams of the word 'limit' is $\{ 'lim', 'imi', 'mit' \}$, and w' is $\{ '\#l\#i\#m\#*', '*\#i\#m\#i\#*', '*\#m\#i\#t\#' \}$. For each entry in w', $w'[j], j \in [1, size(w')]$, instead of using all time steps, we only consider the part where the corresponding trigram probably appears. Assume the width of the input image is L, the estimated width of one character in w is $L/len(w)$ ($len()$ calculates the number of characters in the given string), denoted as len_c, so the width of one trigram is estimated as $3 \times len\_c$. We divide the whole time steps into $len(w) - 2$ parts, each of which corresponds to one entry in w'. For $w'[j]$, the time steps we consider are $\{(j-1) \times len\_c + 1, \cdots, min(L, (j+2) \times len\_c)\}$. Because the width of all characters may not be the same, such that the width of two consecutive characters is less than $2 \times len\_c$ or the width of one character is larger than len_c, so we expand each part by len_c at the beginning and the end, and the time steps under consideration are changed to $\{max(j-2, 0) \times len\_c + 1, \cdots, min(L, (j+3) \times len\_c)\}$, where the minimal and maximal value are denoted as s_p and e_p, respectively. For example, if the width of the input word image is 200, and we want to generate the probability for word 'limit', then $len\_c = 40$ and the divided parts of time steps is $\{1 : 160, 1 : 200, 41 : 200\}$. Due to the expansion, the time steps for $w'[j]$ may include other parts of characters, so we allow any-character symbol $'*'$ appearing at the beginning or the end of the trigram as shown in Line 3,4 of Algorithm 2.

Given $w'[j]$, s_p, and e_p, we construct a matrix V_i $(i = 1, 2)$, with the size of $len(w'[j]) \times (e\_p - s\_p + 1)$, which is initialized to $-Inf$ for all elements. $V(\cdot, t)$ is updated based on the values in $V(\cdot, t-1)$ to get the optimal character sequence at each time step. $Score_i$ is obtained based on the output of Net_i, however, when the length of $w'[j]$ is bigger, we will accumulate more probabilities in V_i, so $score_i$ is normalized by the length of $w'[j]$. Assume n_i is the number of $w'[j]$ appearing in $train\_data_i$ dividing by the total number of trigrams in $train\_data_i$, then $\omega_i = (n_i + 1)/(n_i + n_{1-i} + 2)$, so that we give more trust to the network which is trained on the data set containing more occurrences of the trigram. We add the sum of the weighted scores to P. However, the probability of word w also depends on its length. At last, P is normalized by the size of w', as the returning probability. Therefore, we calculate the probabilities of all words in the dictionary, and return the one with the highest probability.

3.6 Experiments and Results

3.6.1 Experimental Setup

The experiment data consists of word images segmented correctly and containing English words with more than 4 characters, totally 21332 word images and 1601 distinct words. For words containing less than 4 characters, most of them are stop words, so we do not test them in our experiments.

Algorithm 2 Modified CTC Token Passing Algorithm Combining Outputs of Two Trained Networks

```
1: $P = 0$
2: Input word $w = c_1c_2...c_n, w \in Dict$
3: $w' = \{'\#c_1\#c_2\#c_3\#*', '*\#c_2\#c_3\#c_4\#*', \cdots,$
4:      $'*\#c_{n-2}\#c_{n-1}\#c_n\#'\}$
5: for $i = 1 \to size(w')$ do
6:     $tri = w'[i]$
7:     $l\_t = len(tri)$
8:     $s\_p = max(i-2, 0) \times len\_c + 1$
9:     $e\_p = min(L, (i+3) \times len\_c)$
10:    $l = e_p - s_p + 1$
11:    $V_1(r, c) = 0$ $for$ all $r \in [1, l\_t]$ and $c \in [1, l]$
12:    $V_1(1, 1) = Prob_1(tri[1], s\_p)$
13:    $V_1(2, 1) = Prob_1(tri[2], s\_p)$
14:    for $t = s\_p + 1 \to e\_p$ do
15:        for $p = 1 \to l\_tri$ do
16:            $Best = V_1(p, t-1)$
17:            if $p > 1$ then
18:                $Best = Best \bigcup V_1(p-1, t-1)$
19:                if $p > 2$ and $tri[p] \notin \{\#, tri[p-2]\}$ then
20:                    $Best = Best \bigcup V_1(p-2, t-1)$
21:                end if
22:            end if
23:            $V_1(p, t) = max(Best) * Prob_1(tri[p], t)$
24:        end for
25:    end for
26:    $V_2$ is constructed by the same manner based on $Prob_2$
27:    $Score_1 = log(max\{V_1(l\_t, l), V_1(l\_t - 1, l)\})/l\_t$
28:    $Score_2 = log(max\{V_2(l\_t, l), V_2(l\_t - 1, l)\})/l\_t$
29:    $P = P + (Score_1 * \omega_1 + Score_2 * \omega_2)$
30: end for
31: return $Score = P/size(w')$
```

The recognition network is BLSTM, which has 9 input nodes and 53 output nodes, containing 52 lower-case and capital characters and one more node for 'blank', the symbol $'\#'$ in Algorithm 2. The forward and backward hidden layers both have 100 LSTM memory blocks, and the number of total weights is 99253, which are initialized with a Gaussian distribution of mean zero and standard deviation 0.1. Gradient descent is used for training the network, with learning rate 0.0001 and momentum 0.9. In the decoding, we only use a closed dictionary, without any language model.

The performance is measured on the character error rate (CER):

$$CER = 100 \times (\frac{\text{insertions} + \text{substitutions} + \text{deletions}}{\text{total number of characters in the testing data}}) \tag{3.1}$$

where all the counts are summed over the whole test set.

Word error rate (WER) is also recorded, which is defined as the number of words recognized wrongly divided by the total number of words in the test set.

3.6.2 Results on Randomly Selected Training and Testing Data

First, we randomly select 20% of the 1601 distinct words and make the corresponding word images as testing data. Moreover, we try to avoid adding words, which have the same relatively long prefix into the test set simultaneously, such as 'writes' and 'write'. In the all remaining word images, we randomly select 70% for training, and 30% for validating. We repeat the above procedure and do the experiment for five times, and record the average results. The number of distinct words and the corresponding word images for 5 experiments, Exp_i, $i = 1, 2, ..., 5$, are shown in Table 3.2.

Table 3.2: The number of distinct words and the corresponding word images in each data set.

	Exp_1	Exp_2	Exp_3	Exp_4	Exp_5
Train	1288/12479	1275/12403	1292/11949	1253/11529	1272/11904
Validate	1201/5348	1199/5315	1191/5121	1172/4941	1189/5101
Test	310/3505	325/3614	308/4262	347/4862	328/4327

In each entry, the value at the left of '/' is the number of distinct words in the corresponding data set, and the right value is the total number of word images.

In Fig. 3.5, the character error rates for validation dataset of one experiment during the first 100 training iterations are shown, where Net is trained on $train\_data$, and Net_i is trained on $train\_data_i$, $i = 1, 2$. The error rates of Net_1 and Net_2 drop faster than

Net in the first 10 iterations and $Net2$ is converged much faster than Net and $Net1$. We can see that $Net1$ and $Net2$ have higher character error rates than Net in the following iterations, because they are trained on only half of the whole training word images. We choose the networks with the best character error rates on validation data for decoding in our experiments, for example, the best networks for Net, $net1$ and $Net2$ are in the 60, 61, 31 iterations in the figure 3.5, respectively.

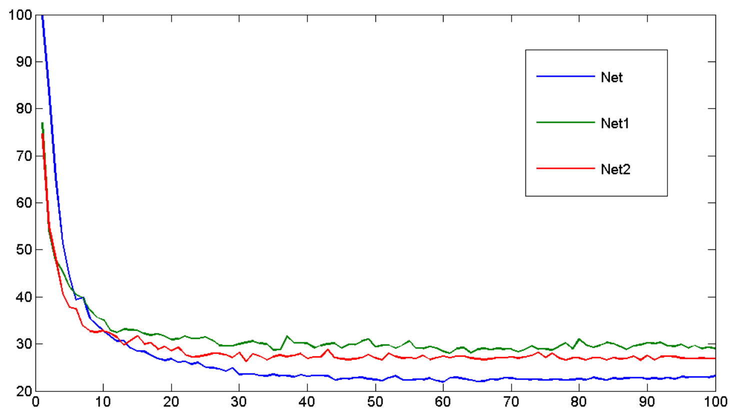


Figure 3.5: Character error rate on the validation data over first 100 iterations.

We do the experiments for 5 times on 5 different sets of training and testing data, as described in Table 3.2. The results for Net are based on the CTC token passing algorithm in (45) for single words and the results for $Net_1 + Net_2$ are based on our method. As shown in Table 3.3 and Table 3.4, our methods have better results according to both CER and WER. For our method, the two trained networks are trained on different sets of distinct words, in order to capture characteristics of different sets of trigrams separately, in contrast to training one network on the whole training data, containing large variations. In the decoding, for each possible trigram, we give more trust to the network which is trained on more word images containing the trigram, i.e. the trigram appears much more times in the corresponding training data. Therefore, we get the probability for each word in the dictionary by the outputs of two networks, both of which give their results with more confidence.

Table 3.3: Character Error Rate (CER%)

Test Set	Exp_1	Exp_2	Exp_3	Exp_4	Exp_5	Average
Net	13.82	11.21	12.46	13.51	11.28	12.46
$Net_1 + Net_2$	9.14	8.37	8.69	9.38	8.62	8.84

Table 3.4: Word Error Rate (WER%)

Test Set	Exp_1	Exp_2	Exp_3	Exp_4	Exp_5	Average
Net	16.46	14.17	13.76	14.83	15.79	15.04
$Net_1 + Net_2$	13.35	11.27	10.94	11.57	13.29	12.08

3.6.3 Results on Writer Independent Training and Testing Data

Besides using randomly selected training and testing data, we also test our algorithm on the public dataset used for Large Writer Independent Text Line Recognition Task (58). We put 21332 word images into the corresponding sets, and totally 16500 words are used. The separation of the dataset is shown in Table 6.1, on which Net will be trained:

Table 3.5: Writer Independent Dataset for Net

Set Name	Number of Word Images	Number of Writers
Train	11642	283
Validation 1	1269	46
Validation 2	1389	43
Test	2200	128

For our algorithm, we split the training data using Algorithm 1, and based on set_1 and set_2 we get, each validation dataset is also split into two sets, each of which only contains the words in set_1 or set_2. Table 3.6 shows the dataset we use for our method. In set_1, there are 1214 distinct trigrams, and in set_2, there are 1231 distinct trigrams. The number of common trigrams in the two sets are 371.

We train Net_1 and Net_2 on two sets of validation dataset, and test on the same test dataset in Table 6.1. The results are shown in Table 3.7, which are the average

Table 3.6: Writer Independent Dataset for Net_1 and Net_2

Set Name for Net_1	No. of Word Images	No. of Distinct Words
Train	5856	751
Validation 1	643	279
Validation 2	606	273
Set Name for Net_2	No. of Word Images	No. of Distinct Words
Train	5786	737
Validation 1	626	278
Validation 2	783	326

results after doing experiments for 5 times, i.e. different weight initializations for the networks. Because of writer independence in the training and testing dataset and large writing variations among different writers, the CER and WER for Net, $Net1$ and $Net2$ are all higher than our experiments in Section 3.6.2. However, our method, combining two trained networks, also has better recognition results. We also split the training set randomly, i.e. put each training word image into either $train\_data_1$ or $train\_data_2$ with equal probability, and use the same value 0.5 for both ω_1 and ω_2 in Algorithm 2. As shown in the last row in Table 3.7, the results are worst. Because of splitting the training data randomly, in each set, the variations are large, but the training data is half, so that, the trained networks cannot fit to the large variations very well.

Table 3.7: Resutls on Large Writer Independent Dataset

	Validation 1		validation 2	
	CER(%)	WER(%)	CER(%)	WER(%)
Net	14.2	20.5	13.6	18.5
$Net_1 + Net_2$	11.7	16.8	10.1	14.3
$Net_1 + Net_2$ (Randomly)	16.2	23.6	15.3	22.8

3.7 Conclusion

We proposed a new method combining the outputs of two networks, which are trained on the subset of the training data separately. The splitting of the training data into two subsets satisfies the condition that different words in the two sets are exclusive and the two word sets have as few common trigrams as possible. Our method for decoding is a modified version of the Token Passing Algorithm and we only focus on spotting trigrams instead of the whole character sequence for a word in a Dictionary. In the experiments, we select the training and testing data from a collection of word images randomly and also test on the Writer Independence Recognition Task dataset. Our method has better results both on the character error rate and word error rate. Moreover, our modified CTC token passing algorithm can also be used to get better recognition results by combining two trained networks, which are trained on different sets of training data, than using each network individually.

In the future, we try to apply our method directly on text line images, and also try to reduce the time cost for decoding. What is more, we would like to test on other databases, especially for other languages.

Chapter 4

Handwritten Word Image Matching

In this chapter, a novel method for word image matching will be presented, which is based on Heat Kernel Signature (HKS) and Triangular Mesh Structure. HKS can tolerate large variations in handwritten word images and can also capture local features of important points based on their near neighbours. Moreover, the triangular mesh structure of the detected keypoints is used to represent global characteristics. The proposed method does not need pre-processing steps, including binarization, skew and slant correction.

4.1 Introduction and Related Works

In the previous chapter, word recognition method is used to convert the handwritten document images into text format, which can be used for future retrieval tasks. However, degradation, noise and various unconstrained writing styles always hinder OCR in providing satisfactory recognition results. In (64), an alternative way to OCR was proposed to retrieve useful information for users, especially when it can be used for spotting query text or word images. Furthermore, only features extracted from the query images are needed, without knowledge of ASCII content.

A method which is commonly used to achieve handwritten word image matching is extracting geometrical features in each column of the word images from left to right (65) and applying Dynamic Time Warping (DTW) (66) to calculate the distance between two sequences of feature vectors. However, pre-processing steps are always needed and very crucial, including binarization, skew or slant correction, and normalization, therefore the accuracy highly depends on the reliable pre-processing results. Moreover, column features only take the current column into account and ignore the context information. Consequently, DTW based on column feature sequences may not deal with word images with large variations, which is always the case in handwritten documents. In order to consider context information of the consecutive strokes in handwritten words, (67) extracted features from a sliding window instead of only one column.

Some other widely used methods are based on Scale Invariant Feature Transform (SIFT) (68), which has been successfully used in computer vision and object recognition, and also shows its robustness and reliability to be invariant to multi-scaling and -rotation of objects in different images. Some variations of SIFT feature are gradually used for document analysis. In (69), a new feature sequence is proposed using the local gradient histograms based on the idea of SIFT. Each histogram is extracted from each cell of a sliding window. However, this method also requires pre-processing steps.

In handwritten documents, infinite writing styles may occur, so that different writers, or even the same writer, may write the same word in large variant styles, just like the same word image is deformed by non-rigid deformations in any part of the strokes. (70) shows that SIFT can deal with affine-invariant situations quite well, but cannot handle non-rigid deformations. On the other hand, Heat Kernel Signature (HKS) (71) is proved to be invariant to non-rigid deformations and illumination changes. Motivated by this observation, we propose a new method for handwritten word image matching based on HKS. We also propose a new similarity measurement approach to calculate the distance between two sets of keypoint descriptors, based on triangular mesh structure, which can capture global spatial relations of keypoints. Moreover, our method do not need pre-processing steps, such as binarization, normalization, and skew or slant correction.

In this chapter, we will first introduce the localization of keypoints and how the HKS descriptor is extracted from each keypoint in a word image, which will construct the descriptors for the word image. Then, we will present our new method for computing the distance between two sets of descriptors for two word images, based on the triangular mesh structure and score matrix. Finally, we will test our method on large variant handwritten word images.

4.2 Descriptor based on Heat Kernel Signature

Heat kernel Signature (HKS) was first proposed for 3D shape recognition or classification (72). Based on the 3D coordinates of all the points on a shape surface and their triangular mesh structure, heat kernel can capture the characteristics of the surface with the Laplace-Beltrami operator. The heat kernel is isometric invariant, due to the invariance property of the Laplace-Beltrami operator. Therefore, the heat kernel can even match the same human or animal with different poses. Moreover, due to its multi-scale property, we can capture the features in its near neighbourhoods in small time scales or on the global shape as time progresses, so that for two points, which have similar features in their small neighbourhoods, they may have very different features on the whole shape. Therefore, we can match two points by the features from their small neighbourhoods and extend to a larger domain. In the rest of this section, we will introduce how to calculate HKS descriptors.

4.2.1 Keypoints Detection and Selection

When HKS is used for shape segmentation or recognition, all the points on the surfaces are used to measure the distances between two shapes. However, for handwritten documents, generating descriptors for all the points in the word images (73) and finding the optimal matching between two sets of descriptors are very time consuming and unnecessary (74). In order to localize informative keypoints in the word image, we apply the keypoint detector for SIFT proposed by D. G. Lowe (68).

In handwritten documents, we focus on the keypoints located on the strokes, namely, we remove the keypoints in the background or near the contour of strokes according to their intensity values. Therefore, word images are first smoothed by a Gaussian kernel, and then a set of keypoints in each word image are located, such as the red points shown in Figure 4.1(a). Finally, the keypoints with the intensity values smaller than a threshold are removed, which are always located in the background or along the boundaries of strokes. Figure 4.1(b) shows the final selected keypoints, the descriptors of which will be generated, as described in the next section.

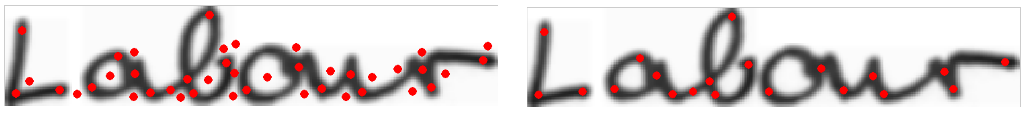

(a) Keypoints detected by SIFT detector. (b) Removing uninteresting keypoints.

Figure 4.1: Keypoints selection.

4.2.2 Heat Kernel Signature

After the keypoints are detected in the word images, a HKS descriptor will be extracted from a local patch centered at each keypoint. Heat Kernel Signature can capture local geometry of 3D shapes with short time scales and gradually represent global characteristics as time becoming larger (70). The heat kernel has the properties of invariant to isomeric, and stable to non-rigid deformations. Moreover, it can capture both local and global characteristics.

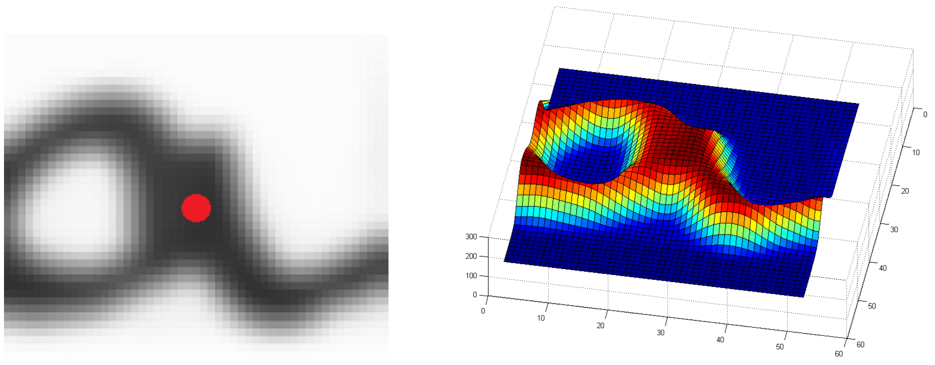

(a) (b)

Figure 4.2: Embed 2D image into 3D manifold. (a) illustrates the patch centered at the 6th keypoints in Figure 4.1(b) (assuming all the keypoints are sorted from left to right). The keypoint is marked as the red dot. (b) shows the 3D surfaces embedded from the 2D patch in (a). The intensity values are in the range of $[0, 255]$.

In order to apply HKS to word images, we should first embed a patch in a 2D word image into a 3D surface. We assume that P is a patch, with the size of $N \times N$, extracted from a word image I and centered at a keypoint, as shown in Figure 4.2(a).

The 3D Riemannian manifold M is the 2D surface embedded into 3D space from P, satisfying the condition that if (x, y) is a point in P, then there is a point (x, y, z) on M, where z is the intensity value of (x, y) in P (70), as shown in Figure 4.2(b). The heat diffusion geometry of patch P is obtained by using the Laplace-Beltrami operator over the manifold M (75):

$$(\Delta_M + \frac{\partial}{\partial t})u(\mathbf{x}, t) = 0, \tag{4.1}$$

where $\mathbf{x}$ is a point on M, Δ_M is the Laplace-Beltrami operator, and the solution $k(\mathbf{x}, \mathbf{x}', t)$ is named as the heat kernel, presenting how the heat between two points $\mathbf{x}$ and $\mathbf{x}'$ on the same surface diffusing from one to the other at time t, if we assume that the unit heat source is from the position of $\mathbf{x}$ at time $t = 0$. When M is a compact manifold, $k(\mathbf{x}, \mathbf{x}', t)$ can be expressed compactly by the eigenvalues $\{\lambda_i\}$ and eigenvectors $\{\phi_i\}$ of Δ_M (75):

$$k(\mathbf{x}, \mathbf{x}', t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i(\mathbf{x}) \phi_i(\mathbf{x}'). \tag{4.2}$$

Based on Eq. 4.2, HKS is proposed in (72) to present local and global characteristics around a point $\mathbf{p}$ on M as follows:

$$\mathrm{HKS}(\mathbf{p}, t) = k(\mathbf{p}, \mathbf{p}, t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i^2(\mathbf{p}). \tag{4.3}$$

In order to tolerate 2D noise around keypoints (70), the descriptor of $\mathbf{p}$ is constructed from all the points in P, weighted by a Gaussian kernel considering their distances to the center. This descriptor is called Deformation Invariant (DI) descriptor (70):

$$\mathrm{DI}(\mathbf{p}, t) = [\mathrm{HKS}(\mathbf{x}, t). * G(\mathbf{x}; \mathbf{p}, \sigma)]_{\forall \mathbf{x} \in \mathbf{P}}, \tag{4.4}$$

where G is a 2D Gaussian filter centered at p with standard deviation σ. Figure 4.3 shows the DI descriptors of the patch in Figure 4.2(a) at different t values, where the number of eigenvalues and eigenvectors we use in Eq. 4.3 is 100.

4.2.3 Discrete Version of Laplace-Beltrami Operator

Because we have finite number of points in a patch, we apply the discrete version of the Laplace-Beltrami operator based on cotangent scheme (76), in order to obtain $\{\lambda_i\}$ and $\{\phi_i\}$ in Eq. 4.3.

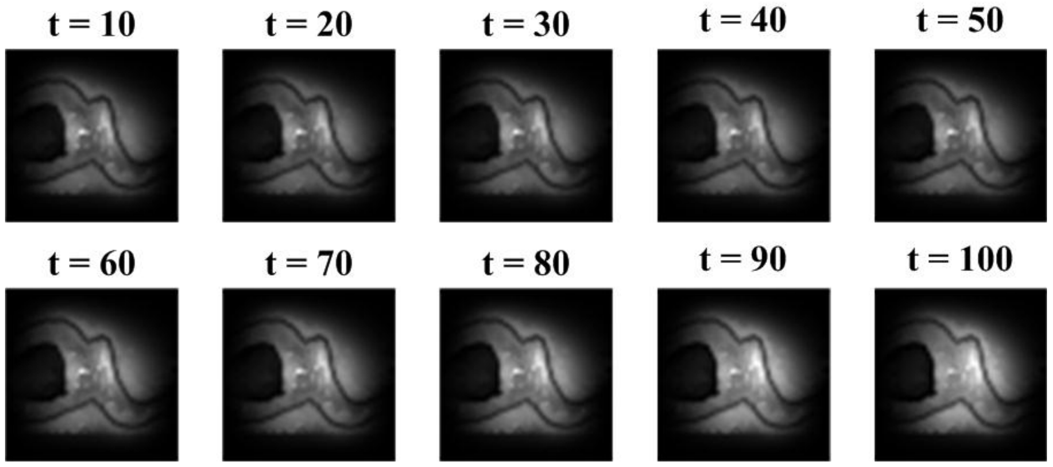


Figure 4.3: DI descriptors for the patch in Figure 4.2(a) with different t.

We assume that $P = \{p_1, p_2, ..., p_v\}$ includes all the pixels in a patch, and the intra-pixels as shown in Figure 4.4. We construct the triangular mesh based on P. The discrete version of the Laplacian matrix L is a $v \times v$ matrix, and computed as the following equations (71):

$$m_{ij} = \begin{cases} \frac{cot\alpha_{ij} + cot\beta_{ij}}{2}, & \text{if } i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

$$L_{ij} = \begin{cases} \sum_k \frac{m_{ik}}{s_i}, & \text{if } i = j \\ -\frac{m_{ij}}{s_i}, & \text{if } i \text{ and } j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where α_{ij} and β_{ij} are the angles depicted in Figure 4.4 (c), s_i is the area of all the triangles having the same vertex p_i. In order to compute the eigenvalues and eigenvectors of the non-symmetric matrix L, let S be a diagonal matrix with $S_{ii} = s_i$, and M with $M_{ij} = m_{ij}$, so that $L = S^{-1}M$. The generalized eigenvalue problem of L can be written as:

$$M\vec{v} = \lambda S\vec{v} \tag{4.7}$$

where λ and $\vec{v}$ correspond to $\{\lambda_i\}$ and $\{\phi_i\}$ in Eq. 4.3. The computational time cost for HKS is always very high, especially the larger the patch size is, the longer the time is needed to get eigenvalues and eigenvectors in Eq. 7. Therefore, we can apply the fast computation for heat kernel presented in (77).

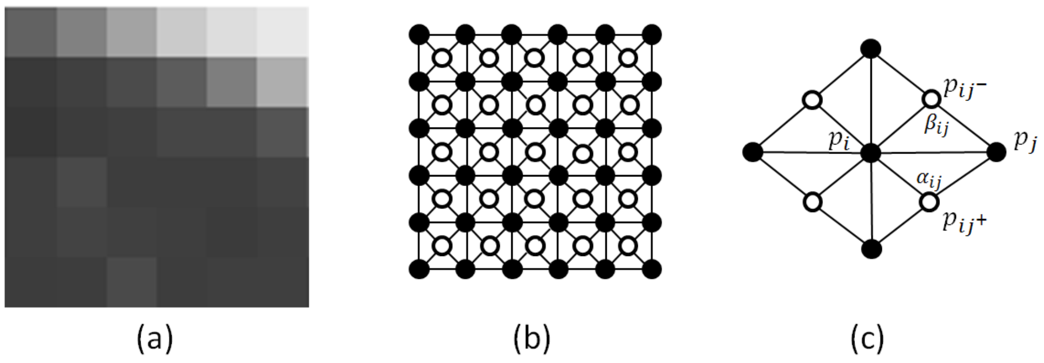


Figure 4.4: (a) A 6×6 patch. (b) The black dots are the centres of the pixels in (a), and the circles are intra-pixels. The lines between pixels represent the triangular mesh in the (x, y) dimensions. (c) A portion of the triangular mesh.

4.2.4 Scale Invariant HKS

Because we extract features directly from word images, without any pre-processing steps, word images may have different intensity values along strokes, namely, the manifold M maybe scaled along the intensity axis. Moreover, one disadvantage of HKS is its sensitivity to scale changes. Therefore, in order to remove the dependence to scale, (78) proposed a scale invariant version of HKS, named SI-HKS. SI-HKS is transformed by HKS in the following steps. First, instead of sampling heat kernel in time t, we sample logarithmically, with respect to $log_\alpha(t)$. Second, take the logarithm of HKS and make derivative with respect to $log_\alpha(t)$. At last, the discrete-time Fourier transformation of the descriptors generated in the second step finally removes the effect of scaling changes successfully. (70) shows that SI-HKS not only can tolerate isotropic scalings, but also can remove the dependence of HKS to scaling only in the intensity dimension. Therefore, we apply SI-HKS to achieve handwritten word image matching instead of HKS. Therefore, replacing HKS by SI-HKS in Eq. 4.4, the final descriptor of each point $\mathbf{p}$ is named as Deformation and Light Invariant (DaLI) descriptor and defined as below (70):

$$\text{DaLI}(\mathbf{p}, \omega) = [\text{SI-HKS}(\mathbf{x}, \omega). * G(\mathbf{x}; \mathbf{p}, \sigma)]_{\forall \mathbf{x} \in \mathbf{P}} \tag{4.8}$$

where DaLI$(\mathbf{p}, \omega)$ will be denoted as DaLI$(\mathbf{p})$ for short in the rest of this chapter. The Gaussian kernel is only applied after the computation for DaLI descriptors, and is used to weaken the effects of the points far away from the center keypoint in the patch. Therefore, applying Gaussian kernel will not destroy the invariance property of HKS.

4.2.5 Distance between two Descriptors

Note that we consider all the points within one patch. Moreover, word images are not normalized and also not corrected for skew and slant. Therefore, in order to tolerate in-plane rotation of points in the patches due to skew and scaling of illuminations along strokes, we generate differently rotated and scaled copies of the original DaLI descriptors to obtain the minimal distances between two DaLIs. Therefore, the distance d between DaLI($\mathbf{p}$) and DaLI($\mathbf{q}$) is (70):

$$d(\mathbf{p}, \mathbf{q}) = min_{\{\theta_i, s_j\}} \| T_{\theta_i, s_j}(\text{DaLI}(\mathbf{p})) - \text{DaLI}(\mathbf{q}) \| \tag{4.9}$$

where θ_i and s_j are selected in discrete sets, the descriptor is rotated by angle θ_i and scaled by s_j by the operation T_{θ_i, s_j}, and $\|\cdot\|$ is the L2-norm.

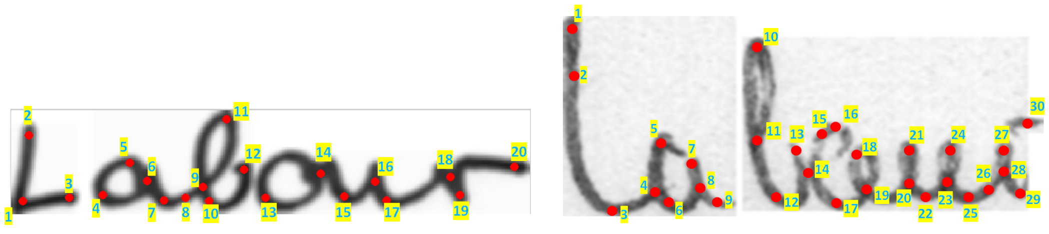

(a) (b)

Figure 4.5: A word 'Labour' written by two writers.

In order to match two word images without any preprocessing steps, the descriptors of the keypoints must be robust enough to tolerate the variations of different writing styles. The reason why we apply HKS descriptors is the fact that it gives the best performances for both the synthetic deformations and real deformations with illumination changes in the real scene images as presented (70). In handwriting scenarios, the same letter may have quite different appearances if written by two writers, and this situation can be considered as a case when the letter written by the second writer is the deformed version of the letter written by the first writer. Figure 4.5 shows two images containing the same word 'Labour' written by two writers, and we choose all the detected keypoints to show how HKS descriptors can tolerate different writing styles. There are 20 keypoints in Figure 4.5(a) and 30 keypoints in Figure 4.5(b), and we match the keypoints in Figure 4.5(a) to the ones in Figure 4.5(b) based on the distances between their DaLI descriptors. For example, if the first keypoint in Figure 4.5(a) is to be matched, we will compute the distances between its DaLI descriptors and the DaLI descriptors of all the keypoints in Figure 4.5(b). The distances will be sorted in the ascending order, and the rank of the true matched keypoint, i.e. the third keypoint in Figure 4.5(b), will be recorded. All the ranks are plotted in Figure 4.6. We can see than although the variations of the two words are very large, DaLI descriptors can tolerate the variations much better than SIFT features. Most of the true matched keypoints are ranked on the top positions in the ranking list if we apply DaLI descriptors, however, SIFT features cannot be robust enough to match two keypoints which have large variant writing styles. The matching method presented in (79) depends on the robustness of SIFT features on scales and rotations. However, in handwriting scenarios, two matched keypoints cannot be treated as a matching pair based on the Euclidean distance between their SIFT features as shown in Figure 4.6, because SIFT cannot be invariant to the non-rigid deformations, i.e. different writing styles.

Although DaLI descriptors cannot guarantee that all the true matched keypoints can be ranked on the top first, the triangular mesh structure of the keypoints will constrain the region where the true matched keypoints could appear, so that we can get the correct matching pairs by our proposed matching method, which will be described in the next section.

4.3 Word Image Matching

Keypoints are extracted from all the query and candidate images, and DaLI descriptors are calculated on the patches centered at each keypoint. Then Triangular Mesh structure is generated based on the spatial relations of all the keypoints in each word image. All DaLI descriptors and the Triangular Mesh structure are stored in a database for further usage. Given a query image, we calculate the distances between two sets of DaLI descriptors, using the method we will introduce in Section 4.3.2 and all candidate

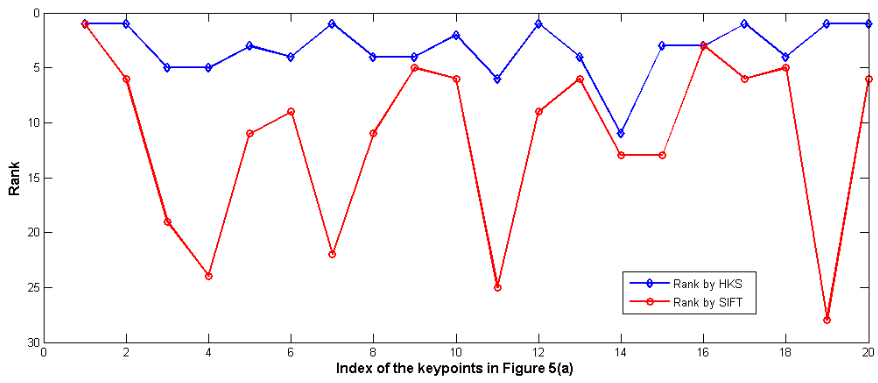


Figure 4.6: For each keypoint in Figure 4.5(a), we calculate the distances between its descriptor and the descriptors of all the keypoints in Figure 4.5(b). All the distances are sorted in the ascending order, and we only plot the position on which the true matched keypoint is in the ranking list. We plot the ranks for both the DaLI descriptors and SIFT features.

images are sorted with respect to their scores in the ascending order. Only images on the top of the ranking list are returned. The whole procedure is shown in Figure 4.7.

4.3.1 Structure of Keypoints

Keypoints may appear in any part of the word images and two keypoints may be located in the same column. As a result, we cannot simply concatenate all descriptors, sorted by their vertical locations from left to right to compare two sets of descriptors. This means that DTW or other methods measuring distance between two feature sequences will not be applicable.

As illustrated in Figure 4.1, the selected keypoints are mostly located at the start or end of strokes, intersection of two strokes or the locations strokes tending to change orientations. The local patches around these points can capture local characteristics of word images very well, and the spatial relations of all the keypoints can be used to represent the global structure of word images. Due to cursive handwritten styles, some keypoints belonging to different characters may have similar descriptors, but they may have different spatial relations with respect to their neighbouring keypoints. We apply the triangular mesh structure based on Delaunay triangulation algorithm, to connect spatial related keypoints together, as shown in Figure 4.8, where two keypoints

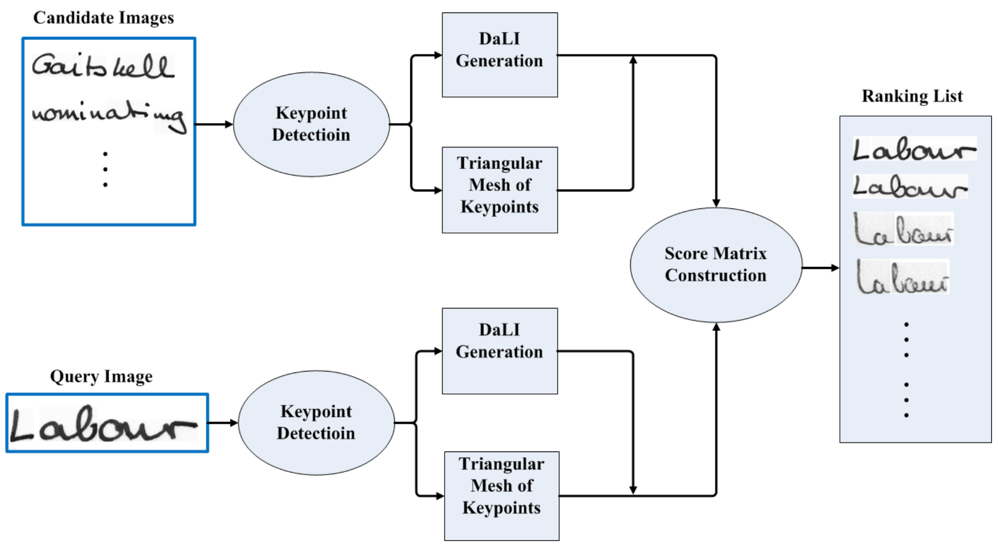


Figure 4.7: Procedure of our method for handwritten word image matching.

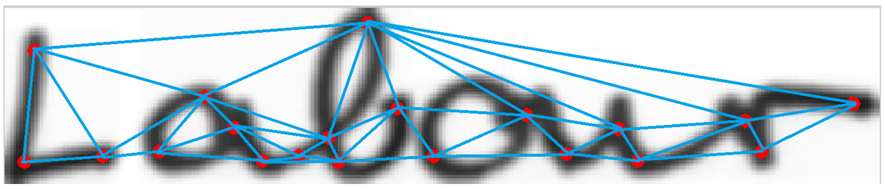


Figure 4.8: The triangular structure of keypoints.

connected by a blue line are treated as neighbours. It should be noted that some long connections between keypoints are not shown in Figure 4.8 because they will be partially overlapped with other short connections. If we include all the connections, the triangular mesh will not be shown very clearly.

Assume that we find n unique keypoints $\{kp_i\}$ in I, each of which with a coordinate $(x_i, y_i), i \in [1, n]$, sorting by y_i in ascending order, presenting the location of each keypoint. By applying Delaunay Triangulation algorithm to coordinates of all $\{kp_i\}$, we can get a triangular mesh for all keypoints, aiming at connecting neighbouring keypoints together, and constructing a triangular mesh structure (TMS) to represent I. From TMS, we generate two functions. One is $neighors(kp_i)$ which returns a list containing the indexes of all the neighbours of kp_i. The other is $Adj(kp_i, kp_j)$ which returns 1 if kp_i and kp_j are neighbours or 0 otherwise.

4.3.2 Score Matrix

After we obtain the DaLI descriptors and the triangular mesh structure of all the keypoints we detect for the query image and a candidate image, Score Matrix (SM) is used to calculate the distance between two sets of descriptors, namely measure the similarity between the two word images.

Algorithm 3 Construct SM and find the optimal matching score

```
1: $D(i,j) = \|\text{DaLI}(\mathbf{kp'_i}) - \text{DaLI}(\mathbf{kp_j})\|$, for all $i \in [1, n_c]$, $j \in [1, n_q]$, $\|\cdot\|$ is the L2-norm
2: Initialize $SM_s$ to $Inf$ and $SM_h$ to $NULL$ for all components
3: for $i = 1 \to n_c$ do
4:     $SM_s(i,1) \leftarrow D(i,1)$
5:     $SM_h(i,1) \leftarrow i$
6: end for
7: for $j = 2 \to n_q$ do
8:     for $i = 1 \to n_c$ do
9:         $Adj \leftarrow \{z | z \in neighbors'(kp'_i)$ and $i \notin SM_h(z, j-1)$ and
10:                $SM_s(z, j-1) \neq Inf\}$
11:        $z^* \leftarrow argmin_{z \in Adj} SM_s(z, j-1)$
12:        if $Adj$ is not an empty set then
13:            $SM_s(i,j) \leftarrow SM_s(z^*, j-1) + D(i,j)$
14:            $SM_h(i,j) \leftarrow [SM_h(z^*, j-1), kp'_i]$
15:            %comment: $kp'_i$ is appended at the tail
16:        end if
17:    end for
18: end for
```

We assume that Q is a query image having n_q keypoints, denoted by kp_j, with coordinates (x_j, y_j), $j \in [1, n_q]$, and C is a candidate image in the collection, having n_c keypoints, denoted by kp'_i, with coordinates (x'_i, y'_i), $i \in [1, n_c]$. Our task is that given a sequence of keypoints in Q, we should find the optimal matching sequence of keypoints in C and calculate the matching score. In order to present the triangular mesh structure of the query image, we first reorder all kp_i in a new sequence. From the first keypoint kp_1, we choose one of its neighbours and check if the neighbour has been already in the new sequence. If not, we append it into the new sequence, otherwise, another neighbour is chosen and checked. When a new keypoint is added into the new sequence, we will continue to check its neighbours and add one neighbour into the new sequence, until all the keypoints are in the sequence and no one is duplicated. If not all the keypoints are included in the new sequence, but all the neighbours under consideration cannot be included, we will back trace to the previous step. For example, if the two latest added keypoints are kp_s, kp_t, and all the neighbours of kp_t cannot be added into the new sequence, we will remove kp_t and check another neighbour of kp_s, which has never been checked. Because a keypoint may have more than one neighbours, we always consider the one with smallest Euclidean distance first. Consequently every two consecutive keypoints kp_i and kp_{i+1} are neighbours in the new sequence. Therefore, in the corresponding sequence of keypoints we will find in the candidate image, two consecutive keypoints should be also neighbours.

SM is a score matrix with the size of $n_c \times n_q$, each component of which contains two types of information: one is the optimal matching score $SM_s(i, j)$, and the other one is the optimal matching history $SM_h(i, j)$ from kp_1 to kp_{j-1}, if kp'_i is matched to kp_j. SM is constructed as described in Algorithm 3, and the aim is to find a matching sequence including the keypoints in C, each component of which can be matched to the corresponding keypoint in the query image optimally. In a local area of the query image, matched keypoints in the candidate image should have similar spatial relations to those in the query image, because at each matching step, we only consider the neighbours according to the triangular mesh structure, as stated in *Line* 9 in the Algorithm 3 and shown in Figure 4.9(a). One advantage of our algorithm is that it can remove dissimilar images more effectively, because in such situations, similar triangular mesh structure cannot be found, most of the entries of SM_s will be Inf, especially the right half of columns. We can discard these images quickly by only finding the matching score in the right-bottom part of SM_s, controlled by the thresholds T_c and T_r, as shown in Figure 4.9(b).

After constructing SM, from the last column n_q of SM_s to column T_c, we find the first column of SM, which at least has one entry, with score unequal to Inf, and row number bigger than T_r, where T_c and T_r are used to discard images which do not have enough matching points. Assume that the index of the entry with the minimum score in the selected column is (i, j), the final score of the candidate image is $SM(i, j)/j$. If no such entry exists, the candidate image is also discarded.

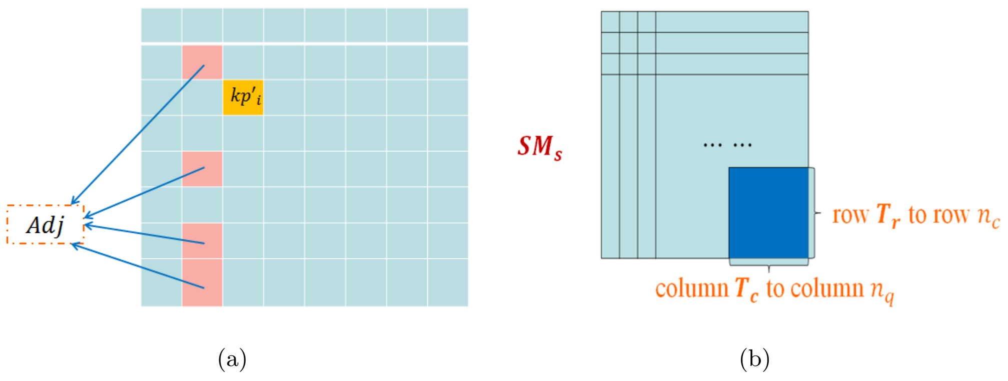


Figure 4.9: Score Matrix Construction. (a) We only consider the neighbors of the keypoint under consideration. (b) We only search the optimal matching score in the right-bottom of SM.

For each keypoint in the query image, we enforce every candidate image to have a matching keypoint which satisfies some constrains in our algorithm. Therefore, if a candidate image containing a different word in it either has Inf score, when we cannot find enough matching keypoints, or a very high score compared with other similar images, when the sequence of matching keypoints are quite different from the corresponding keypoints in the query image. As shown in Figure 4.10, for the candidate image in 4.10(a) at the bottom, we can get an optimal matching score at the right-bottom part of SM_s, so that we can also obtain an optimal matching sequence. However, for the candidate image in 4.10(b), most of the elements in the right half of SM_s have Inf values, so that we discard the candidate image as the irrelevant image.

Furthermore, as shown in Figure 4.11, another advantage of our algorithm is that matching pairs for two similar images can be found more correctly than Best-Bin-First (BBF) algorithm used in (68) which discards many correct matching pairs for word images with large variations.

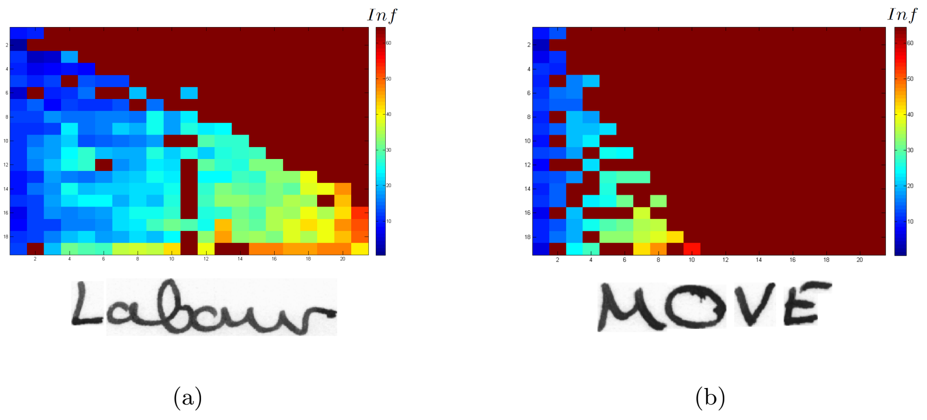


Figure 4.10: (a)SM_s of the candidate image, also containing the word 'Labour'. (b) SM_s of the candidate image, containing a very different word.

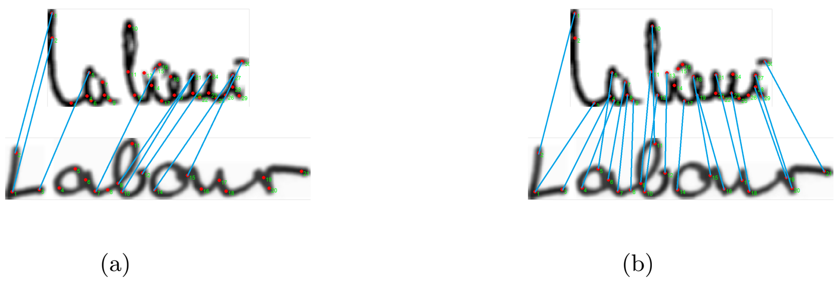


Figure 4.11: Examples of matching keypoints of two word images. (a) Matching keypoints by BBF. (b) Matching keypoints by our proposed method.

4.4 Experiments and Results

4.4.1 Experimental Setup

The dataset we use is IAM public handwritten database (58), with very large variations, and we choose 4000 commonly used handwritten word images written by different writers as a collection of candidate images and another 69 word images as queries, each of which appears more than 10 times in the collection. For comparison, we also carry out the experiments by the following methods: DTW with column features and local gradient histogram features (69), keypoints matching based on SIFT descriptor with BBF, and SIFT descriptor with SM. Assuming the width of one query image is w, in the pruning step, all the methods will discard the images with the width smaller than $0.5 * w$, or larger than $2w$.

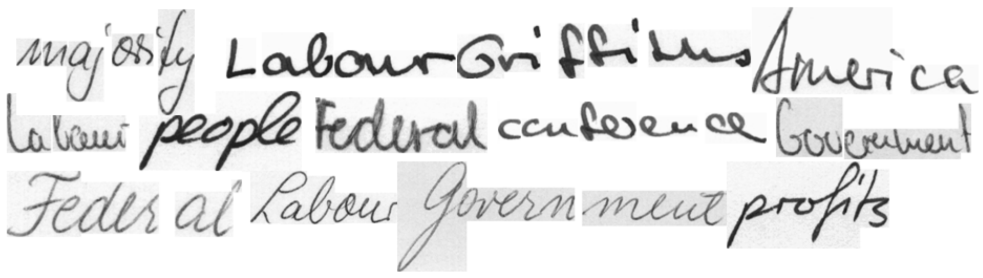


Figure 4.12: Examples of word images in our experiments.

Before we extract a sequence of 9 geometric features from each column of the word images used in (80), all the word images are binarized, slant and skew correction. Each word image is represented by a sequence of 9-dimensional vectors, which is denoted as column features (CF). For local gradient histogram features (LGH), after applying the same pre-processing steps, the sliding window is with height H and width $\omega = 32$, where H is the height of the word image. At each position, the sliding window is divided into 4×4 cells, and for each possible gradient orientation, one of the 8 orientation bins is assigned (69). The similarity measurement with both the features are calculated by DTW.

After preprocessing steps, column features (CF) and local gradient histogram features (LGH) are extracted based on the methods in (80) and (69) respectively, and the similarity measurement is calculated by DTW. SIFT features are extracted by the codes provided by D. G. Lowe online (68). Keypoints are matched by BBF, and word images in which the query image cannot find matches for more than 30% of its keypoints are discarded. The matching score for each candidate image is the sum of all the distances between matching keypoints.

Preprocessing steps are only needed for CF and LGH, but not needed for DaLI descriptors. The parameters needed for DaLI descriptors are set according to the experimental results shown in (70). In all the experiments, we set the size of the patch is $S \times S = 51 \times 51$, $\alpha = 2$, t is logarithmically sampled from 1 to 25 with increments of 1/16, $\sigma = \frac{S}{4}$, $\phi = \{-20, -10, 0, 10, 20\}$, $s = \{0.8, 1, 1.2\}$. Only the first 20 lowest frequencies of SI-HKS are used as the descriptors. If the number of keypoints in the query image and one candidate image is n_c and n_q respectively, $T_c = 0.8 * n_q$ and $T_r = 0.8 * n_c$.

4.4.2 Results and Discussion

Given a query image, all the distances or matching scores are sorted in the ascending order, and top n candidate images are returned, where n is the number of the matching images in the ground truth. The percentage of the number of correct matching candidate images in the final ranking list is used to compare different methods, named as Matching Rate (MR). For each query image, when we only keep top n returning word images, precision and recall are equal, and MR is also equal to the precision and recall. For example, there are 34 occurrences for the word 'Labour', so that from the returning list, we only keep the top 34 returned candidate word images to compare different methods.

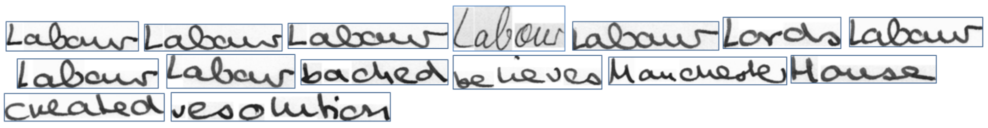
4.4.2.1 Comparison with the methods based on DTW

We first do the experiments to compare our proposed method with the methods based on DTW. As shown in Table 4.1, our proposed method outperforms the others. In Figure 4.13, the top 15 candidate images returned by different methods for the query word image in Figure 4.8 are shown. We can see that our method can return correct matching candidate word images much better, even for the images with large variations and different illumination changes along strokes. However, column features and local gradient histogram features with DTW can only match very similar word images as shown in Figure 4.13(a) and Figure 4.13(b), and fail when some word images cannot be de-skewed or de-slanted correctly. The matching results for another query image in Figure 4.14(a) are shown in Figure 4.14.

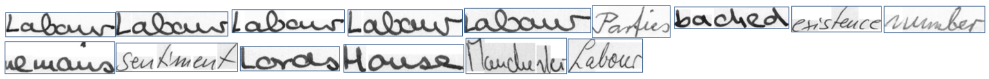
Moreover, binarization is always needed for column features. In our testing dataset, due to the different levels of illuminations along strokes, some words will lose their important parts after binarization. Consequently, some true matching word images with much degradation will not be returned in the top of the returning list.

Table 4.1: Experimental Results with Comparison to DTW-based Methods

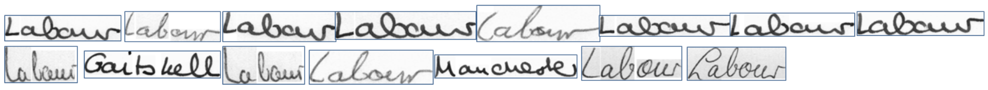
Methods	CF + DTW	LGH + DTW	Our Method
MR(%)	28.598	22.996	**37.827**



(a) Column features with DTW.



(b) Local Gradient Histogram features with DTW.

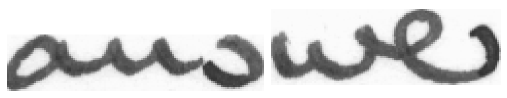


(c) Our proposed method.

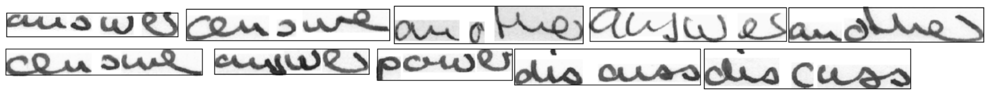
Figure 4.13: Top 15 candidate images returned by different methods for the query word image in Figure 4.8.

4.4.2.2 Comparison with the methods based on keypoints

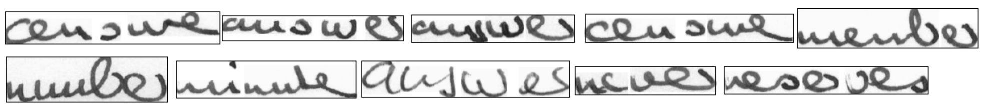
We also compare our proposed method with the methods based on keypoints. In Figure 4.15, the top 15 candidate images returned by different methods for the query word image in Figure 4.8 are shown. The same set of keypoints for each word image are used, but with different keypoint descriptors and different matching algorithms. Our proposed method also can return more correct matches on the top positions in the returned ranking list. However, for two word images containing large variations, if we only apply BBF to find matching keypoints based on their SIFT features, many correct matching pairs are discarded or keypoints are matched incorrectly. As shown in Figure 4.15(a), some very dissimilar word images are returned. If combining SIFT with SM, instead of finding matching pairs throughout the whole word image, which always brings in errors, the searching range is limited to a local area, and global consistency is also taken into account. Therefore, the matching results are improved. As shown in



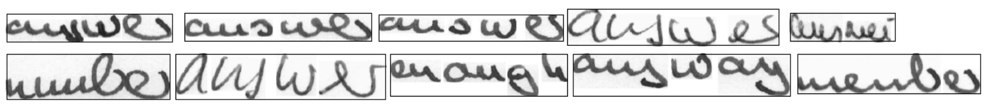
(a) A query word image 'answer'.



(b) Column features with DTW.



(c) Local Gradient Histogram features with DTW.



(d) Our proposed method.

Figure 4.14: Top 10 candidate images returned by different methods for the query word image in (a).

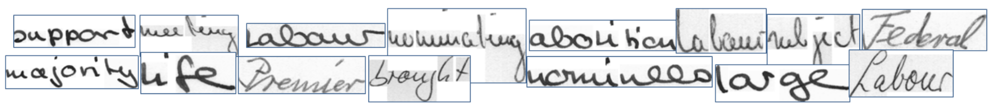
Table 4.2, both methods with SIFT cannot provide more satisfactory results than our method, which shows that DaLI descriptor can be more robust to tolerate variations of writing styles.

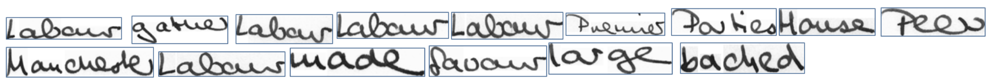
In figure 4.16, we can see that for the query word image in Figure 4.16(a), SIFT features with BBF cannot even return one correct matching word image in the top 10 of the returning list. If we apply our proposed matching algorithm, the matching results are improved obviously. However, the results of our proposed method shown in Figure 4.16(d) still indicates our best matching performance.

Table 4.2: Experimental Results of Keypoint-based Methods

Methods	SIFT+BBF	SIFT+SM	Our Method
MR(%)	8.126	21.584	**37.827**

Based on our proposed matching methods, we also do the experiments to compare different keypoint detection algorithms. There are four widely used keypoint detection


(a) SIFT features with BBF.


(b) SIFT features with SM.

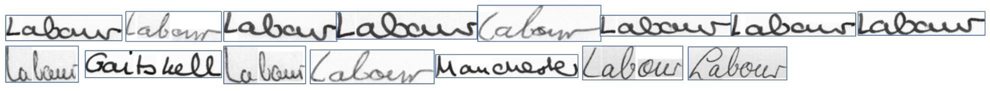
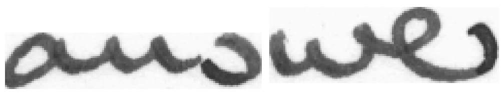

(c) Our proposed method.

Figure 4.15: Top 15 candidate images returned by different methods for the query word image Figure 4.8.

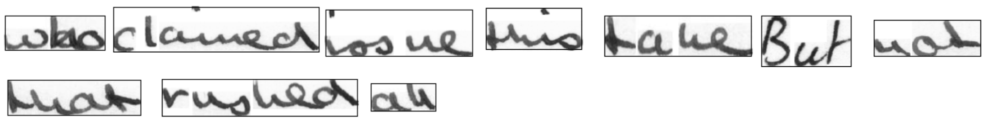
algorithms: Harris algorithm (81), LoG (Laplacian of Gaussian) algorithm (82), Gilles algorithm (83) with radius 4, and SIFT (Scale Invariant Feature Transform) keypoint detector (68) we use in previous experiments. The keypoints detected by different algorithms are shown In Fig. 4.17.

For one set of keypoints detected in a word image by one of the four keypoint detection algorithms, we extract DaLI descriptors for each keypoint, and apply our proposed matching algorithm to find matching candidate word images. We do the same word matching experiments based on the four keypoint detection algorithms, and compare their results based on MR, shown in Table 4.2. The numbers of the keypoints detected by Harris and LoG are much more than the numbers of keypoints detected by SIFT and Gilles. However, as shown in Table 4.3, more keypoints do not give us better matching results, but even much worse performance.

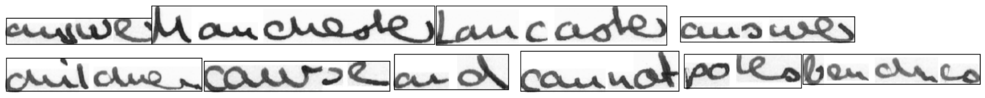
For the keypoints detected by Harris, most of the keypoints are on the boundaries of strokes, so that most of the keypoints can only be connected with the ones on the same strokes, but on the other side of the boundaries. Consequently, the triangular mesh structure of the keypoints cannot represent the global geometric structure of the keypoints in the whole word image. On the other hand, some of the keypoints detected by LoG are located very close to each other, only one of which is needed to construct the triangular mesh structure. More keypoints may lead to bad effects to the matching



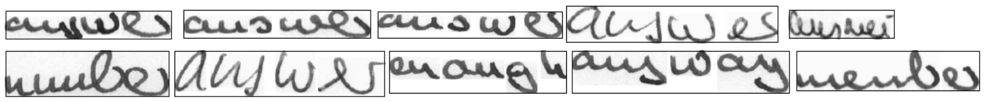
(a) A query word image ’answer’.



(b) SIFT features with BBF.



(c) SIFT features with SM.



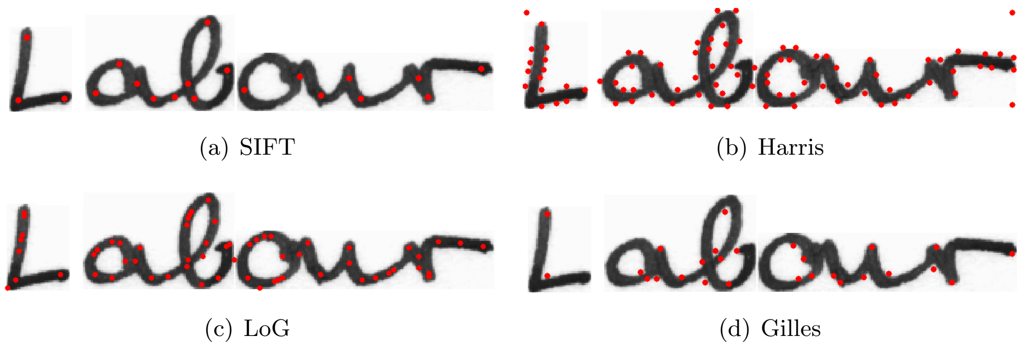
(d) Our proposed method.

Figure 4.16: Top 10 candidate images returned by different methods for the query word image in (a).

performance. Gilles can detect a similar set of keypoints as SIFT, but the keypoints are not located along the center axis of the strokes. The keypoints detected by SIFT are always located along stokes, and can contain the ones which are very important for matching performance, such as the ends of strokes, or the intersections of strokes. SIFT can detect enough keypoints and do not contain unnecessary keypoits which may have bad effects on the triangular mesh structure, so that in our experiments, using the keypoints detected by SIFT provides the best performance.

Table 4.3: Experimental Results of Different Keypoint Detection Methods

Methods	SIFT	Harris	LoG	Gilles
MR(%)	**37.827**	13.419	11.989	34.770



(a) SIFT (b) Harris

(c) LoG (d) Gilles

Figure 4.17: Keypoints detected by different methods.

4.5 Conclusion

Instead of extracting column features from binarized word images, or calculating gradient based features from the detected keypoints, we extract HKS descriptors from the local patches centered at every keypoint in the query and candidate images. HKS is isometric invariant and robust to non-rigid deformations, which is very suitable for handwriting scenarios, but is rarely used in handwritten document analysis. Moreover, instead of applying DTW to compare two sequences of features, we propose a new similarity measurement method based on the triangular mesh structure of all the keypoints in the word images. HKS descriptors are used to capture the local features, and triangular mesh structure is used to keep the global consistency, so that our proposed method not only can find the optimal matching pairs of keypoints, but also can quickly discard dissimilar candidate images due to different global structure. As shown in our experiments, our proposed new method can capture more robust and reliable features for word images and outperforms other commonly used methods. Besides, we also test different kinds of keypoint detection algorithms, and SIFT keypoint detector can work the best with HKS descriptors for our proposed method.

In future, efforts should be put on how to find stable keypoints, so that HKS and triangular mesh structure can be made full use of. Moreover, more sophisticated method should be proposed to find optimal alignment of two sets of DaLI descriptors for rotated images. We also would like to work on how to tolerate missing keypoints.

Chapter 5

Segmentation-free Keyword Spotting

In this chapter, we will present a new segmentation-free method for keyword spotting in handwritten document images based on Heat Kernel Signature (HKS), which has been already described in Chapter 4. After all the keypoints are located on the document pages and the query image, HKS descriptors are extracted from a local patch centered at each keypoint. In order to locate the positions where the query image appears in document pages, we propose a searching method which tries to locate a local zone which contains enough matching keypoints corresponding to the query image.

5.1 Introduction and Related Works

Due to the limitations of OCR-based methods, Keyword Spotting becomes an alternative method to spot query words in a large collection of document images. Unlike Optical Character Recognition (OCR), which tries to translate the whole documents into ASCII text, keyword spotting always returns a ranking list containing similar word images corresponding to the query image, along with bounding boxes on the documents, but with no need of knowing the ASCII content. Because of large variations in handwritten documents, degradation of historical manuscripts, or real scene images containing both text and non-text content with complex layout, OCR always gives us poor recognition results. Thus, word spotting can be used as an alternative way to retrieve our interesting information in these situations.

Keyword spotting in handwriting was first introduced in (84) and treated as an alternative way to index handwritten documents. Because of unconstrained writing styles, consecutive characters may be connected together or long ascenders and descenders belonging to different text lines are touched, segmenting characters correctly is always difficult and impossible. (85) indexed historical documents based on word image matching, without character segmentation.

Because intra-space within words and inter-space between different words are also difficult to differentiate, some methods try to apply keyword spotting directly to text lines, avoiding word segmentation errors. In (80), the trained HMMs can return the possibility of how likely one text line containing the query image, and text lines with possibility higher than a threshold are returned as positive matches. Besides, in (45), BLSTM is used for keyword spotting, with a modified CTC token passing algorithm, and outperforms the methods based on DTW and HMM models. However, HMM and BLSTM both depend on a large number of training samples, in some cases, training data and corresponding transcripts cannot be available.

Moreover, features used in the methods described above should be extracted from normalized word or text line images, namely, all images should be cleaned or binarized, with skew or slant correction, in order to eliminate variations of different writing styles, and especially, some features are language specific. Further more, word segmentation or text line segmentation errors cannot be avoided. (86) indexed ancient manuscripts without any segmentation and is language independent. The query images and documents are described as a collection of zones of interest (ZOI) points or guides, which denote the vertical strokes of text. The beginning ZOI point of the query image is matched to every guide on the document images, and other ZOIs are matched to best local positions with small displacements. The score of the guide is the sum of the distances between every pair of matched ZOIs and guides based on their gradient angles. Finally, guides with scores smaller than a threshold are returned as positive hits.

In (73), segmentation-free word spotting is based on bags of features of Scale-Invariant Feature Transform (SIFT) (68) descriptors extracted from densely sampled patches, so that every word can be covered by at least one patch. Each patch is presented by a SIFT feature vector, and Latent Semantic Indexing (LSI) is used to refine feature descriptors in document images, by transforming the feature descriptor space to a topic space. For a given query image, the feature descriptor is projected onto the topic space of every document. Patches with enough evidences that the query word most likely appears are the final retrieved zones.

In this chapter, we propose a segmentation-free keyword spotting method based on HKS, which can tolerate large variations in handwriting scenarios as presented in Chapter 4. Instead of extracting features for each pixel on the documents, we only extract HKS descriptors from local patches centered at keypoints, which locates along text strokes. In the spotting phase, we only focus on the local zones where the query word appears most likely, based on the similarity between keypoints on the document pages and the ones in the query image, instead of measuring densely sampled patches. We also propose a new method to find the optimal matching path of keypoints in each local zone, in order to obtain the matching score and further discard irrelevant local zones which does not have enough matching keypoints along the optimal path.

5.2 Historical Manuscripts written in English

5.2.1 Keypoint Detection

We detect a set of keypoints on the candidate documents and the query word using the keypoint detector for SIFT in (68). We only focus on text strokes, therefore we remove the keypoints with lower intensity values than a threshold, which most likely lie in the background. Moreover, some strokes may be blurred and unclear because of degradation. By fixing an appropriate set of parameters for the detector, we try to find enough keypoints for all the query word images and candidate documents, which are sampled along text strokes. Fig. 5.1 shows the keypoints found in a query image containing the word "Company".

5.2.2 Keyword Spotting

For a given query image, after the keypoints are detected, they are sorted by their vertical locations from left to right, and DaLI descriptors are calculated for all the keypoints. DaLI descriptors are calculated in the same way as described in Chapter 4. Assume that, there are m keypoints in the query image Q, denoted by q_k, and their corresponding DaLI descriptors are presented by $\mathbf{DaLI}(q_k)$, $k \in \{1, 2, ..., m\}$. For the document pages, the jth keypoint in the ith document D_i is denoted as kp_j^i, and its DaLI descriptor is $\mathbf{DaLI}(kp_j^i)$.

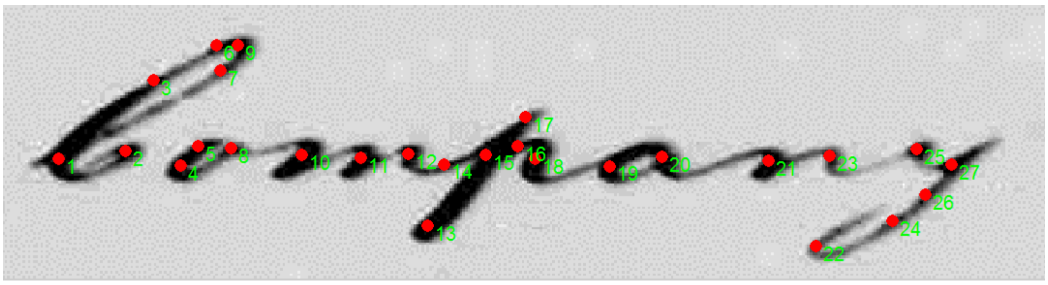


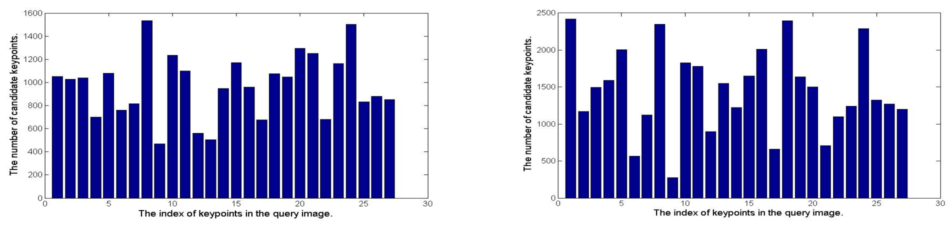
Figure 5.1: An example of keypoints found in a query image. The numbers are the indexes of keypoints according to their vertical locations in the image.

5.2.2.1 Candidate Keypoints

In order to find locations on the document that may match with the query image, we should search through all keypoints in each document and check whether there is one or more local zones containing enough similar keypoints compared with the ones in the query image. For a given document D_i with n keypoints, we first calculate the distances between all its keypoints and every keypoint in Q. The distance is defined as:

$$d_k^{i,j} = d(q_k, kp_j^i) = \| DaLI(q_k) - DaLI(kp_j^i) \| \tag{5.1}$$

where $k \in \{1, 2, ..., m\}$, $j \in \{1, 2, ..., n\}$, and $\| \cdot \|$ is the L2 norm. For each q_k, after sorting all distances $d_k^{i,j}$ in the ascending order, we should decide how many keypoints in D_i we should consider as the candidate keypoints for q_k. The local patch centered at q_k always contains the important part of a character or parts of neighbouring characters, which may appear many times in one page. For different patches, the number of appearances may be different in one document page, and for the same patch, it may appear different times in different pages. Thus, we cannot just keep top n candidate keypoints for all q_k, where n has a predefined value. For example, if character 'm' appears 50 times in one document page, and character 'p' appears only 10 times, we choose the top 20 keypoints on the document as the candidates for both 'm' and 'p', we will miss many locations where 'm' actually appears and bring in many irrelevant keypoints of 'p'. Therefore, we decide the threshold used to discard irrelevant keypoints based on the distribution of all distances $d_k^{i,j}$ in D_i, $j \in \{1, 2, ..., n\}$, for q_k. We first calculate the minimum distance value min and the maximum distance value max of



(a) (b)

Figure 5.2: The plot of the number of candidate keypoints for each keypoint in Fig. 5.1 for two documents. (a) The number of candidate keypoints for each keypoint in Fig. 5.1 for the left document in Fig. 5.4. (b) The number of candidate keypoints for each keypoint in Fig. 5.1 for the right document in Fig. 5.4.

$d_k^{i,j}$, and the threshold is chosen as $min + 0.2 \times (max - min)$, which is experimentally tested. In Fig. 5.2, we can see that in different document pages, the same keypoint in the query image has different candidate keypoints, and different keypoints have different number of candidate keypoints in the same document page. We try not to bring in too many irrelevant keypoints and keep all of the relevant keypoints as candidates.

Thus, all the keypoints with distances larger than the threshold are discarded, and the rest is treated as candidate keypoints for q_k. In order to record the positions containing the candidate keypoints, we use a matrix $Mark$, initialized to 0 for all its elements, and with the same size as the document image. For each q_k, we mark the positions of its candidate keypoints as k in $Mark$, and record the corresponding distances into another matrix $Dist$. In some cases, one kp_j can be the candidate keypoints for more than one keypoints in the query image, for example, in the word 'manual', 'a' appears twice, keypoints which locate at the strokes of 'a' on the document pages will be the candidate keypoints for both 'a' in 'manual'. In such situations, the same component of $Mark$ will be marked several times. In order to avoid replacing the value of one element by a later assignment, before we mark one component, we first check if its value is equal to 0, namely, it has not be marked yet. If not, we will mark one of its 8 neighbours as the new value, instead of replacing its previous value, based on the fact that two keypoints will not locate next to each other, i.e. the 8 neighbours are always free for assignment.

5.2.2.2 Matching Score of Local Zones

A local zone for kp_j^i is defined as an area around kp_j^i, and treated as a positive matching against the query image, only if it contains enough matching candidate keypoints. In other words, for each position of the keypoints in $Mark$, we check whether its local zone contains at least $0.7 * m$ different numbers. If so, the local zone is returned and its matching score against Q is the sum of the distances of the corresponding components in $Dist$. In order to narrow the scope of the search, we only check the local zones centered at $(r, c)^T = (x_j, y_j)^T + (t_1, t_2)^T$, where $(x_j, y_j)^T$ is the coordinate of a candidate keypoint on the document for a keypoint in Q, which locates near to the center pixel, such as the 16th keypoint in Fig. 5.1, and $(t_1, t_2)^T$ is the position shift from the selected keypoint in Q to the center pixel of Q. According to our experiments, which keypoint is chosen near to the center pixel will not affect the results greatly, because the size of the local zone is determined by font, so that all word instances can be covered, and a little position shift due to different selected keypoints in Q is always not significant. We can freely choose any keypoint near to the center, which always locates along the strokes of the same character. In our experiment, we only choose the one which is nearest to the center.

There is another issue in that given a local zone, how to decide the actual number of matching keypoints. We cannot just count the amount of different numbers in the corresponding area in $Mark$, because the order in which these numbers appear from left to right is also an important aspect in deciding whether the local zone under consideration is a positive matching or not. Thus, we propose a new method based on dynamic programming to find the actual matching path of the candidate keypoints. Here, local zones with the matching paths larger than $0.7 * m$ are treated as positive matchings. Assume that in a local zone, there are n' keypoints, sorted by their vertical locations in the ascending order, and for each position of the keypoints, it has at most 8 different numbers in $Mark$ at its surroundings, namely, we assume each keypoint on the document can be mapped to at most 9 different keypoints in the query image. We construct a matrix C, with the size of $9 \times n'$, and the ith column contains the indexes of keypoints in the query image, which the ith keypoint in the local zone is mapped to. Based on C, we apply the following algorithm in Algorithm 4 to construct another matrix H, with the same size as C, and each component stores the optimal longest path of different ascending numbers from the beginning to the current position:

Algorithm 4 Construct the matrix H, and find the optimal matching path and score

```
1:  $H(1,:) = C(1,:), H(2:n,:) = 0$
2:  for $j = 2 \to n$ do
3:      for $i = 1 \to 9$ do
4:          for each $C(i',j')$, $i' \in [1,9]$, $j' \in [max(1, j-5), max(1, k-1)]$ do
5:              if $C(i',j') = C(i,j)$ and $| H(i',j') | > | H(i,j) |$ then
6:                  $H(i,j) = H(i',j')$
7:              end if
8:              if $C(i',j') < C(i,j)$ and $| H(i',j') | +1 > | H(i,j) |$ then
9:                  $H(i,j) = [H(i',j'), C(i,j)]$
10:                 % $C(i,j)$ is added to the last position in $H(i,j)$
11:             end if
12:         end for
13:     end for
14: end for
```

The keypoints in the query image are sorted by the vertical positions in the ascending order from left to right, therefore, in our algorithm we try to find a longest path with the values of all components also in the ascending order, each of which is chosen from 3×3 area in $Mark$ centered at a position of one keypoint in the local zone, in order to take the geometry of the keypoints into account. Furthermore, because all the matching keypoints may not appear one by one exactly the same as the ones in the query image, some noise keypoints may appear in the local zone, and parts of long ascenders or descenders in different text lines may also appear in the local zones, we enlarge the searching range to the previous 5 keypoints, avoiding missing any possible optimal matching sequence. An example of our algorithm is shown in Fig. 5.3, the optimal path is $[1, 2, 3, 5, 6]$.

After we finish constructing H, we find the longest path stored in H as the actual matching sequence, and the length of the path is used to represent the amount of matching keypoints for the local zone. The matching score is the sum of the corresponding distances in $Dist$ of each component in the path, divided by the length of

		27	36								
			2								
1							5	10	25		
								2	26		
		46									
		8									6
				46	6	8					4
					3	11					
					28	13					

1	46	27	46	5	6
	8	36	6	10	4
		2	8	25	
			11	26	
			3	2	
			13		
			28		

Figure 5.3: The left figure is an example of matrix $Mark$. Each component in the dark grey is the position where a keypoint kp_j^i on the document image D_i appears, and the numbers in 3×3 area are indexes of keypoints in the query image each kp_j^i is mapped to. The right figure is the corresponding matrix C. Each column of C records different numbers at and around the positions of every keypoint in $Mark$.

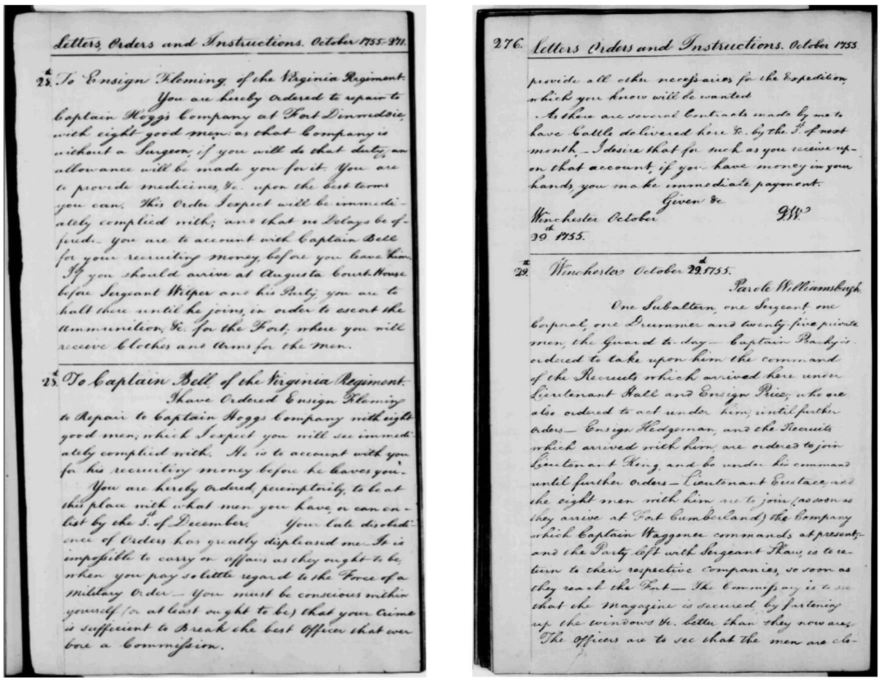
the path. If there are more than one path with the same largest length, we choose the one with smallest matching score. After all positive matching zones in all document pages are located, we sort them by their matching scores in the ascending order. The top ones are most similar with respect to the query image.

5.2.3 Experiments and Results

5.2.3.1 Experimental Setup

We test our method on the George Washington (GW) dataset, containing 20 pages and 4860 words (87). Fig. 5.4 shows two pages in the GW dataset. The transcript and bounding box for each word is provided in the ground truth. Because words with smaller lengths are always stop words, we only consider words larger than 5 characters in our experiments, and compare our results with the performance in (73).

First, we generate query word images from pages based on the bounding boxes in the ground truth, and remove strokes belonging to other words manually. Then, DaLI descriptors are extracted from all pages and query word images. The size of each patch P is $S \times S = 50 \times 50$, the standard deviation of Gaussian function is $\frac{S}{4}$, and we take the first $T = 15$ lowest frequencies for DaLI. In order to make the comparison, we spot all words with length larger than 5 in all pages, and the performance is evaluated by the average precision and recall. All the thresholds and parameters are tested based on experiments to provide the best results.



Letters, Orders and Instructions. October 1755-271.

28th. To Ensign Fleming, of the Virginia Regiment.

You are hereby Ordered to repair to Captain Hoggs Company at Fort Dinwiddie, with eight good men: as that Company is without a Surgeon, if you will do that duty, an allowance will be made you for it. You are to provide medicines, &c. upon the best terms you can. This Order I expect will be immediately complied with; and that no Delays be offered. You are to account with Captain Bell for your recruiting money, before you leave him. If you should arrive at Augusta Court-House before Sergeant Wilper and his Party, you are to halt there until he joins, in order to escort the Ammunition, &c. for the Fort; where you will receive Clothes and Arms for the men.

28th. To Captain Bell, of the Virginia Regiment.

I have Ordered Ensign Fleming to Repair to Captain Hoggs Company with eight good men; which I expect you will see immediately complied with. He is to account with you for his recruiting money before he leaves you.

You are hereby Ordered, peremptorily, to be at this place with what men you have, or can enlist by the 1st of December. Your late disobedience of Orders has greatly displeased me. It is impossible to carry on affairs as they ought to be, when you pay so little regard to the Force of a Military Order. You must be conscious within yourself (or at least ought to be) that your Crime is sufficient to Break the best Officer that ever bore a Commission.

276. Letters Orders and Instructions. October 1755

provide all other necessaries for the Expedition, which you know will be wanted

As there are several Contracts made by me to have Cattle delivered here &c. by the 1st of next month, I desire that for such as you receive upon that account, if you have money in your hands, you make immediate payment.

Given &c.

Winchester October 29th 1755. G.W.

29th. Winchester October 29th 1755.

Parole Williamsburgh.

One Subaltern, one Sergeant, one Corporal, one Drummer and twenty-five private men, the Guard to-day. Captain Peachy is ordered to take upon him the command of the Recruits which arrived here under Lieutenant Hall and Ensign Price; who are also ordered to act under him, until further orders. Ensign Hedgeman, and the Recruits which arrived with him, are ordered to join Lieutenant King, and be under his command until further orders. Lieutenant Eustace, and the eight men with him are to join (as soon as they arrive at Fort Cumberland) the Company which Captain Waggoner commands at present; and the Party left with Sergeant Shaw, is to return to their respective Companies, so soon as they reach the Fort. The Commissary is to see that the Magazine is secured by fastening up the windows &c. better than they now are.

The Officers are to see that the men are clo-

Figure 5.4: Two pages in the GW dataset.

5.2.3.2 Results

All the words in the dataset with lengths larger than 5 are used as the query images. Our proposed method has the mean precision 62.47% and the mean recall 92.38%. As shown in Fig. 5.5, most of the positive matching instances are returned. Moreover, only based on the length of the matching path, most of the irrelevant local zones are discarded, and others can be further discarded due to their large matching scores.

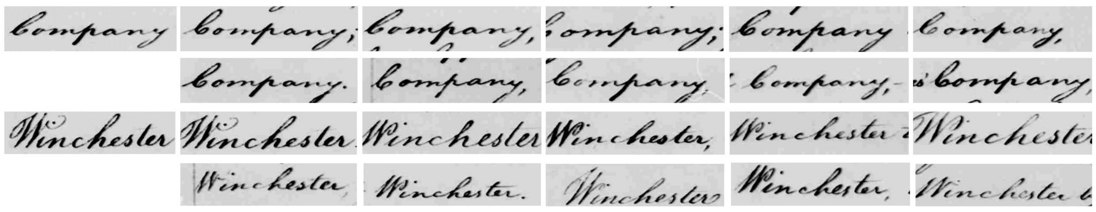


Figure 5.5: Top 10 possitive mathching local zones for two query images.

Our method is based on the descriptors of keypoints, and their geometric locations. Without extracting features from densely sampled small regions in a local zone, our method can save much computation time. Furthermore, because we only check local zones for the candidate keypoints of a small set of selected keypoints in the query image, there is a substantial reduction in the search space.

From the results, we notice than in some cases, an instance of the query word cannot be spotted because we do not detect enough keypoints by SIFT detector, due to low resolution, even though, all the detected keypoints are matched correctly. Thus, compared with (73), which got the average precision 53.76% and the average recall 93.39%, our average recall is a little lower, however, we have much higher average precision.

5.3 Handwritten Bangla Documents

Besides English handwritten documents, we would like to apply our keyword spotting method to the documents written in different languages. In this section, we will present how to make our proposed method spot query word images on Bangla documents with satisfactory results.

5.3.1 Descriptor Generation

5.3.1.1 Localization of Keypoints

In document images, regardless of printed or handwritten documents, the important information we can use to retrieve useful information is the characteristics of the appearing characters or words. Densely extracting features from all points on the strokes is very time consuming, not only for extracting and storing the large features, but also for similarity matching. Therefore, how to localize the important keypoints, which can capture the maximum information the documents can provide, is one of the most crucial works we should consider.

There are four normally used keypoint detecting algorithms: Harris algorithm (81), SIFT (Scale Invariant Feature Transform) keypoint detector (68), LoG (Laplacian of Gaussian) algorithm (82), and Morphological operation (MO) on binary images (88). In Fig. 5.6, the keypoints detected by these four algorithms are shown. We can see that in Fig. 5.6(a), Harris always find keypoints along the boundary of strokes, and SIFT in Fig. 5.6(b) detect some keypoints in the background, which are not useful. In Fig. 5.6(c), all the keypoints are along the strokes, but some keypoints are located very near to each other, the features of which may be very similar, leading to duplication. The most important characteristics the strokes contain are always located at the start or end of strokes, intersections of strokes, or the positions where the strokes try to change its orientation, so the end points and intersect points by morphological operations in Fig. 5.6(d) are always the desirable points we want. However, in some cases, some keypoints may be missed. Therefore, in order to include all the keypoints we want, we combine the results of SIFT and MO.

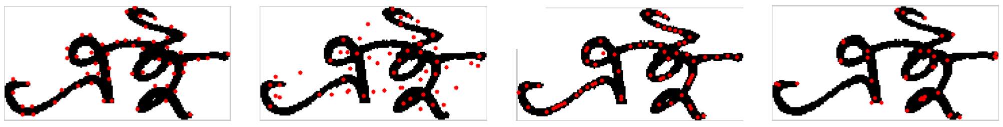

(a) Harris (81) (b) SIFT (68) (c) LoG (82) (d) MO (88)

Figure 5.6: Keypoints detected by different algorithms.

For the keypoints detected by SIFT detector, we remove the keypoints in the background, by checking the intensity values. If the images are binary (1 is the foreground, 0 is the background), we can easily remove the keypoints with the intensity value of 0, as shown in Fig. 5.7(a). After combining the keypoints with MO, as shown in Fig. 5.7(b), we can find out that some keypoints are located in the very near positions. Therefore, we group keypoints with small Euclidean distances to each other together, and only keep one of them, which is located nearest to their center. The final keypoints we use in our experiments are shown in Fig. 5.7(c).

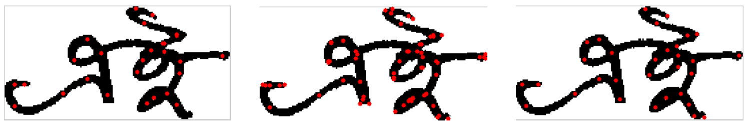

(a) (b) (c)

Figure 5.7: The final Keypoints we will use in the experiments. (a) Keypoints detected by SIFT detector after removing keypoints in the background. (b) Combining the keypoints in Fig. 5.7(a) with the ones in Fig. 5.6(d). (c) Removing near keypoints.

5.3.1.2 Size of Local Patch

The descriptors are extracted from the local patch centered at each keypoint we detect, because in handwritten documents, the same word may be written in variant writing styles, fixing the size of all patches may bring in unwanted parts of strokes, so that the descriptors of the similar keypoints in different occurring same words may have large difference.

In order to automatically decide the size of patch centered at one keypoint, we use the Entropy values in different sizes of patches centered at the same keypoint to find the optimal one. (86) also used the entropy to compute the size of ZOI (zones of interest). Centered at a keypoint kp, we calculate HKS descriptor HKS(kp), from a patch P with the size of $(2 * r + 1) \times (2 * r + 1)$, where $r \in [5, 50]$. For each r, we get the Entropy as following equations:

$$Energy = \mathrm{HKS}(kp, t = 0) \tag{5.2}$$

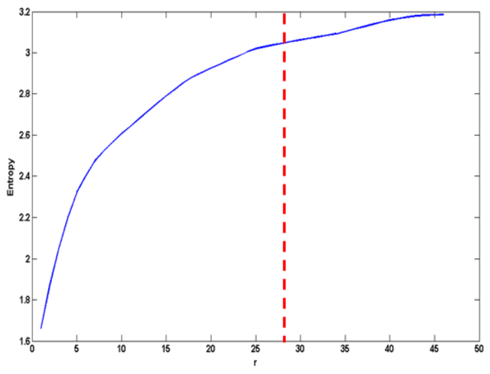
$$Entropy_r = -\frac{1}{E} \times \sum_{p_i \in S} Energy(p_i) * ln \frac{Energy(p_i)}{E} \tag{5.3}$$

where S is the set of all points on the strokes in P, $Energy(p_i)$ is the Energy value of p_i, and $E = \sum_{p_i \in S} Energy(p_i)$.

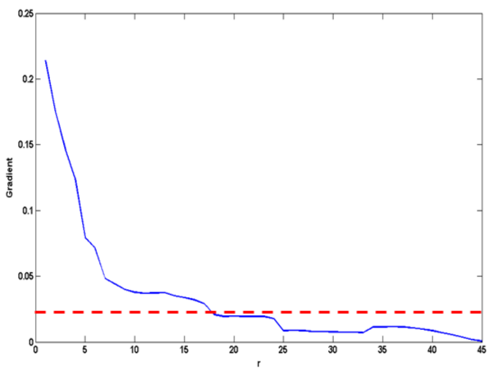
While r is increasing, more points are included in S, so that the energy becomes large, so does the entropy value. However, the increase of entropy values is large at the beginning, but slows down when enough information is included. Fig. 5.8(a) shows the entropy values of the patches centered at the left most keypoint in Fig. 5.7(c), when r is increasing from 5 to 50. We can see that at the beginning, the entropy value increases very fast, but slow down when r is bigger than 20. In order to track when the entropy starts slowing down, namely, the patch has already include enough information, we calculate the gradient values shown in Fig. 5.8(b), and choose the position when the gradient begins smaller than ϵ. If we choose $\epsilon = 0.02$, the optimal size of the patch is 23, as indicated as the red dot lines in Fig.5.8.

5.3.1.3 Patch Normalization

The similarity measure of the distance between two HKS descriptors requires that the descriptors have the same dimension, however the size of patches may be different, so



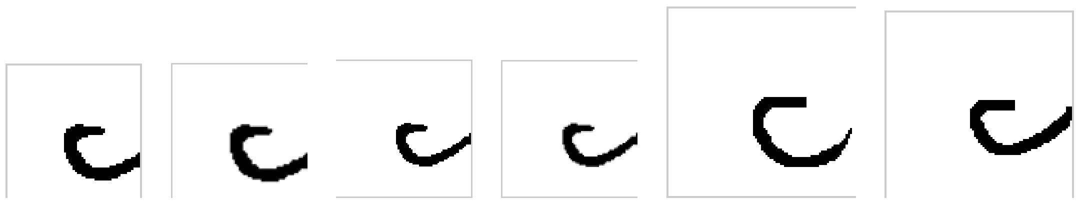
(a) Entropy values.



(b) Gradient of Entropy values.

Figure 5.8: Sizing the local patch

we should normalize all the extracted patches to be the same size and then calculate the HKS descriptors. If we directly resize all the patches to a predefined size, the width of strokes will vary differently based on the size of original patches. As shown in Fig. 5.9, Fig. 5.9(a) and Fig. 5.9(c) are the two patches centered at the same keypoint, but with different r, if we resize these two patches to the same size 91×91, in Fig. 5.9(b) and Fig. 5.9(d) respectively, the width of strokes are different, obviously the stroke in Fig. 5.9(b) is thicker.



(a) $r = 30$ (b) 91×91 (c) $r = 40$ (d) 91×91 (e) (f)

Figure 5.9: Resizing patches with different r to the same size will lead the width of stokes different. (e) Normalize the patch in 5.9(b). (f) Normalize the patch in 5.9(d).

In order to keep the width of strokes unchanged, we first apply the thinning method (89) on the strokes in the resized patch, so that the width of strokes is 1, and then remove the spurious pixels to smoothen the strokes. We then apply dilation twice, and at last only keep the foreground pixels in a circle with the same radius as the side length of the patch, in order to be invariant to the rotations. The normalized patches of Fig. 5.9(b) and Fig. 5.9(d) are shown in Fig. 5.9(e) and 5.9(f) respectively, and the widths of the strokes are the same.

Therefore, for every keypoint, we get the local patch with the size determined by the method in Section 5.3.1.2, and normalize all the patches to the same size, so that, descriptors are all in the same dimension. Based on the descriptors, in the next Section, we will introduce how to spot the locations where the query word appears on the document.

5.3.2 Keyword Spotting

5.3.2.1 Candidate Keypoints

Given a query image and a document, we should first search throughout the whole document, and find the possible matching areas. In order to relieve the effort of checking all the parts of the document, such as densely moving a sliding window, we only consider the areas containing similar keypoints, with respect to the ones in the query image. Therefore, if we assume that the query image has n_q keypoints, denoted as kp_i, $i \in [1, n_q]$, and the document has n_d keypoints, denoted as kp'_j, $j \in [1, n_d]$, we first calculate the distance between every pair of kp_i and kp'_j by the following equation:

$$d_i^j = d(kp_i, kp'_j) = \parallel \text{HKS}(kp_i), \text{HKS}(kp'_j) \parallel \tag{5.4}$$

where $\parallel \cdot \parallel$ is the L2-norm.

For every kp_i, we can get a list of distances d_i^j, $j \in [1, n_d]$, and then all the distances are normalized by scaling to $[0, 1]$. Fig. 5.10 shows an example of the sorted distances between the descriptor of one keypoint in query image and the descriptors of all the keypoints in one document. We can see that the distances increase very fast at the beginning, and then slow down for most of the keypoints. At last, the distances increase dramatically. Therefore, we can divide the change of the sorted distances into three parts: the first part contains most of the similar keypoints with respect to the one in the query image, the second part contains the keypoints with few small similar parts, and the third part contains most of the dissimilar keypoints. In order to get the candidate keypoints, we only consider the first part, so that the keypoints in the first parts and with the distances in the top 20% smallest are considered as the candidate keypoints. Using this threshold method, different keypoints in the query image may have different numbers of candidate keypoints on a document, because the patch centered at a keypoint always contains a part of a character, which can also be a part of other different characters. Moreover, the number of occurrences of different characters is different due to specific language or different content on the document. Some patches may appear much fewer times than the others. Our threshold method tries to capture all the similar patches, but brings in the dissimilar patches as few as possible.

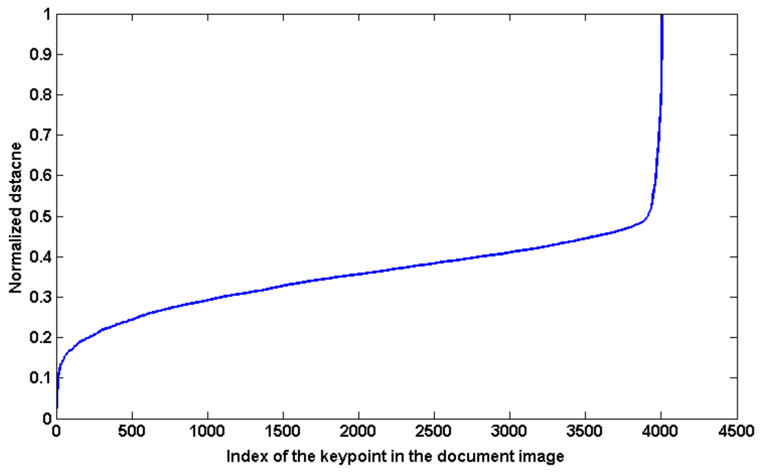


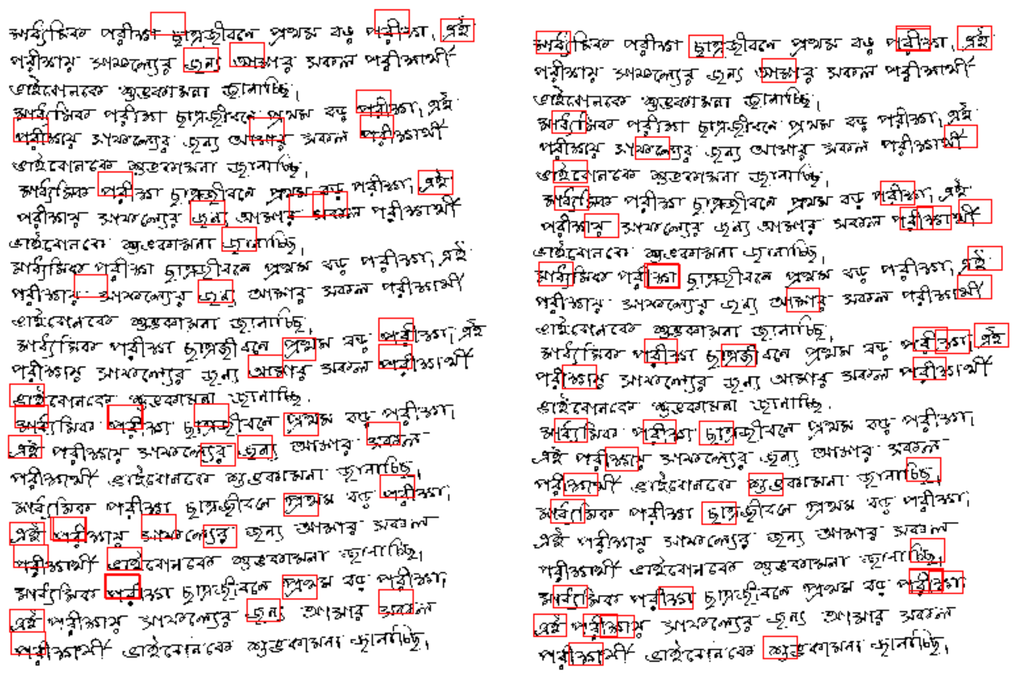
Figure 5.10: An example of the sorted distances of one keypoint in the query image with respect to all the keypoints in one document.

Candidate keypoints for one keypoint kp_i in the query image is denoted as $Candi(kp_i)$, and we also record the coordinates of the candidate keypoints, which will be used in the next section to locate the candidate matching zones. $M(kp'_j)$ stores the keypoints in the query image, to which kp'_j is similar, namely, kp'_j is the candidate keypoint of every keypoint in $M(kp'_j)$. In addition, a matrix $Mark$, with the same size of the document is used to record the positions where the candidate keypoints appear, by setting the values to 1, and others to 0.

5.3.2.2 Localization of Candidate Local Zones

Instead of densely moving a sliding window throughout the whole document, we try to reduce the searching effort, and focus on the areas with candidate keypoints.

Assume the size of the query image is $m \times n$, and the coordinates of the keypoints kp_i in the query image is denoted as (x_i, y_i). For each kp_i, the candidate local zones are defined as the windows with the bounding box of $((x'_j - x_i, y'_j - y_i), m, n)$, which denotes the coordinate of the left-bottom position, the height, and the width, and (x'_j, y'_j) are the coordinates of $kp'_j \in Candi(kp_i)$. In order to avoid missing any candidate local zones, we extract windows for all kp_i, because we cannot guarantee that all the similar keypoints can be included in the candidate keypoints, if we only consider the candidate keypoints of subset of kp_i, some positive matching zones may be missed, as shown in Fig. 5.11. Fig. 5.11(a) shows the candidate local zones for the first keypoint in Fig. 5.7(c), and some positive matching area are not included. However, in Fig.5.11(b), which shows the candidate local zones of the second keypoint, some missing areas are spotted. Therefore, using the candidate local zones for all the keypoints in the query image can spot as many positive matching areas as possible.



(a) Candidate local zones of the first keypoint in 5.7(c). (b) Candidate local zones of the second keypoint in 5.7(c).

Figure 5.11: Candidate local zones.

5.3.2.3 Matching Score

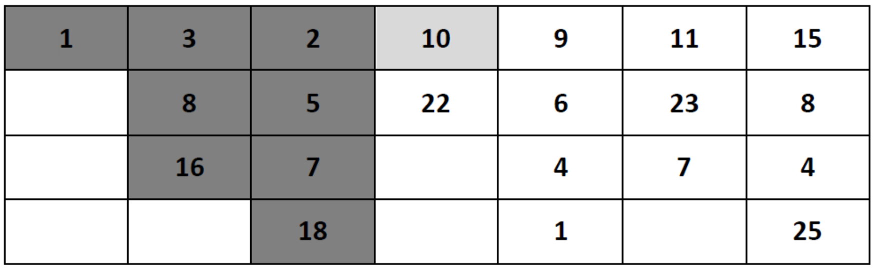
Given a candidate local zone, the matching score is calculated based on the candidate keypoints inside. Based on the bounding box of the candidate local zone, and the matrix $Mark$, we can get a set of keypoints on the documents, denoted as LZ, each of which is at least a candidate keypoint of one keypoint in the query image, and all of which are sorted by their vertical position values. Our aim is to get a matching path in the candidate local zone from left to right, so we use two variables to record the matching process: $Hist$ storing the matching path, and $Score$ storing the optimal matching score. Our proposed method for calculating the matching score is shown in Algorithm 5.

For each kp'_j in LZ, we first store $M(kp'_j)$ in one column of MS, and then initialize the first column $Hist$ to $MS(LZ[1])$, storing the corresponding matching distances to the first column of $Score$. Fig. 5.12 shows an example of MS, each column of which are the indexes of the keypoints in $M(kp'_j)$, such as in the second column, $LZ[2]$ is the candidate keypoint of the 3rd, 8th, and 16th keypoint in the query image. $Hist$ and $Score$ always have the same size of MS. For each cell of MS, we check all the cells in the several previous columns, for example, if the first cell in the forth column of MS in Fig. 5.12 is marked as light gray, we check all three previous columns in the dark gray. Because some keypoints may not be detected or included in the candidate keypoints, or some noise keypoints are located in the candidate local zone, in order to get optimal matching path and to tolerate missing or noisy keypoints, we look backward more than one column in MS, namely, we consider more than one previous keypoint, which is already marked as a positive matching.

For a cell (c, j) in MS under consideration, only in two conditions we will update the values in $Hist(c, j)$ and $Score(c, j)$, based on $Hist(c', j')$ and $Score(c', j')$ under checking: one is the length of the matching path is longer, the other one is the matching score is smaller. Therefore, when the Algorithm 5 terminates, we find the longest matching path in $Hist$, and choose the one with the minimum score in $Score$. So that we get a pair of $(score, path)$. In order to enhance the effect of the length of the matching path, the final matching score for a candidate local zone is $\frac{score}{size(path)^2}$.

Algorithm 5 Calculate matching score

```
1: $w = max(size(M(kp'_j))), kp'_j \in LZ$
2: $MS = zeros(w, size(LZ))$
3: for $j = 1 \rightarrow size(LZ)$ do
4:     for $c = 1 \rightarrow size(M(LZ[j]))$ do
5:         $MS(c, j) = M(LZ[j])[c]$
6:     end for
7: end for
8: % Initialization
9: for $c = 1 \rightarrow size(M(LZ[1]))$ do
10:     $Hist(c, 1) = [MS(c, 1)]$
11:     $Score(c, 1) = d(MS(c, j), LZ[1])$
12: end for
13:
14: for $j = 2 \rightarrow size(LZ)$ do
15:     for $c = 1 \rightarrow size(M(LZ[j]))$ do
16:         for each $MS(c', j')$, $j' \in [max(1, j - 5), j - 1]$ do
17:             if $MS(c', j') > MS(c, j)$ or $| Hist(c', j') | < | Hist(c, j) |$ then
18:                 $Continue$
19:             end if
20:             if $MS(c', j') == MS(c, j)$ and $| Hist(c', j') | > | Hist(c, j) |$ then
21:                 $Hist(c, j) = Hist(c', j')$
22:                 $Score(c, j) = Score(c', j')$
23:             else
24:                 if $| Hist(i', j') | +1 > | Hist(i, j) |$ or $Score(c', j') + d(MS(c, j), LZ[j]) < Score(c, j)$ then
25:                     $H(i, j) = [H(i', j'), MS(i, j)]$
26:                     $Score(c, j) = Score(c', j') + d(MS(c, j), LZ[j])$
27:                 end if
28:             end if
29:         end for
30:     end for
31: end for
```



1	3	2	10	9	11	15
	8	5	22	6	23	8
	16	7		4	7	4
		18		1		25

Figure 5.12: An example of MS. The indexes in the jth column is the indexes of the keypoints in the query image, of which $LZ[j]$ is the candiditate keypoints.

5.3.2.4 Removing Overlapping Returned Results

Because we consider the candidate local zones for all the keypoints in the query image, many local zones are overlapped with each other. In order to remove the overlapping zones, all the local zones are first sorted by the length of the optimal matching path in the descending order and then sorted by their matching score in the ascending order. From the top one, we remove all the other zones with more than 50% overlapping areas, until we only have zones with no overlapping areas in the spotting list.

Furthermore, we remove the ones with lengths of matching paths smaller that $0.5 * n_q$. For the remaining zones, we normalize their matching scores by scaling to [0,1], and discard the ones with larger matching scores. For example, as shown in Fig. 5.13, we can see that after the 8th returned zone, the matching score increases dramatically, therefore, we only keep the first 8 zones.

5.3.3 Experiments and Results

5.3.3.1 Experimental Setup

We test our proposed method on Bangla handwritten documents, with two examples shown in Fig. 5.14. We select one document of each writer and segment it into word images, which will be used as query images to be spotted on all the other documents for the same writer in the experiments. The word spotting results are evaluated by the average precision and recall.

For comparison, first we densely move a sliding window throughout the whole document, which has the same size as the query image. The sampling step in horizontal

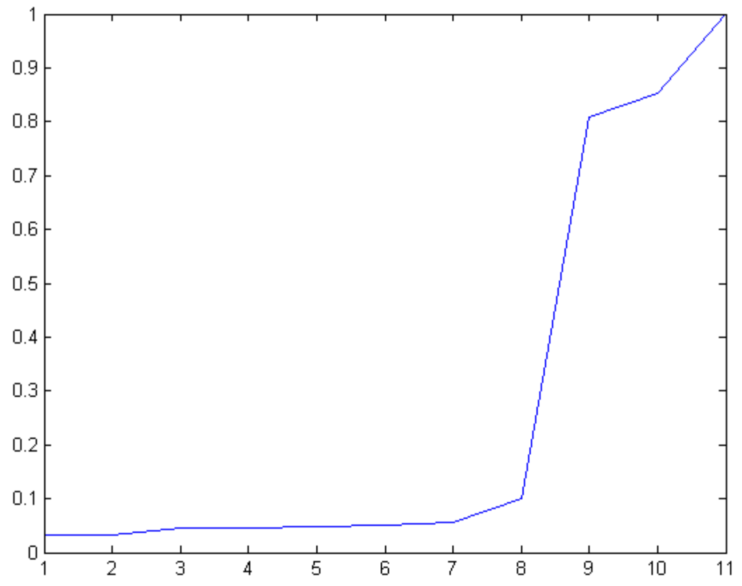
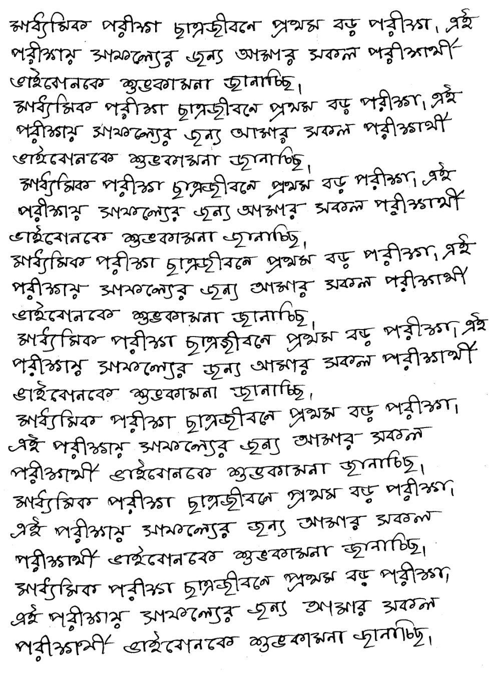


Figure 5.13: Plot of the scores of returning zones. The horizontal axis is the index of the zones in the spotting list, and the vertical axis is the noramlzied matching score.

and vertical directions are set to be half of the height of the query image. At each position, we extract Histogram of Gradients (HoG) features, as described in (90). We also extract same features from the query image, and the similarity between one candidate local zone and the query image is measured by the L2 norm distance between HoG features. Secondly, we use SIFT features for the keypoints and apply our proposed matching method on the same set of query word images and documents.

5.3.3.2 Results

The experimental results are shown in Table 5.1. Our method has both highest precision and recall than using sliding windows combined with HoG features and applying SIFT features with our spotting method.Because usually HoG features require the word images are normalized, such as skew or slant correction. But for our segmentation-free keyword spotting tasks, we try to avoid the pre-processing steps. Even though the sliding window is densely moving throughout the whole documents and any location the query image appears can be checked, HoG cannot be robust enough to find true


(a)

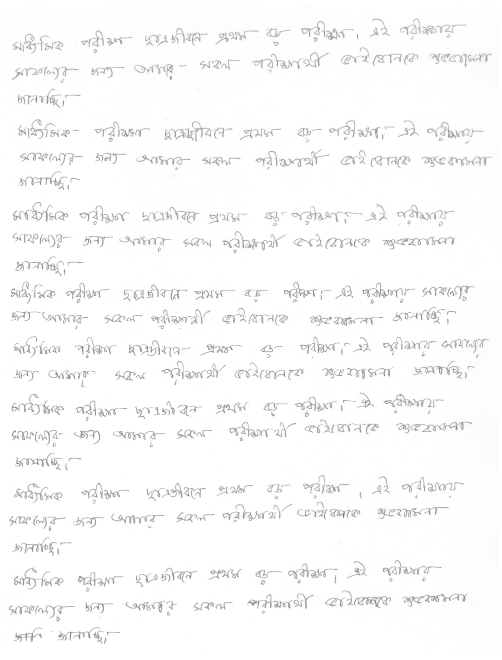

(b)

Figure 5.14: Two examples of Bangla handwritten documents.

matched local zones due to large writing styles. HKS can tolerate variations better than HoG and SIFT, even the locations of the corresponding keypoints in the same occurring words are slightly different. Moreover, our method for deciding the size of the local patches can avoid bringing in noise into the patches, and includes as much information as possible. For SIFT, some keypoints have very large scales, so that the SIFT features are extracted from a large local region and always contain the parts of the surrounding characters. In this situation, these keypoints will not be matched correctly.

Table 5.1: Experimental Results

Methods	Average Precision	Average Recall
Sliding window with HoG features	37.8	55.4
SIFT with our proposed matching method	70.9	84.8
Our proposed method	**77.2**	**94.8**

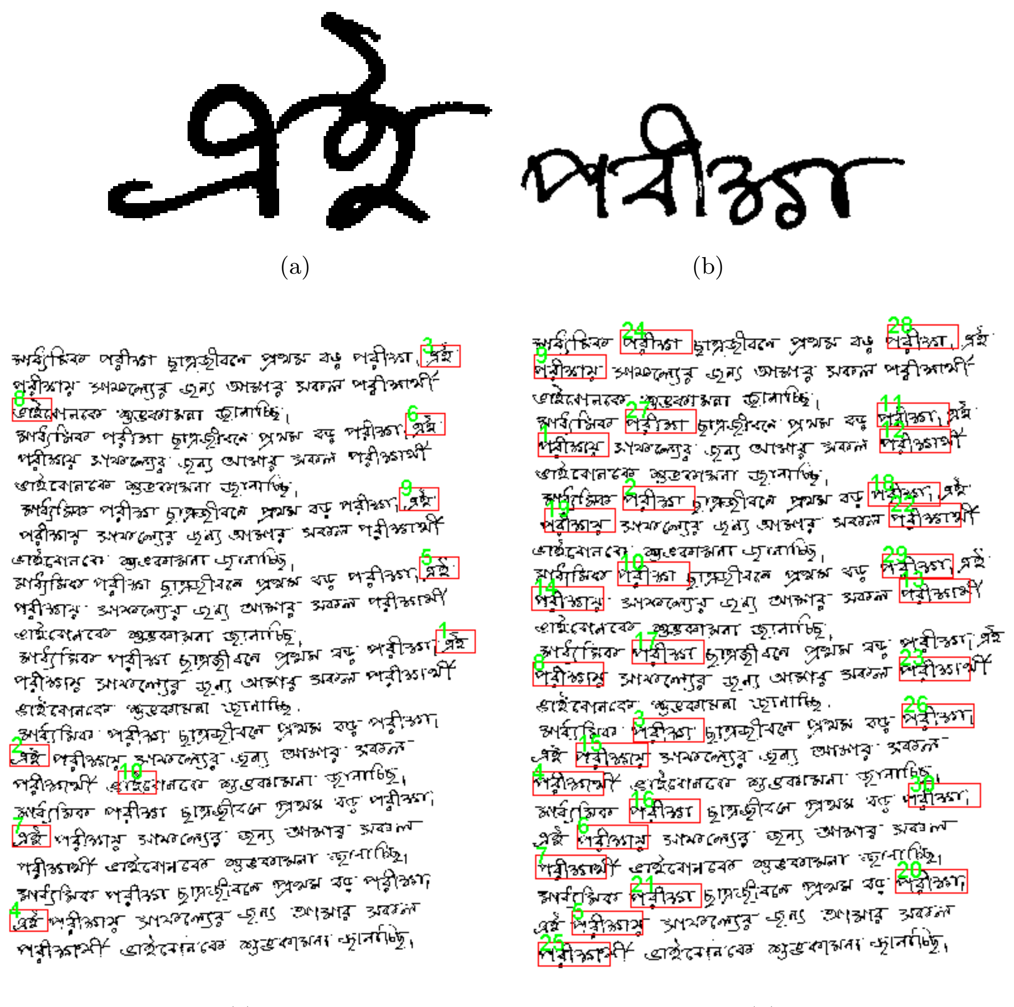
In Fig. 5.15(c) and 5.15(d), there are two spotting results for the query images in Fig. 5.15(a) and 5.15(b). We can see that all the instances of the query images are spotted based on our method, but in Fig. 5.15(c), 10 local zones are returned, two of which contain only one character in the query image, and in Fig. 5.15(d), much more local zones are returned, because most of the parts in the query image appear in different words, even though only 8 instances are exactly the same to the query image, but all the other returned zones have no more than one character different. For other longer query word images, with few common parts in the other words, the spotted zones in the spotting list are exactly the instances of the query images, as shown in Fig. 5.16.

For the documents in GW dataset, all the documents were written by President George Washington, and the size of the same words is very consistent. However, in Bangla documents, the variations of the same words written by the same writer are much larger, so that we need more important keypoints and we cannot extract features from the fixed-size patches. Moreover, the width of stokes is another important aspect which may affect the extracted DaLI descriptors, therefore, we also normalize the the width of all strokes.

Due to the much larger variations in Bangla documents, some true positive matching candidate keypoints may not be ranked in the top, so that we cannot only check the candidate keypoints of one keypoint in the query image to locate candidate local zones, as what we do for GW documents. We need to take into account the candidate keypoints for all the keypoints in the query image, in order to find all possible true positive matching zones.

5.4 Conclusion

We proposed a segmentation-free keyword spotting method for handwritten documents. Document images are presented by HKS descriptors of all detected keypoints. In the spotting process, in order to narrow down the searching scope, we only consider the local zones containing similar keypoints with respect to the ones in the query image, and the local zones with enough matching keypoints are returned, sorted by the matching scores in the ascending order. According to our experimental results, HKS can tolerate

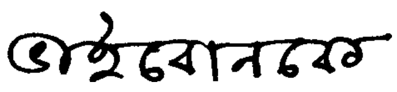

(a) (b)

(c) (d)

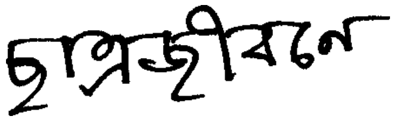
Figure 5.15: (c) the spotting results of (a). (d) the spotting results of (b). The number marked around the spotted rectangle box is the position in the spotting list, namely, the smaller the number is, the more similar to the query image.

the variations much better, and our proposed method has higher average precision and recall.

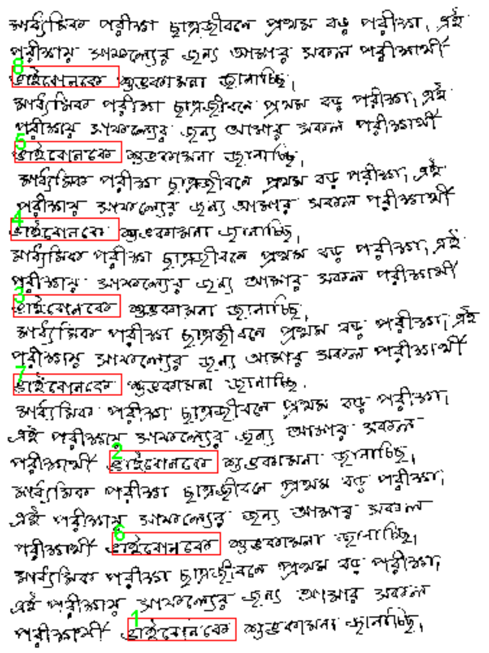
In future, we will research on reducing the dimension of HKS descriptors, and speed up the searching process. Then, we would like to use heuristics to discard irrelevant keypoints quickly, to avoid calculating all distances between two keypoints. At last, we would like to work on rotated or curved document pages based on HKS.



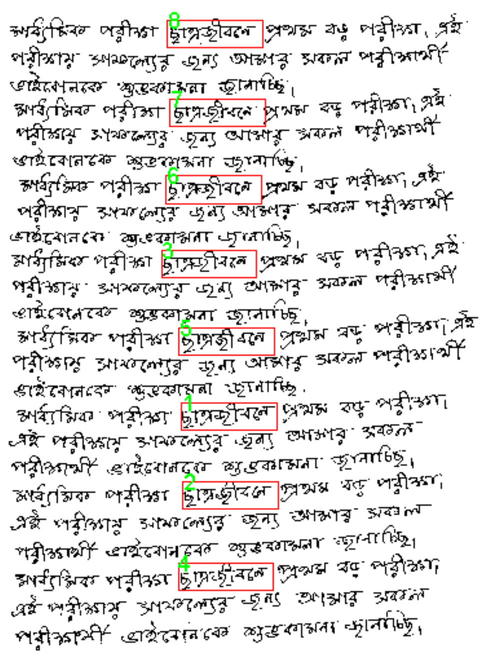
(a)



(b)



(c)



(d)

Figure 5.16: (c) the spotting results of (a). (d) the spotting results of (b).

Chapter 6

Handwritten Document Image Retrieval based on Keyword Spotting

In this chapter, we will present a method to retrieve relevant handwritten documents based on our proposed keyword spotting in the previous chapter. Document retrieval can be achieved by the traditional text retrieval methods based on ASCII content. The frequencies of every unique word can indicate how the relevant one document is according to the query document. However, in some cases, the ASCII content is not available, moreover, due to the degradation and unconstrained cursive handwriting styles, the recognized ASCII content may contain errors, which can affect the retrieval results. Therefore, if we first segment the query document into connected components, which do not need to be whole words, then we can just spot how many times each component appears on the candidate documents. Based on the spotting results, we can also get the frequencies of every component on the candidate documents, which can be used to measure the relevance.

6.1 Introduction and Related Works

Nowadays, a large amount of imaged documents are stored in digital databases and libraries, and many organizations have developed different systems to index and retrieve these document image repositories. Textual content based on Optical Character

Recognition (OCR) can be used to achieve the indexing and analysis. However, large variations in cursive writing styles and degradation of historical documents always have poor OCR results, which lead to bad retrieval performance. Content-based document Image Retrieval (CBIR), on the other hand, can retrieve the relevant documents without the textual content, only with the features extracted directly from the document images. Therefore, CBIR can be used to retrieve multilingual handwritten documents, and complex document collections. All the non-textual contents lead to the limitations of OCR-based retrieval methods.

One of the most useful applications of CBIR is retrieving the documents written by the same writer, or having similar patterns, such as the documents from the same historical period. (91) indexed and retrieved handwritten documents based on two kinds of features. One is the Micro features, including gradient, structural, and concavity features. The other is the Macro features, which are extracted directly from the entire document, or on a line-by-line basis. Combining these two kinds of features, the retrieval based on writing styles can be achieved at document level, partial image level, and word level.

Besides, instead of using the individual sets of different features, (92) extracted a global feature from the entire documents, named as Curvelets. The curvelet transform is applied to all the documents, then the curvature and orientation are generated for each pixel. Every document is represented by a signature, defined as a matrix of all occurring (curvature, orientation) pairs. Therefore, the relevance between signatures is calculated based on the normalized correlation similarity.

But Curvelets were originally developed for the signals in the continuous domain, so that in the discrete domain, such as our document images with pixelization, there are challenges for the implementation of Curvelets due to the discrete sampling. Therefore, (4) developed a new directional multiresolution transform, named as Contourlets, which are directly constructed from the discrete domain. Because the directionality and anisotropy of contourlets, which are powerful for capturing writing styles, (93) proposed an content-based document image retrieval based on writers, using the contourlet transform. Four levels of Laplacian Pyramid (LP) are used, and the descriptor for each document is constructed by the energy and standard deviation on each subband. The matching between two descriptors of document images are achieved by the Canberra distance. As the results shown, contourlets have better performance than curvelets, according to both the precision and recall. In addition, the representation of the document images based on contourlets are more compact than that based on curvelets (92).

However, as presented in (94), the contourlet transform is not shift-invariant due to the downsamplers and upsamplers, and how to design good filters for the contourlet transform is a challenging task. In (94), the authors proposed a overcomplete transform, named as Nonsubsampled Contourlet Transform (NSCT), which is fully shift-invariant, multiscale and multidirection expansion. Besides, the authors solved the filter design problems. Therefore, we use NSCT in our proposed method, instead of the contourlet transfrom (CT).

Moreover, the relevance between two documents is only measured based on writing styles, but the actual content, namely, the appearing words, should be also considered for measuring the relevance. In this chapter, we propose a retrieval method to retrieve the handwritten documents not only based on writers, but also on the content relevance. Therefore, the returned documents on the top of the ranking list should contain more similar words with respect to the query document, without use of textual content, but with the spotting results of the appearing words in the query document. In addition, our spotting method is independent of segmentation, because we cannot get correct segmentation results for all kinds of documents. Because we make no assumption about the language, our method is also language-independent. In the experiments, we will test our methods on four databases of handwritten documents written in different languages.

6.2 Features

In order to measure how relevant a document is according to the query document, we need to extract important features from the documents, which can be invariant for the documents written by the same writer, but have large variations among different writers. In this section, we will introduce two kinds of features which can be used for writer identification, and can also be used to spot segmented components.

6.2.1 Curvelet

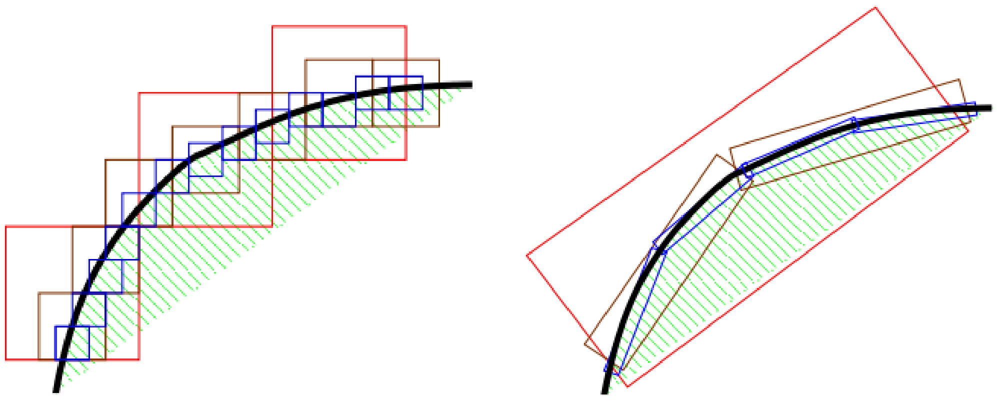
In order to overcome the failure cases for smooth edges and contours based on wavelet, curvelet (95) can exploit the smoothness of the contours more effectively, using different elongated shapes, as shown in Figure 6.1. Assume in R^2, $W(r)$ and $V(t)$ are the radial window and angular window respectively, where r and t are both polar coordinates in the frequency domain. These two windows satisfy the following conditions (92):

$$\sum_{j=-\infty}^{\infty} W^2(2^j r) = 1, r \in (3/4, 3/2) \tag{6.1}$$

$$\sum_{l=-\infty}^{\infty} V^2(2^j t - l) = 1, t \in (-1/2, 1/2) \tag{6.2}$$

Then, U_j, which is a frequency window with $j \geq j_0$, is defined as:

$$U_j(r, \theta) = 2^{-3j/4} W(2^{-j} r) V(\frac{2^{\lfloor j/2 \rfloor}}{2\pi}) \tag{6.3}$$



(a) Wavelet (b) Curvelet

Figure 6.1: For a 2D smooth contour, wavelet needs much more redundant square-shapes to describe the contour, but curvelet can represent the contour more efficiently by enlarged shapes,with different directions (4).

All the curvelets at scale 2^{-j} are obtained from the "mother" curvelets, using different rotations and translations. Based on the computed curvelet transform, each pixel on the document is represented by the most significant curvature and orientation in its corresponding curvelet coefficients.

6.2.2 Contourlet

The contourlet transform was proposed in (4). It is defined directly on discrete domain, and can capture the directionality and anisotropy of the images, unlike wavelet or curvelet, which were initially developed in the continuous domain. Laplacian pyramid (LP) is first applied to obtain a multi-scale decompositions of the original image, and then a 2D directional filter band (DFB) is implemented for each subband to capture the smooth contours and directional edges, which have higher frequencies. Combining LP and DFB, the original image is decomposed into directional subbands, with multi-scales. Assume I is the original document, and H and G are the low pass analysis and synthesis filters respectively. With a sampling matrix M, Figure 6.2 shows one level of decomposition by LP, in which a is the coarse approximation of I, and b is the difference between I and the prediction. In each level $l(l > 1)$ of LP, the low pass filtered version of I_{l-1} is denoted as L_l and the prediction is denoted as P_l, so that the different between L_{l-1} and the prediction is $E_l = P_l - L_l$, where $l = 0$, $L_0 = I$. Then, DFBs with multiple directions are implemented for each E_l to get a set of directional subbands. In Figure 6.3, for 3-levels DFB, the frequency domain is divided into $2^3 = 8$ frequency bands, which can be used to represent multidirectional features for each l-level decomposed image L_l.

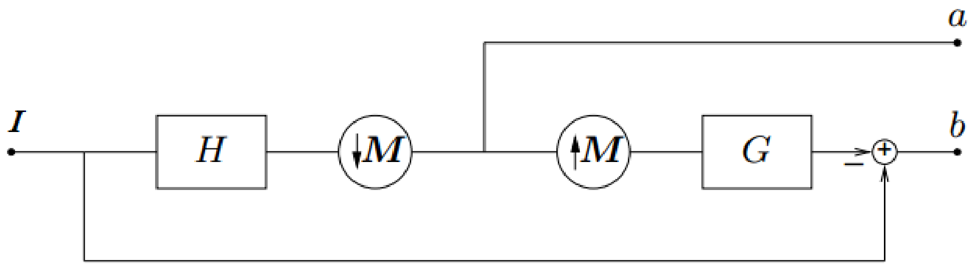


Figure 6.2: One level of decomposition by Laplacian pyramid (4).

The energy and standard deviation in each subband is used to construct the feature vector of one document. If E_k and σ_k are the energy and standard deviation in the k^{th}

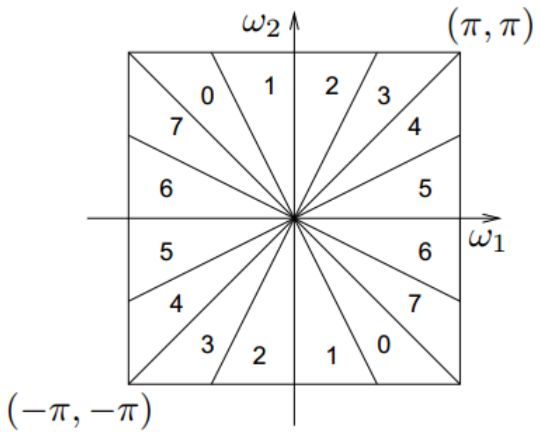


Figure 6.3: Directional filter bank with $l = 3$ and $2^3 = 8$ frequency bands.

subband, $f_E = [E_1, E_2, \cdots, E_n]$, and $f_\sigma = [\sigma_1, \sigma_2, \cdots, \sigma_n]$. After normalizing f_E and f_σ to mean 0 and standard deviation 1, the final feature vector is $[\overline{f_E}, \overline{f_\sigma}]$ (93).

6.3 Retrieval Model

In addition to retrieving the documents based on writing styles, more relevant documents based on content on different documents written by the same writer should be ranked higher in the ranking list. Only the visual features are used with no use of recognition, therefore, in order to optimal approximate the content of the documents, we apply our proposed word spotting method in the previous chapter to spot the appearing connected components in the query document. The document images are represented by a bag of visual features, extracted from a local patch centered at each detected significant keypoint, instead of extracting the global features on the entire document. With no use of OCR, the frequencies of the appearing words in the query document on the documents in the database can be used to measure the content relevance.

6.3.1 Writer identification

The important keypoints on all the documents are detected by the SIFT keypoint detector. Based on the average height AH of all the connected components (CC) on each document, NSCT is applied to a local patch with the size of $(2 * AH + 1) \times (2 *$

$AH + 1)$ centered at each keypoint. The whole document image is presented by the energy and the standard deviation in each subband as proposed in (93). If there are n keypoints on one document, and Sub_k^i denotes the kth subband of the ith keypoint, then the energy of the kth subband is:

$$f_{E_k} = \frac{\sum_i \sum_{s,t} |Sub_k^i(s,t)|}{n * (2 * AH + 1)^2} \tag{6.4}$$

and the stardard deviation of each kth subband is:

$$f_{Sigma_k} = \sqrt{\frac{\sum_i \sum_{s,t} (Sub_k^i(s,t) - mean_k)^2}{n * (2 * AH + 1)^2}} \tag{6.5}$$

$$mean_k = \frac{\sum_i \sum_{s,t} Sub_k^i(s,t)}{n * (2 * AH + 1)^2} \tag{6.6}$$

Therefore, the document is represented by following for writer identification:

$$[f_{E_1}, f_{E_2}, \cdots, f_{E_s}, f_{Sigma_1}, f_{Sigma_2}, \cdots, f_{Sigma_s},] \tag{6.7}$$

where s is the total number of subbands and the representation is then normalize by the method in (93).

6.3.2 Keyword spotting

The query document is first convolved by a Gaussian kernel to connect the horizontal consecutive characters together, and segmented into individual CCs after binarization. Very large and very small CCs are discarded, because they are always the background noise, which can be determined by checking their width or height. Each remaining CC is treated as a keyword to be spotted on all the other documents. We do not assume each CC contains exactly one word, which may contain several words or only a portion of one word. Our method can spot multiple words in a CC together, and can also spot each individual word separately on the document. Moreover, we make some modifications on our proposed method in the previous chapter to be more flexible with no assumption about the size of the candidate local zones. The aim of the spotting is to spot any possible appearing similar sequence of characters according to the query connected components, and the length of the matching sequence and the corresponding similarity are used as the measure for content relevance.

We assume that $\{kp_i = (x_i, y_i)\}$ is a set of keypoints detected inside the bounding box of a keyword K on the query document, where $i \in [1, n]$. All $\{kp_i\}$ are sorted by the vertical positions, and their corresponding features are denoted as $F(kp_i)$. On the other hand, the keypoints on the documents in the database are denoted as $\{kp'_j = (x'_j, y'_j)\}, j \in [1, m]$, and the corresponding features are $F'(kp'_j)$. After generating the features of all the keypoints, we calculate the distances d_i^j between each pair of kp_i and kp'_j. The distances are calculated based on all the values in each subband.

Based on the distances, we set a threshold t_i for each kp_i, so that kp'_j with distances smaller than t_i are considered as the candidate keypoints of kp_i, denoted as $C(kp_i)$, which we will use to spot the positions where the query keyword probably appears. In our experiments, we set all t_i to 50. In order to store the locations of each set of candidate keypoints, we use a set of matrix $\{M_i\}, i \in [1, n]$, each of which has the same size of the document, and initialized to 0 for all the elements, then if kp'_j is the candidate keypoint of kp_i, $M_i(x'_j, y'_j)$ is set to 1. Besides, H_i is used to record the optimal matching path for each element with the value of 1 in M_i. Each $H_i(x'_j, y'_j)$ is a list and initialized to have one element, kp'_j, where $kp'_j \in C(kp_i)$.

For each candidate keypoint kp'_j in $C(kp_2)$, we check its N nearest neighbouring keypoints on the document. If at least one of its neighbours has 1 value in M_1, we will add the elements in H_1 of its neighbours to $H_2(x'_j, y'_j)$. There are two constrains we are considering when generating the optimal matching path:

1. Geometrical structure: In order to capture the geometrical structure of keypoints, we constrain the consecutive matching keypoints on the documents to have similar relative positions with respect to the corresponding keypoints in K. For example, if we assume kp'_1, kp'_2 and kp'_3 are going to match the keypoints kp_1, kp_2 and kp_3 respectively,then we calculate the angle between the two vectors $\overrightarrow{kp'_2kp'_1}$ and $\overrightarrow{kp'_2kp'_3}$, and the angle between $\overrightarrow{kp_2kp_1}$ and $\overrightarrow{kp_2kp_3}$. If the difference between these two angles is smaller than ϵ, we will treat this matching as a positive matching, otherwise, this matching is discarded.

2. Missing keypoints: The correct candidate keypoints may be missed, either in the detection step or the threshold t_i is not chosen properly. Therefore, in order to tolerate missing candidate keypoints, for each kp'_j in $C(kp_i)$, we check all its N nearest neighbours in M_{i-l}, where $l \in [1, 2]$. Each candidate keypoint may have

various matching paths, but we always keep the matching path with the longest length and the minimum matching distance as the optimal one.

After checking M_2, the above matching procedure is implemented for M_3 to M_n. After we finish checking M_n, each element of H_i records the optimal matching path of the corresponding candidate keypoint, and the matching score, which can be easily calculated by summing over all the distances between each pair of the matching keypoints and dividing by the square of the length of the matching path.

6.3.3 Document representation

For each keyword on the query document image, we obtain $H_i, i \in [1, n]$ by the methods in the previous section, and we only keep the elements in H_i, which has the length longer than $n/2$, namely, at least half of the number of keypoints in the keyword are matched to some keypoints on the document. We denote the set of matching paths of the i^{th} keyword K_i as $M\_path(K_i)$, and the corresponding scores as $S\_path(K_i)$. We will use n_i to denote the number of keypoints in K_i.

Each element in $M\_path(K_i)$ is an evidence, which indicates the whole or partial parts of K_i appearing somewhere on the document. Therefore, the length of each matching path diving by n is treated as the appearing count of K_i, and the total number of appearing times of K_i on the document is the sum of all the counts. The count for each appearance may be less than 1, because the keypoints in K_i may not be totally matched, so that the longer the matching path is, the more convincing evidence the matching path provides. The count of the appearances of K_i on the j^{th} document D_j is defined as:

$$tf_{K_i}^{D_j} = sum_{p \in M\_path(K_i)} \frac{length(p)}{n_i} \tag{6.8}$$

and the normalized count is defined as:

$$ntf_{K_i}^{D_j} = \frac{tf_{K_i}^{D_j}}{\sum_k tf_{K_k}^{D_j}} \tag{6.9}$$

Assume M is the total number of the documents in the database, and no_i is the number of documents having at least one matching path for K_i, so that

$$tf \times idf_{K_i}^{D_j} = \frac{ntf_{K_i}^{D_j} \times log(\frac{M}{no_i})}{\sqrt{\sum_k ntf_{K_k}^{D_j} \times \frac{M}{no_k}}} \tag{6.10}$$

Finally, the feature vector of D_j with respect to the query document image is constructed as following:

$$f(D_j) = [tf \times idf_{K_1}^{D_j}, tf \times idf_{K_2}^{D_j}, \cdots, tf \times idf_{K_Q}^{D_j}]' \tag{6.11}$$

where Q is the total number of keyword on the query document.

6.4 Experiments

In order to evaluate our proposed method, we first do the experiment on the IAM offline handwriting database (58), which contains various large writing styles. Then, we also test our method on three sets of historical manuscripts, written in different languages.

6.4.1 IAM database

In IAM offline handwriting database, we choose 13 writers who wrote 5 forms in the database, totally 65 forms. We compare our method with the one proposed in (93), where CT is applied to all the forms and each form is then represented by a feature vector. We take every form as a query and rank other forms based on the Canberra distance:

$$Canb(X, Y) = \sum_{i=1}^{d} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{6.12}$$

where $X = [x_1, x_2, \cdots, x_d]'$, and $Y = [y_1, y_2, \cdots, y_d]'$, if X and Y are the feature vectors of two documents.

For comparison, we also extract the NSCT features from the entire document images, and the relevance between two documents is also measured by Eq. 6.12.

In our method, we extract the NSCT features from a local patch of each detected keypoint on all the forms. Then, for the query document, the keywords are generated, and spotted in all the forms, including the query document, so that we can get the document feature vector as shown in Eq. 5, counting the weighted number of appearances of each keyword. The distance between two document feature vectors are measured by L2-norm.

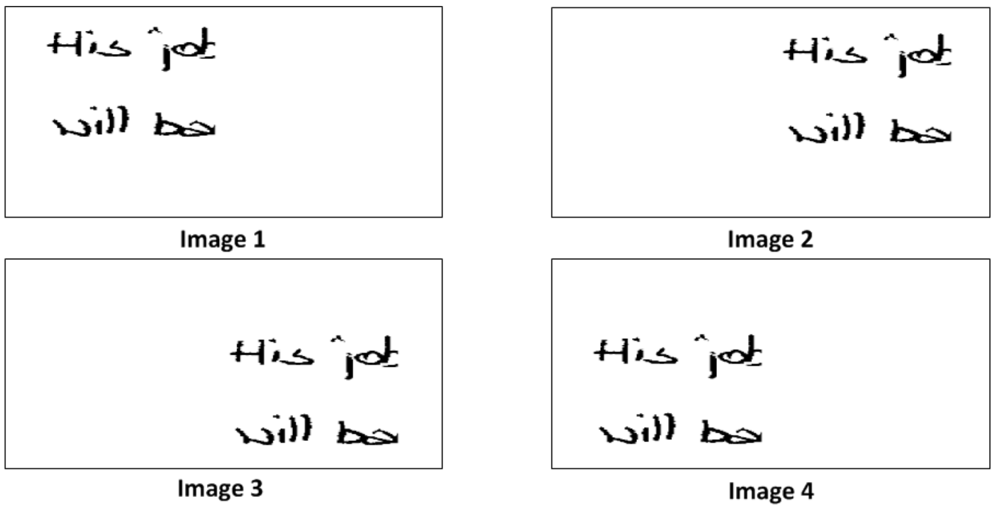
Table 6.1 shows the writer identification results based on three methods, where CT denotes the contourlet transform, $NSCT$ denotes the nonsubsampled contourlet transform, and P, R are the average precision and recall respectively. In table 6.1, we record the average precision and recall for Top1 to Top5 individually, as denoted in the left most column. We can see that NSCT has much better results than CT. As shown in Fig. 6.4(a), the four documents contain exactly the same content, except that the positions of the content are different. The CT of these four documents are different, as shown in Fig. 6.4(b), which is not what we want. On the other hand, the NSCT are exactly the same, which is invariant to the position changes, as shown in 6.4(c). For our method, because we get the feature vector for one document, based on the NSCT features from each keypoint, we only consider the features around strokes, discarding those in the background. The NSCT features around the strokes are enough to capture the writing styles, so that our method has exactly the same results as applying NSCT to the entire document.

Table 6.1: Writer Identification Results

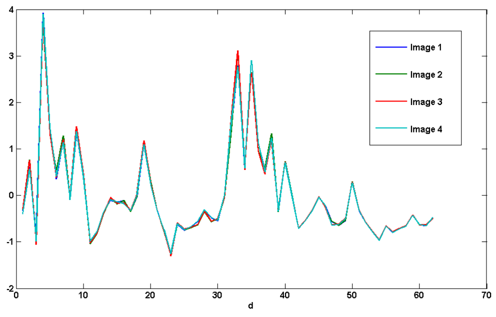
	CT		NSCT		Our method	
Top	P(%)	R(%)	P(%)	R(%)	P(%)	R(%)
1	100.00	20.00	100.00	20.00	100.00	20.00
2	63.85	25.54	100.00	40.00	100.00	40.00
3	48.72	29.23	99.49	59.69	99.49	59.69
4	41.54	33.23	99.23	79.38	99.23	79.38
5	36.00	36.00	98.46	98.46	98.46	98.46

In order to compare how the different methods rank all the documents written by the same writer according to the content relevance, we apply Ranked-biased Overlap (RBO) (96) to compare different rankings with respect to the ground-truth. In the returned ranking list, there are only 4 ranked documents. The ground-truth is obtained by applying $tf-idf$ scheme with lemmatization on the transcriptions of the documents, and the returned documents are ranked according to the L2-norm distances between their feature vectors and the one of the query document, sorted in the ascending order. On the documents written by the same write, each one is treated as a query, and average results are recorded.

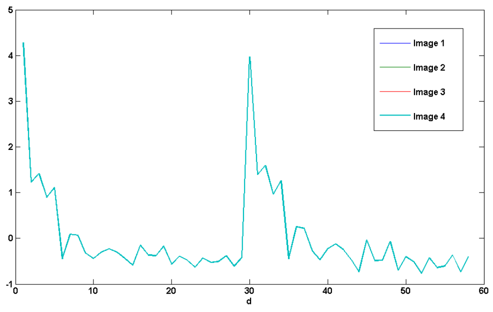
As shown in Table 6.2, NSCT performs a litter better than CT according to the content relevance of different documents, but our method have an obvious better results. Because NSCT and CT can only capture the global features, but cannot capture the



(a) The same content appears at the different positions in the four document images, with the same size.



(b) CT



(c) NSCT

Figure 6.4: (b) and (c) are CT and NSCT of the documents in (a).

local features, which can indicate the appearing different words. On the other hand, our method is based on the keyword spotting, and can obtain the estimated appearing times for each keyword, just like the $tf - idf$ scheme on the ASCII content, even though we spot all the keywords without recognition.

Table 6.2: Content relevance Retrieval Results

Methods	CT	NSCT	Our method
RBO	0.970	0.974	**0.991**

6.4.2 Historical manuscripts

We also test our method on the historical manuscripts, including George Washington Database, Parzival Database, and Saint Gall Database, written in English, German, and Latin respectively. We choose the first 20 pages in each database for our experiments, and every document is used as a query. These three databases have very different appearances, so that CT, NSCT, and our method has the same perfect results for retrieving other documents from the same database to which the query document belongs. So that, if we return top $n, n \in [1, 19]$ documents, the average precision and recall for the three methods are 100%, and $n/19$.

Besides, we also test different methods on how the retrieval based on content relevance can be achieved in each database individually. The results are also evaluated by comparing the ranking lists based on RBO, and average results are recorded if every document in one database is used as a query. Table 6.3 shows the retrieval results for the three databases based on the content relevance, and there are 19 ranked documents in the returned list. We can see that NSCT has a little better performances than CT for George Washington and Saint Gall database, but worse than CT for Parzival database. On the other hand, our method has better results for all the three databases.

Table 6.3: Content relevance retrieval results

Database	CT	NSCT	Our method
George Washington	0.793	0.814	**0.877**
Parzival	0.758	0.727	**0.812**
Saint Gall	0.757	0.759	**0.828**

6.5 Conclusion

NSCT can be applied to the local patches centered at each detected keypoint, instead of applying NSCT on the entire documents, so that we can achieve both writer identification and content relevance retrieval. NSCT has much better results than CT for writer identification, but if applying NSCT on the entire document, the retrieval results for content relevance are similar to CT. On the other hand, our keyword spotting based retrieval method has much better results, because we estimate the appearing times of each keyword on the query document, instead of only considering the global features.

In future, we would like to work on the documents containing complex content and layout, and also on proposing an efficient indexing method to speed up the retrieval process.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Handwritten documents are always stored as whole images, and in order to achieve retrieval tasks, pre-processing steps are need at first. A text line segmentation method for handwritten documents based on seam carving was presented in this thesis. However, unlike the previously proposed method which first used seam carving to extract text lines, we constrain the energy flowing directions, so that the energy can be mainly passed to neighbouring points in the same text lines, and jumping across different text lines can also be avoided. Moreover, by only calculating the energy map once, we can extract all the text lines, instead of recomputing after each text line is extracted.

After obtaining segmented text lines, word images can be extracted based on the distances of the inter- and intra-spaces between connected components. Using the word images, we proposed a novel word recognition method combining the outputs of two networks, which are well trained on the subset of the training data. The splitting of the training data into two subsets satisfies the condition that the different words in the two sets are exclusive and the two word sets have as few common trigrams as possible. Our method for decoding is a modified version of the Token Passing Algorithm and we only focus on spotting trigrams instead of the whole character sequence for an input word. In the experiments, we select the training data and testing data from a collection of word images randomly and also test on the Writer Independence Recognition Task dataset. Our method has better results both on the character error rate and word error rate. What is more, our modified CTC token passing algorithm can also be used to get better recognition results by combining two trained networks, which are trained on different sets of training data, than using each network individually.

Besides, in order to avoid the limitations of the methods based on supervised learning, we extracted HKS descriptor for every keypoint in the query and candidate images and proposed a new similarity measurement method based on a triangular mesh structure, in order to keep global structure consistency. As shown in our experiments, our new method can capture local and global features more robustly and reliably and outperforms other commonly used methods. The proposed method can be directly applied to any given word images, without having a specific recognizer in advance.

However, in some cases, segmentation of text lines and word images are difficult due to degradation, noise or complex structure of content, we also proposed a segmentation-free keyword spotting method for handwritten historical manuscripts and Bangla documents. Document images are represented by HKS descriptors of all detected keypoints. After comparing the keypoints in the query image with the ones in document pages, we search throughout every given document to locate local zones. The local zones which contain enough matching keypoints, most likely contain the query word. Shown as the experiment results, HKS shows its power of tolerating non-rigid invariant and illumination changes, and our searching method can obvious reduce the searching scope for spotting.

Based on our keyword spotting methods, we can also achieve writer identification and content relevance retrieval based on the same set of features, in which the query can be a whole document image. The feature we apply on the documents is NSCT, which is extracted from the local patches centered at each detected keypoint. Previously, NSCT can be used to retrieve the documents written by the same writer or having similar patterns, when NSCT is extracted from the whole documents. But if we extract NSCT from local patches, we not only can achieve writer identification, but also can retrieve relevant documents according to the query document based on content relevance by our proposed keyword spotting method.

7.2 Future Work

In future work, we would like to improve our energy accumulation process to reduce the computation time for text line segmentation. Moreover, we will improve the performance of splitting large components which touch multiple text lines, and we will also work on gray level documents, which have more challenges.

For word recognition, we will try to apply our method directly on text line images, and also try to reduce the time cost for decoding. What is more, other databases, especially containing different languages, will be tested on.

Besides, more efforts should be put on how to find stable keypoints, so that HKS and triangular mesh structure can be made full use of for word image matching. Moreover, more sophisticated method should be proposed to find optimal alignment of two sets of DaLI descriptors for rotated or scaled images, in which, missing keypoints may occur. So that how to tolerate missing keypoints is also an important issue we should work on.

In practice, efficiency is a much more important aspect for retrieval, therefore we will research on presenting the document and query images more compactly, so that the computation time of searching throughout the documents can be reduced. In addition, we would like to use heuristics to discard irrelevant keypoints quickly, to avoid calculating all the distances between two keypoints. Moreover, we also would like to work on the documents containing complex content and layout, and propose an efficient indexing method to speed up the retrieval process.

Publications arising from this work

1. **Xi Zhang**, Chew Lim Tan, ”Handwritten Word Image Matching based on Heat Kernel Signature”, The 15th International Conference on Computer Analysis of Images and Patterns (CAIP), 2013. **(Oral)**

2. **Xi Zhang**, Chew Lim Tan, ”Segmentation-free Keyword Spotting for Handwritten Documents based on Heat Kernel Signature”, The 12th International Conference on Document Analysis and Recognition (ICDAR), 2013.

3. **Xi Zhang**, Chew Lim Tan, ”Unconstrained Handwritten Word Recognition based on Trigrams Using BLSTM”, The 22nd International Conference on Pattern Recognition (ICPR), 2014.

4. **Xi Zhang**, Chew Lim Tan, ”Text Line Segmentation for Handwritten Documents Using Constrained Seam Carving”, The 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2014.

5. **Xi Zhang**, Umapada Pal, Chew Lim Tan, ”Segmentation-free Keyword Spotting for Bangla Handwritten Documents”, The 14th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2014.

6. Chew Lim Tan, **Xi Zhang**, Linlin Li, ”Chapter 24 Image Based Retrieval and Keyword Spotting in Documents” in ”Handbook of Document Image Processing and Recognition”, 2014.

7. **Xi Zhang**, Chew Lim Tan, ”Handwritten Word Image Matching based on Heat Kernel Signature”, Pattern Recognition, 2014.

References

[1] Nikolaos Stamatopoulos, Basilis Gatos, Georgios Louloudis, Umapada Pal, and Alireza Alaei, "Icdar 2013 handwriting segmentation contest," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on.* IEEE, 2013, pp. 1402–1406.

[2] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[3] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855–868, 2009.

[4] Minh N Do and Martin Vetterli, "The contourlet transform: an efficient directional multiresolution image representation," *Image Processing, IEEE Transactions on*, vol. 14, no. 12, pp. 2091–2106, 2005.

[5] EA Galloway and VM Gabrielle, "The heinz electronic library interactive on-line system: An update," *The Public-Access Computer Systems Review*, vol. 9, no. 1, 1998.

[6] M. Ohta, A. Takasu, and J. Adachi, "Retrieval methods for english-text with misrecognized ocr characters, proc. of icdar'97," *Ulm, Germany*, pp. 950–956.

[7] Y. Ishitani, "Model-based information extraction method tolerant of ocr errors for document images," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on.* IEEE, 2001, pp. 908–915.

[8] S. M. Harding, W. B. Croft, and C. Weir, “Probabilistic retrieval of ocr degraded text using n-grams,” *The 1st European Conference Research and Advanced Technologies for Digital Libraries*, pp. 345–359, 1997.

[9] D. Lopresti and J. Zhou, “Retrieval strategies for noisy text,” in *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval.* Las Vegas, 1996, vol. 269.

[10] Kazem Taghva, Julie Borsack, and Allen Condit, “Expert system for automatically correcting ocr output,” in *IS&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology.* International Society for Optics and Photonics, 1994, pp. 270–278.

[11] A. Takasu, “An approximate string match for garbled text with various accuracy,” *the Fourth International Conference on Document Analysis and Recognition*, vol. 2, pp. 957–961, 1997.

[12] D. Doermann and S. Yao, “Generating synthetic data for text analysis systems,” *In Symposium on Document Analysis and Information Retrieval*, pp. 449–467, 1995.

[13] K. Tsuda, S. Senda, M. Minoh, and K. Ikeda, “Clustering ocr-ed texts for browsing document image database,” in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on.* IEEE, 1995, vol. 1, pp. 171–174.

[14] T. Kameshiro, T. Hirano, Y. Okada, and F. Yoda, “A document image retrieval method tolerating recognition and segmentation errors of ocr using shape-feature and multiple candidates,” in *Document Analysis and Recognition, 1999. ICDAR’99. Proceedings of the Fifth International Conference on.* IEEE, 1999, pp. 681–684.

[15] H. Fujisawa and K. Marukawa, “Full text search and document recognition of japanese text,” in *Symposium on Document Analysis and Information Retrieval*, 1995, pp. 55–80.

[16] Katsumi Marukawa, Tao Hu, Hiromichi Fujisawa, and Yoshihiro Shima, “Document retrieval tolerating character recognition errorsevaluation and application,” *Pattern Recognition*, vol. 30, no. 8, pp. 1361–1371, 1997.

[17] K. Katsuyama, H. Takebe, K. Kurokawa, et al., "Highly accurate retrieval of japanese document images through a combination of morphological analysis and ocr," in *Proc. SPIE, Document Recognition and Retrieval*, 2002, vol. 4670, pp. 57–67.

[18] T. Kameshiro, T. Hirano, Y. Okada, and F. Yoda, "A document retrieval method from handwritten characters based on ocr and character shape information," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on.* IEEE, 2001, pp. 597–601.

[19] Larry Spitz, "Duplicate document detection," in *Electronic Imaging'97.* International Society for Optics and Photonics, 1997, pp. 88–94.

[20] A.F. Smeaton and A.L. Spitz, "Using character shape coding for information retrieval," in *Proceeding of the 4th International Conference Document Analysis and Recognition*, 1997, pp. 974–978.

[21] A.L. Spitz, "Shape-based word recognition," *International Journal on Document Analysis and Recognition*, vol. 1, no. 4, pp. 178–190, 1999.

[22] A.L. Spitz, "Progress in document reconstruction," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on.* IEEE, 2002, vol. 1, pp. 464–467.

[23] F.R. Chen and D.S. Bloomberg, "Summarization of imaged documents without ocr," vol. 70, no. 3, pp. 307 – 320, 1998.

[24] J.M. Trenkle and R.C. Vogt, "Word recognition for information retrieval in the image domain," in *Symposium on Document Analysis and Information Retrieval*, 1993, pp. 105–122.

[25] Y. Lu, L. Zhang, and C.L. Tan, "Retrieving imaged documents in digital libraries based on word image coding," 2004.

[26] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, and S. Theodoridis and, "Keyword-guided word spotting in historical printed documents using synthetic data and user feedback," *International Journal on Document Analysis and Recognition*, vol. 9, no. 2, pp. 167 – 177, 2007.

[27] S. Lu, L. Li, and C.L. Tan, "Document image retrieval through word shape coding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 130, no. 11, pp. 1913–1918, 2008.

[28] S. Lu and C.L. Tan, "Retrieval of machine-printed latin documents through word shape coding," *Pattern Recognition*, vol. 41, no. 5, pp. 1799–1809, 2008.

[29] A. Murugappan, B. Ramachandran, and P. Dhavachelvan, "A survey of keyword spotting techniques for printed document images," *Artificial Intelligence Review*, pp. 1–18, 2011.

[30] R.F. Moghaddam and M. Cheriet, "Application of multi-level classifiers and clustering for automatic word spotting in historical document images," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 511–515.

[31] B. Gatos and I. Pratikakis, "Segmentation-free word spotting in historical printed documents," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 271–275.

[32] Luc Vincent, "Google book search: Document understanding on a massive scale," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE Computer Society, 2007, vol. 2, pp. 819–823.

[33] G. Agam, S. Argamon, O. Frieder, D. Grossman, and D. Lewis, "Content-based document image retrieval in complex document collections," in *Proc. SPIE*. Citeseer, 2007, vol. 6500.

[34] S. Marinai, E. Marino, and G. Soda, "Font adaptive word indexing of modern printed documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1187 – 1199, 2006.

[35] J. Li, Z.G. Fan, Y. Wu, and N. Le, "Document image retrieval with local feature sequences," in *2009 10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 346–350.

[36] Y.H. Tseng and D.W. Oard, "Document image retrieval techniques for chinese," in *Symposium on Document Image Understanding Technology*, 2001, pp. 151–158.

[37] Y. Lu and C.L. Tan, "Chinese word searching in imaged documents," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 2, pp. 229–246, 2004.

[38] S. Senda, M. Minoh, and K. Ikeda, "Document image retrieval system using character candidates generated by character recognition process," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on.* IEEE, 1993, pp. 541–546.

[39] M.W. Sagheer, N. Nobile, C.L. He, and C.Y. Suen, "A novel handwritten urdu word spotting based on connected components analysis," in *2010 International Conference on Pattern Recognition.* IEEE, 2010, pp. 2013–2016.

[40] W. Magdy, K. Darwish, and M. El-Saban, "Efficient language-independent retrieval of printed documents without ocr," in *String Processing and Information Retrieval.* Springer, 2009, pp. 334–343.

[41] Y. Leydier, F. Le Bourgeois, and H. Emptoz, "Omnilingual segmentation-free word spotting for ancient manuscripts indexation," in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on.* IEEE, 2005, pp. 533–537.

[42] Y. Xia, B.H. Xiao, C.H. Wang, and R.W. Dai, "Integrated segmentation and recognition of mixed chinese/english document," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on.* IEEE, 2007, vol. 2, pp. 704–708.

[43] Y. Lu and C.L. Tan, "Information retrieval in document image databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1398–1410, 2004.

[44] Toni M Rath and Raghavan Manmatha, "Word image matching using dynamic time warping," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on.* IEEE, 2003, vol. 2, pp. II–521.

[45] Volkmar Frinken, Andreas Fischer, R Manmatha, and Horst Bunke, "A novel word spotting method based on recurrent neural networks," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 2, pp. 211–224, 2012.

[46] Manivannan Arivazhagan, Harish Srinivasan, and Sargur Srihari, "A statistical approach to line segmentation in handwritten documents," in *Electronic Imaging 2007*. International Society for Optics and Photonics, 2007, pp. 65000T–65000T.

[47] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju, "A steerable directional local profile technique for extraction of handwritten arabic text lines," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 176–180.

[48] Georgios Louloudis, Basilios Gatos, and Constantin Halatsis, "Text line detection in unconstrained handwritten documents using a block-based hough transform approach," in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. IEEE, 2007, vol. 2, pp. 599–603.

[49] Fei Yin and Cheng-Lin Liu, "Handwritten chinese text line segmentation by clustering with distance metric learning," *Pattern Recognition*, vol. 42, no. 12, pp. 3146–3157, 2009.

[50] Zaidi Razak, Khansa Zulkiflee, Mohd Yamani Idna Idris, Emran Mohd Tamil, Mohd Noorzaily Mohamed Noor, Rosli Salleh, Mohd Yaakob, Zulkifli Mohd Yusof, and Mashkuri Yaacob, "Off-line handwriting text line segmentation: A review," *International journal of computer science and network security*, vol. 8, no. 7, pp. 12–20, 2008.

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

[52] Shai Avidan and Ariel Shamir, "Seam carving for content-aware image resizing," in *ACM Transactions on graphics (TOG)*. ACM, 2007, vol. 26, p. 10.

[53] Gordon Wilfong, Frank Sinden, and Laurence Ruedisueli, "On-line recognition of handwritten symbols," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 9, pp. 935–940, 1996.

[54] U-V Marti and Horst Bunke, “Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 01, pp. 65–90, 2001.

[55] Sanparith Marukatat, Thierry Artières, R Gallinari, and Bernadette Dorizzi, “Sentence recognition through hybrid neuro-markovian modeling,” in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on.* IEEE, 2001, pp. 731–735.

[56] Émilie Caillault, Christian Viard-Gaudin, and Abdul Rahim Ahmad, “Ms-tdnn with global discriminant trainings,” in *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on.* IEEE, 2005, pp. 856–860.

[57] Joachim Schenk, Gerhard Rigoll, et al., “Novel hybrid nn/hmm modelling techniques for on-line handwriting recognition,” in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[58] U.V. Marti and H. Bunke, “The iam-database: an english sentence database for offline handwriting recognition,” *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[59] Alessandro Vinciarelli and Juergen Luettin, “A new normalization technique for cursive handwritten words,” *Pattern Recognition Letters*, vol. 22, no. 9, pp. 1043–1050, 2001.

[60] U.V. Marti and H. Bunke, “Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system,” *IJPRAI*, vol. 15, no. 1, pp. 65–90, 2001.

[61] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” 2001.

[62] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[63] Stephen John Young, NH Russell, and JHS Thornton, *Token passing: a simple conceptual model for connected speech recognition systems*, Citeseer, 1989.

[64] R. Manmatha, Chengfeng Han, and E. M. Riseman, "Word spotting: A new approach to indexing handwriting," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1996, pp. 31 – 637.

[65] T.M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on.* IEEE, 2003, pp. 218–222.

[66] T.M. Rath and R. Manmatha, "Word spotting for historical documents," *International Journal on Document Analysis and Recognition*, vol. 9, no. 2, pp. 139–152, 2007.

[67] H. Bunke, S. Bengio, and A. Vinciarelli, "Offline recognition of unconstrained handwritten texts using hmms and statistical language models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 709–720, 2004.

[68] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[69] J.A. Rodrıguez and F. Perronnin, "Local gradient histogram features for word spotting in unconstrained handwritten documents," in *Int. Conf. on Frontiers in Handwriting Recognition*, 2008.

[70] F. Moreno-Noguer, "Deformation and illumination invariant feature point descriptor," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE, 2011, pp. 1593–1600.

[71] R.M. Rustamov, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *Proceedings of the fifth Eurographics symposium on Geometry processing.* Eurographics Association, 2007, pp. 225–233.

[72] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Computer Graphics Forum.* Wiley Online Library, 2009, vol. 28, pp. 1383–1392.

[73] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós, “Browsing heterogeneous document collections by a segmentation-free word spotting method,” in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE, 2011, pp. 63–67.

[74] Xi Zhang and Chew Lim Tan, “Segmentation-free keyword spotting for handwritten documents based on heat kernel signature,” in *Document Analysis and Recognition (ICDAR), 2013 International Conference on*. IEEE, 2013.

[75] M. Reuter, F.E. Wolter, and N. Peinecke, “Laplace–beltrami spectra as shapednaof surfaces and solids,” *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006.

[76] U. Pinkall and K. Polthier, “Computing discrete minimal surfaces and their conjugates,” *Experimental mathematics*, vol. 2, no. 1, pp. 15–36, 1993.

[77] Giuseppe Patané, “wfem heat kernel: Discretization and applications to shape analysis and retrieval,” *Computer Aided Geometric Design*, vol. 30, no. 3, pp. 276–295, 2013.

[78] M.M. Bronstein and I. Kokkinos, “Scale-invariant heat kernel signatures for non-rigid shape recognition,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1704–1711.

[79] Duk-Ryong Lee, Wonju Hong, and Il-Seok Oh, “Segmentation-free word spotting using sift,” in *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*. IEEE, 2012, pp. 65–68.

[80] A. Fischer, A. Keller, V. Frinken, and H. Bunke, “Hmm-based word spotting in handwritten documents using subword models,” in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 3416–3419.

[81] Chris Harris and Mike Stephens, “A combined corner and edge detector.,” in *Alvey vision conference*. Manchester, UK, 1988, vol. 15, p. 50.

[82] Tony Lindeberg, “Feature detection with automatic scale selection,” *International journal of computer vision*, vol. 30, no. 2, pp. 79–116, 1998.

[83] Dr Sébastian Gilles, *Robust description and matching of images*, Ph.D. thesis, University of Oxford, 1999.

[84] R. Manmatha, C. Han, and E.M. Riseman, “Word spotting: A new approach to indexing handwriting,” pp. 631–637, 1996.

[85] T.M. Rath and R. Manmatha, “Word spotting for historical documents,” *International Journal on Document Analysis and Recognition*, vol. 9, no. 2, pp. 139–152, 2007.

[86] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz, “Towards an omnilingual word retrieval system for ancient manuscripts,” *Pattern Recognition*, vol. 42, no. 9, pp. 2089–2105, 2009.

[87] V. Lavrenko, T.M. Rath, and R. Manmatha, “Holistic word recognition for handwritten historical documents,” in *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on.* IEEE, 2004, pp. 278–287.

[88] T Yung Kong and Azriel Rosenfeld, *Topological algorithms for digital image processing*, Access Online via Elsevier, 1996.

[89] Louisa Lam, Seong-Whan Lee, and Ching Y Suen, “Thinning methodologies-a comprehensive survey,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 14, no. 9, pp. 869–885, 1992.

[90] Oswaldo Ludwig Junior, David Delgado, Valter Gonçalves, and Urbano Nunes, “Trainable classifier-fusion schemes: an application to pedestrian detection,” in *Intelligent Transportation Systems, 2009 12th International IEEE Conference on.* IEEE, 2009, pp. 1–6.

[91] Sargur Srihari, Chen Huang, and Harish Srinivasan, “Content-based information retrieval from handwritten documents,” in *1st International Workshop on Document Image Analysis for Libraries (DIAL2004)*, 2004, pp. 188–194.

[92] Guillaume Joutel, Véronique Eglin, Stéphane Bres, and Hubert Emptoz, “Curvelets based queries for cbir application in handwriting collections,” in *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on.* IEEE, 2007, vol. 2, pp. 649–653.

[93] MS Shirdhonkar and Manesh B Kokare, "Writer based handwritten document image retrieval using contourlet transform," in *Advances in Digital Image Processing and Information Technology*, pp. 108–117. Springer, 2011.

[94] Arthur L Da Cunha, Jianping Zhou, and Minh N Do, "The nonsubsampled contourlet transform: theory, design, and applications," *Image Processing, IEEE Transactions on*, vol. 15, no. 10, pp. 3089–3101, 2006.

[95] Emmanuel J Candes and David L Donoho, "Curvelets: A surprisingly effective nonadaptive representation for objects with edges," Tech. Rep., DTIC Document, 2000.

[96] William Webber, Alistair Moffat, and Justin Zobel, "A similarity measure for indefinite rankings," *ACM Transactions on Information Systems (TOIS)*, vol. 28, no. 4, pp. 20, 2010.