8-9-2014

Document Image Analysis Techniques for Handwritten Text Segmentation, Document Image Rectification and Digital Collation

Dhaval Salvi
University of South Carolina - Columbia

Document Image Analysis Techniques for Handwritten Text Segmentation, Document Image Rectification and Digital Collation

by

Dhaval Salvi

Bachelor of Engineering
University of Mumbai 2007

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2014

Accepted by:

Song Wang, Major Professor

Michael N. Huhns, Committee Member

Manton M. Matthews, Committee Member

Yan Tong, Committee Member

David L. Miller, External Examiner

Lacy Ford, Vice Provost and Dean of Graduate Studies

Dedication

To my parents, Mr. Nathuram Salvi and Mrs. Suchitra Salvi, who have made numerous sacrifices so that I could follow my dream.

Acknowledgments

There are more people I would like to acknowledge for helping me during this dissertation than I can name. I would like to thank my advisor, Dr. Song Wang, for guiding me during this thesis work and for his support during all these years of research under his guidance. I would also like to thank my committee, Dr. Huhns, Dr. Matthews, Dr. Tong, and Dr. Miller whose guidance and feedback was invaluable during this thesis work. I would like to acknowledge Dr. Wilder for his research collaboration and support. I am also grateful to Mrs. Catherine Matthews for her support during these years.

The computer vision lab is a great place to do research and I cannot understate the importance of having labmates who have helped and made working on this dissertation fun. A special thanks to Dr. Pahal Dalal, Dr. Andrew Temlyakov, Dr. Yu Cao, Dr. Zhiqi Zhang, Dr. Jarrell Waggoner, Dr. Matthew Fawcett, Kenton Oliver, Jun Zhou, Yuewei Lin, Youjie Zhou, Xiaochuan Fan and Kang Zheng, who have been supportive throughout this journey. My heartfelt gratitude to the Computer Science and Engineering staff including Mrs. Barbara Ulrich, Mrs. Jewel Rogers and Ms. Randi Baldwin. There are many friends who have helped to keep me going and made the graduate life experience, a best part of my life.

Last but not least, I would like to thank my parents, my elder brother Devendra and my sister-in-law Karthika for their support and understanding throughout these years and their never wavering faith in my work.

Abstract

Document image analysis comprises all the algorithms and techniques that are utilized to convert an image of a document to a computer readable description. In this work we focus on three such techniques, namely (1) Handwritten text segmentation (2) Document image rectification and (3) Digital Collation.

Offline handwritten text recognition is a very challenging problem. Aside from the large variation of different handwriting styles, neighboring characters within a word are usually connected, and we may need to segment a word into individual characters for accurate character recognition. Many existing methods achieve text segmentation by evaluating the local stroke geometry and imposing constraints on the size of each resulting character, such as the character width, height and aspect ratio. These constraints are well suited for printed texts, but may not hold for handwritten texts. Other methods apply holistic approach by using a set of lexicons to guide and correct the segmentation and recognition. This approach may fail when the domain lexicon is insufficient. In the first part of this work, we present a new global non-holistic method for handwritten text segmentation, which does not make any limiting assumptions on the character size and the number of characters in a word. We conduct experiments on real images of handwritten texts taken from the IAM handwriting database and compare the performance of the presented method against an existing text segmentation algorithm that uses dynamic programming and achieve significant performance improvement.

Digitization of document images using OCR based systems is adversely affected if the image of the document contains distortion (warping). Often, costly and precisely

calibrated special hardware such as stereo cameras, laser scanners, etc. are used to infer the 3D model of the distorted image which is used to remove the distortion. Recent methods focus on creating a 3D shape model based on 2D distortion information obtained from the document image. The performance of these methods is highly dependent on estimating an accurate 2D distortion grid. These methods often affix the 2D distortion grid lines to the text line, and as such, may suffer in the presence of unreliable textual cues due to preprocessing steps such as binarization. In the domain of printed document images, the white space between the text lines carries as much information about the 2D distortion as the text lines themselves. Based on this intuitive idea, in the second part of our work we build a 2D distortion grid from white space lines, which can be used to rectify a printed document image by a dewarping algorithm. We compare our presented method against a state-of-the-art 2D distortion grid construction method and obtain better results. We also present qualitative and quantitative evaluations for the presented method.

Collation of texts and images is an indispensable but labor-intensive step in the study of print materials. It is an often used methodology by textual scholars when the manuscript of the text does not exist. Although various methods and machines have been designed to assist in this labor, it still remains an expensive and time-consuming process, often requiring travel to distant repositories for the painstaking visual examination of multiple original copies. Efforts to digitize collation have so far depended on first transcribing the texts to be compared, thus introducing into the process more labor and expense, and also more potential error. Digital collation will instead automate the first stages of collation directly from the document images of the original texts, thereby speeding the process of comparison. We describe such a novel framework for digital collation in the third part of this work and provide qualitative results.

Table of Contents

List of Tables

List of Figures

Introduction

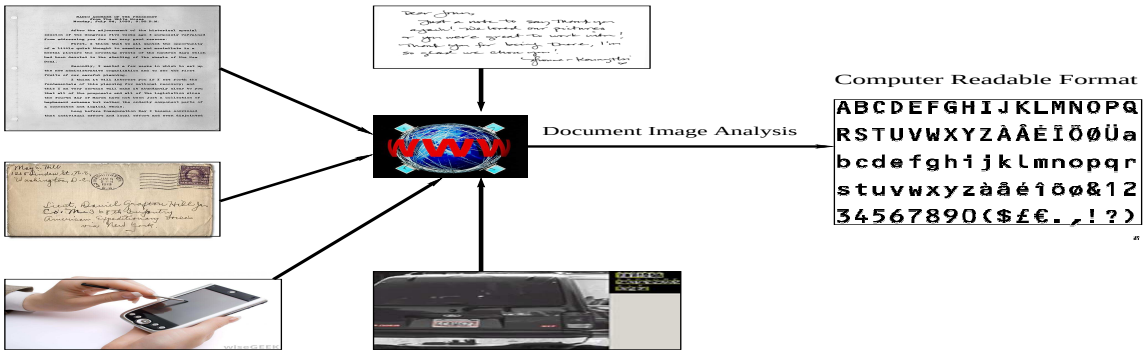


Figure 0.1: An illustration of application of Document Image Analysis.

Document Image Analysis encompasses all the algorithms and processes through which computers can automatically read and represent the information present in an image of a document. As illustrated in Fig.0.1, a wide range of information which has been conventionally stored as documents on paper is now being converted into digital form for permanent and better storage. These documents can include machine printed documents such as memos and technical reports, handwritten documents such as personal letters or addresses on postal mails, online handwritten documents such as PDA accquired documents, or video documents such as images from a video containing text etc. The algorithms and processes developed in document image analysis enable computers to automatically decipher the basic content of the documents. The world wide web has grown exponentially over the past few decades, with more and more information publicly available. To utilize this information which often can be in form of images of documents, it must be first digitized into computer readable format, thus enabling the data to be searched and items of interest to be retrieved. Document image analysis provides the means to accomplish this task. In this work

we focus on three such document image analysis techniques, namely (1) Handwritten text segmentation (2) Document image rectification and (3) Digital Collation.

Document image analysis techniques can be broadly categorized into either (a) Optical Character Recognition (OCR) based techniques or (b) Non-OCR techniques. OCR has various applications including applications such as banking, health care, digital libraries, digital repositories, legal industry, license plate recognition, handwritten letters recognition etc. Significant research has been done in recent decades, leading to a generally high recognition rate, but these recognition rates can drop severely in presence of image artifacts such as distortion introduced during scanning of the documents etc. Typically an OCR system is applied to printed texts or handwritten texts and can substantially reduce the time needed to convert a document to computer readable form, as compared to the time a human needs to manually enter the same data. Even though in recent years much research effort has been put into developing algorithms and methods to automatically convert documents into digital format, many documents that are easy to read for humans can achieve approximately only 92% recognition accuracy. This makes the automatic document conversion process still unreliable to entirely remove humans from the process, thereby increasing the time and cost required in digitization of document images. Most commonly these low accuracy rates can be attributed to image degradations that exist in these document images, which are caused in the process of printing, scanning, photocopying etc of the documents. One solution might be to make the OCR systems more robust against these degradations, but even this might not work. Therefore it is desirable to develop algorithms which will handle these degradations even before the step of OCR is utilized. In this work, we describe one such novel non-OCR technique to handle distortion caused during scanning of printed document etc.

Earlier research in the field of document image analysis focussed primarily on pre-processing techniques such as image accquisition [17, 75, 2], binarization [61, 29,

20, 65], layout analysis [69, 34], feature extraction [27, 35, 63] and classification. Most often in OCR systems, before performing any character recognition to decipher the text in document images, the first step is to binarize the image to convert the gray-scale document image into a binary document image. This is required to aid the algorithms which need to be utilized further such as de-skewing and layout analysis. Binarization is a difficult problem especially when the document images are accquired from old historic documents and thus contain a lot of noise and artifacts that make textual content blurry, faint or not legible. The main objective of binarization is to automatically choose a proper threshold that separates the foreground from the background. Selection of such a threshold is often the challenging part due to the degradations that might exist such as bleeding through, variance in illumination etc. Even though this topic has receieved much attention in the recent past, it is still an open problem. There have been great advances in this research and can be witnessed in recent binarization contests such as [56, 55].

Another important area of earlier research in document image analysis related to printed texts was to estimate the skew that exists in the scanned image. The skew of a document image specifies the tilt that may exist in the text lines from the horizontal axis. This occurs primarily during scanning or copying of the document. It is an inevitable process and its detection is critical for improving the performance of an OCR system. De-skewing is the process in which the image tilt angle is calculated and the image is rotated by that angle in clockwise or counter-clockwise direction with respect to the x-axis to properly orient the text lines. After the de-skewing process, the image has a zero skew angle [81, 15, 85]. Layout analysis research concerned determining the different components and regions of interest such as text body, tables, illustrations and symbols in the given printed text. Extracting relevant features from a document image for a given task is a challenging problem. Feature extraction and classification research focussed on extracting such features relevant to the given task

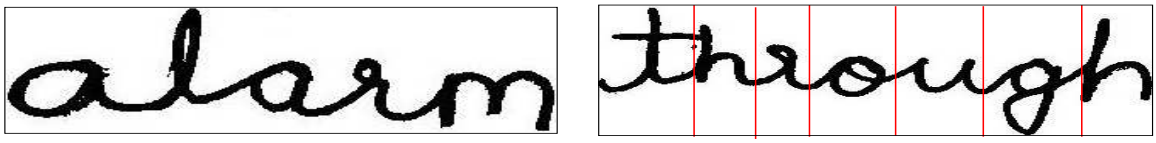
and used them to classify the documents.

Another scenario where the OCR systems can perform poorly is in the recognition of handwritten letter document images. This failure can be attributed to the inherent complexity of the handwritten documents which makes it impossible to design a universal OCR system or classifier for such an application. In this work we present several novel techniques which aim to address these complexities and thereby improve OCR system performance on handwritten texts by a substantial amount.

Document image analysis has enabled converting scanned document images into systematically searchable documents than just being ordinary image files. This has particularly helped in not just digitizing important documents such as old historic books which inevitably degrade over time and might be lost forever, but also extract information and develop various concepts which might be helpful to humanists. Non-OCR based techniques help to extract relevant information from printed text document images and enter them into an appropriate database which normally would have required manually finding and entering this information into such databases. As rare historical documents are digitized in greater numbers, we must develop tools that take advantage of this form of access. The challenges that attend digitization, including data loss and long-term preservation, have been studied, and the best practices to conserve digital assests have been identified by several museums and libraries. Still, the methods and equipment used to digitize materials as well as the policies and standards of various repositories vary widely. This variability in the conditions of collection presents major challenges to the comparison and analysis of materials from multiple repositories. Collation of printed texts and images is an indespensible but labor-intensive step in the study of print materials. Textual scholars have designed ingenious methods and machines to assist in this labor, but it remains both expensive and time consuming, requiring travel to distant repositories for the painstaking visual examination of multiple original copies. Therefore even the best editorial practices

recommended by textual scholars have rarely been implemented with only a few major libraries adopting such practices with the canonical texts held at such libraries. As scanning costs come down, we have an opportunity to change the way textual editing is done. Efforts to computerize collation have so far depended on first transcribing the texts to be compared, introducing into the process a layer not only of labor and expense but also of potential error. Digital collation will instead automate the first stages of collation directly from the scanned images of the original printed texts, dramatically speeding the process of comparison. it will also afford new functionalities for coping with variations in the quality and rendering of digital materials captured in different ways at different times. We describe such a novel framework for digital collation in this work.

Handwritten Text Segmentation



(a) (b)

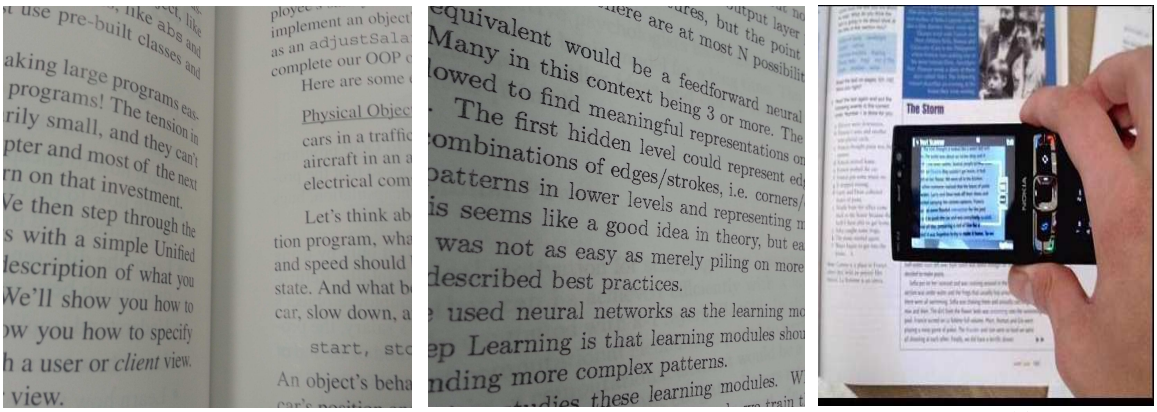
Figure 0.2: An illustration of the challenges in handwritten text segmentation. Note that the two "a"s show different sizes and shapes in (a) and there are different kinds of connection strokes in (b), where red vertical lines are the segmentation *boundaries* between neighboring characters.

Offline handwritten text recognition has a wide range of applications, such as automatic bank check processing and handwritten postal address recognition. One major challenge in handwritten text recognition is that neighboring characters within a word usually are connected when written, as shown in Fig. 0.2. Many OCR tools are built to recognize individual characters [24, 50, 82, 30]. As a result of this, to achieve handwritten text recognition, we often need to segment a connected word (or

words) into individual characters [66], which we call *handwritten text segmentation* in this work.

Handwritten text segmentation is a very difficult problem because there is a large variation in handwriting styles. For example, people may write the same character in different ways, including in different shapes and sizes, even within the same word, as shown in Fig. 0.2(a). As a result, it is usually difficult to ascertain the number of characters in a handwritten text to be segmented. Furthermore, the various ways in which two neighboring characters could be connected make it very difficult to determine the *boundary* that separates neighboring characters by evaluating only local stroke geometries. In Chapter 1, we introduce the second part of our proposed work which presents a new global non-holistic method to segment handwritten text.

Document Image Rectification



(a) (b) (c)

Figure 0.3: Examples of distortions: (a) Distortion at book bindings, (b) Perspective projection in camera captured image. (c) One of the applications of proposed method: Mobile document scanner (Image taken from google).

OCR research over the last few decades has led to highly accurate digitization of documents. However, there is a severe drop in the performance of OCR systems in the presence of distortion (warping) in the scanned/photographed document image as shown in Fig. 0.3. These systems rely on the document image being planar and

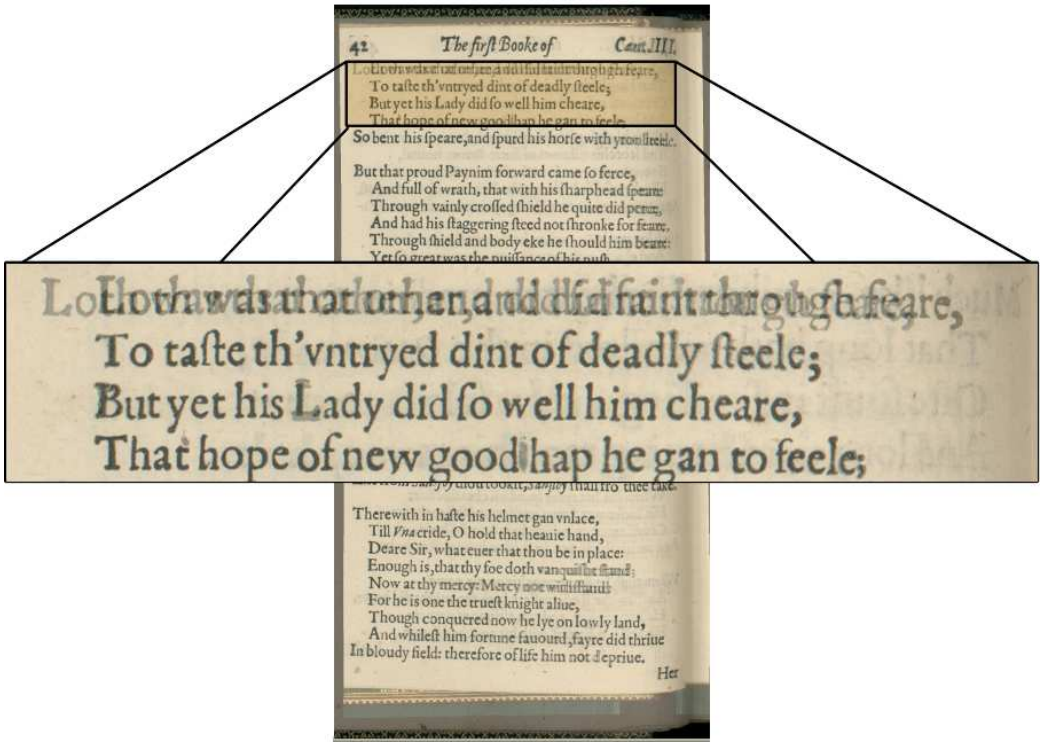


Figure 0.4: Collation of two pages, visualized as an overlay.

having straight horizontal text lines. Therefore, it is critical to remove any distortion that might exist in the document image. Distortion may be introduced in a variety of ways during the process of acquiring a document image. For example, book bindings may lead to warping on part of a page, and perspective projection may be introduced if using a camera instead of a scanner to obtain the document images. Ideally, a document image should be obtained from a flatbed scanner one page at a time. This is often not possible if the documents being digitized are from an historic book, which may be damaged by flattening. In Chapter 2, we describe the second part of our work which addresses Document image rectification.

Digital Collation

Collation is an often used methodology for humanists studying ancient documents, and is extensively used by historians of the book, bibliographic scholars, and those in-

terested in the material culture of print, particularly when the underlying manuscript of the text is nonexistent. Simultaneously examining multiple witnesses (copies) of a work enables the rich annotation of a corpus and can reveal subtle but important details such as printing errors and textual differences among witnesses. The existing mechanisms in place for performing collation have traditionally been limited by requiring two original copies to be compared (often obtained by flattening pages of a bound book, risking damage to the binding and adjacent pages), demanding a rigorous character-by-character manual inspection, and/or employing complex optical systems.

Filled with uncertainty, collation is always a binary process: instead of establishing truth, collation uses comparison to establish the difference between two printed texts . In performing collation, researchers isolate difference as a series of binary judgments, building alteration sequentially by comparing many individual witnesses to a given "control copy" that arbitrarily fixes the text to a given state. While many of the differences between texts consist chiefly of mechanical or human error—errors in typesetting and mechanical errors common to early presses and the methods for aligning and securing type within the formes—there are more than a few instances of variance *within an edition* that simply cannot be explained in terms of error. For example, in the 1590 *Faerie Queene*, there are variants with whole words inserted or deleted, lines abridged or added, sonnets relocated or re-ordered. Coupled with the lack of a stable underlying manuscript to fix the ground of the text, collational variance becomes part of the assemblage of the text, irreducibly part of the play of its intricate meaning. Despite its importance, collation remains a slow process filled with inefficiencies for researchers and their objects of study.

Digital collation has risen to prominence recently [57, 46], as it addresses many deficiencies of existing methods. Operating on highly-detailed digital photographs of the original documents, which can be taken without flattening the book binding

them, eliminates risk of harm to the source documents. Partially automating the inspection process and crafting better visualizations of the collated documents can expedite the annotation of large works when contrasted with fully manual inspection using cumbersome optical devices. In chapter 3 of this work, we describe such a novel framework.

Related Work

Handwritten text segmentation related work

Earlier methods for handwritten text segmentation evaluate the local stroke geometry for segmentation boundaries [44, 19]. For example, Liang et al. [41] propose two different types of projections to construct a segmentation, and optimize this segmentation using a dynamic recursive algorithm and contextual information. Between the top and bottom sides of the text image, Wang et al. [77] find paths with the minimum number of stroke pixels, and use such shortest paths as the text-segmentation boundaries. However, these overly simplified criteria for determining the segmentation boundaries work only for printed texts, but not handwritten texts.

Recent methods use character recognition for aiding text segmentation. In these methods, for each resulting text segment, a *character likeliness* is first defined to measure how likely the segment is a valid character using a character recognition algorithm. Text segmentation is then achieved when the resulting characters show large character likeliness. For example, in [59] the text image is described by a feature graph and the text segmentation is achieved by identifying subgraphs with large character likeliness. Martin et al. [48] use a sliding window approach to scan horizontally over the text image, and use a neural network classifier to measure the character likeliness of the subimage within the sliding window. Recently, the award winning paper (best student paper of ICDAR 2009) by Roy et al. [60] used local

stroke geometry to identify a set of candidate segmentation boundaries, and then used dynamic programming to decide the final segmentation boundaries that lead to a maximum total character likeliness.

Many holistic methods have also been developed for text image segmentation and recognition. In these methods, it is assumed that the texts presented in the image are valid words from a set of lexicons [78, 3]. The text image segmentation and word recognition problems are then solved simultaneously by using features from the whole text image. For example, Arica et al. [5] extend the method used by [37] to segment characters by running a series of constrained shortest-path algorithms, and use a Hidden Markov Model to do word-level recognition. Lee et al. [38] extend the method developed in [48] by using a cascade neural network classifier. However, if the texts presented in an image are not valid words (for example, in the application of finding typos), or the lexicon domain is insufficient, the above holistic methods will fail.

Document image rectification related work

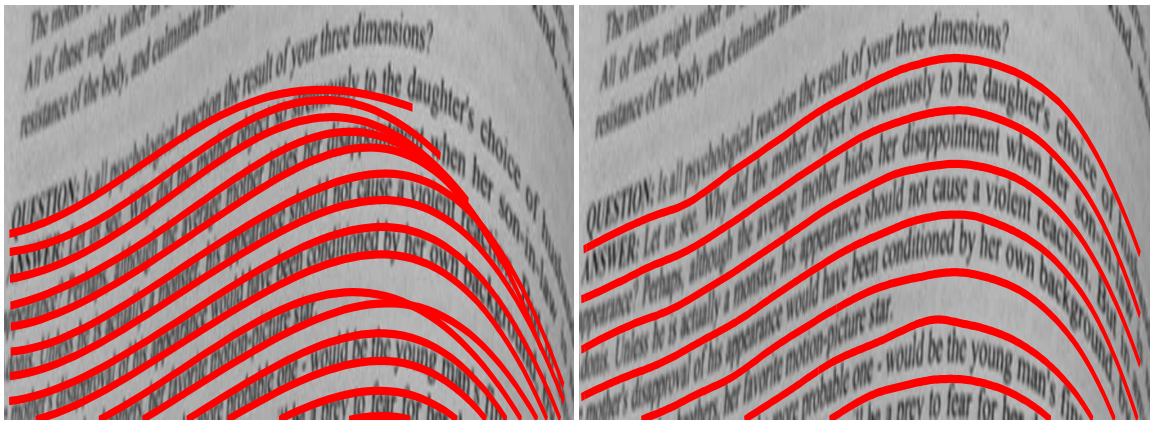
Previous literature on handling this distortion problem can be categorized in three ways. The first category of related works rely on special hardware, such as stereo cameras [71] [84], laser scanners [54] or structured light devices [10] to solve the distortion problem. These precisely-calibrated systems acquire a 3D shape model of the distorted document, and use the obtained model to rectify the distortion. Although such systems are shown to be highly accurate, the cost and size of such hardware makes them an impractical option for many applications.

The second category of approaches [14, 86, 88, 36, 26, 80, 72, 12] rely on inferring the distortion from the text lines present in the document. These approaches pre-process images using techniques such as binarization, connected component analysis, or character segmentation to estimate the 2D distortion grid. The accuracy of such

methods is highly sensitive to the results obtained from the aforementioned preprocessing techniques, and thus may obtain poor results if the underlying preprocessing yeilds unsatisfactory results. Ulges et al. [72] use a priori layout information and local textline estimation with RAST [9] which relies on connected component analysis. Bukhari et al. [12] use Gaussian matched filters to enhance text lines and employ a ridge detection technique to find the center of the text line, after which an active contour model is used to find the top and bottom part of the horizontal text lines.

The third category of methods employ shape-based models to perform document dewarping. These works can be further divided into methods which: a) have a rigid predetermined shape model, and b) use a deformable shape model to capture more variation among the possible distortion. Methods which use a rigid shape model include Cao et al. [14], who propose a model-based method to rectify document image distortion, which makes the assumption that the book surface is a cylindrical surface. Also, Fu et al. [28] extend this work by removing the constraint that the camera lens must be strictly parallel to the book surface, and additionally introduce textual information for reconstructing the 3D shape. Meng et al. [49] present a metric rectification method which employs a general cylindrical surface to model the page shape, and use the horizontal textline information and vanishing point to sample a grid on this cylindrical surface. Such rigid model based methods are able to rectify a scanned document image but mostly fail to rectify document images captured with a camera, where the distortion does not necessarily follow a rigid model.

Methods which use a deformable shape model, relaxing the fixed-shape constraint, are able to better handle such distortions. Such methods include Liang et al. [39], which relax the strong shape assumption by asserting that the document image can be approximated by a developable surface. This assumption is intuitive because document pages can be unrolled without stretching or tearing. Shao et al. [62] extend this idea by formulating a locally deformable surface instead of a globally deformable



(a) (b)

Figure 0.5: An illustration of line tracings. (a) Example of wrongly estimated horizontal text lines as described in [70], (b) White space based line tracing of proposed method.

surface. Tan et al. [68] propose a shape-from-shading (SFS) method to recover the 3D shape of the document.

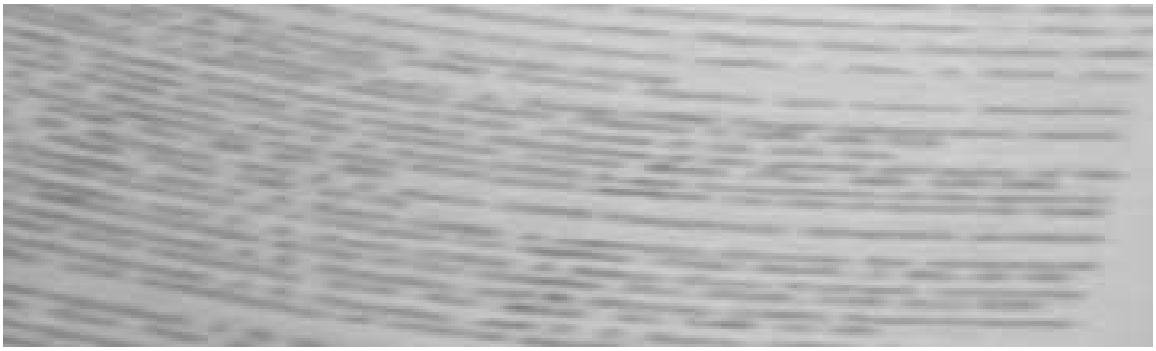


Figure 0.6: Example of distortion perception based on text line and white space between text lines.

Tian et al. [70] propose a new state-of-the-art 3D rectification framework which utilizes a similarity measure based horizontal line tracing and uses vertical stroke statistics to estimate the vertical direction. As shown in Fig. 0.5 (a), this method may suffer from inaccurate line tracing when the similarity measure mistakenly associates two patches within the image, thereby affecting the final result.

The presented method falls in the shape-from-X class of approaches. We humans

can easily infer the 2D distortion based on the text lines and the surrounding white space between these text lines, as shown in Fig. 0.6, independent of the language or fonts present in the document. Based on this intuition, we utilize the white space that exists between the text lines, rather than the actual lines themselves, to estimate the 2D distortion grid. We use this approach because preprocessing methods such as binarization, connected components etc. can affect the information extracted from the text lines, whereas the white space surrounding these text lines is less affected by such preprocessing. We present a distance transform (DT) based line propagation technique guided by an open active contour algorithm, to robustly estimate the horizontal line tracings and build a 2D distortion grid based on the white space. A typical approach to represent this notion of white space is to use DT of the binarized document image. A DT assigns each pixel of the image a distance based on the proximity of the given pixel to its nearest foreground text pixel. [1, 6] utilize the DT for a different application based strictly on horizontal, straight lines. In our proposed method, we make no such straight-line assumption, and formulate the 2D distortion grid estimation based on the DT.

Digital collation related work

Much work has been done both in identifying differences among similar documents and registering documents so that they can be compared. Tan et al. [67] and Marinai et al. [47] propose a text retrieval solution that works without doing character recognition, much as our collation solution operates without having to do recognition. Marinai et al. [64] handle the problem of clustering in a similar image-based manner. The system proposed by Beusekom et al. [74] is similar in nature to the problem of collation, but only focuses on revision detection and does not handle cases of non-affine document warping. There are also a number of works, both old and new, that focus on similar preprocessing steps that our dataset requires [31, 43, 22].

There are a number of works that focus on doing various types of comparison among documents. [53, 23] focus on extracting and identifying differences among a subset of documents; specifically "drop caps" (the enlarged and often stylized letter at the beginning of a document). Baudrier et al. [7] also focus specifically on drop caps, though the difference detection methods they present have broader and less specific application. Brzakovic et al. [11] present a system that targets document falsification (in typewriter-written documents) and uses several forms of unary analysis with the intent to produce a binary decision as to whether a document is falsified. Similarly, Doermann et al. [25] and Caprari et al. [16] target document duplication—the identification of documents with duplicate content within a large dataset. In much the same way, Hull et al. [33] propose a more granular, patch-based system that provides a generic similarity measure. Radke et al. [58] presents a survey of change detection algorithms.

There are several complete systems that attempt to solve similar problems to what we present here. Document Mosaicing is a related problem to our collation solution. In mosaicing, multiple parts of a document are captured with a camera or scanner, where there is some overlap among the images captured. The job of a mosaicing algorithm is to stitch these parts back into a complete document. Our collation application is similar in that we align images under sometimes severe distortion, but the images do not represent subparts of the document, but rather the entire document. In addition, we must handle more complex (non-affine) transforms that mosaicing applications tend not to handle. Earlier mosaicing systems [87] required a fixed camera to reduce perspective distortion, but more modern applications [40] have attempted to eliminate this requirement. Other systems focus on video [52], and can exploit the temporal relationship between frames to improve mosaicing. The equivalent operation in the natural image domain is image stitching, which has been used for years to stitch together photographs into panoramas [51]. Many of these systems rely upon

feature matching and/or correlation metrics to construct the alighment, much as we do in our solution. More generally, [76] focus on the selection of feature points for doing a general document registration, though they have no comparison step as our solution requires.

Research Contribution

The research presented in this work addresses three major problems in the domain of document image analysis as discussed in the previous section. In particular, the three major contributions are: (1) Handwritten text segmentation (2) Document image rectification and (3) Digital collation.

In Chapter 1 of this work which deals with handwritten text segmentation, we develop a new global non-holistic method to segment handwritten text by maximizing the *average* character likeliness of the resulting text segments. Different from many previous methods, this proposed method does not make any limiting assumptions about the number of resulting characters and the size of an individual character. We uniformly and densely sample the text image to construct a set of candidate segmentation boundaries. We reduce the text segmentation to the problem of finding an *average* longest path between the first and the last candidate segmentation boundary. We find that the average longest path in the constructed graph can be found in polynomial time with global optimality. We implement such an algorithm, and test it on real handwritten text images taken from the IAM handwriting database.

In Chapter 2 of this work which addresses document image rectification, we present a novel 2D distortion grid estimation method to rectify distorted images based on the white space lines that are present between text lines. A distance transform-based line propagation approach is presented to obatain a robust estimation of the white space lines. These white space lines are then used to build a 2D distortion grid, which is used in a 3D rectification algorithm to dewarp a document image.

We demonstrate the robustness and accuracy of our method by providing qualitative and quantitative evaluations, and compare with a state-of-the-art method, achieving better results.

In Chapter 3 of this work, we present a solution to a time-honored problem in the humanities-that of document collation. By solving this problem in the digital domain, many of the downsides of tedious optical methods can be resolved. Furthemore by employing state-of-the-art feature matching and registration, we are able to automatically identify the most important differences for humanists studying a work.

Chapter 1

Handwritten Text Segmentation using Average Longest Path Algorithm

1.1 Method

In this chapter, we consider binary handwritten text images with the border fitting tightly around the text. As shown in Fig. 1.1, the presented method consists of several components.

1. Construct the candidate segmentation boundaries, as shown by the vertical red lines in Fig. 1.1(b).

2. Construct a directed graph where each vertex represents a candidate segmentation boundary, including the left and right image border, where each edge represents the text segment between two candidate segmentation boundaries, as shown in Fig. 1.1(c).

3. Weigh the graph edges by the character likeliness derived from a character recognition algorithm.

4. Find the average longest path between the leftmost vertex (left image border) and rightmost vertex (right image border) in the graph, as shown in Fig. 1.1(d).

5. Take the candidate segmentation boundaries, corresponding to the vertices along the identified average longest path, as the final segmentation boundaries for text segmentation, as shown in Fig. 1.1(e).

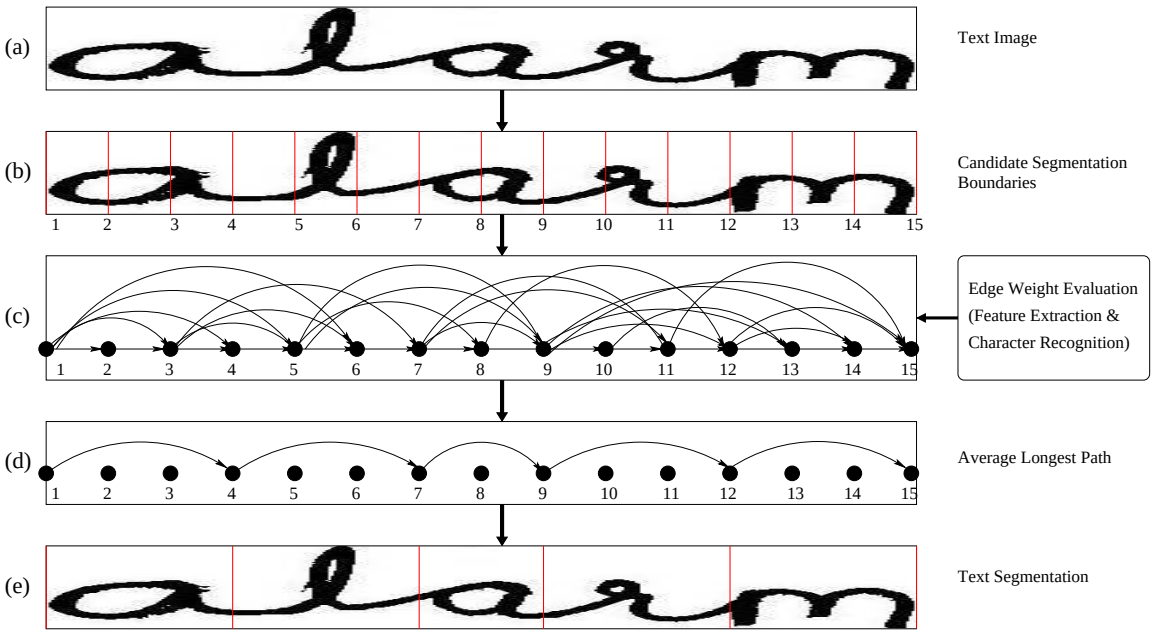


Figure 1.1: An illustration of the components of the presented method.

For Component 1, we uniformly and densely partition the text image, as shown in Fig. 1.1(c), for the candidate segmentation boundaries. When speed is not a factor, we can even partition the text image into single-column text segments. Note that the candidate segmentation boundaries constructed using this approach contain a large number of false positives compared to the set of true segmentation boundaries. While most previous methods need a good initial set of candidate segmentation boundaries (e.g., covering all the desired segmentation boundaries with few false positives), the presented method can robustly handle a large number of false positives.

For Component 2, given a set of candidate segmentation boundaries $S_1, S_2, \cdots, S_n$ that are ordered from left to right on the text image, as shown in Fig. 1.1, we construct a directed graph $G = (V, E)$ as follows:

1. Each candidate segmentation boundary S_i will be represented by a vertex v_i in G. Note S_1 and S_n represent the left and right border of the text image.

2. Between each pair of vertices v_i and v_j, where $i < j$, we construct a *directed*

edge $e_{ij} = (v_i, v_j)$ that starts from v_i and ends at v_j.

Note that we construct edges between both neighboring and non-neighboring candidate segmentation boundaries. As mentioned above, each edge represents the text segment between two candidate segmentation boundaries. Therefore, an edge between non-neighboring candidate segmentation boundaries actually represents a text segment that merges multiple partitions from Component 1. An example is illustrated in Fig. 1.1, where edge e_{14} actually indicates that the first three partitions from Component 1 are merged as a text segment. If this edge is included in the identified average longest path, this merged text segment will constitute a single character in the final text segmentation.

In the following, we first present a method that uses character recognition for measuring the character likeliness of a text segment, and use this character likeliness as the edge weight in the constructed graph. We then develop a graph algorithm to find the average longest path for text segmentation.

Character Likeliness Measure

The edge weight $w(e_{ij})$ describes the character likeliness of the text segment between candidate segmentation boundaries S_i and S_j, where $i < j$. The basic idea is to feed this text segment (a subimage) into an adapted character classifier to ascertain its character likeliness: a text segment fully and tightly covering a single character is expected to return a high character likeliness while a text segment covering part of a character, or overlapping multiple characters, is expected to return a low character likeliness. In this chapter, we train a set of support vector machine (SVM) classifiers for this purpose.

In this chapter, we focus on text consisting of the 26 Roman alphabetic characters. Thus we have 26 classes of characters. We train a binary SVM classifier [18] for each class of characters. For this purpose, we collect a set of isolated handwritten

characters as training samples. In training the binary SVM classifier for a specific character class, say "a", the training samples in this class are taken as positive samples and the training samples in the other 25 classes are taken as negative samples. When a new test sample is fed into this binary SVM classifier, we obtain a *class likeliness* associated with this character class. By testing against all 26 SVM classifiers, we obtain the class likeliness associated with each of these 26 character classes, and we simply take the maximum class likeliness as the character likeliness.

More specifically, in this chapter we use the lib-SVM [18] implementation for each binary SVM classifier, which has two outputs: a classification indicator of positive $(+1)$/negative (-1), and a probability estimate p in $[0, 1]$ that describes the confidence of the classification. If the indicator is $+1$, we simply take p as the class likelihood. If the indicator is -1, we take $1-p$ as the class likelihood because, in this case, p is the negative classification confidence.

For a text segment, we extract the HOG (Histogram of Oriented Gradients) based features [45] as the input for the SVM classifiers. We first normalize the size of the text segment to a 28×28 image. Each pixel in the image is assigned an orientation and magnitude based on the local gradient. The histograms are then constructed by aggregating the pixel responses within cells of various sizes. Histograms with cell size 14×14, 7×7 and 4×4 with overlap of half the cell size were used. Histograms at each level are multiplied by weights 1, 2 and 4 and concatenated together to form a single histogram. The details of construction of these feature can be found in [45].

Text Segmentation by Finding Average Longest Path

Based on the above formulation, the major task of text segmentation is to identify a subset of candidate segmentation boundaries $S_{f_1}, S_{f_2}, \cdots, S_{f_m}$, where $1 = f_1 < f_2 < \cdots < f_m = n$, as the final segmentation boundaries. The number of final segmentation boundaries, m, is unknown, and to be discovered in text segmentation. The general

principle is that the text segments defined by boundary pairs S_{f_i} and $S_{f_{i+1}}$, $i = 1, 2, \cdots, m$ should show large character likeliness. In another words, the graph edges between v_{f_i} and $v_{f_{i+1}}$ should have a large weight. In [60], this is formulated as an optimization problem

$$\begin{aligned} & (m^*, f_i^*; i = 1, 2, \cdots, m^*) \\ = & \arg \max_{m, f_i; i=1,2,\cdots,m} \sum_{i=1}^{m-1} w_{f_i, f_{i+1}}, \end{aligned}$$

where $w_{f_i,f_{i+1}} = w(v_{f_i}, v_{f_{i+1}})$ is the edge weight between v_{f_i} and $v_{f_{i+1}}$. A dynamic programming algorithm can be applied to find the global optimal solution efficiently. The major issue with this method is its undesired bias toward more segmentation boundaries, i.e., larger m, which may result in an oversegmentation of the text. This can be illustrated by a simple example in Fig. 1.2, where the text segments between S_1 and S_3 have a large character likeliness ($w_{13} = 1$), but this dynamic programming based method may still choose a segmentation boundary S_2 between S_1 and S_3 because $w_{12} + w_{23} > w_{13}$.

Figure 1.2: An illustration of the oversegmentation problem of dynamic programming.

In this chapter, we propose to address this problem by defining a new cost function

$$\begin{aligned} & (m^*, f_i^*; i = 1, 2, \cdots, m^*) \\ = & \arg \max_{m, f_i; i=1,2,\cdots,m} \frac{\sum_{i=1}^{m-1} w_{f_i, f_{i+1}}}{m-1}. \end{aligned}$$

which finds a path between v_1 and v_n with the maximum average total weight. In this chapter, we call this path the *average* longest path. Clearly, by introducing the

path length in the denominator in Eq. (2.1), we remove the bias toward a larger m and avoid oversegmentation of the text. In the following, we show that the average longest path in the constructed graph G can be found in polynomial time, and present such a polynomial time average longest-path algorithm.

First, on the graph G we define a second edge weight $l_{ij} = l(v_i, v_j) = 1$ for each (v_i, v_j). We then construct an augmented *directed* edge (v_n, v_1) that starts from v_n and ends at v_1, as shown in Fig. 1.3(b). We also set the weight for this augmented edge as $w(v_n, v_1) = l(v_n, v_1) = 0$. This way, finding the average longest path in G is reduced to the problem of identifying a *directed* cycle C in the augmented graph G' with a maximum cycle ratio

$$\frac{\sum_{e \in C} w(e)}{\sum_{e \in C} l(e)}.$$

We then transform the edge weight w to W by $W(e) = 1 - w(e)$ for all edges in the augmented graph G'. The problem is then reduced to finding a cycle with a minimum cycle ratio

$$\frac{\sum_{e \in C} W(e)}{\sum_{e \in C} l(e)}.$$

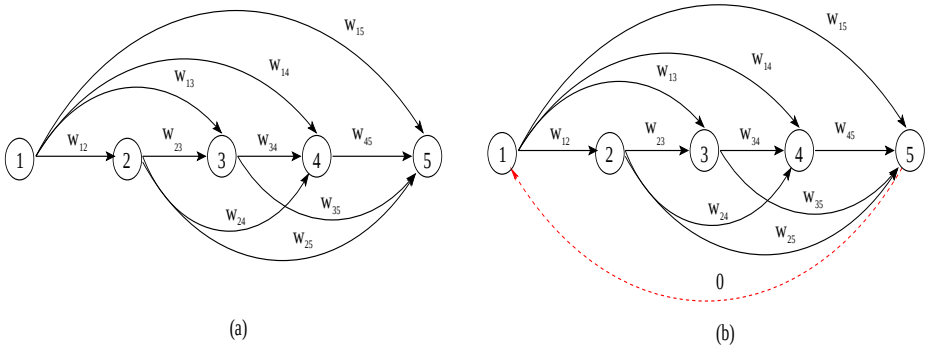


Figure 1.3: An illustration of adding an augmented edge for finding the average longest path in G. (a) Original graph G. (b) Graph augmented by an edge (v_n, v_1).

It is easy to prove that the desired cycle with the minimum cycle ratio is invariant to the edge weight transformation

$$W(e) \leftarrow W(e) - b \cdot l(e) \tag{1.1}$$

for any constant b. Clearly, there exists an optimal constant b^* such that the linear transform in Eq. (1.1) leads to $\sum_{e\in C} W(e) = 0$. In this case, the problem is reduced to finding a zero-weight cycle with $\sum_{e\in C} W(e) = 0$. To search for this optimal b^* and this zero-weight cycle, we can use the sequential-search algorithm [4] shown in Algorithm 1 on G'.

Algorithm 1: Sequential-search Algorithm:

1: Initialize $b = \max_{e\epsilon E} W(e) + 1$. We know that $b^* < b$
2: Transform the edge weights using Eq. (1.1) and then detect a negative cycle C, i.e., $\sum_{e\in C} W(e) < 0$. For the initial b, there must exist such a negative cycle because the current $b > b^*$. If no negative cycle is detected in a later iteration, return the cycle C detected in the previous iteration as a minimum ratio cycle in the augmented graph G'.
3: Calculate the cycle ratio $\frac{\sum_{e\in C} W(e)}{\sum_{e\in C} l(e)}$ using the original edge weights W without applying the linear transformation (1.1), using this calculated cycle ratio as the new b, return to Step 2.

Negative cycle detection is a well-solved polynomial-time problem [21] and has a worst-case running time of $O(n^2 mlog(n))$. Here n is the number of nodes in the graph and m is the number of edges. The complete presented handwritten text segmentation algorithm can be summarized by Algorithm 2.

Algorithm 2: Handwritten Text Segmentation:

1: Construct candidate segmentation boundaries by uniformly and densely sampling the text image.
2: Construct graph G representing candidate segmentation boundaries as vertices. Construct forward edges between each pair of vertices.
3: Define the edge weight w to reflect the character likeness.
4: Find the average longest path in the constructed graph by using the graph algorithm described in Section 1.1.
5: Keep the candidate segmentation boundaries whose corresponding vertices are included in the identified average longest path as the final text segmentation boundaries.

1.2 Experiments

In our experiments we use the standard IAM handwriting database. This database consists of 657 different writers and 1539 pages of scanned text. For testing, we randomly selected a set of 300 handwritten words from this IAM handwriting database. These words were collected from a subset of 50 different writers. Each word is made up of 2 to 9 characters, drawn from 26 lowercase characters. The characters in each word are written in a connected fashion, and we apply the presented text segmentation method to segment each word into individual characters. We also collected characters in isolation for these 26 character classes from a training set of 200 words from the same database without any overlap to the 300 words used for testing, and we use these isolated characters as training samples for the SVM classifiers. For each character, we collected 50 training samples giving us a total of 1300 training samples.

In constructing the candidate segmentation boundaries, we uniformly sample each test word with an interval of 10 pixels. To quantitatively evaluate the performance of a text segmentation, we manually construct a ground truth segmentation for each test word. Specifically, we present, to a human evaluator, each test word overlaid by the candidate segmentation boundaries. The human evaluator simply selects a subset of these boundaries that best separate all the characters as the ground-truth segmentation boundaries. To evaluate a segmentation result, we calculate the precision and recall in finding these ground-truth segmentation boundaries. Here a text segment is true positive only if it spatially overlaps with a ground-truth segment perfectly. Table 2.1 shows the average precision, average recall, and their standard deviations in terms of all the 300 test words, together with the average F-score that combines precision and recall.

For comparison, we implement the dynamic programming (DP) based text segmentation method developed in [60]. As discussed above, this method has a bias toward oversegmentation and requires a good initial candidate segmentation with

Table 1.1: Precision/Recall statistics for the presented method and the comparison method on all the 300 test words. AP is average precision, AR is average recall, SP is the standard deviation of the precision, and SR is the standard deviation of the recall.

	AP	SP	AR	SR	F-score
Proposed	0.7968	0.2158	0.7961	0.2390	0.7965
DP [60]	0.6090	0.2203	0.5060	0.2515	0.5526

Figure 1.4: Text segmentation on a subset of the test words. For each test word, we show the segmentation from: (Left) the ground truth, (Middle) the presented method, and (Right) the dynamic programming based method [60]. The characters below the test words for the presented method (Middle column) are the recognition results from the character class corresponding to the character likeliness. The characters below the test words for ground truth (Left column) are the ground-truth characters for these words.

few false positives. Following the basic ideas described in [60], we adopt the following two strategies to prune more candidate segmentation boundaries before applying dynamic programming for text segmentation.

1. Prune all the candidate segmentation boundaries that enter stroke pixels more than twice when scanning from the top to the bottom.

2. Do not allow the dynamic programming algorithm to consider text segments that have an aspect ratio greater than 1.2.

For fair comparison, we use the same histogram of oriented gradients feature [45] for this comparison method as we used for the presented method (see Section 3.1). The performance of this dynamic programming method is also shown in Table 2.1. We can clearly see that the presented method outperforms this dynamic programming based method. Sample segmentation results are shown in Fig. 1.4, where the character recognition results for the presented method (shown below each word) are obtained from the character class corresponding to the character likeliness. We can see that, even with the additional strategies reducing false positives, the dynamic programming based method still produces many oversegmentations. Note that characters such as "L" in the word "Labour" and "B" in the word "Bell" are still recognized using the same classifier which is trained only on 26 lowercase characters.

Using the above precision/recall metric does not well quantify the segmentation discrepancy in pixels. To address this issue, we further evaluate the spatial overlap between the ground-truth segmentation and the segmentation from a test method. Specifically, given each text segment T resulting from a test method, we find its best overlapped ground-truth text segment T_G and calculate their spatial-overlap difference as

$$\phi(T, T_G) = 1 - \frac{T \cap T_G}{T \cup T_G}.$$

A lower spatial overlap difference indicates a better segmentation. We calculate the average of this overlap difference over each obtained text segment, and then over all the 300 test words. The performance of the presented method and the comparison method, in terms of this overlap difference, is shown by the box plot in Fig. 1.5. Clearly, the presented method achieves much better text segmentation.

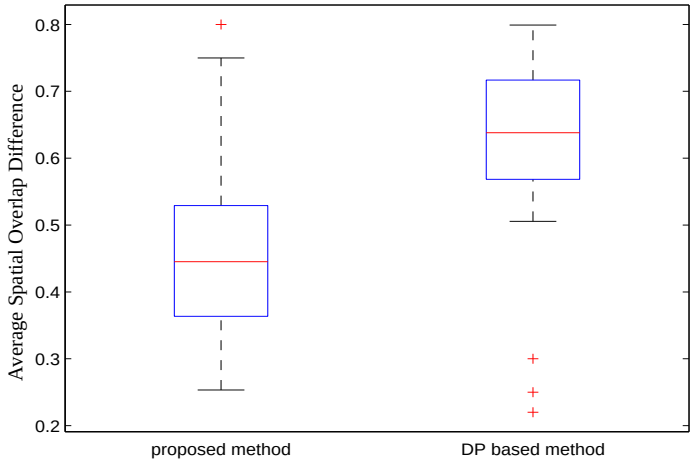


Figure 1.5: Average spatial overlap difference over all the 300 test words, using the presented method and the comparison method based on dynamic programming

A fair comparison with holistic methods against presented method is not possible since holistic methods utilize trained models for each word to recognize each test word according to a fixed set of lexicon. The presented method does not employ such a lexicon based word classifier and this will enable the application of our method to more general application domains where the handwritten texts cannot be modeled by a fixed set of lexicons. Since the 300 IAM handwritten texts used in our experiments are valid English words, we conduct a simple experiment to see how many words out of these 300 can be correctly recognized by applying the spelling-check tool (in the Microsoft Office) to the segmentation and character recognition results from the presented method. If any one of the top four candidate words provided by the spelling check is correct, we count this text as correctly recognized. We found that using this

spelling check, we can get 65% of these 300 test words correctly recognized. One previous holistic method [73] reported 73.45% of recognition rate on 300 test words from IAM database. However, it is unclear to us which 300 test words are used in [73] and what kind of the lexicons are used in achieving this rate.

Without setting any limiting constraints on the size and aspect ratio of the segmented characters, the presented method has an advantage in segmenting handwritten texts in which character size varies substantially from one to another. This is particularly useful in applications where the text is distorted in scanning. For example, when the page to be scanned is not tightly pressed on the scanner, some texts may become smaller and thinner in the scanned text image. Figure 1.6 shows three such examples, where the right side of each word is condensed horizontally and the characters become thinner from left to right in each word. We found that the presented method can successfully segment these words as shown in Fig. 1.6, even without training the SVM classifiers using any such distorted characters.

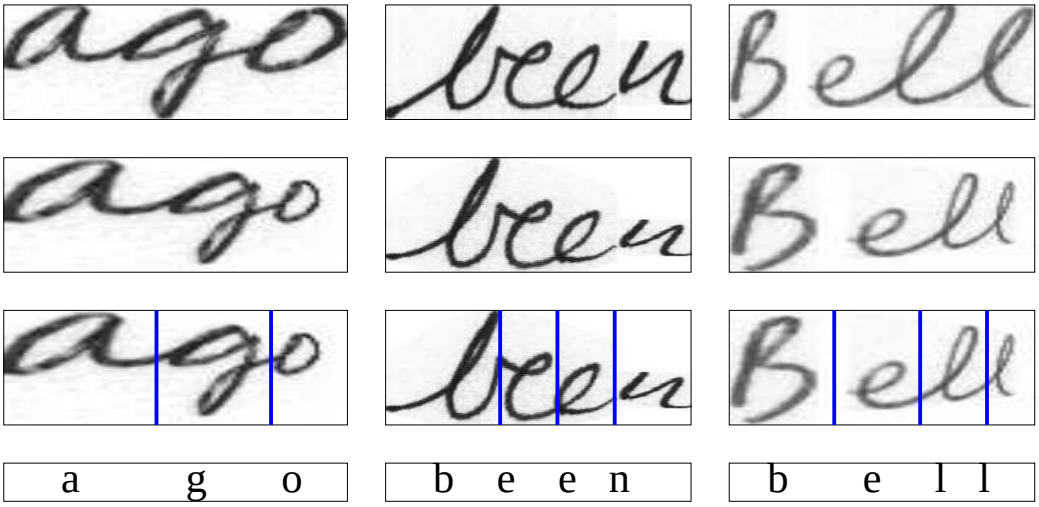


Figure 1.6: Examples of segmenting three distorted texts. (Top) Original texts. (Second row) Distorted texts. (Third row) Text segmentation using the presented method. (Bottom) Character recognition from the class corresponding to the character likeliness.

We also conduct experiments to show the effectiveness of character likeliness, as

defined by the output of multiple binary SVM classifiers (see Section 3.1), in distinguishing characters and non-characters. In the test words, we randomly select 400 text segments, in which 200 are characters from the ground truth, and 200 are non-characters constructed by either taking the left or right half of a character, or merging the right half of one character to the left half of the next character in the same word. We evaluate their character likeliness, which is shown in Fig. 1.7. We can see that the 200 characters (blue) show a much higher average character likeliness than the 200 non-characters (red). However, we can also see that some characters show low character likeliness and some non-characters show high character likeliness, because of the ambiguity and complexity underlying the handwriting. For example, the left half of the character "W" is indistinguishable from the character "V" and has a high character likeliness. However, in Fig. 1.7 we always count half of a character as a non-character.

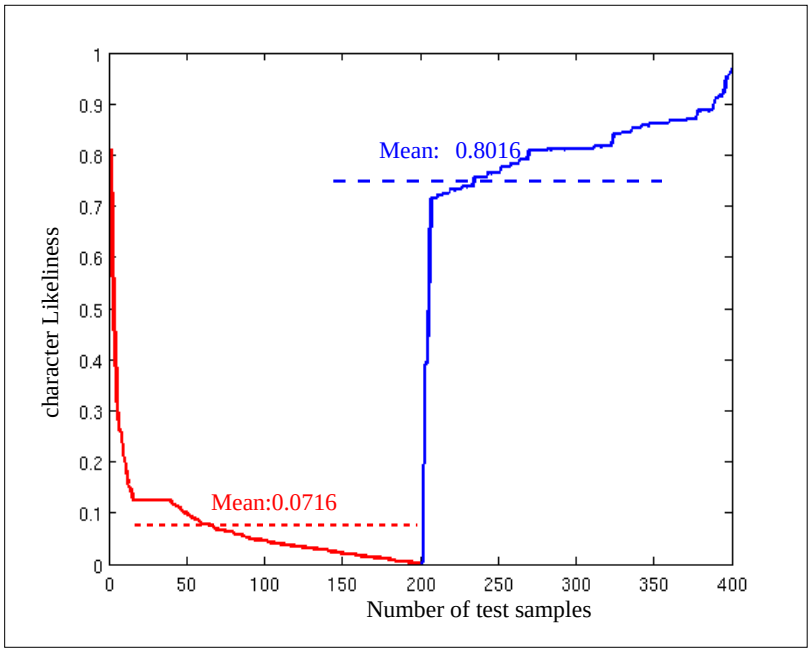


Figure 1.7: Character likeliness calculated for 200 characters and 200 non-characters.

This kind of ambiguity can lead to failures of the presented method in text segmentation. Several examples are shown in Fig. 1.8. In Fig. 1.8(a), the character

"w" is oversegmented into a "n" and a "v", both of which show high character likeliness. In Fig. 1.8(b), two contiguous characters "ub" are not correctly segmented because their combination resembles the character "w" and bears a high character likeliness. Similarly, in Fig. 1.8(c), two contiguous characters "ur" are not correctly segmented because their combination also resembles character "w" and bears a high character likeliness. Such ambiguity cannot be well addressed by considering only the text-image information or the stroke shape.

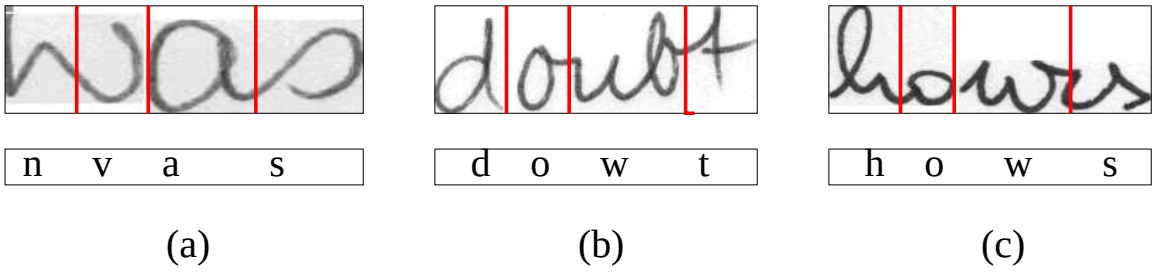


Figure 1.8: Examples where the presented text segmentation method fails.

Dictionary based post-processing extension

The above mentioned ambiguity can be considerably resolved if the domain based lexicon is available. For example if the words being processed are from techincal letters related to a specific problem domain, having such knowledge and lexicon based dictionary, we can post-process the output of the presented algorithm to obtain a better word recognition. We present such a simple post-processing extension to help address the before mentioned ambiguity. Specifically, given the output of the presented method and a domain based lexicon, we compute the Levenshtein distance between the given output and each word in the dictionary. The levenshtein distance is a metric used to measure the difference between any two given sequences. Given two words, the levenshtein distance is measured as the minimum number of single character insertions, deletions or substitutions required to change one word into the

other [79]. The word in the dictionary with the lowest levenshtein distance is chosen as the word correction. In case of a tie in the levenshtein distance, the post-processing extension chooses the character substitution according to the alphabetical order. To

Table 1.2: Word recognition accuracy for the IAM dataset for Microsoft Word (MS-word) based word correction and the presented post-processing extension.

Method	Word recognition Acccuracy
MS-word top-1	0.5667
MS-word top-2	0.6200
MS-word top-3	0.6367
MS-word top-4	0.6467
MS-word top-5	0.6500
Post-processing extension	**0.7633**

demonstrate the effectiveness of this approach, we compare our post-processing extension word recognition accuracy against a commerically available spell-check tool such as Microsoft office. We choose different settings to manually apply the Microsoft office spell check utility. Specifically, we choose the top 1, top 2, top 3, top 4 and top 5 suggested word replacements and replace the output with the suggested word if it exists in any of these words. For our dataset, we created a ground-truth dictionary for the 300 words used, and used it to guide our post-processing extension. As shown in table 1.2, we can see that even after using top 5 word sugggestions from the MS-word spell check utility, our post-processing extension significantly outperforms the MS-word spell check utitlity. This suggests that a domain specific lexicon based dictionary can significantly improve the word recognition accuracy. However some ambiguities caused by the inherent complexity of handwritten text cannot be addressed by even a dictionary based extension. Consider the word 'foot' which is segmented by the presented method as 'fod' because of the merging of letters 'o' and 't' which highly resemble the letter 'd'. Even if we use the given dictionary based extension, the closest word replacement will be 'food' instead ot the actual word 'foot'. Such an ambiguity cannot be resolved by a dictionary based lexicon alone and

requires further natural language processing to resolve it.

Speed

In evaluating the speed of the presented method, we implemented the entire algorithm in Matlab, and run on a 2GHz Linux workstation with 8 GB of RAM. For a text image with roughly 30 candidate segmentation boundaries, the presented method takes about 10 seconds to complete. For the 300 test words, the presented method takes an average of 14 seconds per word for text segmentation.

1.3 Limitations and Future Work

Although the presented method performance can be significantly improved by using a post-processing dictionary based extension, there are certain ambiguities caused by the inherent complexity of handwritten text which cannot be resolved. In our future work, to resolve such ambiguities, we will apply more complex natural language processing techniques to utilize the sentence structure to determine the word replacement. In our future work, we also plan to apply the presented method to the well known 'Captcha' system to determine the words that are present and obfuscated by purpose.

Chapter 2

Distance Transform Based Active Contour Approach for Document Image Rectification

2.1 Method

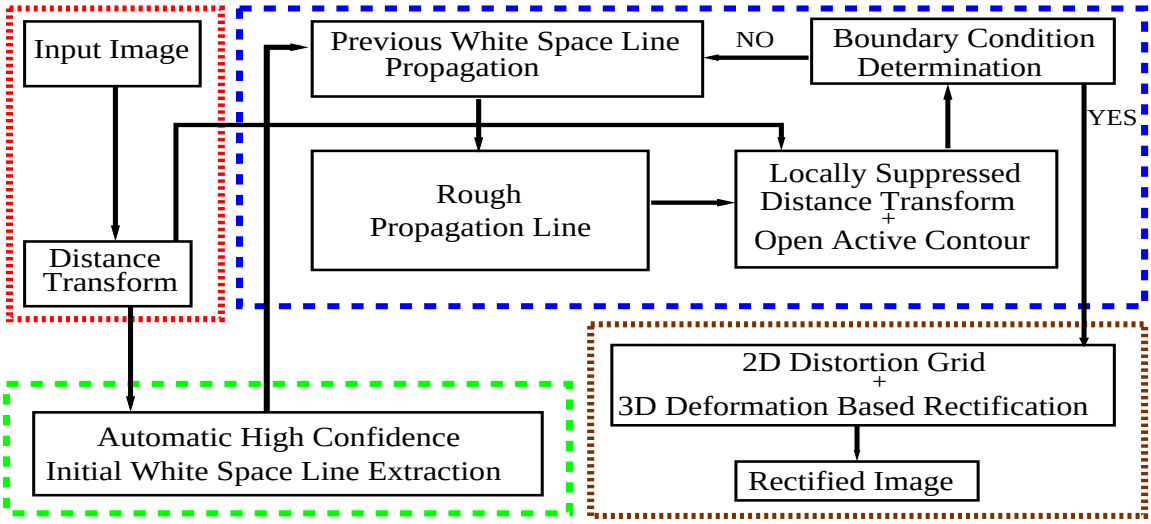


Figure 2.1: System diagram of the presented method.

Our method for document image rectification can be summarized by the pipeline shown in Fig. 2.1 and includes the following steps:

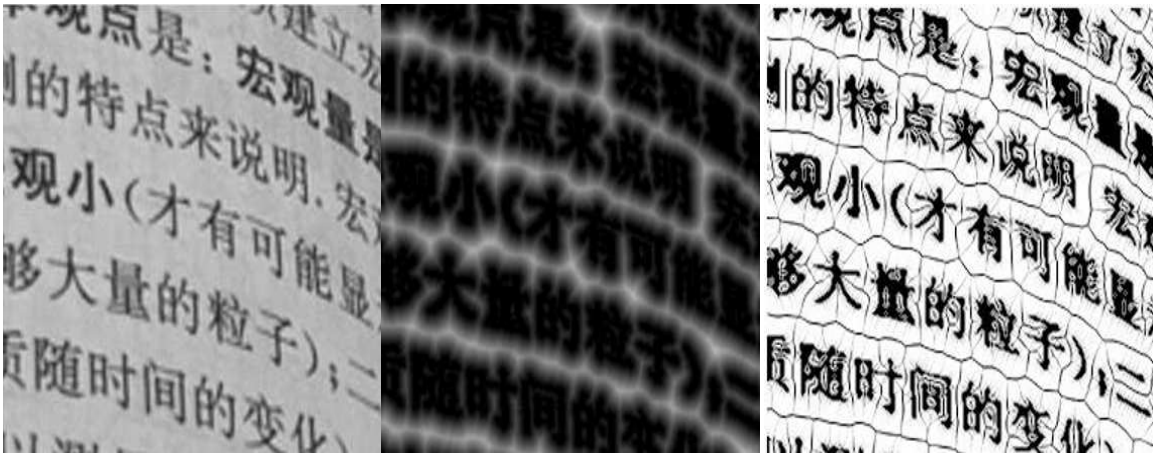
1. Compute the distance transform (DT) for given input image as described in Section 2.1, as shown in Fig. 2.1 with the red dashed box.

2. Extract a high confidence initial white space(WS) line automatically using a bank of Gaussian line filters approach as described in Section 2.1, as shown in Fig. 2.1 in the green dashed box.

3. Use an iterative WS line propagation process (shown in blue dashed box in Fig. 2.1) as described in Section 2.1 and Section 2.1, to extract all WS lines in the document image and refine them using open active contour algorithm.

4. Construct a 2D distortion grid from the extracted WS lines and use it to rectify the input image using a 3D dewarping algorithm as described in Section 2.2, as shown in Fig. 2.1 by the brown dashed box.

Distance transform (DT)



(a) (b) (c)

Figure 2.2: An illustration of distance transform (DT). (a) Original image, (b) DT of the original image and, (c) Gradient map of the DT.

As discussed above, our key intuition is to use the white space between the text lines in the document image, to obtain a better 2D distortion grid. A typical approach to represent this notion of white space is to use a distance transform (DT) of the binarized document image. The binary image is filtered to remove any noise which might affect the DT. The WS line tracing is then formulated as a process of finding maxima in the Euclidean distance transform $D(\mathbf{x})$ from a binarization of the original image $I(\mathbf{x})$, where $\mathbf{x} = (x, y)$ represents the pixel position. The DT can be visualized

as an intensity image, where the intensity at a pixel indicates its distance to the nearest text pixel, as shown in Fig. 2.2 (b). With this DT, it is more convenient to locate a WS line between any two text lines. This is evident from the gradient map of the DT, as shown in Fig. 2.2 (c), as the WS line between two text lines tends to fall on the DT maxima, and appears in the gradient map as an edge between the two text lines.

Automatic extraction of high confidence initial WS line

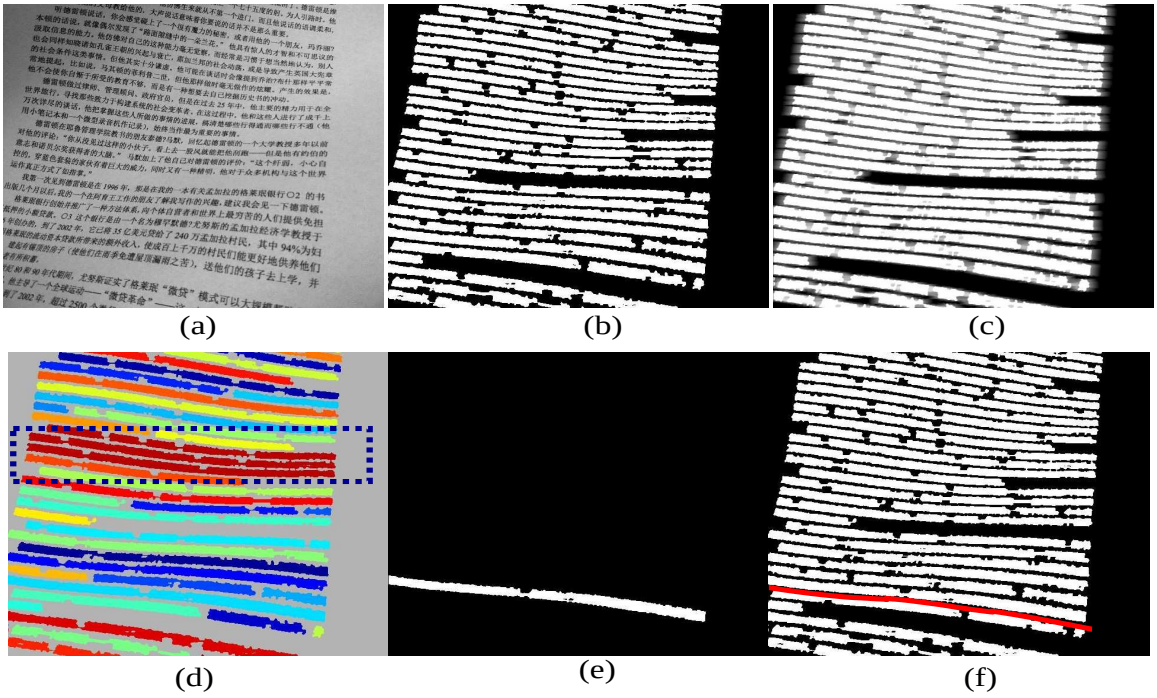


Figure 2.3: An illustration of automatic initial WS line extraction with high confidence. (a) Original text, (b) Binary image accquired by thresholding DT, (c) Output of bank of Gaussian line filters, (d) Connected components result, blue dashed box highlights merging of textual lines in a highly warped region, (e) Largest width connected component, and (f) High confidence initial WS line L_0 shown in red.

A document image (as shown in Fig. 2.3 (a)), usually contains a set of WS lines. In this section, we propose an approach to extract one of these WS lines with high confidence as the initialization L_0 (as shown by the red line in Fig. 2.3 (f)). With this

initial WS line, we can propagate upward and downward iteratively to extract all the WS lines, which will be discussed later in Section 2.1. Specifically, a modified version of the bank of Gaussian line filters algorithm described in [13] is used to extract L_0. First we binarize the DT by a small threshold δ_1, such that pixels with $D(\mathbf{x}) < \delta_1$ result in a binary image as shown in Fig. 2.3 (b). Next the algorithm described in [13] is used to obtain a smooth image as shown in Fig. 2.3 (c). A high threshold δ_2 is used to threshold and binarize the image shown in Fig. 2.3 (c) and connected components analysis is performed on this binarized image, resulting in blobs as shown in Fig. 2.3 (d). The amount of document warping might vary across the page, and may result in the merging of multiple textual lines in a highly warped region into a bigger component after using the algorithm from [13], as shown in Fig. 2.3 (d) by the red component inside the blue dashed box. To address this problem, an average height is calculated using all the connected components and any components with height greater than this average height are discarded. A component with the largest width as shown in Fig. 2.3 (e) is selected from the remaining components to estimate L_0. This width is estimated from the length of the major axis of an ellipse fitted to the component. The pixels at the bottom edge of this connected component are used as a rough estimation $\hat{L}_0$, which is refined into initial WS line L_0 using the open active contour algorithm described further in Section 2.1.

Iterative WS line propagation

Once the high confidence initial WS line L_0 described in Section 2.1 is obtained, it is used to extract the remaining WS lines in the document image using an iterative propagation process, performed in upward and downward directions relative to L_0. Let's consider the downward propagation that extracts the WS line L_1 just below L_0. Given the WS line L_0 as shown by the red curve in Fig. 2.4 (a), each pixel $\mathbf{x} = (x, y)$ on this line is displaced downward such that it is located on a new local DT maxima

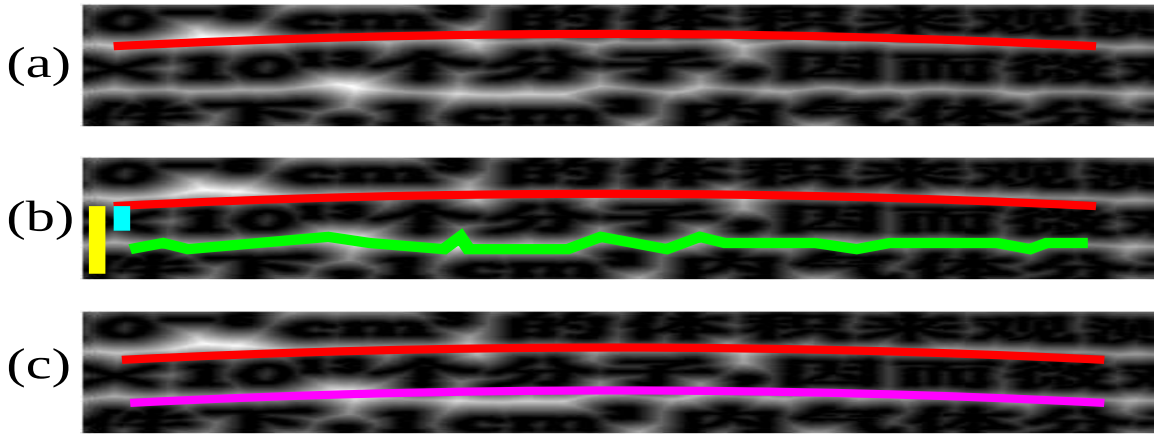


Figure 2.4: An illustration of WS line propagation. (a) Initial WS line L_0 shown by red curve on top of DT, (b) WS line propagation to obtain a rough propagation line $\hat{L}_1$ shown in green curve on top of the DT, with displacement bounds t_1 and t_2 shown by cyan and yellow vertical lines on the left of (b) respectively, and (c) Extracted WS line L_1 shown in magenta on top of DT.

within specific bounds from L_0. These displaced pixels form the rough propagation line $\hat{L}_1$ as shown by the green curve in Fig. 2.4 (b). Specifically, pixel $(x, y) \in L_0$ is displaced to $(x, y^*) \in \hat{L}_1$ where:

$$y^* = y - \arg \max_{\delta_y \in [t_1, t_2]} D\left(x, y - \delta_y\right)$$

t_1 and t_2 are the thresholds that bound the displacement. These thresholds are computed from L_0 by averaging $D(\mathbf{x})$ for each pixel $\mathbf{x}$ along this line as:

$$\bar{D}(L_0) = \frac{1}{|L_0|} \sum_{\mathbf{x}_i \in L_0} D(\mathbf{x}_i)$$

We set $t_1 = 0.6 \cdot \bar{D}(L_0)$ and $t_2 = 1.2 \cdot \bar{D}(L_0)$, as illustrated by cyan and yellow vertical lines in Fig. 2.4 (b). This rough propagation line $\hat{L}_1$ is then refined to the WS line L_1 using the open active contour based refinement process described in Section 2.1. With L_1, we can use the same technique to propagate downward and get the next WS line L_2. This process can be repeated to extract all the WS lines below L_0. Similarly, we can use this propagation technique upwards to extract all the WS lines above L_0.

We evaluate a termination condition after each line propagation. For this termination condition, we compute parallel curves from the last white space line, each separated by a small displacement δ. For each of these generated curves, we compute the Mean Pixel Intensity (MPI) on $I(\mathbf{x})$. A plot of the MPI for these curves demonstrates two particular boundary condition characteristics. The MPI plot for a boundary case as shown in Fig. 2.5(a) has one last valley in the plot before abruptly falling to zero as shown in Fig. 2.5(c). The other boundary condition is shown in Fig. 2.5(b) which has an MPI plot which falls abruptly from the last peak in the plot back down to zero without having a last valley, as shown in Fig. 2.5(d). We terminate our algorithm when we detect these boundary conditions in the MPI plot.

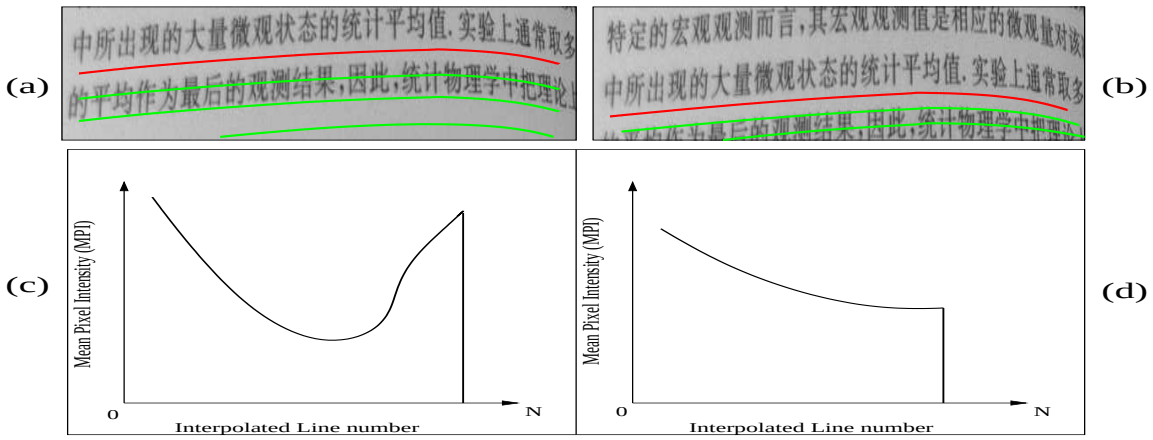


Figure 2.5: (a) Boundary condition where image ends with white space. (b) Boundary condition where the image ends with a text line. The red line indicates the last found white space line and the green lines indicate displaced curves to calculate the Mean Pixel Intersity (MPI). (c) The MPI plot for case shown in (a). (d) The MPI plot for case shown in (b).

Open Active Contour based Line Refinement

Active contours, also known as "*Snakes*," are energy minimizing deformable splines and widely used for computer vision tasks, such as segmentation. Snakes are generally defined by an internal elastic energy term which defines the bending enery of the

snake and an external edge-based energy term, which guides them towards the desired contour. For each of the rough propagated line $\hat{L}_i$ described in Section 2.1, as shown in Fig. 2.4 (b) by the green curve, we apply an open active contour algorithm to remove any small local noise and obtain the smooth WS line L_i as shown in Fig. 2.4(c) by the magenta curve. We use a Gradient Vector Flow (GVF) based open snakes algorithm

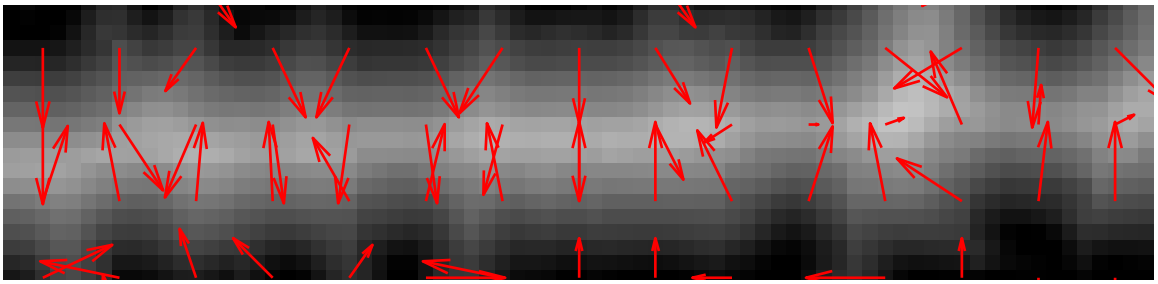


Figure 2.6: An illustration of a Gradient Vector Flow (GVF) field based on the DT.

to obtain the WS line refinement. The DT described in Section 2.1 is used for defining the external energy force, to guide the open snake towards the DT maxima. As illustrated in Fig. 2.6, the gradient vector flow field is directed towards the DT maxima. The external edge energy term is formulated using the DT as follows:

$$E_{edge} = -\left|\nabla D\left(\mathbf{x}\right)\right|^2$$

We still have to address one special case when using the DT and open active contour to find the WS lines, as illustrated in Fig. 2.7 (a) and (b). A short text line will cause extra white space at the end of the text line, as shown by the red dashed box in Fig. 2.7 (a). This introduces an artifical DT maxima that will be misinterpreted by the open snake-based line estimation as the location for the WS line, as illustrated inside the red box in Fig. 2.7 (b). Such an error will introduce an undesired deformation in the constructed 2D distortion grid. To address this problem, a DT suppression is performed near the rough WS propagation line $\hat{L}_i$, before using the open snake to obtain the WS line L_i. To perform this local DT suppression for a given rough WS

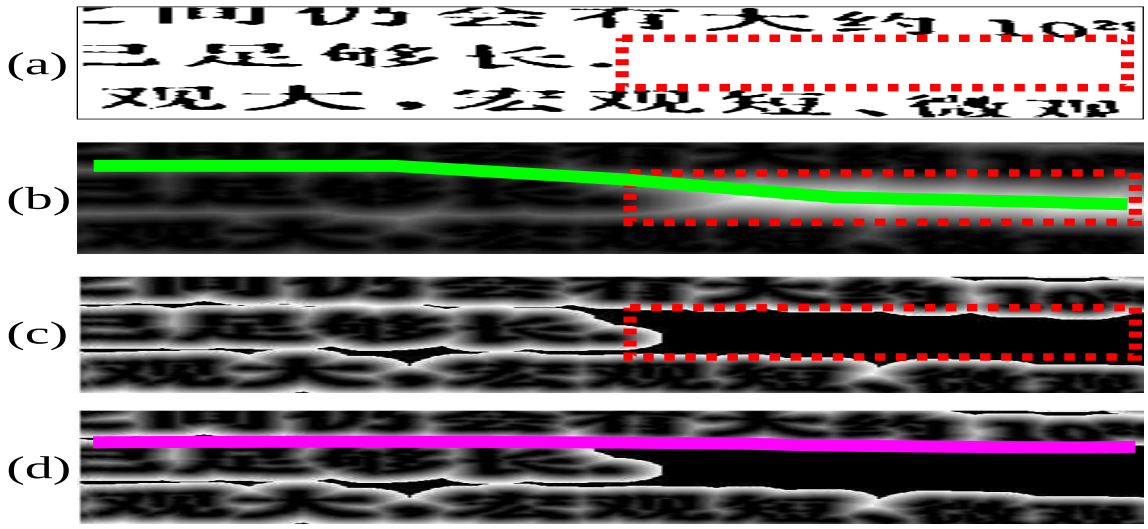


Figure 2.7: An illustration of a special case using the DT. (a) Document image containing a short text line, red dashed box highlights the extra white space at the end of the short line. (b) Open snake attracted towards artificial DT maxima, as shown by green curve on top of DT inside the red dashed box. (c) Supressed DT shown in the red dashed box. (d) Open snake fitting the suppressed DT maxima, resulting in a refined WS line.

propagation line $\hat{L}_i$, we suppress the DT as follows:

$$D(\mathbf{x}) = \begin{cases} 0 & \texttt{if } D(\mathbf{x}) > \bar{D}(L_{i-1}) + \epsilon \\ D(\mathbf{x}) & \texttt{otherwise} \end{cases}$$

where L_{i-1} is the previous WS line and ϵ is a small integer value, which is set empirically to 5 in our experiments. This operation attenuates the artifical DT maxima caused by the extra white space, and we get a suppressed DT as illustrated in Fig. 2.7 (c). The open snake now converges to the suppressed DT maxima and generates the desired refined WS line L_i as illustrated in Fig. 2.7 (d).

2.2 Text Orientation Estimation and 3D Dewarping

We use the 3D dewarping algorithm from [70] for image rectification, which requires not only horizontal (WS) lines, but also vertical lines to form a complete 2D distortion grid. After robustly estimating the WS lines as described in Section 2.1, we use the DT

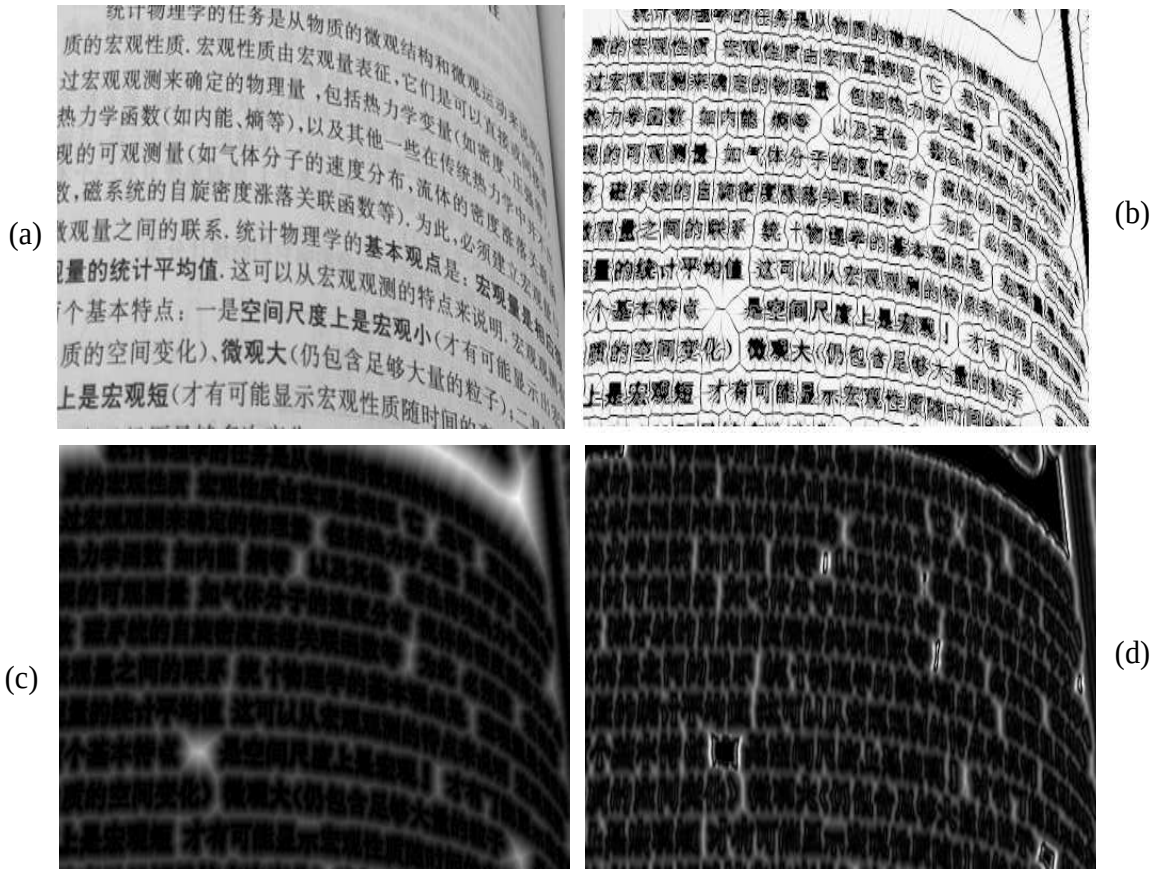


Figure 2.8: An illustration of vertical direction estimation based on the DT. (a) The original image, (b) gradient map of the DT, (c) DT of original image and (d) Gabor filter map of the DT.

to deduce the vertical directions. [70] makes the assumption that alphabets in many languages including English, Chinese etc., contain vertical strokes, e.g., "d","l","k" and "t" in English, and can be utilized to infer the dominant vertical directions in overlapping local regions in the document image. This assumption may not hold true for all languages. Instead the presented method utilizes the DT to infer the dominant vertical directions. As illustrated in Fig. 2.8(b), we can see from the gradient map of the DT that not only does the white space between the textlines contains useful information to estimate the horizontal lines, but also the vertical direction can be infered from the white space between the individual characters irrespective of the language used. Gabor-filter based methods have been applied in a variety of image

processing applications such as texture segemntation, edge detection and recognition of handwritten numerals etc. To infer these local dominant vertical directions we first apply a gabor filter bank to enhance the vertical information contained in the DT. Specifically, we apply a set of vertical orientation biased filters to the DT which is shown in Fig. 2.8(c), which enhance the vertical direction information contained in the DT. We obtain the gabor filtered map as shown in Fig. 2.8(d). Next we use a similar optimization as discussed in [70] to find the dominant vertical directions in overlapping local regions in this gabor filter map. The estimated local vertical dominant directions are illustrated in Fig. 2.9.

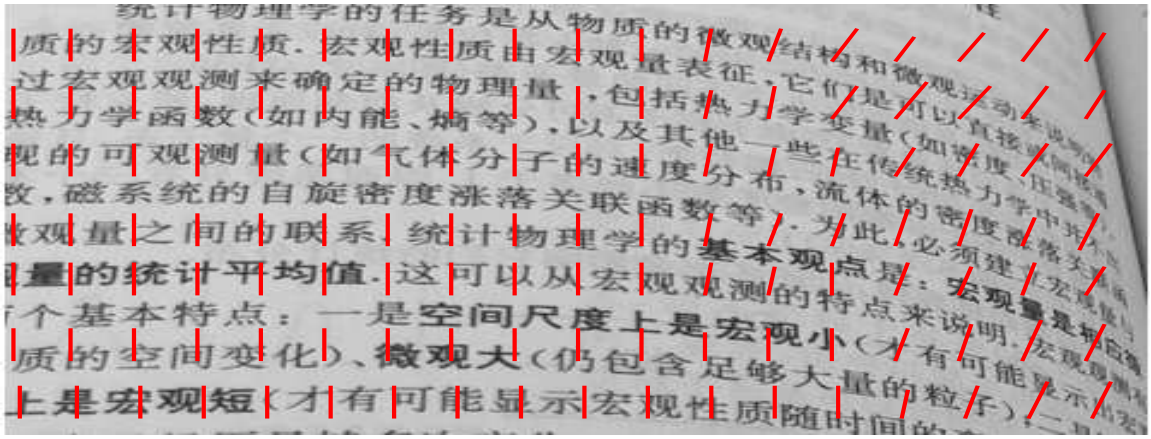


Figure 2.9: An illustration of vertical direction estimation based on the gabor filtered map of the DT.

A 2D distortion grid is created based on these infered vertical directions and the previously extracted WS lines. Next, a 3D deformation of the distorted document is estimated from the 2D distortion grid. It is assumed that the camera projection is perspective, and each cell of the 2D distortion grid is a parallelogram in 3D space. A 3D parallelogram mesh is constructed from the 2D distortion grid. Let $\mathrm{V}_i = (X_i, Y_i, Z_i) = (x_i Z_i, y_i Z_i, Z_i)$ denote the 3D location of i-th grid vertex, where (x_i, y_i) is its 2D coordinates. As shown in Fig. 2.10 (a), The four vertices $V_{1:4}$ form a parallelogram P_j if $\Delta(P_j) \equiv \mathrm{V}_1 + V_3 - V_2 - V_4 = 0$. The following optimization is per-

formed using Singular Value Decomposition to obtain the globally optimal solution for location of each Vertex V_i , as described in [70]:

$$Q\left(\{\mathrm{V}_i\}_{i=1}^{n}\right) \quad = \quad \sum_{j=1}^{N_p} \|\Delta(P_j)\|^2 \quad + \quad \alpha \sum_{i=1}^{n} \left(X_i - x_i Z_i\right)^2 \quad + \quad \left(Y_i - y_i Z_i\right)^2 \quad (2.1)$$

We use this formulation to take the 2D distortion grid created using our WS line esti-

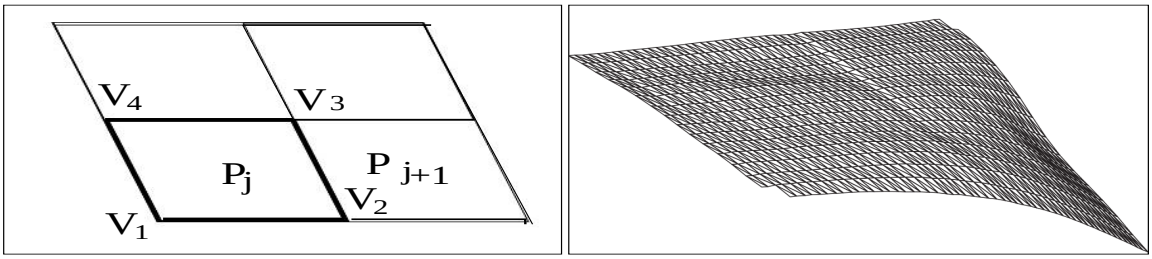


Figure 2.10: An illustration of 3D deformation estimation as described in [70]. (a) Each 2D distortion cell is a parallelogram in 3D space. (b) 3D parallelogram mesh.

mation described in Section 2.1, to create a 3D deformation and rectify the associated document image.

2.3 Experiments

In our experiments, we evaluate three methods, namely:

1. **No-dewarp**: In this method, we use the original image without performing any dewarping.

2. **TL-dewarp**: The text line based dewarping described in [70].

3. **presented-dewarp**: presented WS line based dewarping described in this paper.

For TL-dewarp, we use the code and parameters as used in [70]. We choose a freely available OCR engine (onlineocr.net) to perform OCR on the dewarped images and

use the OCR results for quantitative evaluation of each method disccused above. Three evaluation datasets are collected as follows:

Dataset 1: This dataset consists of 15 English document images from [70].

Dataset 2: This dataset consists of 15 English document images that we collected ourselves.

Dataset 3: This dataset contains 20 international document images, consisting of images in Hindi and Chinese Languages. It contains equal number of documents from [70] and images we captured overselves. This dataset is used only for qualitative evaluation since the OCR used does not work on international document images.

Evaluation criteria: Dataset 1 and Dataset 2 are used for quantitative evaluations using all three methods, for which we use the evaluation method described in [83]. First the ground truth text is created for each image by manually typing the text contained in the image. This evaluation method then uses a text alignment scheme by which it aligns the OCR result for a given method against the ground truth text on the word/character level. Once the text's are aligned, word and character recognition accuracies are estimated. The OCR accuracy metric is defined as $\frac{w}{t}$, where w is the total number of matching words/characters in the alignment and t is the total number of words/characters in the ground truth. We tabulate the results of

Table 2.1: Quantitative evaluation results for No-dewarp, TL-dewarp [70] and presented-dewarp. For each image in Dataset 1 and Dataset 2, we compute the OCR word and character accuracy as described in [83] and average it over the number of images in the respective datasets.

	Dataset 1 (word)	Dataset 1 (character)	Dataset 2 (word)	Dataset 2 (character)
No-dewarp	0.4151	0.6029	0.4398	0.6141
TL-dewarp [70]	0.5460	0.6359	0.5437	0.6406
Prop.-dewarp	**0.6558**	**0.7604**	**0.7001**	**0.8109**

running No-dewarp, TL-dewarp and presented-dewarp on Dataset 1 and Dataset 2 in Table 2.1. In this table, we report the word level and character level OCR accuracy,

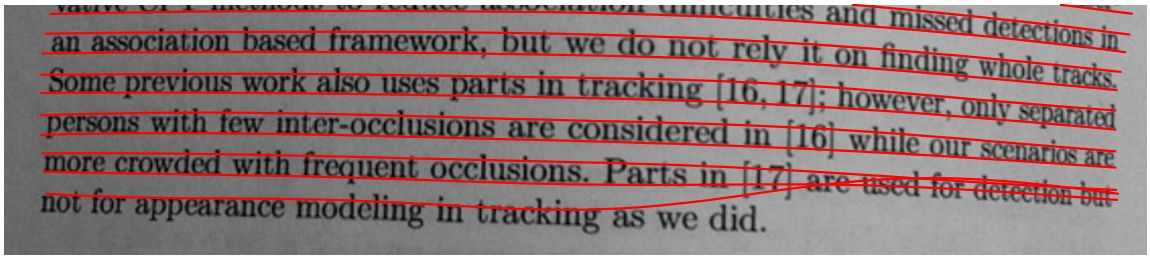
averaged over the respective entire datasets. We can see that the presented-dewarp method performs the best amongst the three methods discussed. This table also demonstrates the effectiveness of using a dewarping algorithm in general to improve the OCR accuracy of a distorted original image. We perform an additional simple experiment using an extra spell-check step. A word processor software (Microsoft Word) is used to spell-check and correct the OCR results. These spell-checked OCR results for No-dewarp, TL-dewarp and presented-dewarp are then evaluated as previously discussed using method described in [83]. We tabulate these results in Table 2.2. The above experiment demonstrates that even after using an OCR with additional

Table 2.2: Quantitative evaluation results for No-dewarp, TL-dewarp [70] and presented-dewarp, using additional spell check (Microsoft Word is used for spell-check). OCR word and character accuracy is calculated for each image in Dataset 1 and Dataset 2 as described in [83], and averaged over the number of images in the respective datasets.

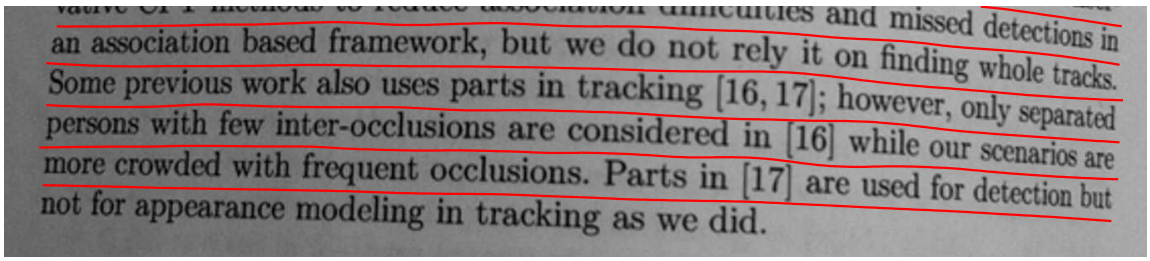
	Dataset 1 (word)	Dataset 1 (character)	Dataset 2 (word)	Dataset 2 (character)
No-dewarp	0.4292	0.6071	0.4486	0.6192
TL-dewarp [70]	0.5585	0.6390	0.5532	0.6427
Prop.-dewarp	**0.6742**	**0.7659**	**0.7201**	**0.8190**

spell-check, presented-dewarp method still outperforms No-dewarp and TL-dewarp.

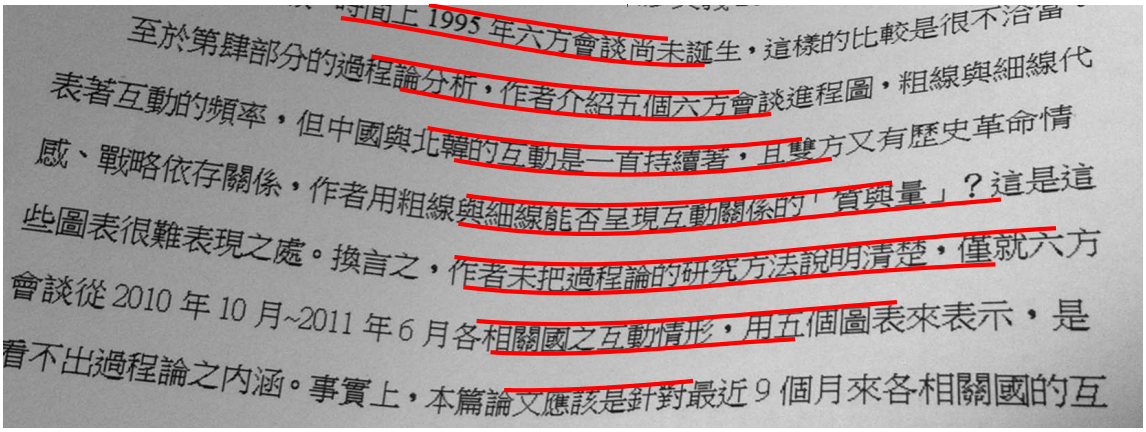
Fig. 2.11 shows some qualitative results where the TL-dewarp method's text line estimation either fails to properly trace the 2D distortion grid lines, or fails to trace enough lines for the 2D distortion grid to cover the image. As mentioned in [70], we varied the step size parameter in a range of 1 to 5, and selected the best possible result. In comparison, presented-dewarp robustly estimates the horizontal WS lines for the 2D distortion grid on the same set of images. We provide more qualitative results in Fig. 2.12 for some select images from all three datasets, where we show the original image plotted with text lines from TL-dewarp, dewarping result for TL-dewarp, original image plotted with WS lines for presented-dewarp, and de-



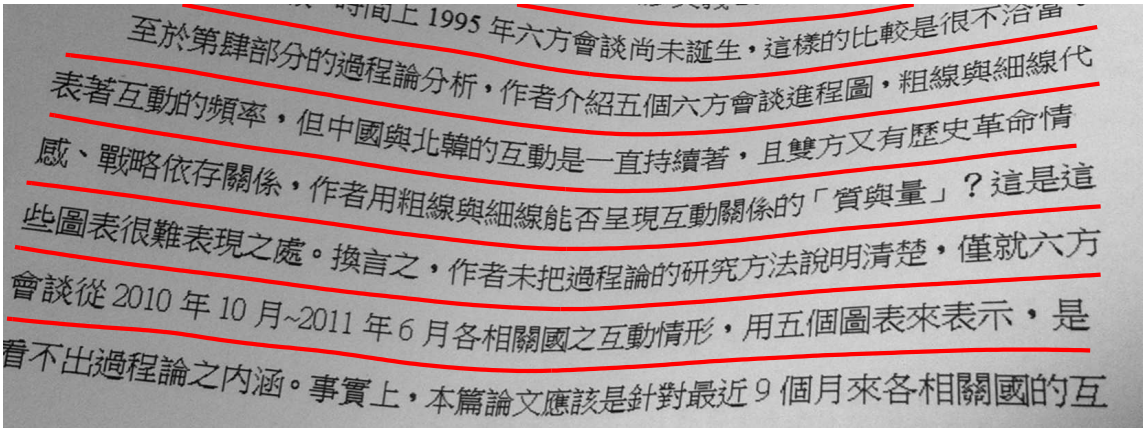
(a)



(b)



(c)



(d)

Figure 2.11: An illustration of the failure cases of [70]. The first and third row show the failure case for TL-dewarp. The second and fourth row show the result for presented-dewarp for the same set of images from Dataset 2.

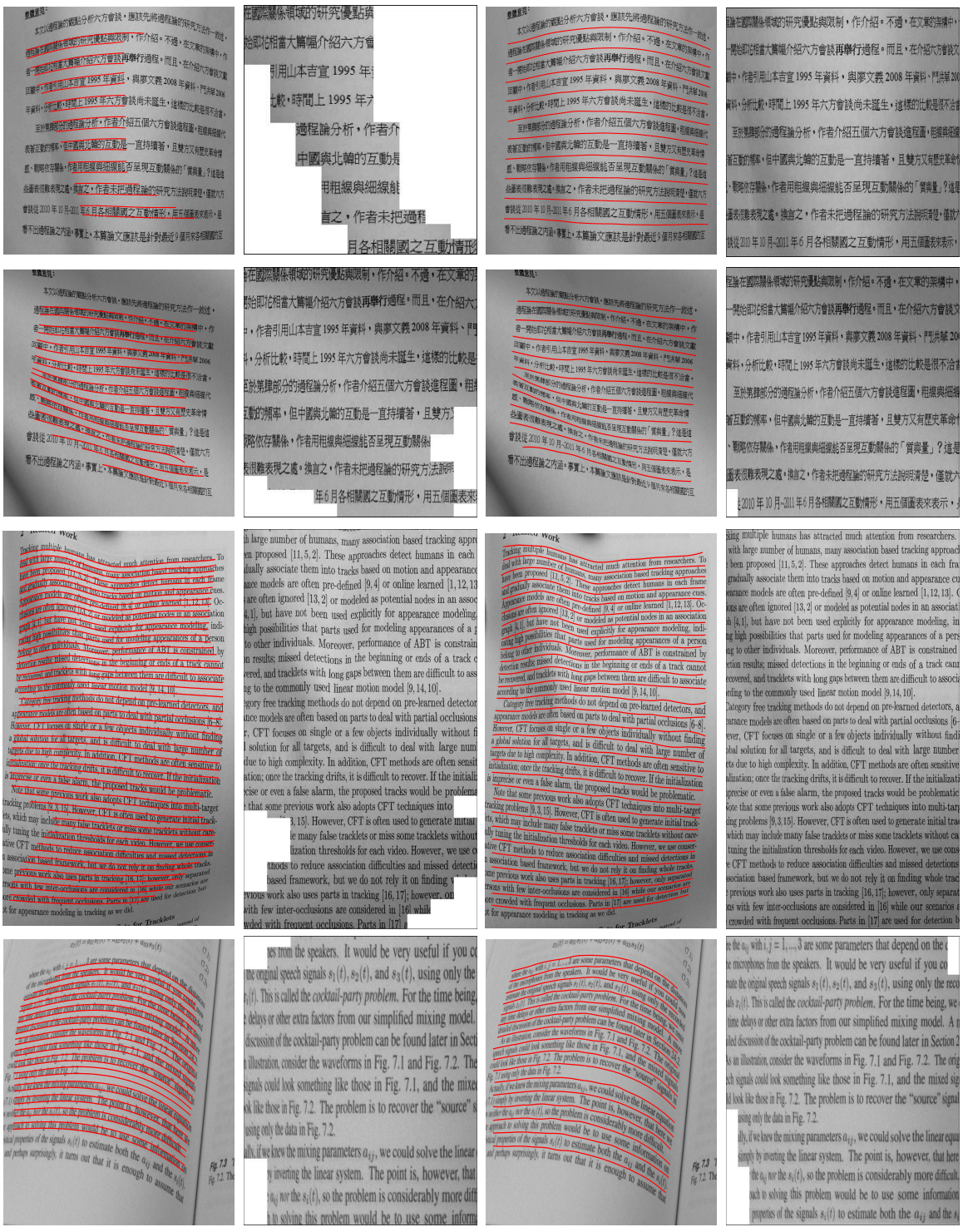


Figure 2.12: An illustration of qualitative results for TL-dewarp and presented-dewarp. The original distorted images plotted with TL-dewarp text lines are shown in the first column. The rectification results for TL-dewarp are shown in the second column. The original distorted images plotted with presented-dewarp WS lines are shown in the third column. The rectification results for presented-dewarp are shown in the fourth column.

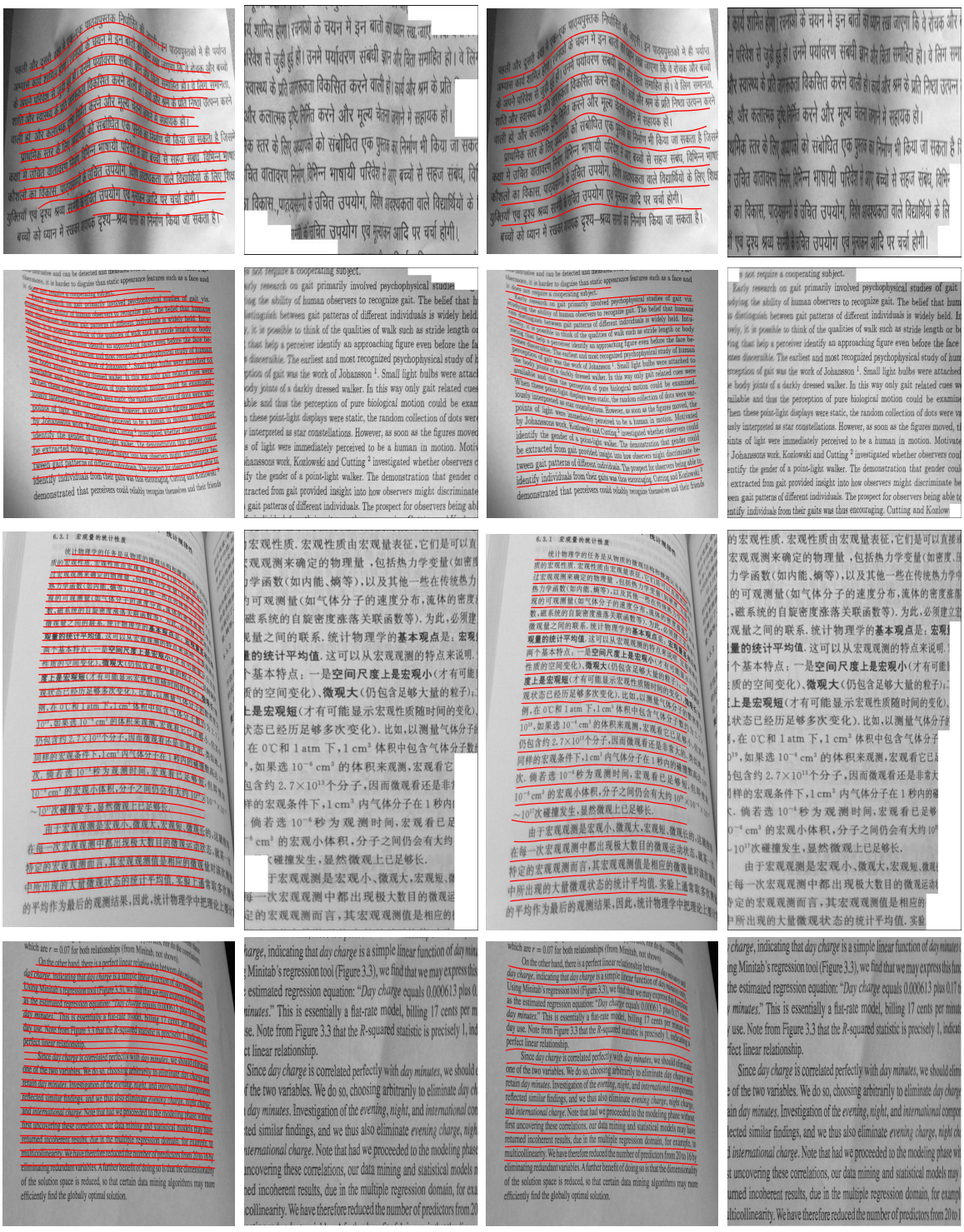


Figure 2.13: An illustration of qualitative results for TL-dewarp and presented-dewarp. The original distorted images plotted with TL-dewarp text lines are shown in the first column. The rectification results for TL-dewarp are shown in the second column. The original distorted images plotted with presented-dewarp WS lines are shown in the third column. The rectification results for presented-dewarp are shown in the fourth column.

warping result for presented-dewarp. We can see from these qualitative results that the presented-dewarp is able to locate more meaningful horizontal 2D distortion grid lines than TL-dewarp, and gives a better dewarping result.

Bank of Gaussian line filters parameters: For the bank of Gaussian line filters described in 2.1, we empirically selected range of values for $\theta = \pm 10^{\circ}$ and l in range of 5 to 200.

Snake parameters: For the GVF snakes in our experiments, we used $\alpha = 0.5$, $\beta = 1$ and $\tau = 0.5$ for all the snakes with the number of iterations fixed at 60.

Performance: The code for presented method is implemented in MATLAB and is not optimized for achieving best runtime performance. Average run time for presented-dewarp is similar to TL-dewarp [70] and both take roughly 2-3 minutes for an image of 2592×1936 pixel resolution. All experiments are performed on a quad-core Intel 3.0 Ghz machine with 8G memory.

2.4 Limitations and Future Work

The presented method currently cannot handle document images with tables and embedded figures. In future work, we will formulate an approach to utilize the same initutive white space approach to estimate the distortion grid taking into account these tables and figures which might exist in the document image.

Chapter 3

Digital Collation

3.1 Method

In this work we formulate the collation problem as 2D image registration–a classic computational problem in the field of computer vision. Repeating our registration in a pairwise fashion, we transform an ensemble of witnesses(copies) to a single coordinate system. The process is made more complex because we seek to register digital photographs rather than flattened, idealized copies of a work. These photographs may display the documents under different lighting conditions, with varying levels of warping near the spine of the book, and with differing margins around the text area, among other inconsistencies that are unimportant to humanists studying the work. We handle this large variation among photographs by employing novel techniques for general feature detection and matching, while simultaneously introducing new constraints on the matching problem specific to this domain to yield a registration that is robust against variations in the source data.

Collation in the humanities domain roughly corresponds to a comparison of registered images. This approximation, however, fails to capture the nuances in creating a utility that is of benefit to humanists working with collated documents. Because we seek to create a tool for humanists to identify appropriate differences, we err on the side of finding all appropriate differences, even when it generates false positives. Since any reduction in the search space of the document means that a human would no longer be required to examine every character in the document, the necessity of

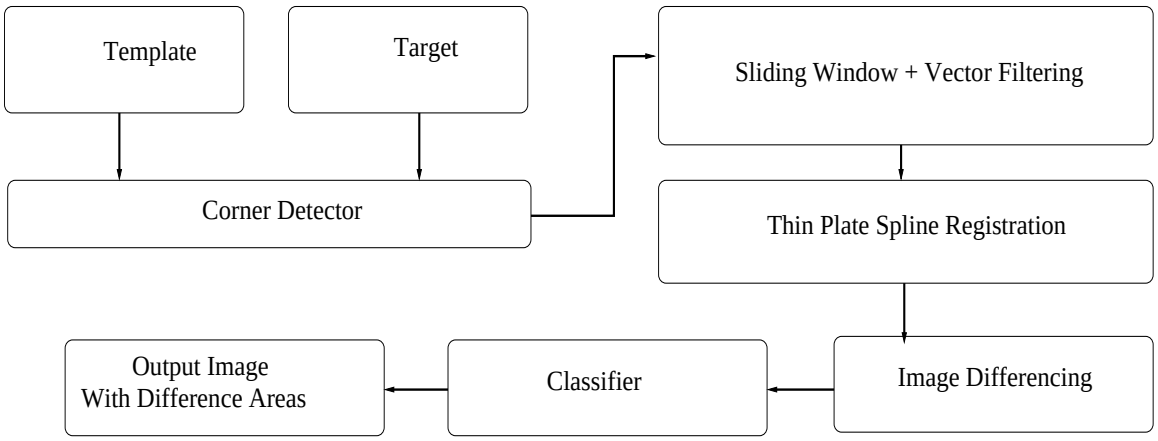


Figure 3.1: System diagram of the digital collation method.

examining both true and false positives is no more of a burden than it would be to examine the entirety of the document.

We expand the scope of our input to include a large variety of very noisy input data. Unlike other methods [74], we choose to handle input documents that are warped in non-affine ways. In addition, we seek to unify the way text and images are handled, so differences between both can be analyzed seamlessly and simultaneously. Finally, ancient documents can be highly degraded and may show intensity differences between witnesses due to age or the imaging process, and we do not want these factors to influence the collation. As shown in Fig. 3.1, the proposed method consists of several components.

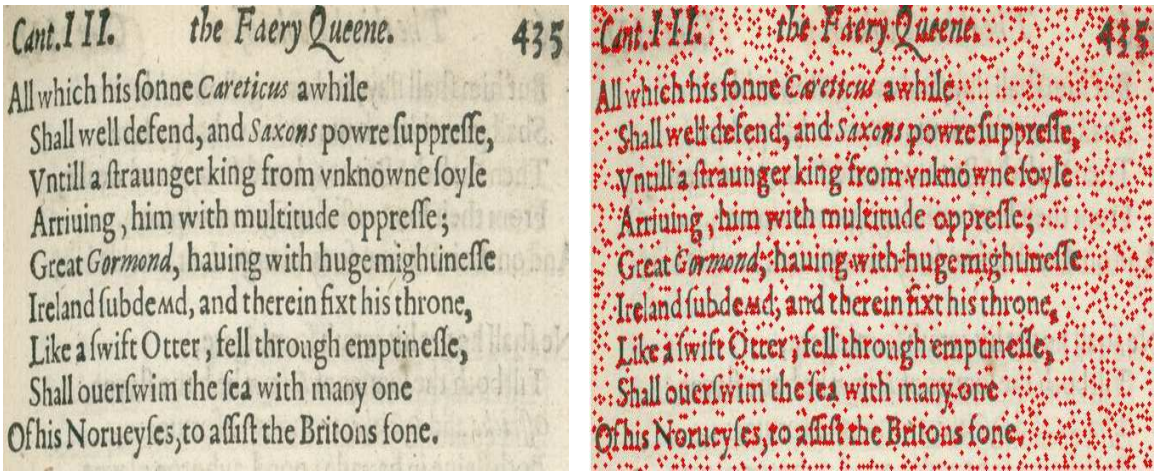
1. Run corner detector on both the template and target image as in Fig. 3.2.

2. Run Sliding window plus vector filtering algorithm on the output of corner detection, as shown in Fig. 3.3.

3. Apply thin plate spline algorithm on the features obtained from previous step.

4. Perform image differencing operation, as shown in Fig. 3.6.

5. Run classifier to identify true positive areas of differences, as shown in Fig. 3.7.



(a) (b)

Figure 3.2: An illustration of the application of Harris corner detector on (a) the template and (b) the corner detector output. The red dots indicate the corner points estimated by the detector.

Feature extraction using corner detector

For a pair of copies to be collated, one of the pages is selected arbitrarily as the *template*, which will not be warped, and the other as the *target*, which will be warped to match the template. To initiate the digital collation pipeline, we first need to estimate the feature points common between both template and target, and use these feature points to warp the target.

Normally the state-of-the-art SIFT feature detector [42] designed for use with natural images is applied to obtain the feature points. However, based on empirical data, we find that this feature detector is not suitable for digital collation, as it tends to produce sparse feature points across the image. Therefore, we present a novel patch-based feature detector which is particularly suitable for digital collation. Interest point detection is a well researched topic in the field of computer vision and refers to the detection of feature points which are required for subsequent processing. Corner detector is a type of interest point detector, where the primary goal is to obtain robust well defined image features. As the first step in the digital pipeline, we

use a Harris corner detector [32] to detect dense corner points on the template image as shown in Fig. 3.2.

Feature matching using sliding window and feature vector filtering

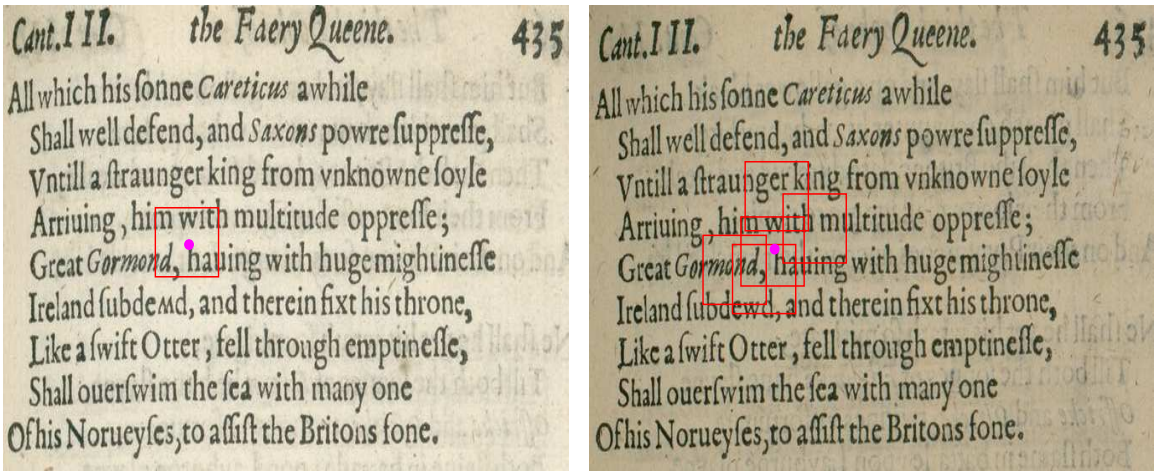


Figure 3.3: A sliding window approach to match the feature points obtained by the corner detector.

The previous step estimates a dense sampling of feature points on the template image. Next for a successful subsequent registration step in the digital collation, we need to estimate a corresponding point in the target image for each of these feature points in the template image. We present a novel sliding window based approach to obtain the desired correspondence. Specifically, for each corner point in the template image, we use a patch centered at this point and find a corresponding matching patch in the target image. To look for the best matching patch, we use a sliding window approach to search inside the target image. We displace the patch window centered at the chosen template feature point, approximately at the same point coordinates in the target image. We can make this assumption since we pre-process the template and target images to approximately align in the same feature coordinate space. To

compute the patch similarity we use a well known similarty measure known as Jaccard index. Based on this similarity measure we compute the nearest matching patch in the target image and use the center of this patch as the corresponded target feature point as illustrated in Fig. 3.3. These corresponded points do contain bad matches

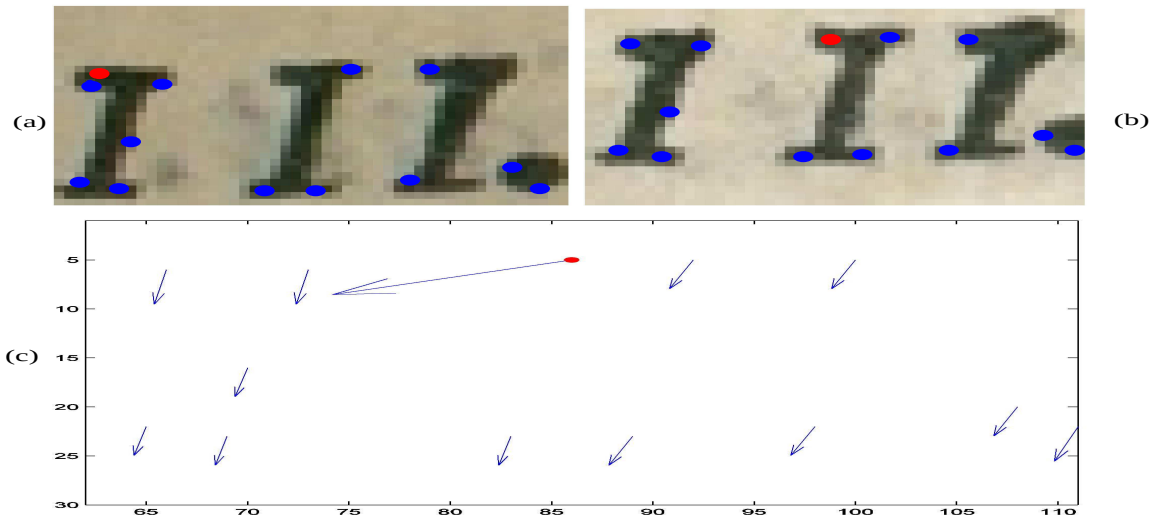


Figure 3.4: (a) Feature points on Template, (b) Feature points on Target and (c) The vector field calculated based on feature point correspondence.

which must be filtered out before performing the registration process. We present a simple vector filtering step for this pruning process. In this process we compute a distance vector by between each corresponded template and target feature point by subtracting the template point coordinates from target point coordinates. Thereafter, we perform a consistency check on each such vector by examining whether the vector is consistent with its neighboring vectors. To do this, inside a small window, we check the length and the orientation of each vector, comparing it with other vectors inside the window. We set a threshold for the vector length and orientation difference based on the average vector lengths and orientations in this window. If the length difference or the orientation difference is above the threshold, we consider current vector as an outlier, and therefore prune out the matching points associated with this vector. As illustrated in 3.4(a) and (b), one such outlier pair is indicated by a red

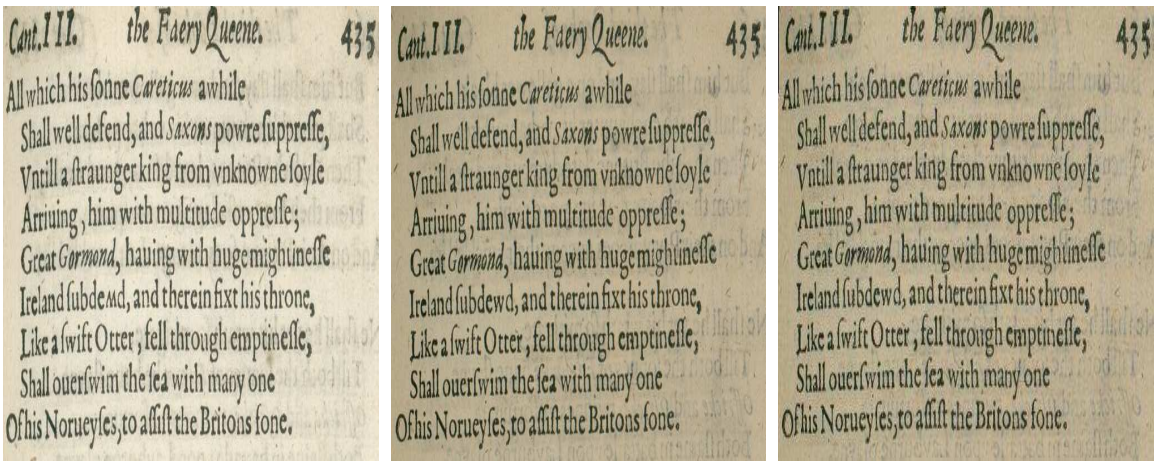


Figure 3.5: Illustration of Image registration using thin plate spline. (a) Template image, (b) Target image and (c) Registered image.

dot in the target and template image. The remaining matching feature points are considered as correct. The vector lengths and orientations for each of these matching points are shown in Fig. 3.4(c). We can clearly see that the vector associated with the outlier pair of matching points is inconsistent in length and orientation from neighboring vectors. So we can safely consider this matching pair of points in the template and target as an outlier and discard them. We perform this process in fixed size overlapping windows within the entire template image and obtain a robust and consisten matching pair of feature points. Next we use these matched feature points to warp the target into the template by performing image registration using thin plate spline.

Image registration using thin plate spline

To allow non-affine document warping to better characterize the registration between pages and account for any warping of the pages, we employ a 2D thin plate spline transform [8] using the matches we have already obtained. The feature matching described in Section 3.1 is strictly pairwise, so we approach registration in a

similar manner. We fix one page as the template page U and, pairwise, set each page to be registered to the template as the target page V. The features are then $U_F = \{\mathbf{u}_1, \ldots \mathbf{u}_i, \ldots \mathbf{u}_n\}$ and $V_F = \{\mathbf{v}_1, \ldots \mathbf{v}_i, \ldots \mathbf{v}_n\}$ where all $\mathbf{u}_i \leftrightarrow \mathbf{v}_i$ correspond to matching feature points, and $\mathbf{u}_i = (u_{ix}, u_{iy})$ and $\mathbf{v}_i = (v_{ix}, v_{iy})$. The newly computed point $\hat{\mathbf{v}}_i$ has been shown [8] to be given by

$$\hat{\mathbf{v}}_i = a_1 + a_2\hat{v}_{ix} + a_3\hat{v}_{iy} + \sum_{j=1}^{n} w_j\, U(|(u_{xj}, u_{yj}) - (x - y)|) \tag{3.1}$$

where the weights $a_1, a_2, a_3, w_{1\ldots n}$ can be computed from

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w_{1\ldots n} \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \\ \mathbf{0} \end{bmatrix}$$

given that

$$D(r) = -r^2 \log r^2,$$

$$r_{ij} = |\mathbf{v}_i - \mathbf{v}_j|,$$

$$K = \begin{bmatrix} 0 & D(r_{12}) & \cdots & D(r_{1n}) \\ D(r_{21}) & 0 & \cdots & D(r_{2n}) \\ \cdots & \cdots & \cdots & \cdots \\ D(r_{n1}) & D(r_{n2}) & \cdots & 0 \end{bmatrix},$$

and

$$P = \begin{bmatrix} 1 & u_{1x} & u_{1y} \\ 1 & u_{2x} & u_{2y} \\ \cdots & \cdots & \cdots \\ 1 & u_{nx} & u_{ny} \end{bmatrix}.$$

After computing the requisite components, we use Equation 3.1 to compute new values from V for the transformed $\hat{V}$, which is possible for even non-control points

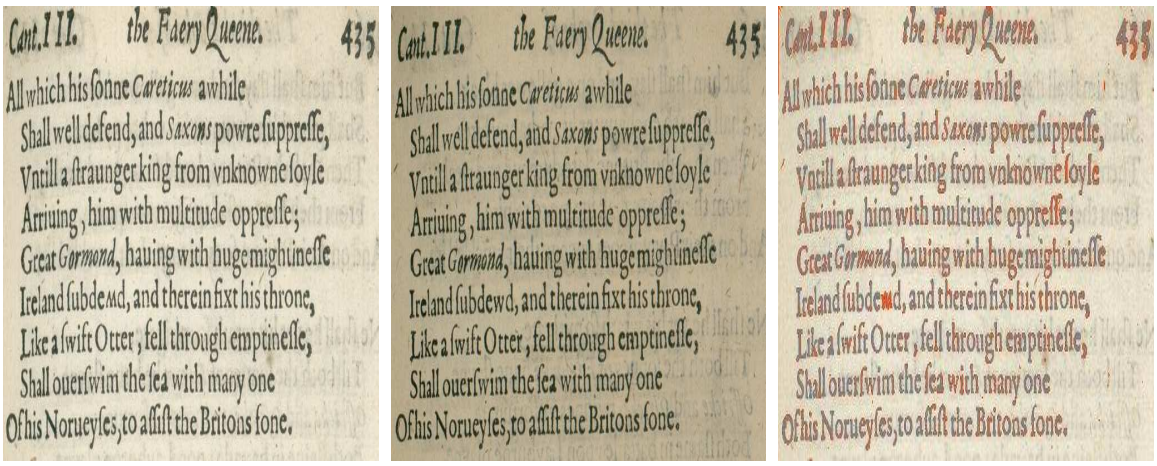


Figure 3.6: Collation of two pages, visualized as an overlay.

because the thin plate spline is an interpolating transform. Points in $\hat{V}$ mapped from outside the boundary of V are simply set to zero. Figure 3.5 shows that warped target which we call the registered image.

Difference area classification

Next, to locate the significant differences, we first convert the template image and registered image to binary images. We perform an XOR operation on these binary images to obtain a difference image. This difference image can be overlayed on the template and visualized as shown in Fig. 3.6(c). For all the pixels valued one in this difference image, we find the connected component that these pixels belong to, i.e., the character or the pattern in the original template image or registered target image. We compare these accquired patches from template image and registered image using three similarity measures, namely (a) structural similarity index (SSIM), (b) Jaccard index and (c) Image correlation, to get three scores. The SSIM similarity score is computed based on three terms: luminance, contrast and structure. The resulting value is between 0 and 1. Higher score is associated with more similarity. Jaccard index, also known as Jaccard coefficient, is a statistic used for comparing the

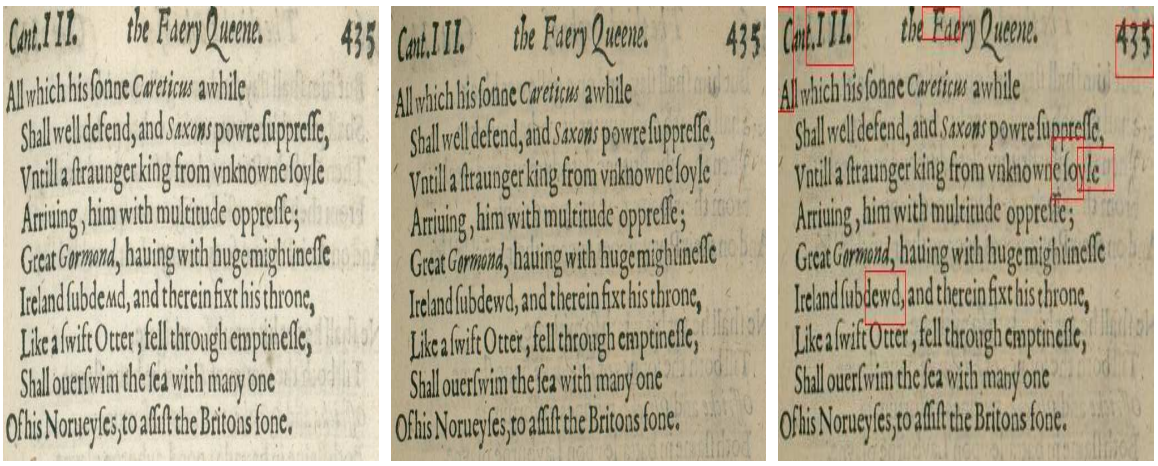


Figure 3.7: An illustration of the areas of significant differences found by the classifier. Colored bounding boxes represent the areas of significant differences overlayed on top of the template image in the third column.

similarity and diversity of sample sets. The Jaccard coefficient is also in the range between 0 and 1. The Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the two sample sets. Image correlation measures the correlation coefficient between two images. The value is between -1 and 1, where -1 and 1 indicate full correlation and 0 indicates non-correlation. We use these similarity measures to distinguish significant differences from the insignificant ones. If any two of these scores are higher than an empirically predefined threshold, we classify the difference as significant. These significant differences can be then overlayed on top of the Template image as colored bounding boxes so as to facilitate experts to analyze them further.

3.2 Implementation

We implement all the components showin in Fig. 3.1 using a client/server architecture which allows us to meet the different interaction needs of the expert users. All the algorithms are implemented in Python/C++ using NumPy/SciPy and OpenCV

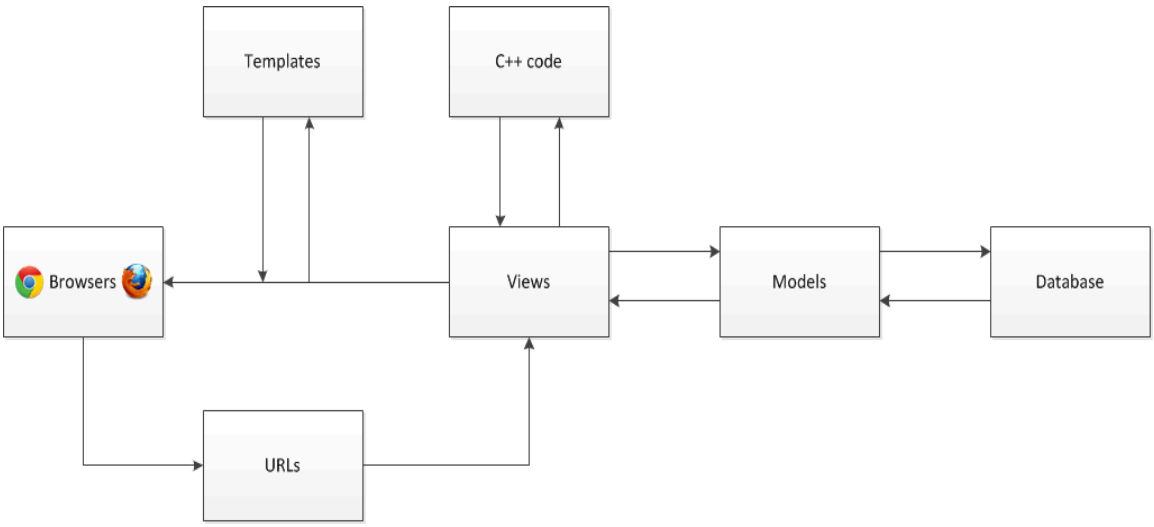


Figure 3.8: Django web framework for Paragon. Client side consists of the Web browser running embedded JavaScript code. Server side consists of the C++ code modules and Database. The Django M-V-C framework consists of the Views, Model, URLs and Templates which provide the required functionality between the client side and the server side.

libraries. We design the Digital Collation user interface (UI) as an interactive interface, built as a web application using the Django web framework as shown in Fig. 3.8. In this framework, the client side consists of a custom single page web application embedded with JavaScript code running in a web browser which is dynamically updated. The server side consists of all the Python and C++ modules running the algorithms shown in Fig. 3.1 along with a database to store the required document images. The Django framework is a M-V-C type of framework which is well suited for developing a single page web app and consists of View, Model, URLs and Templates which provide the required functionality between the client side and the server side.

Interface

As shown in Fig. 3.9, the client allows the user to interact with different functionalities provided by the Digital Collation UI as following:

1. A login interface to register for using the digital collation software. This al-

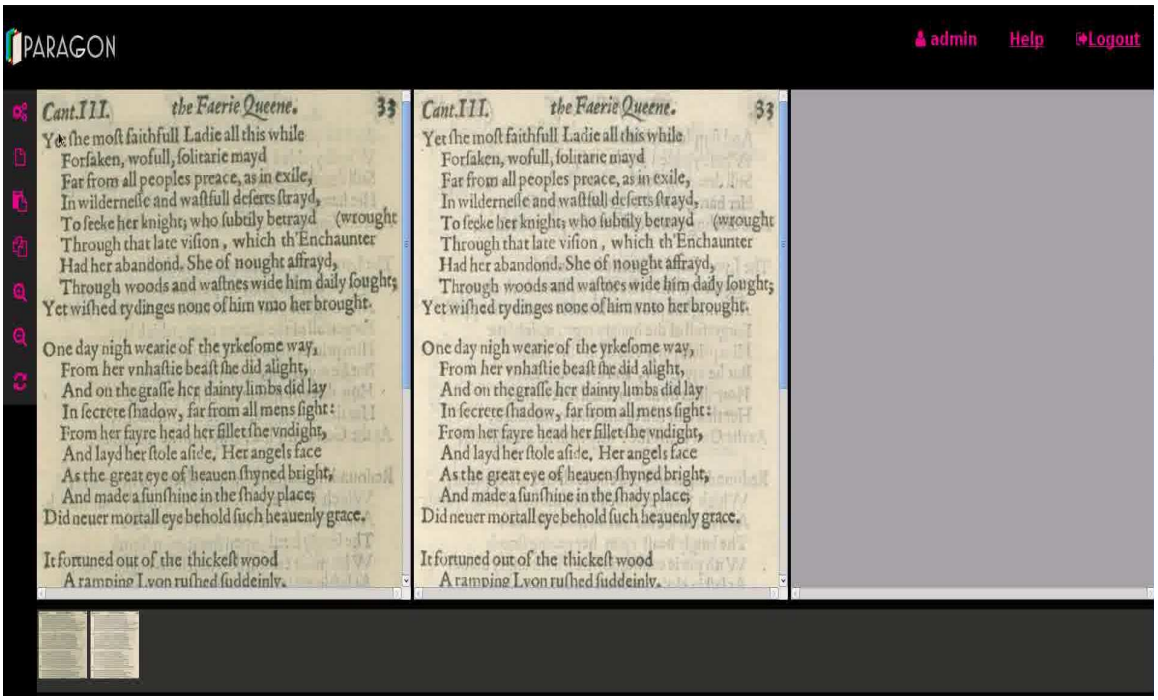


Figure 3.9: An illustration of the web based User Interface for Digital Collation. Here the user has the option to upload multiple template and target images and select as needed. The Template is displayed in the first panel, the target in the second panel and the registered/classifier output image in third panel.

lows multiple users to interact seamlessly and simultaenously with the Digital Collation software.

2. Provides users the options of loading multiple images at a time and selecting any of them as the target and template for registration and collation.

3. The algorithms of the patch-based matching and TPS registration are abstracted from the user and are presented as single-click buttons on the GUI.

4. The user also has an option to analyze the matching/registration results and to manually correct the matching errors as illustrated in Fig. 3.10.

5. Zoom functionality to inspect the results in fine grained detail by zooming in or zooming out the images.

6. Reselect target and template and redo the collation process.

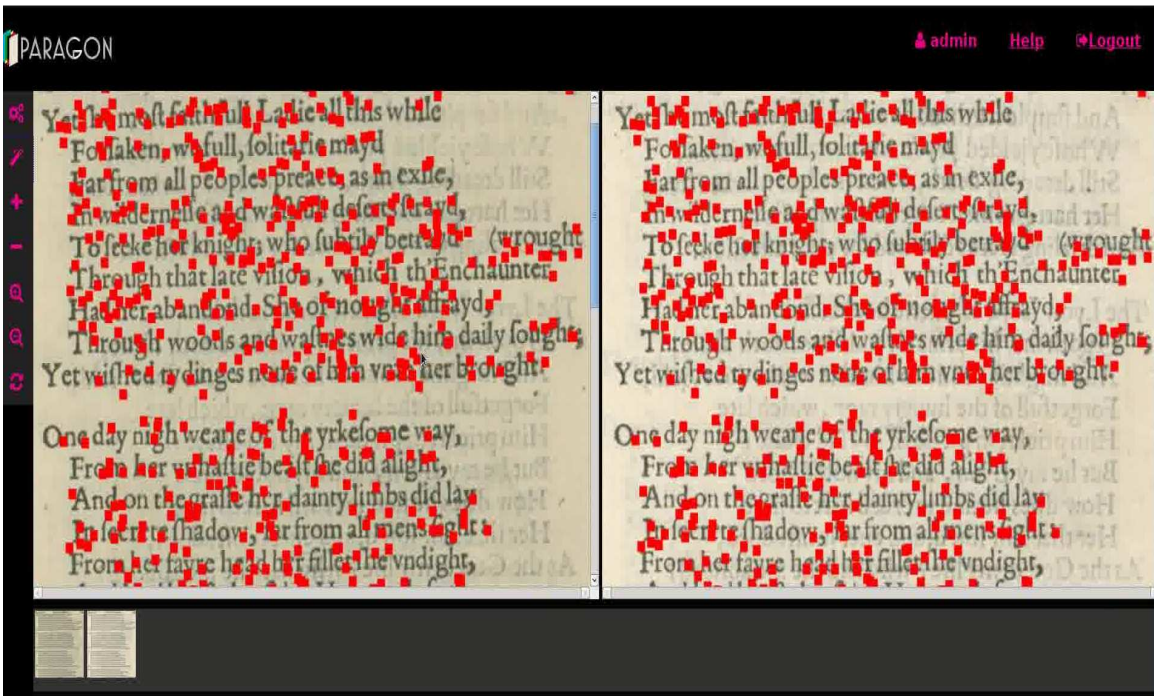


Figure 3.10: An illustration of feature point selection and add/delete functionality provided by digital collation UI. The user can interact with the UI and manually add/delete feature points if needed.

7. Display the results in a manner in which the user can inspect the template,target and collation result side by side as shown in Fig. 3.11 and Fig. 3.12.

3.3 Experiments and results

Collation is often based on subjective inspection by experts, therefore we perform only qualitative evaluations on following three datasets:

Dataset 1: Variorum Gatsby. Original materials furnished by Dr. James L. W. West, Pennsylvania State University, and digitized by the University Libraries Digital Collections Department at the University of South Carolina. This dataset contains 9 prints of The Great Gatsby spanning the years 1925 to 1974. Each book is scanned into 200 high-resolution (2000x2000) images without any warping.

Dataset 2: English Broadside Ballad Archive (EBBA). Digital images furnished

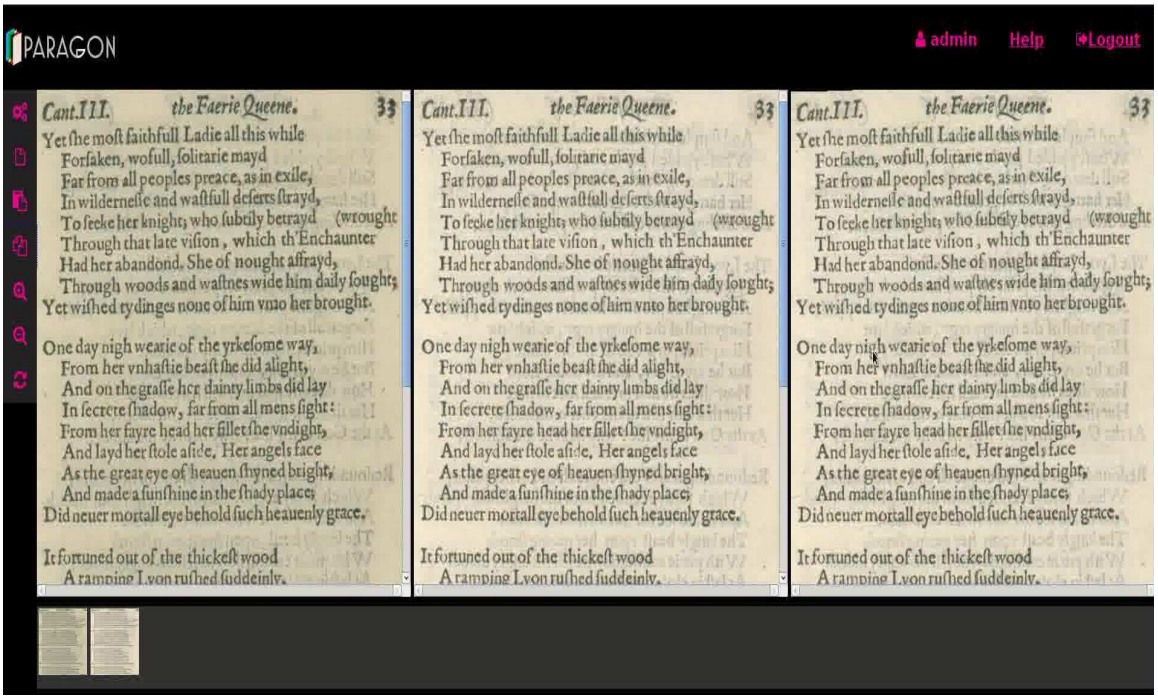


Figure 3.11: An illustration of the registered image shown in the digital collation UI. The registered image is shown in the third panel.

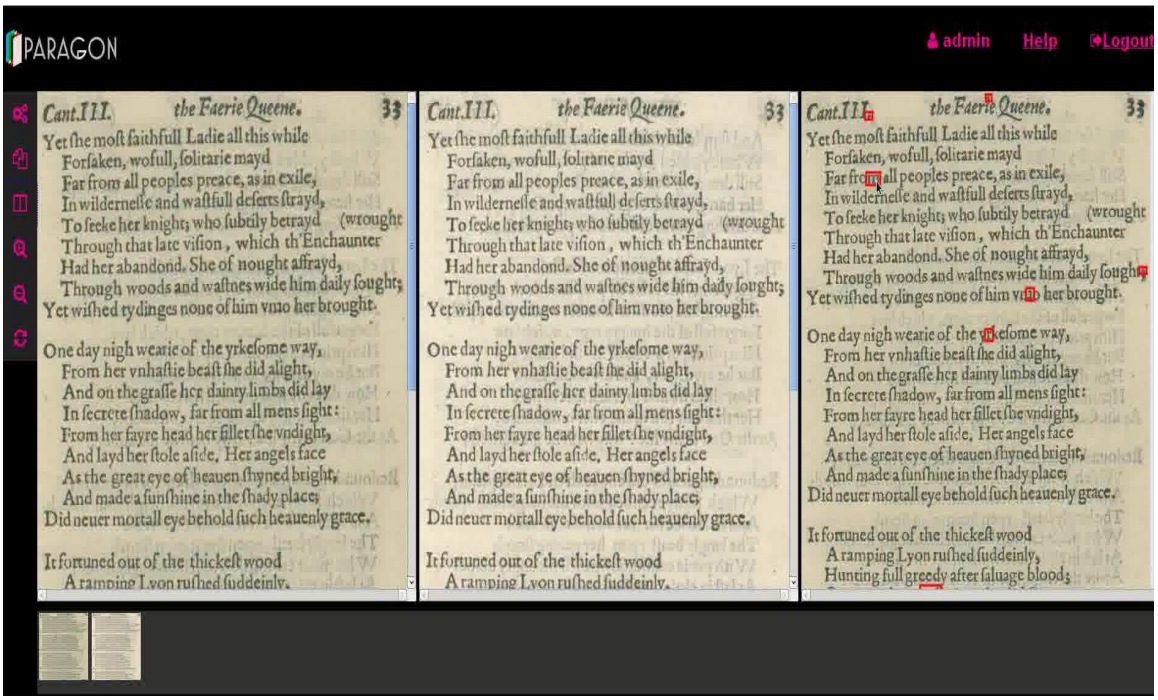


Figure 3.12: An illustration of the classifier ouput shown as colored bounding boxes on top of the template in the third panel. These bounding boxes highlight the areas that an expert can focus on to look for differences.

by Eric Nebeker, University of California, Santa Barbara. This dataset contains three subsets of high-resolution (¿2000x2000) images: close prints, damaged prints and exact prints. Close-print subset contains 10 different broadsides with 2-3 pages in each. Damaged-print subset contains 90 broadsides with 2 pages in each. Exact-print subset contains 35 broadsides with 2 pages in each. This dataset contains various degrees of warping in each subset.

Dataset 3: The Fairie Queene, images furnished by the Spenser project at Washington University in St. Louis and the University of South Carolina. This dataset contains 14 books with approximately 600 pages each. This dataset features considerable warping due to page curvature at the gutter of each opening. The size of each image is approximately 1600x1600.

We show some of the qualitative results for one of these datasets in Fig. 3.13, Fig. 3.14, Fig. 3.15, Fig. 3.16 and Fig. 3.17. As shown in these images, the digital collation highlights specific parts of the images that demand attention from the experts. This partial automation of the inspection process of collated documents can expedite the annotation of large works as compared to full manual inspection using physical optical devices. Because we seek to discover all the meaningful variations, we err on the side of reporting false positives. Sorting of the true positives from false positives is a far less time consuming process than to examine an entire document. In Fig. 3.13, the word "Dronke" in the template is shifted towards the left margin and is correctly recognized by the digital collation as the area of difference. This shift could be either intentional or by mistake, and should be categorized as either by an expert. In Fig. 3.14, the template contains the word "all", while the target contains the word "eche" instead. This is correctly identified by digital collation. Fig. 3.15 shows an example, where the character "f" from the word "falfe" is missing in the template and causes a shift in the entire subsequent sentence. This causes the classifier to identify the entire shift in sentence as an area of difference. Fig. 3.16 shows an

example where the template image contains a single upside-down character "w" and is rightly identified by the digital collation. As illustrated in Fig. 3.17, if either the target or the template image contains distortion usually caused by bending at the spine of the book while scanning, it can result in a significant area of the collation being highlighted as the difference area. This can lead to failure in classification of the difference areas and resulting in high number of false positives.

3.4 Limitations and Future Work

The nonrigid warping featured in some of the datasets substantially increases the difficulty in collating two text images. While weak nonrigid transform can be well handled by the patch-based matching algorithm described in Section. 3.1, strong warping needs a special-purpose dewarping algorithm. In future work, we plan to apply the dewarping algorithm described in Chapter 2. of this work and handle such strong warping.

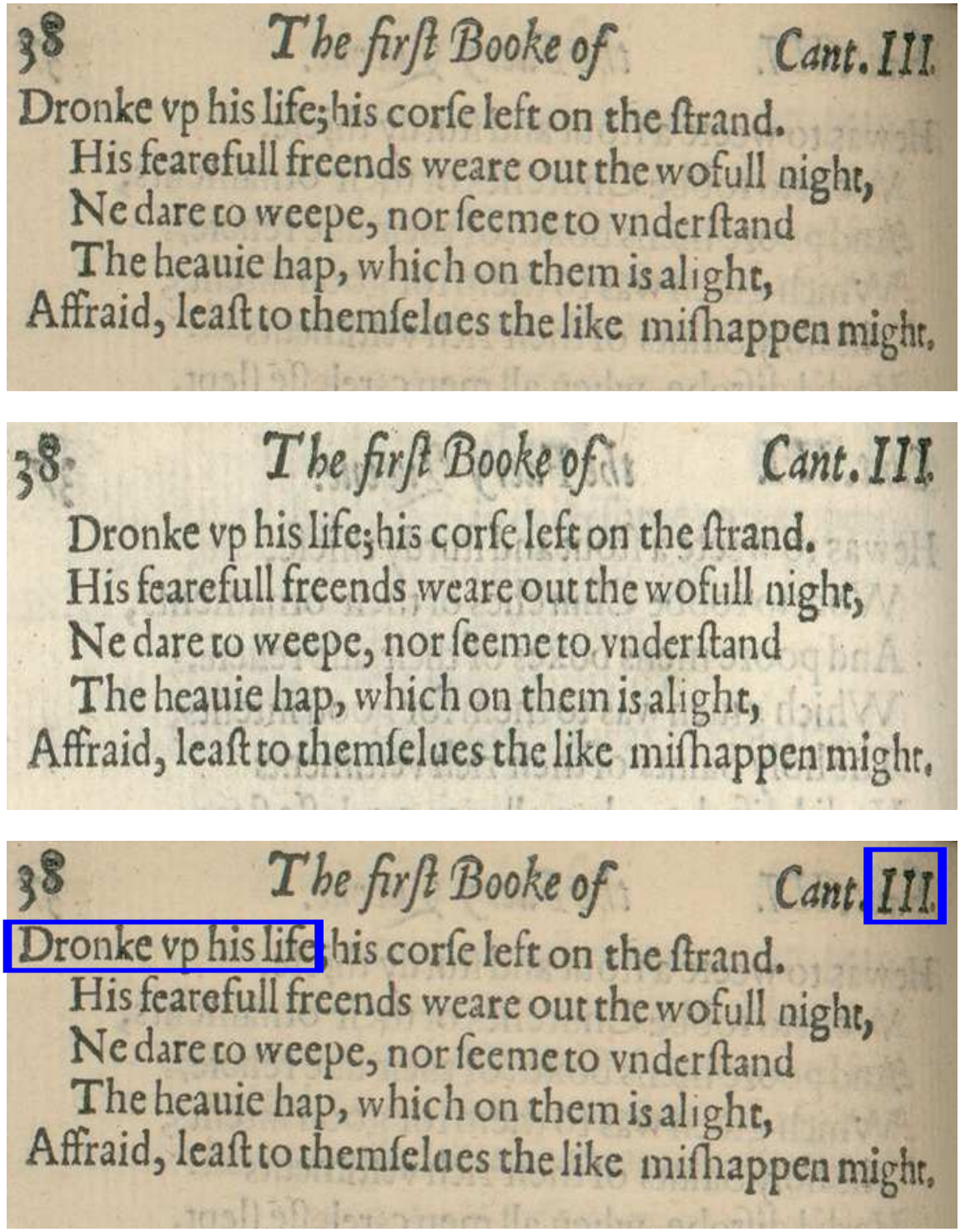


Figure 3.13: Collation result for a pair of images from Dataset 3. The template image is shown in the first row, the target in the second row, and the classifier output in the third row. The blue bounding boxes highlight the areas of differences found by the digital collation.

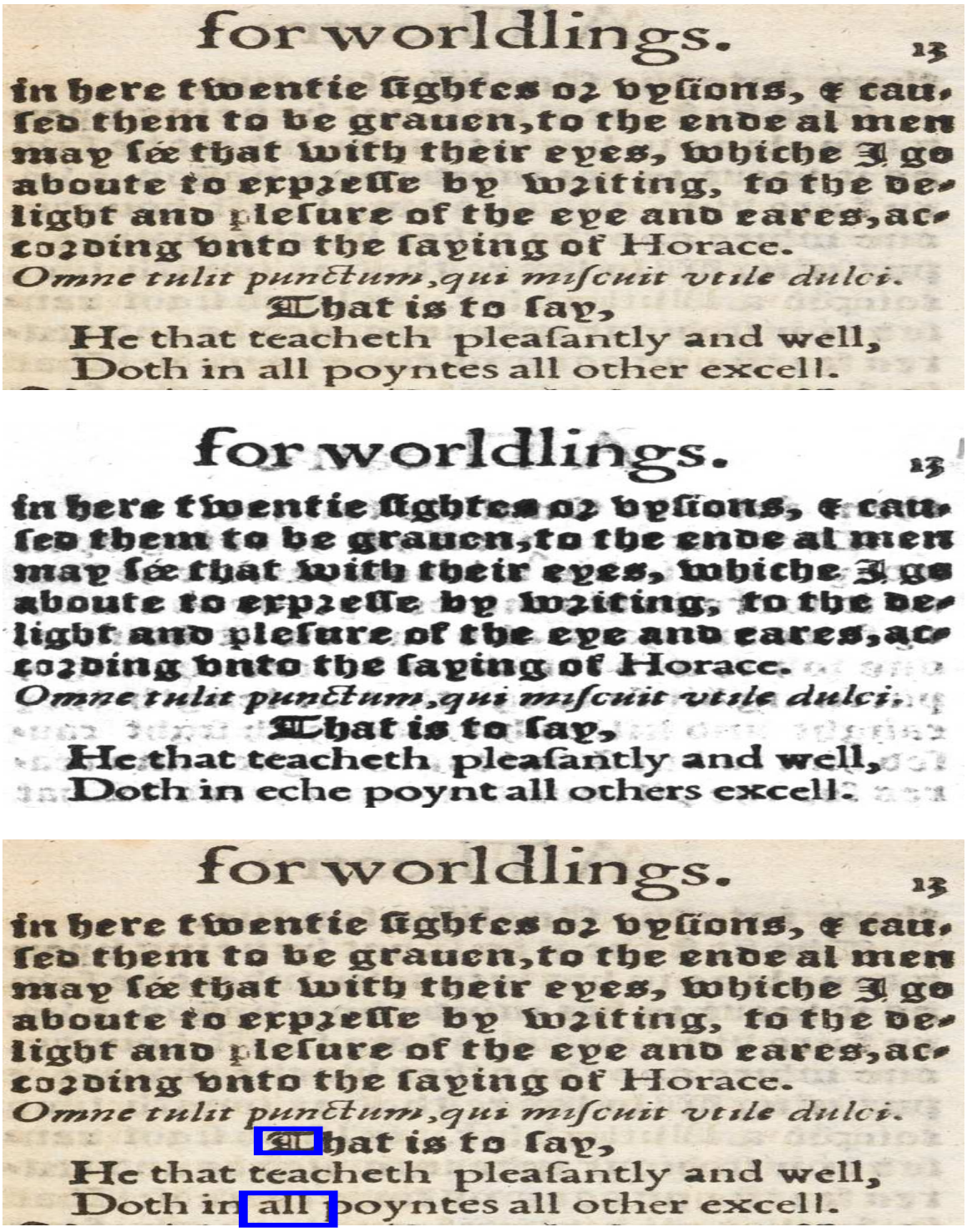


Figure 3.14: Collation result for a pair of images from Dataset 3. The template image is shown in the first row, the target in the second row, and the classifier output in the third row. The blue bounding boxes highlight the areas of differences found by the digital collation.

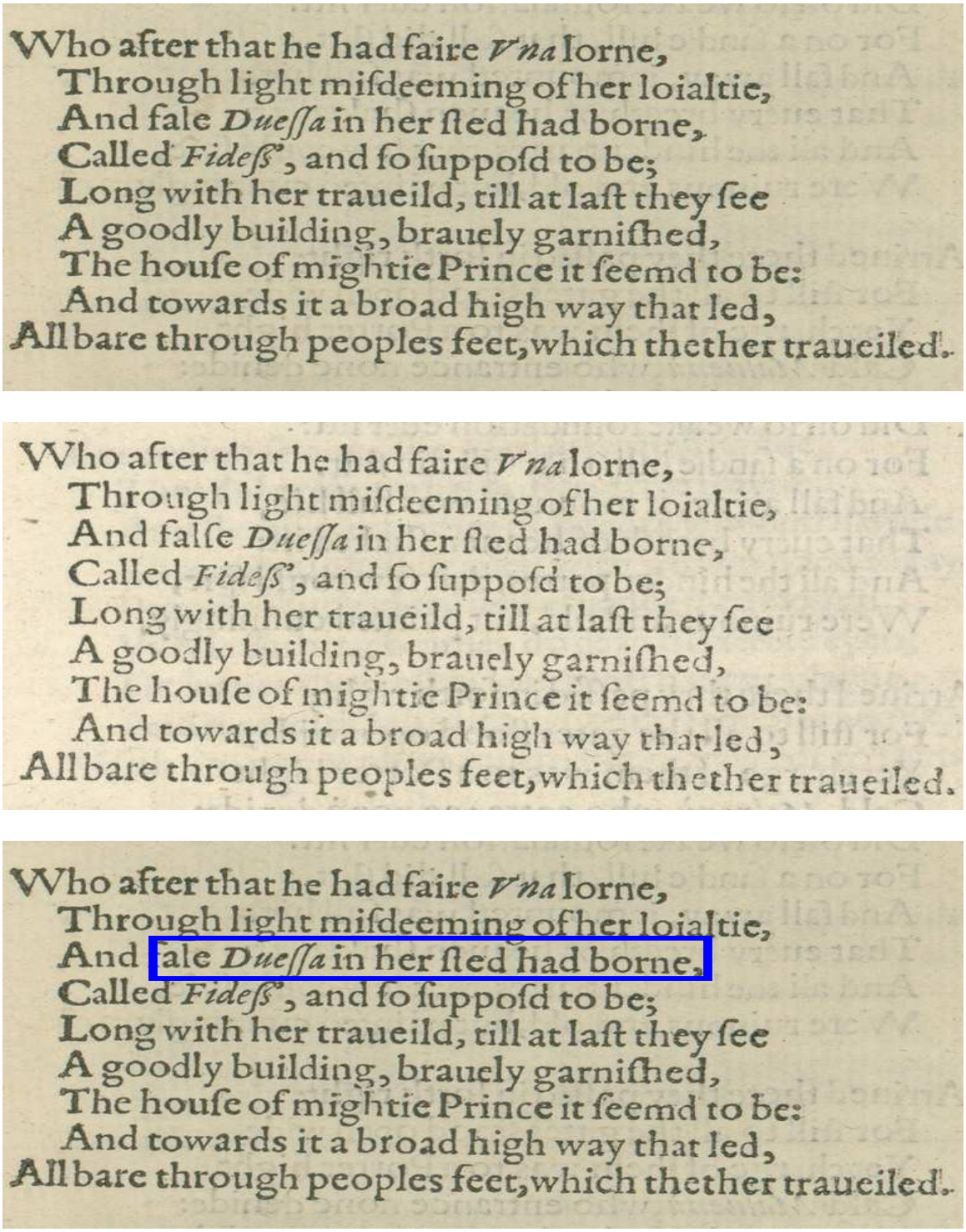


Figure 3.15: Collation result for a pair of images from Dataset 3. The template image is shown in the first row, the target in the second row, and the classifier output in the third row. The blue bounding box highlights the area of difference found by the digital collation.

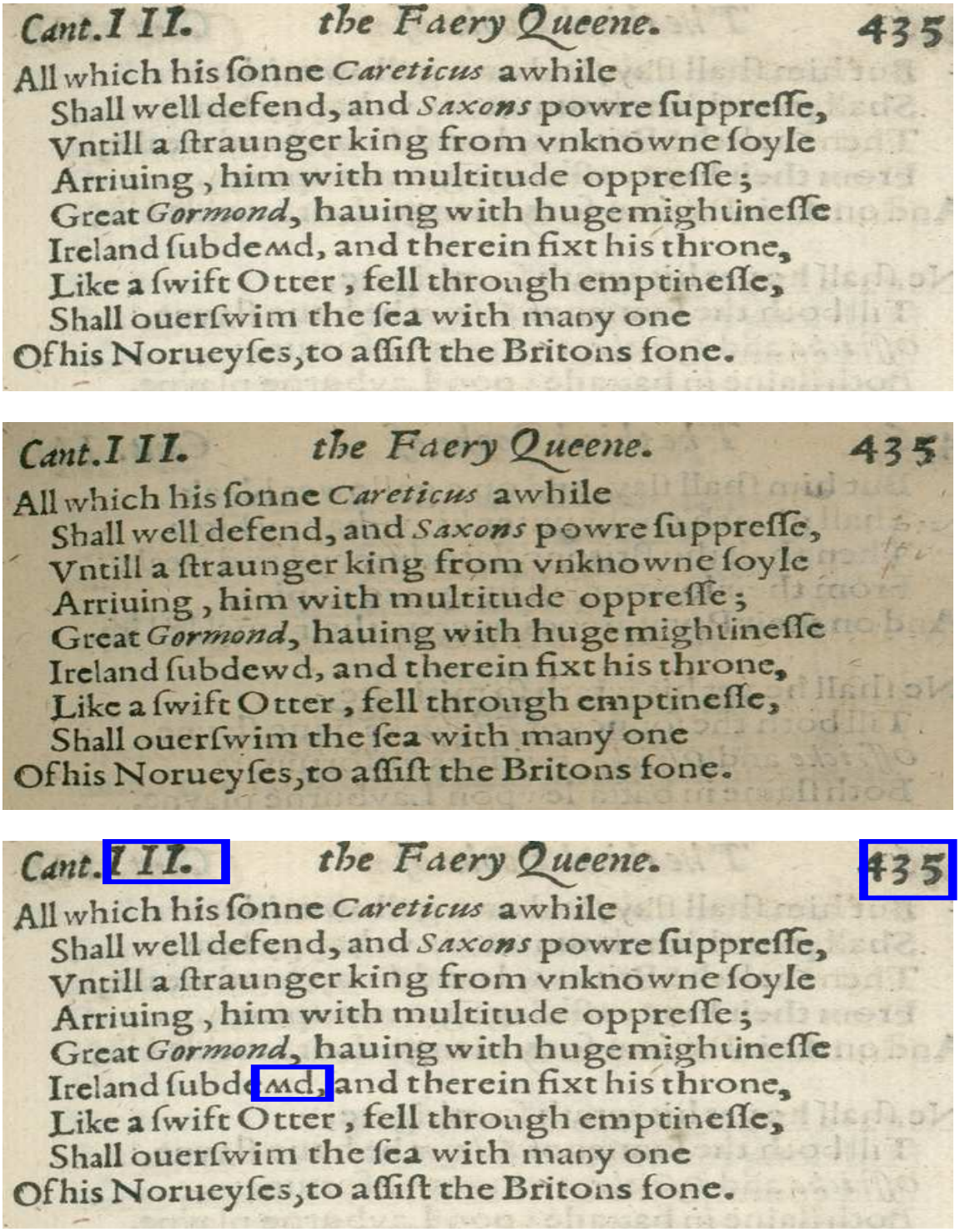


Figure 3.16: Collation result for a pair of images from Dataset 3. The template image is shown in the first row, the target in the second row, and the classifier output in the third row. The blue bounding boxes highlight the areas of differences found by the digital collation.

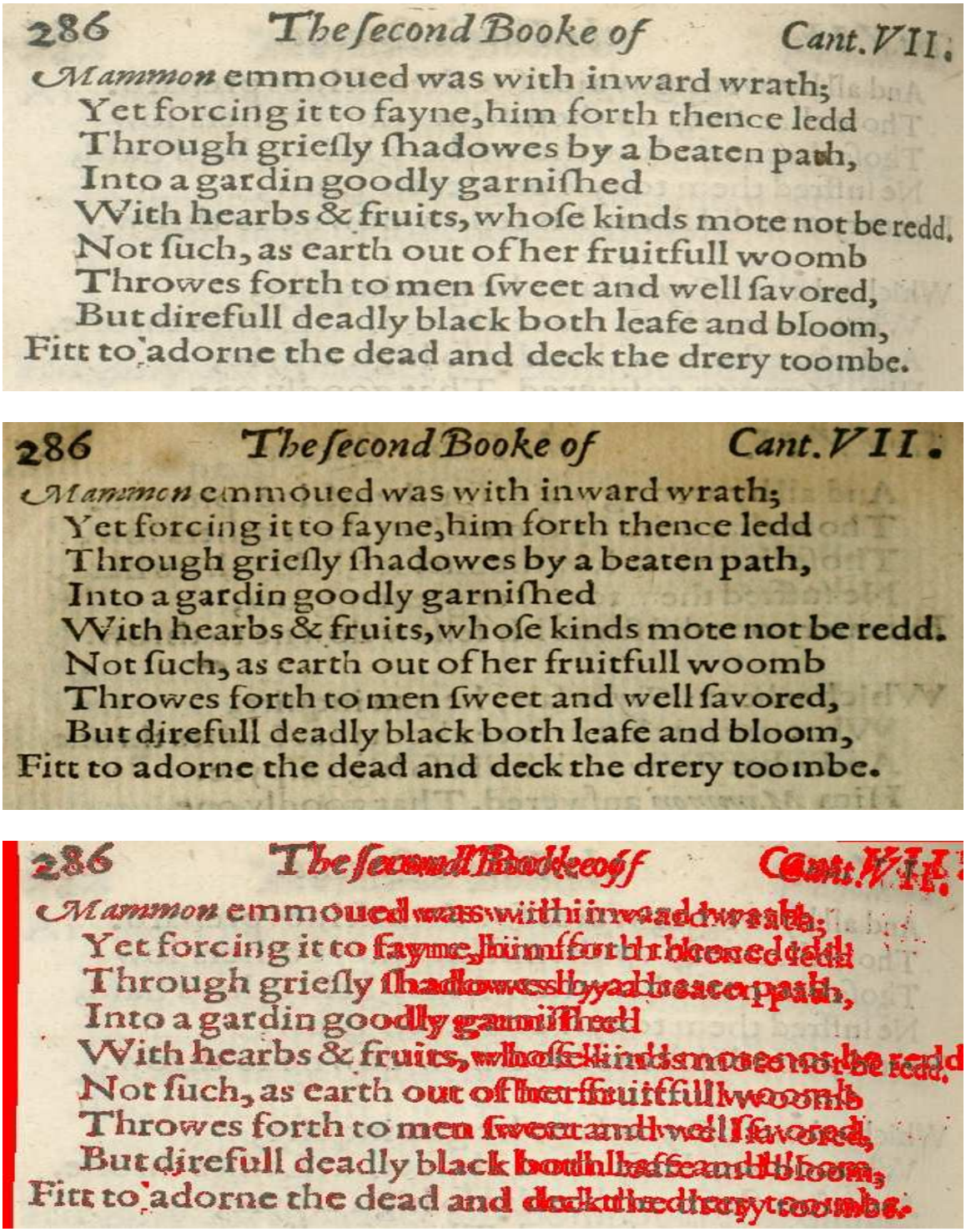


Figure 3.17: An illustration of a failure case for Digital collatin for a pair of images from Dataset 3. The template image is shown in the first row, the target in the second row, and the difference image in the third row. The area shaded in red highlights the area of differences found by the digital collation.

Conclusion

In this work, we developed three new techniques for document image analysis. In the first part of this work which concerns the challenging problem of handwritten text segmentation, we developed a new graph-based method to segment connected handwritten text into individual characters for text recogntion. Different from previous methods, the presented method does not require a good initial set of candidate segmentation boundaries, and does not make any limiting assumptions on the number,size, width, height or aspect ratio of the characters. We adapted a character recognition algorithm using SVM classifiers to measure the character likeliness of each text segment, and then searched for a text segmentation that leads to a maximum average character likeliness. This avoids any possible bias towards an oversegmentation that encumbers previous methods. Specifically we developed a graph algorithm to find the optimal segmentation with the maximum average character likeliness. We collected a set of real, handwritten text images for both training and testing, and found that the performance of the presented method is superior to a pervious method that uses dynamic programming.

We then presented a novel 2D distortion grid estimation method in second part of this work, to rectify distorted images based on white space lines that are present between a pair of text lines. We proposed a distance transform-based line propagation approach to obtain a robust estimation of the white space lines. We also estimated the vertical direction of these text lines. We thne used these white space lines and estimated vertical directions to build a 2D distortion grid which is used in a 3D rectification algorithm to dewarp a document image. We demonstrate the robustness and

accuracy of our method by providing qualitative as well as quantitative evaluations on three different datasets, and compared it against a state-of-the-art method and acheieve significantly better results.

Finally, we proposed a solution to a time-honored problem in the humanities-that of document collation. By solving this problem in the digital domain, many of the downsides of the tedious optical methods can be resolved. Furthermore by developing and employing novel feature matching and registration, we are able to automatically identify the most important differences for the humanists studying a work. We also provide a software implementation for the digital collation. This implementation provides several functionalities for the experts to interact with the digital collation process as well as make it easier to analyze a large number of works at a much faster rate than before.

Bibliography

[1] Sergey Ablameyko and Oleg Okun, *Printed text segmentation using distance transform*, CAIP, 1993, pp. 599–603.

[2] Sergey Ablameyko and Tony Pridmore, *Document image acquisition*, Machine Interpretation of Line Drawing Images, Springer London, 2000, pp. 45–56 (English).

[3] Zaher Al Aghbari and Salama Brook, *Hah manuscripts: A holistic paradigm for classifying and retrieving historical arabic handwritten documents*, Expert Systems with Applications **36** (2009), no. 8, 10942 – 10951.

[4] R.K. Ahuja, T.L Magnanti, and J.B Orlin, *Network flows: Theory, algorithms, and applications*, Prentice Hall, 1993.

[5] N. Arica and F.T. Yarman-Vural, *Optical character recognition for cursive handwriting*, IEEE PAMI **24** (2002), no. 6, 801 –813.

[6] Abedelkadir Asi, Raid Saabni, and Jihad El-Sana, *Text line segmentation for gray scale historical document images*, Proceedings of the 2011 Workshop on Historical Document Imaging and Processing, ACM, 2011, pp. 120–126.

[7] E. Baudrier and A. Riffaud, *A method for image local-difference visualization*, ICDAR, vol. 2, 23-26 2007, pp. 949–953.

[8] F. L. Bookstein, *Principal warps: Thin-plate splines and the decomposition of deformations*, IEEE PAMI **11** (1989), no. 6, 567–585.

[9] Thomas M. Breuel, *Robust least square baseline finding using a branch and bound algorithm*, in Document Recognition and Retrieval VIII, SPIE, 2002.

[10] M.S. Brown and W.B. Seales, *Document restoration using 3d shape: a general deskewing algorithm for arbitrarily warped documents*, ICCV, 2001, pp. 367–374.

[11] D. Brzakovic and J. T. Tou, *An approach to computer-aided document examination*, International Journal of Parallel Programming **14** (1985), no. 6, 365–385.

[12] S Bukhari, F Shafait, and T Breuel, *Dewarping of document images using coupled-snakes*, Proceedings of the third International workshop on camera-based document, 2009, pp. 34–41.

[13] Syed Saqib Bukhari, Faisal Shafait, and Thomas M. Breuel, *Text-line extraction using a convolution of isotropic gaussian filter with a set of line filters*, ICDAR, 2011, pp. 579–583.

[14] Huaigu Cao, Xiaoqing Ding, and Changsong Liu, *Rectifying the bound document image captured by the camera: a model based approach*, Proceedings of Seventh International Conference on Document Analysis and Recognition, 2003, pp. 71–75.

[15] Yang Cao, Shuhua Wang, and Heng Li, *Skew detection and correction in document images based on straight-line fitting*, Pattern Recognition Letters **24** (2003), no. 12, 1871 – 1879.

[16] R. S. Caprari, *Duplicate document detection by template matching*, Image and Vision Computing **18** (2000), no. 8, 633–643.

[17] Peter B. Catrysse and Brian A. Wandell, *Optical efficiency of image sensor pixels*, J. Opt. Soc. Am. A **19** (2002), no. 8, 1610–1620.

[18] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM: a library for support vector machines*, 2001.

[19] T.C. Chang and S.Y. Chen, *Character segmentation using convex-hull techniques*, International Journal of Pattern Recognition and Artificial Intelligence **13** (1999), no. 6, 833.

[20] Mohamed Cheriet, Reza Farrahi Moghaddam, and Rachid Hedjam, *A learning framework for the optimization and automation of document binarization methods*, Comput. Vis. Image Underst. **117** (2013), no. 3, 269–280.

[21] Boris V. Cherkassky and Andrew V. Goldberg, *Negative-cycle detection algorithms*, Proceedings of the Fourth Annual European Symposium on Algorithms (London, UK), 1996, pp. 349–363.

[22] Kok Beng Chu, Li Zhang, Yu Zhang, and Chew Lim Tan, *A fast and stable approach for restoration of warped document images*, ICDAR **0** (2005), 384–388.

[23] Mickael Coustaty, Jean-Marc Ogier, Rudolf Pareti, and Nicole Vincent, *Drop caps decomposition for indexing a new letter extraction method*, ICDAR, 2009, pp. 476–480.

[24] A. Djematene, B. Taconet, and A. Zahour, *A geometrical method for printed and handwritten berber character recognition*, ICDAR, 1997, pp. 564–567.

[25] David Doermann, Huiping Li, and Omid Kia, *The detection of duplicates in document image databases*, ICDAR, 1997, pp. 314–318.

[26] H. Ezaki, S. Uchida, A. Asano, and H. Sakoe, *Dewarping of document image by global optimization*, Proceedings of Eighth International Conference on Document Analysis and Recognition, 2005, pp. 302–306.

[27] George Forman and Evan Kirshenbaum, *Extremely fast text feature extraction for classification and indexing*, CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management (New York, NY, USA), ACM, 2008, pp. 1221–1230.

[28] Bin Fu, Minghui Wu, Rongfeng Li, Wenxin Li, Zhuoqun Xu, and Chunxu Yang, *A model-based book dewarping method using text line detection*, Proceedings of Second International Workshop on Camera Based Document Analysis and Recognition, 2007, pp. 63–70.

[29] B. Gatos, I. Pratikakis, and S. J. Perantonis, *Adaptive degraded document image binarization*, Pattern Recogn. **39** (2006), no. 3, 317–327.

[30] V. Govindaraju and H. Xue, *Fast handwriting recognition for indexing historical documents*, Document Image Analysis for Libraries, 2004, pp. 314–320.

[31] M.M. Haji, T.D. Bui, and C.Y. Suen, *Simultaneous document margin removal and skew correction based on corner detection in projection profiles*, 2009, pp. 1025–1034.

[32] Chris Harris and Mike Stephens, *A combined corner and edge detector*, In Proc. of Fourth Alvey Vision Conference, 1988, pp. 147–151.

[33] Jonathan J. Hull and John Cullen, *Document image similarity and equivalence detection*, ICDAR, 1997, pp. 308–313.

[34] Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin, *Adaptive grid-based document layout*, ACM Trans. Graph. **22** (2003), no. 3, 838–847.

[35] Chuntao Jiang, Frans Coenen, Robert Sanderson, and Michele Zito, *Text classification using graph mining-based feature extraction*, Know.-Based Syst. **23** (2010), no. 4, 302–308.

[36] Hyung Il Koo and Nam Ik Cho, *State estimation in a document image and its application in text block identification and text line extraction*, ECCV, 2010, pp. 421–434.

[37] Seong-Whan Lee, Dong-June Lee, and Hee-Seon Park, *A new methodology for gray-scale character segmentation and recognition*, IEEE PAMI **18** (1996), 1045–1050.

[38] S.W. Lee and Y.J. Kim, *A new type of recurrent neural network for handwritten character recognition*, ICDAR, 1995, pp. 01–38.

[39] Jian Liang, D. DeMenthon, and D. Doermann, *Geometric rectification of camera-captured document images*, TPAMI **30** (2008), 591–605.

[40] Jian Liang, Daniel DeMenthon, and David Doermann, *Camera-based document image mosaicing*, International Conference on Pattern Recognition, 2006, pp. 476–479.

[41] S. Liang, M. Ahmadi, and M. Shridhar, *Segmentation of touching characters in printed document recognition*, ICDAR, 1993, pp. 569–572.

[42] David G. Lowe, *Object recognition from local scale-invariant features*, International Conference on Computer Vision, vol. 2, 1999, pp. 1150–1157.

[43] S. J. Lu, J. Wang, and C. L. Tan, *Fast and accurate detection of document skew and orientation*, ICDAR, 2007, pp. 684–688.

[44] Y. Lu, *On the segmentation of touching characters*, ICDAR, 1993, pp. 440–443.

[45] Subhransu Maji and Jitendra Malik, *Fast and accurate digit classification*, Tech. Report UCB/EECS-2009-159, EECS Department, University of California, Berkeley, Nov 2009.

[46] Agata Mari, *Towards collation with digital images*, Library and Information Science **53** (2005), no. 3, 1–17 (Japanese).

[47] Simone Marinai, Emanuele Marino, and Giovanni Soda, *Exploring digital libraries with document image retrieval*, ECDL (László Kovács, Norbert Fuhr, and Carlo Meghini, eds.), Lecture Notes in Computer Science, vol. 4675, Springer, 2007, pp. 368–379.

[48] Gale Martin, Mosfeq Rashid, and James A. Pittman, *Integrated segmentation and recognition through exhaustive scans or learned saccadic jumps*, International Journal of Pattern Recognition and Artificial Intelligence **7** (1993), no. 4, 831–847.

[49] Gaofeng Meng, Chunhong Pan, Shiming Xiang, and Jiangyong Duan, *Metric rectification of curved document images*, IEEE Trans. Pattern Anal. Mach. Intell. (2012), 707–722.

[50] L. Mico and J. Oncina, *Comparison of fast nearest neighbor classifiers for handwritten character recognition*, Pattern Recognition Letters **19** (1998), no. 3-4, 351–356.

[51] D.L. Milgram, *Computer methods for creating photomosaics*, Computers, IEEE Transactions on **C-24** (1975), no. 11, 1113–1119.

[52] Hiromitsu Miyazaki, Seiichi Uchida, and Hiroaki Sakoe, *Mosaicing-by-recognition: a technique for video-based text recognition*, vol. 2, 2005, pp. 904–908.

[53] R. Pareti, S. Uttama, J.P. Salmon, J.M. Ogier, S. Tabbone, L. Wendling, S. Adam, and N. Vincent, *On defining signatures for the retrieval and the classification of graphical drop caps*, 27-28 2006, pp. 220–231.

[54] Maurizio Pilu, *Deskewing perspectively distorted documents: An approach based on perceptual organization*, HP White Paper (2001).

[55] I. Pratikakis, B. Gatos, and K. Ntirogiannis, *Icdar 2013 document image binarization contest (dibco 2013)*, Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, Aug 2013, pp. 1471–1476.

[56] Ioannis Pratikakis, Basilios Gatos, and Konstantinos Ntirogiannis, *Icdar 2011 document image binarization contest (dibco 2011).*, ICDAR, IEEE, 2011, pp. 1506–1510.

[57] L. Pugin, J. Hockman, J.A. Burgoyne, and I. Fujinaga, *Gamera versus aruspix: Two optical music recognition approaches*, 2008, pp. 419–424.

[58] R.J. Radke, S. Andra, O. Al Kofahi, and B. Roysam, *Image change detection algorithms: A systematic survey*, IEEE Transactions on Image Processing **14** (2005), no. 3, 294–307.

[59] J. Rocha and T. Pavlidis, *Character recognition without segmentation*, IEEE PAMI **17** (1995), no. 9, 903–909.

[60] Partha Pratim Roy, Umapada Pal, Josep Llads, and Mathieu Delalandre, *Multi-oriented and multi-sized touching character segmentation using dynamic programming.*, ICDAR, 2009, pp. 11–15.

[61] J. Sauvola and M. Pietikinen, *Adaptive document image binarization*, Pattern Recognition **33** (2000), 225–236.

[62] Yuanlong Shao, Xinguo Liu, Xueying Qin, Yi Xu, and Hujun Bao, *Locally developable constraint for document surface reconstruction*, ICDAR, 2009, pp. 226–230.

[63] Zhixin Shi, Srirangaraj Setlur, and Venu Govindaraju, *Text extraction from gray scale historical document images using adaptive local connectivity map.*

[64] Giovanni Soda Simone Marinai, Emanuele Marino, *A comparison of clustering methods for word image indexing*, Document Analysis Systems, IAPR International Workshop on **0** (2008), 671–676.

[65] Elisa H. Barney Smith, *An analysis of binarization ground truthing*, Proceedings of the 9th IAPR International Workshop on Document Analysis Systems (New York, NY, USA), DAS '10, ACM, 2010, pp. 27–34.

[66] J.Q.A. Song, Z. Li, M.R. Lyu, and S.J. Cai, *Recognition of merged characters based on forepart prediction, necessity-sufficiency matching, and character-adaptive masking*, IEEE Transactions on Systems,Man and Cybernetics **35** (2005), no. 1, 2–11.

[67] Chew Lim Tan, Weihua Huang, Zhaohui Yu, and Yi Xu, *Imaged document text retrieval without OCR*, IEEE PAMI **24** (2002), 838–844.

[68] Chew Lim Tan, Li Zhang, Zheng Zhang, and Tao Xia, *Restoring warped document images through 3d shape modeling*, IEEE Trans. Pattern Anal. Mach. Intell. (2006), 195–208.

[69] Breuel Thomas M., *Two geometric algorithms for layout analysis*, In Workshop on Document Analysis Systems, Springer-Verlag, 2002, pp. 188–199.

[70] Yuandong Tian and S.G. Narasimhan, *Rectification and 3d reconstruction of curved document images*, CVPR, 2011, pp. 377–384.

[71] Adrian Ulges, Christoph H. Lampert, and Thomas Breuel, *Document capture using stereo vision*, Proceedings of the ACM symposium on Document engineering, 2004, pp. 198–200.

[72] Adrian Ulges, Christoph H. Lampert, and Thomas M. Breuel, *Document image dewarping using robust estimation of curled text lines*, Proceedings of the Eighth International Conference on Document Analysis and Recognition, IEEE Computer Society, 2005, pp. 1001–1005.

[73] U. v. Marti and H. Bunke, *Text line segmentation and word recognition in a system for general writer independent handwriting recognition*, ICDAR, 2001, pp. 159–163.

[74] Joost Van Beusekom, Faisal Shafait, and Thomas M. Breuel, *Image-matching for revision detection in printed historical documents*, 2007, pp. 507–516.

[75] B.A. Wandell, A. El Gamal, and B. Girod, *Common principles of image acquisition systems and biological vision*, Proceedings of the IEEE **90** (2002), no. 1, 5–17.

[76] Jie Wang, Michael S. Brown, and Chew Lim Tan, *Automatic corresponding control points selection for historical document image registration*, ICDAR (2009), 1176–1180.

[77] Jin Wang and Jack Jean, *Segmentation of merged characters by neural networks and shortest-path*, Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing: states of the art and practice (New York, NY, USA), 1993, pp. 762–769.

[78] Xian Wang, Venu Govindaraju, and Sargur Srihari, *Holistic recognition of handwritten character pairs*, Pattern Recognition **33** (2000), no. 12, 1967 – 1973.

[79] Wikipedia, *Levenshtein distance — wikipedia, the free encyclopedia*, 2014.

[80] Changhua Wu and Gady Agam, *Document image de-warping for text/graphics recognition*, Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, 2002, pp. 348–357.

[81] Qi Xiaorui, Ma Lei, Sun Changjiang, and Liu Jiang, *Fast skew angle detection algorithm for scanned document images*, Circuits, Communications and System (PACCS), 2011 Third Pacific-Asia Conference on, July 2011, pp. 1–4.

[82] H.H. Xue and V. Govindaraju, *On the dependence of handwritten word recognizers on lexicons*, IEEE PAMI **24** (2002), no. 12, 1553–1564.

[83] Ismet Zeki Yalniz and R. Manmatha, *A fast alignment scheme for automatic ocr evaluation of books*, ICDAR, 2011, pp. 754–758.

[84] A. Yamashita, A. Kawarago, T. Kaneko, and K.T. Miura, *Shape reconstruction and image restoration for non-flat surfaces of documents with a stereo vision system*, ICPR, 2004, pp. 482–485.

[85] B. Yu and A. K. Jain, *A robust and fast skew detection algorithm for generic documents*, Pattern Recognition **29** (1996), no. 10, 1599–1629.

[86] Ali Zandifar, *Unwarping scanned image of japanese/english documents*, Proceedings of the 14th International Conference on Image Analysis and Processing, IEEE Computer Society, 2007, pp. 129–136.

[87] Anthony Zappalá, Andrew Gee, and Michael Taylor, *Document mosaicing*, Image and Vision Computing **17** (1999), no. 8, 589–595.

[88] L. Zhang and C.L. Tan, *Warped image restoration with applications to digital libraries*, Proceedings of Eighth International Conference on Document Analysis and Recognition, 2005, pp. 192–196.