



All Theses and Dissertations

2013-04-03

Warping-Based Approach to Offline Handwriting Recognition

Douglas J. Kennard

Brigham Young University - Provo

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>



Part of the [Computer Sciences Commons](#)

BYU ScholarsArchive Citation

Kennard, Douglas J., "Warping-Based Approach to Offline Handwriting Recognition" (2013). *All Theses and Dissertations*. 3991.
<https://scholarsarchive.byu.edu/etd/3991>

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Warping-Based Approach to Offline Handwriting Recognition

Douglas J. Kennard

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

William A. Barrett, Chair
Bryan S. Morse
Eric Ringger
Dan R. Olsen
Daniel Zappala

Department of Computer Science
Brigham Young University
April 2013

Copyright © 2013 Douglas J. Kennard
All Rights Reserved

ABSTRACT

Warping-Based Approach to Offline Handwriting Recognition

Douglas J. Kennard
Department of Computer Science, BYU
Doctor of Philosophy

An enormous amount of the historical record is currently trapped in non-indexed handwritten format. Even after being scanned into images, only a minute fraction of the existing records can be manually transcribed / indexed with reasonable amounts of time and cost. Although progress continues to be made with automatic handwriting recognition (HR), it is not yet good enough to replace manual transcription or indexing. Much of the recent HR work has focused on incremental improvements to methods based on Hidden Markov Models (HMMs) and other similar probabilistic approaches. In this dissertation we present a fundamentally new approach to HR based on 2-D geometric warping of word images. The results of our experimentation indicate that our approach is significantly more accurate than an existing whole-word approach used for word-spotting, and may also be better than HMM-based HR approaches. Since it is a completely new method, we also believe there is potential for improvement and future work that builds on this approach. In addition, we demonstrate that the approach can be used effectively in the related application domain of signature verification and forgery detection.

Keywords: Handwriting Recognition, Word Warping / Morphing, Signature Verification, Forgery Detection

ACKNOWLEDGMENTS

I have had the rare opportunity of having two advisors, both of whom have contributed significantly to my progress and learning. I would like to thank Dr. William A. Barrett for his mentorship and guidance. In addition to providing direction, instruction, and advice throughout my graduate school experience, he has also served as an example. His enthusiasm for service and for research that benefits the mission of BYU and its sponsoring institution — The Church of Jesus Christ of Latter-day Saints — has had a profound impact on my own goals and aspirations. I would also like to thank Dr. Bryan S. Morse, who was my advisor for the three years that Dr. Barrett was away as a mission president. I will always be grateful that Dr. Morse was willing to take me on as a student during that time and I appreciate the many things he has taught me. He has also been a great example and mentor, sharing insights and wisdom that have been a great benefit to my research and to me personally.

I also thank the other members of my dissertation committee for their insights, feedback, and willingness to give of their time. Likewise, I thank the other members of the faculty who have been great teachers and examples. I particularly wish to express thanks to Dr. Thomas W. Sederberg for his collaboration with Dr. Barrett and me in the development of the handwriting recognition algorithm described in this dissertation, for his inspiring devotional address, and for his example of commitment and integrity.

I thank others who have provided data, expertise, collaboration, resources, funding, or other support that has benefitted my dissertation research and my other projects. To name just a few: FamilySearch and the Family History Department of the LDS Church including Jake Gehring, Patrick Schone, Heath Nielson, Gordon Clarke, Jimmy Zimmerman, and others; the BYU Harold B. Lee Library including Bill Lund, Don Campbell, Scott Eldredge, and Russ Taylor; Professor Fred E. Woods in the BYU Religion Department; Luke Hutchison for bootstrapping my literature search; Andrew Kent; Renae Bell; the BYU Computer Science Department administrative staff; the Gene Powell BYU Fund for Family

History Research; a BYU Mentoring Grant awarded through the office of Graduate Studies; all the people who traded their handwriting for candy and those who gave it up out of the goodness of their hearts.

Special thanks to the people in the lab that kept me laughing when I needed to, mainly Dave, Carson, and Josh, and to Seth and Kevin for our discussions about politics, ethics, and how to solve the world's problems. And biggest thanks to the family members, friends, leaders, acquaintances, and others who have impacted my life outside of the lab and classroom, making my experience at BYU a life-changing, memorable, and meaningful experience. Thank you, all of you.

Contents

List of Figures	viii
List of Tables	xi
List of Algorithms	xii
1 Introduction	1
1.1 The Offline Handwriting Recognition Problem	2
1.2 General Approach to HR	6
1.3 Previous Work	7
1.4 Recent Progress in HR	13
1.5 Our Warping-Based Approach to Handwriting Recognition	13
1.6 Dissertation Organization	14
2 Word-Warping for Offline Handwriting Recognition	16
2.1 Abstract	16
2.2 Introduction	17
2.3 Related Work	19
2.4 Methods	20
2.4.1 Preprocessing	22
2.4.2 Distance Map and Medial Axis	22
2.4.3 Dynamic Programming for Coarse Mesh Alignment	24
2.4.4 Warping Coordinates	28

2.4.5	Morphing for Warp Mesh Improvement	28
2.4.6	Word Matching Cost	31
2.5	Experiments	32
2.6	Results and Discussion	33
2.7	Conclusion and Future Work	39
3	Offline Signature Verification and Forgery Detection Using a	
	2-D Geometric Warping Approach	40
3.1	Abstract	40
3.2	Introduction	41
3.3	Methods	43
3.3.1	Preprocessing	43
3.3.2	Comparing Signatures by 2-D Warping	44
3.3.3	Computing Classification Threshold	44
3.3.4	Accepting and Rejecting Signatures	46
3.4	Datasets	47
3.5	Experiments	47
3.6	Results	48
3.7	Conclusion	51
4	Additional Analysis, Experiments, and Improvements	53
4.1	Introduction	53
4.2	Analysis of Smith Dataset Recognition Errors	54
4.3	Analysis of Washington Dataset Recognition Errors	60
4.4	Incorporation of Morphing Movement Cost into Word Difference Metric	62
4.4.1	Morphing Movement Cost	63
4.4.2	Results of Incorporating Movement Cost for Handwriting Recognition	64
4.4.3	Adding a Penalty to Movement Cost for Mismatched Word Lengths	66

4.4.4	Results of Incorporating Movement Cost for Signature Verification . . .	68
4.5	Improved Cost Metric	71
4.6	Improved Handling of Distance Map Boundaries	72
4.7	Using $C_{0 \rightarrow 1}$ Instead of Heuristic in <i>placement_cost_{x,y}</i>	72
4.8	Correct Handling of the Last Row and Column of Warp Mesh	74
4.9	Analysis of the Effect of Warp Mesh Size	75
4.10	Parameter Selection for Improved DP Alignment	79
4.11	Using $C_{0 \leftrightarrow 1} = \max(C_{0 \rightarrow 1}, C_{1 \rightarrow 0})$ and $C_{0 \leftrightarrow 1} = \min(C_{0 \rightarrow 1}, C_{1 \rightarrow 0})$	82
4.12	Improved Results for Smith and Washington Datasets	82
4.13	IAMDB Dataset Results	83
4.14	Comparison to Hidden Markov Model (HMM) Approach	85
4.15	Conclusion	90
5	Conclusion and Future Work	91
A	Datasets	94
A.1	Smith Dataset	94
A.2	Washington Dataset	97
A.3	IAM Database (IAMDB version 3.0)	99
A.4	BYU English Signature Dataset (Blind and Casual Forgeries)	100
A.5	SigComp2011 Dutch and Chinese Signatures (Skilled Forgeries)	103
B	Smith dataset errors with different shapes	105
C	Washington dataset errors with different shapes	115
D	Explanation of Code and Documentation	126
	References	127

List of Figures

1.1	Difficulty differentiating letters in a word	2
1.2	HR technology for mail sorting	3
1.3	HR technology for bank check processing	4
1.4	Image degradations and aging artifacts that complicate HR	5
1.5	A typical HR system	6
1.6	Whole word features used by Rath and Manmatha	8
2.1	Overview of HR by matching words with word warping	18
2.2	Reference key to symbols and notation	21
2.3	Distance map and medial axis	23
2.4	Whole word features for coarse alignment	25
2.5	DP table, DP path, Sakoe-Chiba band, and DP alignment	26
2.6	Word profile feature for vertical DP alignment	27
2.7	Warping from rectangular to non-rectangular mesh quads	28
2.8	<i>Improve</i> step of the morphing algorithm	30
2.9	<i>Refine</i> step of the morphing algorithm	32
2.10	Experimental Results – Word Recognition Accuracy	34
2.11	Alignment of medial axes	35
2.12	Examples of recognition errors (Smith dataset)	36
2.13	Examples of correct answer in top 3 matches	37
2.14	Correct answer in top- <i>N</i> results	38
3.1	Various skill levels of forgeries	41

3.2	Inconsistent cropping for Chinese signatures	43
3.3	Comparing two signatures	45
3.4	ROC curves of our method for blind, casual, and skilled forgery datasets	49
3.5	Accuracy of our method as t varies	50
3.6	Comparison to methods from ICDAR 2011 competition	52
4.1	Types of recognition errors - Smith dataset	55
4.2	Smith dataset errors that only differ by capitalization	55
4.3	Smith dataset errors that only differ by one character	55
4.4	Smith dataset errors that have very similar shapes	56
4.5	Smith dataset errors that have moderately similar shapes	56
4.6	Smith dataset errors that have different shapes	56
4.7	Example of filled loops causing a recognition error	57
4.8	Examples of errors caused by poor coarse alignment	58
4.9	Illustration of how $C_{0 \rightarrow 1}$ can be low even with poor alignment	58
4.10	Types of recognition errors - Washington dataset	61
4.11	Washington dataset errors that only differ by punctuation or capitalization	61
4.12	Washington dataset errors that only differ by one character	61
4.13	Washington dataset errors that have very similar shapes	61
4.14	Washington dataset errors that have moderately similar shapes	62
4.15	Washington dataset errors that have different shapes	62
4.16	Effect of including movement cost in the word difference metric	65
4.17	Example of the negative impact of movement cost on recognition	66
4.18	Example of how movement cost favors short words	67
4.19	Effect of word length mismatch penalty on accuracy when $m = 1$	68
4.20	Accuracy on Smith dataset as both movement cost and length penalty vary	69
4.21	Signature verification accuracy for various values of m	70
4.22	Warped medial axis pixels outside of distance map	72

4.23	Correction of how last row and column of mesh are handled	75
4.24	How initial mesh size ratio affects accuracy for Smith dataset	77
4.25	How initial mesh size ratio affects accuracy for Washington dataset	78
4.26	Adjustment of Sakoe-Chiba band constraint for Dynamic Programming	80
4.27	Accuracy for a range of DP band width parameter values	81
A.1	A page image from the Smith diary	95
A.2	Ground truthing program for Smith and Washington datasets	95
A.3	Example word images from the Smith dataset	96
A.4	A page image from the Washington letters	97
A.5	Example word images from the Washington dataset	98
A.6	Example pages from the IAMDB	99
A.7	Genuine signature collection forms	101
A.8	Examples of genuine English signatures	101
A.9	Blind forgery collection forms	102
A.10	Casual forgery collection forms	102
A.11	Inconsistent cropping for Chinese signatures	104
B.1	Errors that have different shapes. (Smith dataset)	105
B.2	Layout of data in this Appendix	106
C.1	Errors that have different shapes. (Washington dataset)	115
C.2	Layout of data in this Appendix	116

List of Tables

3.1	Results of our method	50
3.2	Comparison to methods from SigComp2011	52
4.1	Recognition Accuracy After Improvements and Tuning	83
4.2	Recognition Accuracy on IAMDB Dataset (large, many writers)	84

List of Algorithms

2.1 Morphing algorithm for Word Warping	29
---	----

Chapter 1

Introduction

With the rise of digital libraries and other digital information portals, various government, academic, and private organizations are undertaking massive digitization efforts to convert non-digital materials into formats that can be instantly searched and accessed electronically. A few examples include the American Memory project of the U.S. Library of Congress (<http://memory.loc.gov/ammem/index.html>), online exhibits of the U.S. National Archives (<http://www.archives.gov/exhibits>), Google Book Search (<http://books.google.com>), Project Gutenberg (<http://www.gutenberg.org>), and the FamilySearch Scanning / FamilySearch Indexing projects (<http://indexing.familysearch.org>) of The Church of Jesus Christ of Latter-day Saints (the LDS Church).

A large amount of the material being digitized consists partially or entirely of handwritten information. Most obstacles to digitization of handwriting have been overcome. Scanning technology has advanced to the point that the quality of scans is sufficient, and scanning cost has decreased while scanning speed has increased. Storage costs have decreased dramatically and continue to fall. Preservation can be guaranteed for the foreseeable future by using redundant storage at multiple sites and adhering to a reasonable schedule of migrating data to the most current storage technology. But the cost of manually indexing or transcribing handwritten data as digital text remains extremely high and is non-decreasing. If manual transcription could be replaced with (or significantly reduced by) automatic transcription, the cost for any organization to digitize and provide access to collections that include handwritten materials would be reduced substantially.

The desired solution is to use offline handwriting recognition (HR) to index or transcribe handwritten materials, much like Optical Character Recognition (OCR) is used for materials that consist entirely of machine-printed text. However, HR is much less accurate than OCR and is still largely an unsolved problem except in a few constrained applications, despite the large amount of handwriting recognition research that has been done.

In this dissertation, we present a novel HR approach based on 2-D geometric warping of word images. Our approach is fundamentally different than the Hidden Markov Model (HMM) approaches that have dominated the literature over the past several years. Our analysis and results indicate that our method is more accurate than an existing whole word method used for word-spotting, and may also be more accurate than (or at least comparable to) existing HMM-based HR approaches. We also show that our method can be applied to the distinct problem of signature verification and forgery detection. Our method shows promise as a basis for continued research in HR and related problems.

1.1 The Offline Handwriting Recognition Problem

Unlike machine-printed text with well-formed characters of a few common fonts and predictable spacing, handwriting is variable and often ambiguous. There are almost as many handwriting styles as there are people who write, and even for a specific person, handwriting varies somewhat from one occurrence of a word to another occurrence of the same word. Spacing between words and within words is often inconsistent, and it is sometimes impossible to recognize letters within a word without the context of the whole word (Figure 1.1). Sometimes even whole words cannot be reliably read by humans — much less machines —

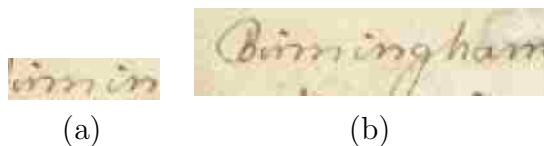


Figure 1.1: Difficulty differentiating letters in a word. It can be nearly impossible to differentiate letters without the context of the entire word. a) i, r, m, and n look nearly identical. b) word-level context disambiguates the letters in Birmingham. (Images from [47])

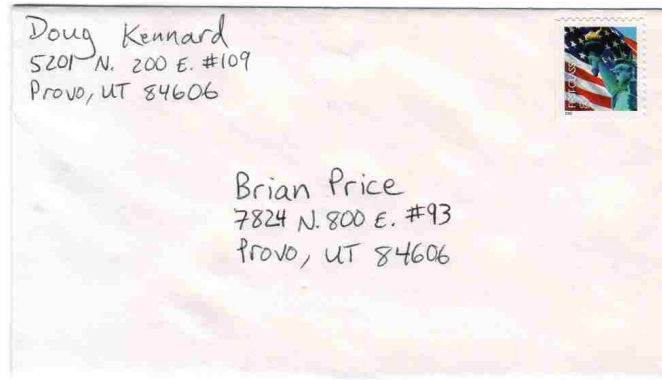


Figure 1.2: HR technology for mail sorting. HR works well because address layout is predictable, and the zip code reduces the lexicon size.

without the context of surrounding words or actual knowledge of the subject being written about. And even contextual knowledge in free-form handwritten text may be of little use in recognizing specific person or place names, which are the most important words in documents such as those in the LDS Church's collections of historical and genealogical materials.

Recognizing handwriting *offline* (after the fact, from scanned documents) is a more difficult problem than recognizing it *online* (while it is written, from a stylus or other device) because there is no temporal information about ink strokes. The difficulty of the offline HR problem (and the accuracy of HR systems) varies widely, depending on the specific application. Some factors that influence how hard a given problem is include:

- quality of the document images being processed
- neatness of penmanship in the images
- size of lexicon (or whether recognition is even constrained to a lexicon at all)
- amount and quality of training data
- whether the system can handle multiple writers or just a single writer

For some constrained applications, such as postal address processing for automatic mail sorting, HR currently works well enough to be very useful. According to a 2012 fact sheet printed by the U.S. Postal Service, sorting equipment is able to read 93% of handwritten letter mail [39]. The layout and format of a postal address is relatively predictable (Figure 1.2),

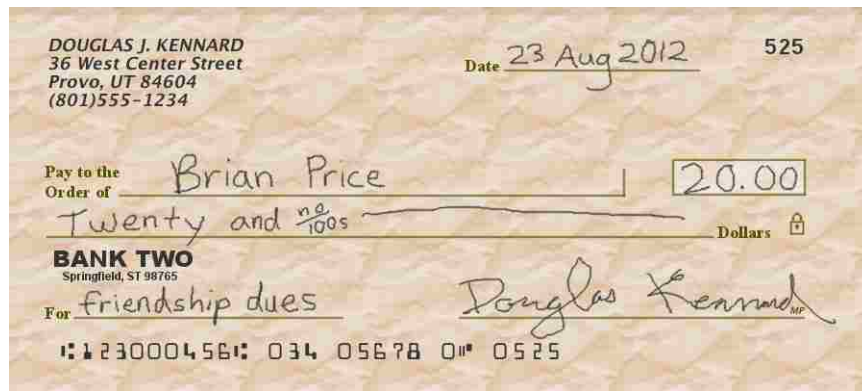


Figure 1.3: HR technology for bank check processing. HR is simplified because the courtesy amount (numerical) and legal amount (written) are redundant, the layout is predictable, and the lexicon is small.

allowing approaches such as that reported by Srihari and Kuebert [43] to parse the address into various components that can be recognized individually (zip code, street number, street name or P.O. box, city, state). Context from one part of the address can be used to make recognition of other parts simpler and more robust. For example, Srihari's system [43] recognizes the zip code and street number before trying to recognize the street name, since knowledge of the two numbers reduces the lexicon to a small number of valid street names — a single street in 69% of (zip,number) pairs, 2.21 streets per pair on average, and 542 streets maximum [42].

Other applications in which HR is currently used very successfully are the processing of bank checks and form processing (e.g., tax forms). With bank checks (Figure 1.3), the numerical courtesy amount and written legal amount are redundant, the layout is predictable, and the lexicon is limited to a few tens of words. In form processing, segmentation and layout are typically not an issue, and the information is usually somewhat predictable, such as individual digits or letters within a specific boxed area.

Recognizing unconstrained handwriting is a much more difficult problem. Segmentation of unconstrained handwriting is difficult because layout is not known in advance. Spacing may not be consistent between lines of text, between words, or even between letters within the words. Words may be written with hand-printed letters, cursive, or a mixture of

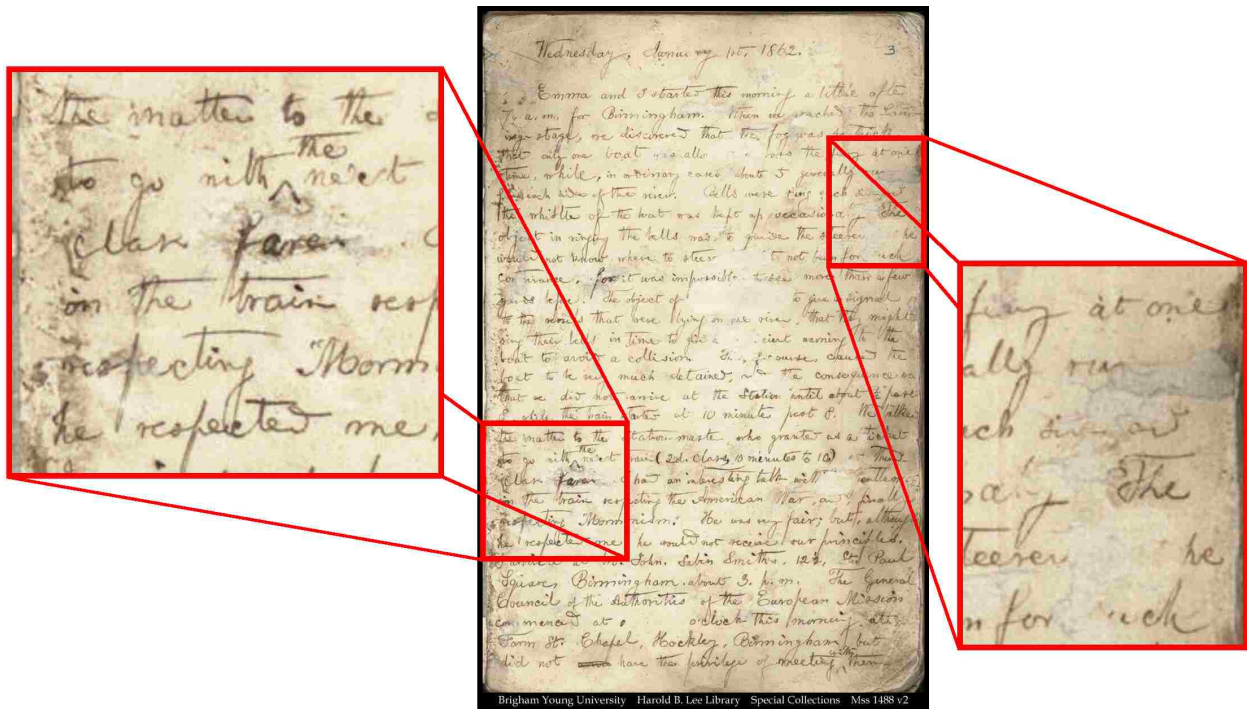


Figure 1.4: Image degradations and aging artifacts that complicate HR. Historical documents can be very difficult to recognize due to fading, discoloration, bleed-through, or other degradation. (Original image from [47])

both. Instead of a small, limited vocabulary, virtually any word or name may be written, as well as abbreviations, acronyms, numbers, and misspellings that do not appear even in very large lexicons. The HR problem becomes even harder when the system must recognize the writing of multiple authors, especially if it must handle the writing of authors for whom it does not have training data.

The difficulties of any HR problem are compounded when the documents being recognized are historical documents, which may have ink that is faded or smeared, bleed-through (or shine-through), unevenly discolored background, and other age-related degradations that can make it extremely difficult to properly segment the handwriting from the background and noise in the image (Figure 1.4).

Due to the many difficulties involved, it is not surprising that unconstrained HR is still largely an unsolved problem. While much progress has been made, there is still great room for improvement and innovation.

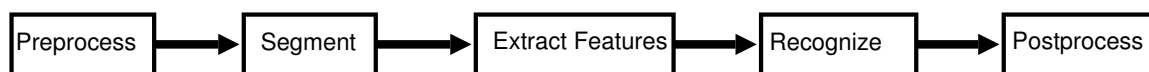


Figure 1.5: A typical HR system.

1.2 General Approach to HR

Most HR systems (e.g., [13, 18, 38]) use some variation of the general method illustrated in Figure 1.5. First, preprocessing tasks are performed, which may include steps such as deskewing the document image, filtering it to remove noise, and binarization or thresholding to separate foreground ink from the background. Next, segmentation of the foreground ink is performed to find handwritten lines, words, or parts of words. Features are extracted at the line, word, or sub-word level, and then recognition is performed. Finally, post-processing using language models such as word bigram/trigram frequencies, spell-checkers, or other methods, are used to correct errors or choose the most likely combination of several recognition hypotheses.

Each of the parts and tasks of an HR system could rightly be considered a research area of its own, and many publications can be found relating to each, either directly or as part of a larger system. For example, binarization (just one of many possible tasks performed during preprocessing) is the focus of a significant amount of research for images in general [40], and for document images specifically (e.g., [9, 10, 19, 51, 52]). Likewise, various methods of segmenting documents into textlines and segmenting textlines into words, characters, or smaller pieces are reported in the literature [14, 25, 48, 26].

Even though most HR systems follow the same general method, the steps of the pipeline in Figure 1.5 are not always divided in the same way. For example, while some systems use a lexicon or language model as a post-processing step to correct recognition hypotheses (e.g., [30]), others use the lexicon or language model as a constraint during the recognition process itself (e.g., [17]). In the case of our HR method, there is not a clear separation of the feature extraction and recognition steps, because we compare whole words to each other instead of extracting features from the words to use with a standard recognition

algorithm. In effect, the features for our method are entire words, and the recognition algorithm is our algorithm for comparing words. However, we do extract features from the words for the purpose of dynamic programming for coarse alignment of the words (discussed in detail in Chapter 2).

The 2-D warping-based method of HR presented in this dissertation encompasses only the feature extraction and recognition portions of an HR system. Although improvements to preprocessing, segmentation, or postprocessing are not within the scope of this dissertation, a complete system that incorporates our HR method would almost certainly benefit from improvements to those parts of the recognition pipeline.

1.3 Previous Work

Since early attempts at automatic HR almost 50 years ago (e.g., [22]), researchers have reported numerous approaches in the literature. Many of the approaches can be loosely grouped into a few broad categories, which we describe in the following subsections. The categories we choose are based primarily on categories mentioned in surveys and other literature, although not all authors categorize the methods in exactly the same way. In reality, there is often significant overlap between categories and approaches, so a strict partitioning of HR approaches into categories may not really be possible. As our list is not exhaustive, we refer the interested reader to several excellent survey papers that provide additional references [5, 20, 30, 44, 49].

Whole Word Recognition

Approaches that perform recognition at the word level are sometimes referred to as *holistic* approaches (as opposed to the *analytical* approaches that recognize based on smaller units such as letters or graphemes). These methods may make use of prominent word features such as ascenders, descenders, loops, dots, and t-crossings [44]. Other word level features

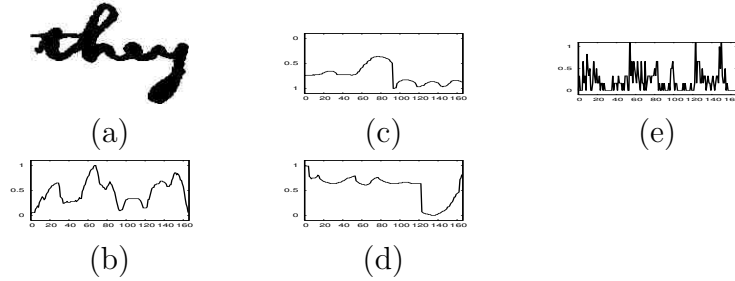


Figure 1.6: Whole word features used by Rath and Manmatha. a) Word image. b) Projection profile. c) Upper indentation profile (inverted for display). d) Lower indentation profile. e) Background-to-ink transition count. (Word image originally from [35])

such as contours, projection profiles, and aspect ratio may also be used. Madhvanath and Govindaraju [24] discuss holistic HR in more detail.

Of particular note in this dissertation is the method used by Rath and Manmatha in their word-spotting research [34]. To compare words, they use dynamic time warping (DTW) with word-level features, including word projection profiles, upper and lower indentation profiles, and background-to-ink transition counts (Figure 1.6).

Current whole word recognition approaches reportedly do not extend well to large vocabularies. As lexicon size increases, there are more words that are similar to each other and are easily confused when using word-level features. As a result, whole word methods have largely been abandoned as stand-alone recognizers. However, whole word methods are often used in large vocabulary systems for lexicon reduction [49]. *Lexicon reduction* means limiting the lexicon used for recognizing any given word to a (usually much smaller) subset of the entire lexicon based on some criteria (e.g., word length, general shape, or other word-level features).

The HR method that we present in this dissertation is a whole word method. We find that it is more accurate than previous whole word methods by comparing it to an approach based on the DTW method used by Rath and Manmatha in their word-spotting research (Chapter 2). We also find that it does extend to larger vocabularies (Sections 4.13–4.14), whereas other holistic methods apparently do not.

Offline HR as Online HR

There are some attempts to transform the offline HR problem into an online HR problem by inferring the order in which ink was written so that online HR methods can be used to recognize the words. Such a strategy seems attractive since online recognition is an easier problem to solve and since there has been success using online methods in devices such as Personal Digital Assistants (PDAs). Some success is seen in restoring writing order in constrained cases. For example, Qiao, et al. report high rates (90%-96%) of restoring writing order using very clean input images of single-stroke data [32], and rates of 94.5% using multi-stroke data [33]. However, we are not aware of any HR systems so far that obtain high recognition rates based on such an offline-to-online conversion strategy for the noisier, more general images that are encountered in practice when performing offline HR.

Segmentation-Based Recognition

Perhaps the most intuitive approach to handwriting recognition is to formulate it into a problem very similar to that of OCR—that is, to segment words into characters (*explicit segmentation*) and recognize the individual characters. One system using this approach for English and Greek handwriting is presented by Kavallieratou, et al. [13]. Results vary from 65.6% to 100% accuracy in their experiments, depending on which image database is used and which experiment is performed. The accuracy is measured at the character level, which means that actual word accuracy is significantly lower than the character accuracy reported.

Many researchers agree that character-level recognition is only practical for hand print, because cursive writing cannot be reliably segmented into individual letters without knowledge of what those letters are (Figure 1.1). This “chicken and egg” problem is noted by many authors ([44] and [49], for example), and is referred to as Sayre’s Paradox.

Instead of attempting to explicitly split words into individual characters, many approaches oversegment words into graphemes that are at least as small as letters (but often smaller). The results of this *implicit segmentation* are then used to determine the most

likely recognition result. For example, Hidden Markov Models (HMMs) or other statistical frameworks can be used with the oversegmented data to determine the likelihood that the given sequence of graphemes corresponds to a particular word or sequence of letters.

One manner of using implicit segmentation results is to merge neighboring graphemes into segmentation hypotheses, and use dynamic programming to determine the optimal segmentation (based on recognition scores using a character recognizer). Camastra proposes a character recognizer for such a system in [6], based on support vector machine (SVM) character classification.

Whether segmenting explicitly or implicitly, segmentation-based recognition methods are limited by the accuracy of their segmentation approaches and how well the segmented pieces can be combined into recognizable chunks. They also depend entirely on local features by design. We believe it is important to take advantage of more global word-level shape information instead of ignoring it as these methods do.

Segmentation-Free Recognition

Some HR approaches do not depend on segmenting words into characters or graphemes. For example, Vinciarelli, Bengio, and Bunke [50] report on an HR system that uses HMMs integrated with N -gram word models. Instead of splitting words into graphemes, a fixed-width sliding window is used to extract features in each column of the line of handwritten text. The window is 16 pixels wide, so features are again very local in nature. For three different data sets and using no N -gram word models, word recognition accuracy ranges from about 76-80%, 29-35%, and 33-43% over a range of lexicon sizes from 10,000 to 50,000 words for the first two data sets and from 10,000 to 30,000 words for the third data set. Accuracy improves significantly to about 88-91%, 45-46%, and 64-65% when using trigram word models for the same data sets and lexicon sizes. As we mention in Section 1.4, many current HR systems are based on segmentation-free HMM approaches such as this.

Multiple Classifier and Ensemble Methods

Some HR work combines more than one HR approach instead of using a single classifier. For example, Bertolami, Halter and Bunke [3] combine three HMM classifiers that individually have recognition rates of 63.06%, 58.71%, and 55.33%. Combining the classifiers results in very slight recognition improvement to 63.85%. When using the authors' rejection and re-recognition strategy, further improvement is seen (64.69% recognition rate).

Classifiers may be combined using voting schemes, weighting the results from all of the classifiers, or using other strategies. It is not necessary that all classifiers work at the same level. For example, in the systems reported by Plessis, et al. [31], a whole word classifier is combined with two different segmentation-based classifiers.

Günter and Bunke [11] test some ensemble methods (bagging, AdaBoost, half-and-half bagging, random subspace, and architecture variation) that have been successfully used for other machine learning applications. They report that improvements are seen for all ensemble methods when certain of the voting schemes they test are used. Their original classifier recognition rate is 66.23% and their best ensemble recognition rate is 68.95%.

Additional HR research that uses ensemble or multiple classifier methods is cited in the survey by Bunke [5].

Human-Inspired HR Models

A significant amount of research exists on how humans read [46]. Although the process is not completely understood, it is evident that we use both holistic and analytical approaches as we read, not just one or the other. Some HR methods take a similar approach, attempting to combine whole word recognition with analytical methods, or top-down with bottom-up approaches [49].

As already mentioned, some systems simply use holistic recognition for lexicon reduction and some use holistic and analytical approaches in a multiple classifier framework. There are also a small number of methods that integrate the two more tightly. One example

is the PERCEPTO system reported by Côté, et al. [7], in which top-down and bottom-up approaches interact with each other by activating hypotheses at the word, letter, and feature level. The cycle is repeated several times to converge toward a solution, at which point a ranked list of candidate words is available. This and other perception-oriented HR models are described in the survey by Steinherz, et al. [44].

Shape Morphing

Although not really a separate category, two papers should be mentioned because of their relevance to our approach in the respect that they use morphing for recognition. In [29], Pavlidis, Singh, and Papanikolopoulos directly use shape metamorphosis (morphing) costs as a metric of how different words and shapes are from each other. The morphing costs are based on those presented by Sederberg and Greenwood [37]. On a small number of handwritten words and simple shapes (107 reference, 428 test), the recognition rate ranges from 86.2% to 99.0%, depending on the handwriting of the author used for the test. The tests are single-author tests, and the method is for online handwriting instead of offline handwritten word images.

In [41], Singh and Papanikolopoulos generalize the method for use with any contour-encoded 2-D shapes, including those extracted from offline images. Tests were performed with extremely small shape vocabulary sizes (15 templates, 50 test shapes) resulting in 93.33% shape recognition. For handwritten words, 4 authors were tested individually using only 10 reference words and 40 test words each. Recognition rates for the 4 authors were, 90.0%, 92.5%, 97.5%, and 100.0%.

Our method also uses morphing for recognition, but in a very different manner than these two papers. In our early work, we experimented with using a morphing cost directly (as these papers do), but we found that we achieved much better accuracy when we only used morphing to align the words and then computed the difference between the aligned words using distance maps (Chapter 2). The morphing algorithm that we use is also very

different than the morphing algorithm used by these two papers. Our morphing algorithm is inspired by later image morphing work of Gao and Sederberg [8] instead of the earlier shape/contour morphing work in [37]. Although the iterative nature of our algorithm is similar to the iterative process in the Gao and Sederberg paper, our morphing algorithm for aligning handwritten words is actually very different even from that paper. This is easily seen by comparing the Gao and Sederberg paper with our algorithm, which we describe in great detail in Chapter 2 of this dissertation. Their morphing algorithm centers around work-based equations, while ours centers around medial axis alignment, measured by using distance maps.

1.4 Recent Progress in HR

For the past several years, almost all significant progress that we find in the HR literature for Latin scripts is primarily due to language models, combinations of recognizers / multiple classifiers, and ensemble methods. The underlying recognizers themselves are usually HMM-based approaches, such as the HMM-based recognizer used by Vinciarelli et al. [50] in their segmentation-free approach (Section 1.3) and the three HMM-based recognizers used by Bertolami et al. [3] in their multiple classifier system (Section 1.3). Very little recent progress is actually due to improvements in the recognition step of the pipeline in Figure 1.5, and few papers introduce fundamentally new recognition algorithms.

1.5 Our Warping-Based Approach to Handwriting Recognition

Our novel approach to HR is based on using 2-D geometric warping to align ink strokes of words, and then using distance maps to compute how different the aligned words are. This is a fundamentally different approach than the HMM-based methods that have dominated the literature over the past several years. The warping approach thereby lays the groundwork for additional research based on our method.

The scope of this dissertation is limited completely to the feature extraction and recognition tasks of the HR pipeline (Figure 1.5). As such, we do not directly address preprocessing, segmentation, or postprocessing with language models in our HR approach. However, improvements to those other tasks in the HR pipeline would benefit our method as part of a complete recognition system.

1.6 Dissertation Organization

In addition to the introductory material in Chapter 1, the conclusions and future work in Chapter 5, and the Appendices, the remainder of this dissertation is organized as follows.

Chapter 2 is based on the paper “Word Warping for Offline Handwriting Recognition,” published in the proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011) [15]. The chapter introduces our novel approach to HR and reports experimental results using two single-author datasets. The chapter also compares our results to those of another whole word recognition method that is based on the word-spotting research of Rath and Manmatha [35, 34], and shows that our method is significantly more accurate.

Chapter 3 is based on the paper “Offline Signature Verification and Forgery Detection Using a 2-D Geometric Warping Approach,” published at the 21st International Conference on Pattern Recognition (ICPR 2012) [16]. The chapter shows that our base HR approach is not limited strictly to the problem of HR, but can be applied to the related application area of signature verification and forgery detection. With virtually no specific optimization for various languages, our method is competitive with other methods found in the literature for both Dutch and Chinese signatures.

Chapter 4 includes significant additional analysis of our HR method. It answers several questions we had about the strengths and weaknesses of our method, reports experimental results for various minor modifications to the method and to the system parameters, and reports the accuracy of our method on a much larger, publicly available, many-author

dataset. Using the experimental results from this larger dataset, we estimate how accurate our method is compared to the base HMM recognizers used in some recent literature. Some portions of the material in Chapter 4 will be used in a future journal article submission to the International Journal on Document Analysis and Recognition (IJ DAR).

Chapter 2

Word-Warping for Offline Handwriting Recognition

In this chapter, we present our word recognition method that uses a novel morphing correspondence algorithm for handwritten words, 2-D geometric warping, and distance maps to compare unknown test words with known training examples. This chapter is based on the paper “Word Warping for Offline Handwriting Recognition,” published in the proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR 2011) [15]. Some minor modifications have been made to the version in this chapter. Of particular note are: 1) inclusion of the Δ term in Equation 2.4, which was inadvertently omitted in the original paper, 2) updated results that reflect a few corrections in the dataset groundtruth labels and some minor code corrections and improvements, and 3) additional figures, text, and modifications that help clarify our method and results.

2.1 Abstract

We present a novel method of offline whole-word handwriting recognition. We use automatic image morphing to compute 2-D geometric warps that align the strokes of each word image with the strokes of word images of training examples. Once the strokes of a given word are aligned to a training example, we use distance maps to quantify the similarity of the two words. Like 1-D Dynamic Programming (DP) methods, our warp-based method is robust to limited variation in word length and letter spacing. However, due to its 2-D nature, our method is also more robust than 1-D DP methods in handling variations caused by additional inconsistencies in character shape and stroke placement. Although we use DP

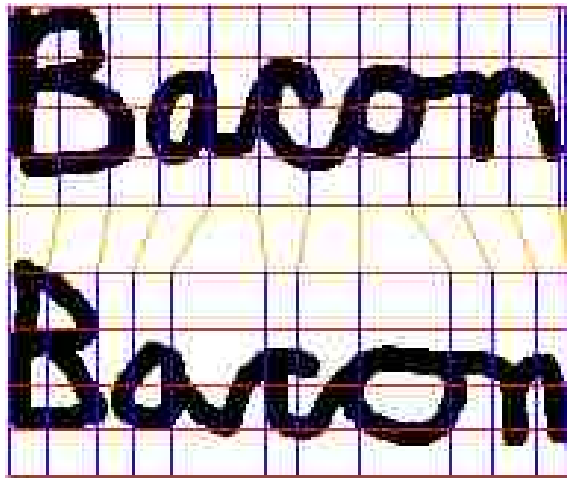
for coarse alignment, the novel contribution of this paper is not 2-D DP, but morphing to automatically discover an actual 2-D mesh-based warp, followed by the use of distance maps to compute similarity between words. Early results are encouraging. On two datasets (1,000 training and 1,000 test words each), we achieve 88.90% and 89.38% recognition accuracy for in-vocabulary words. These are increases of 7.88% and 17.19% above the results of a 1-D DP approach.

2.2 Introduction

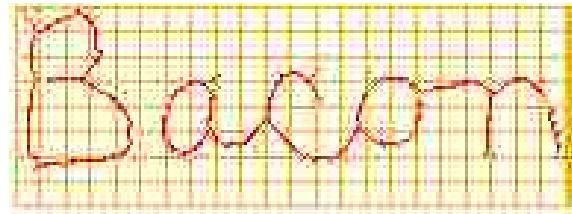
We present a novel offline whole-word recognition method that uses 2-D warping and distance maps to compare words. Our method, “word warping,” successfully handles some of the local variation inherent in handwriting such as inconsistent ink thickness and letters that are unevenly spaced, stretched, compressed, or similarly distorted.

For a given pair of images, we create a regularly-spaced rectangular mesh on the first image and a corresponding warp mesh that defines how to push, pull, bend, and stretch the ink of the first image to align it with the ink in the second image (Figure 2.1b). Aligning the ink allows us to ignore many of the local differences and variations inherent in handwriting and instead compare words at a more structural level. Once the ink is aligned by warping, we use distance maps to quantify the similarity of the two words.

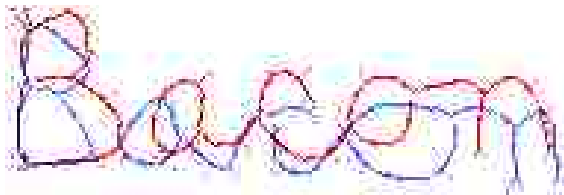
To define the warp mesh used in alignment, we first coarsely align the warp mesh by using 1-D Dynamic Programming (DP) in both the horizontal and vertical directions (Figure 2.1a). After coarse alignment, we perform a more detailed alignment by using an image morphing algorithm (Section 2.4.5) to increase the mesh resolution and iteratively adjust the control points (vertices) of the warp mesh (Figure 2.1b). We only use full-thickness word images (Figure 2.1a) for the coarse alignment. We use medial axis pixels of the words throughout the rest of the process to simplify our morphing algorithm and the distance metric we use to compare words.



(a) Step 1: Coarse alignment



(b) Step 2: Warp mesh by morphing



(c) Medial axes



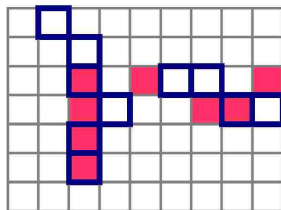
(d) Coarse alignment of medial axes



(e) Step 3: Red medial axis morphed to blue medial axis



(f) Red medial axis morphed to wrong word ("toast")



(g) Overlay of red and blue medial axes from green rectangle in (e)

4	3	2	3	2	3	3	3	2
3	2	1	2	1	2	2	2	1
2	1	0	1	0	1	1	1	0
2	1	0	1	1	1	0	0	1
2	1	0	1	2	2	1	1	1
2	1	0	1	2	3	2	2	2
2	1	1	2	3	4	3	3	3

(h) Step 5: Word matching cost = "distance" from red medial axis (normalized sum of blue squares)

Figure 2.1: Handwriting recognition by matching two instances of the word "Bacon" using word warping. a) Step 1: Coarse alignment of warp mesh using DP with full thickness word images; b) Step 2: Improve warp mesh by morphing using medial axis pixels; c) Medial axis pixels of each image: red=first/top instance, blue=second/bottom; d) Using only coarse alignment to warp the first instance – not as good as morphing; e) Step 3: Warping with mesh improved by morphing gives good alignment; f) Warping "Bacon" to wrong word "toast" does not align as well. g) Overlay of medial axes from rectangular region in e; h) Step 4: Compute distance map from (red) warped medial axis pixels (0=medial axis). Numbers=distance map with respect to red medial axis. Step 5: Compute word matching cost (normalized sum of blue squares) = "distance" from red medial axis.

2.3 Related Work

Numerous HR approaches appear in the literature ([20, 30, 44]), including some whole-word recognition methods such as those described by Madhvanath and Govindaraju in [24]. Whole-word approaches exist that use everything from ascenders, descenders, and loops to contour-based features, profile-based features, and graph-based word descriptions.

Rath and Manmatha [35, 34] show that Dynamic Time Warping (DTW) — a 1-D DP method — can be used to match whole words in the context of word-spotting. We see from their work that DTW is robust to some variation in character spacing, width, and shape in the horizontal direction – the direction of the 1-D alignment. Features from two words are aligned using DTW, and the DP cost for the alignment is used as a metric of how different the words are.

We use their DTW method for the coarse alignment of our warp mesh (Section 2.4.3). We also use DTW as the baseline 1-D DP approach for evaluating the performance of our 2-D warping-based recognizer, as described in Section 2.5. From this evaluation, we see the benefit of moving from 1-D to a recognition method that handles additional 2-D variation.

Pavlidis et al. [29] perform online (not offline) HR by comparing blending costs calculated using the physics based approach to shape blending developed by Sederberg and Greenwood [37] – an algorithm originally used to automatically create smooth graphical blends from one shape to another. Sometimes called shape morphing, the approach models a polygonal shape as a wire that can be bent or stretched into a second shape. The algorithm determines how to manipulate the wire into the second shape using the least amount of work. Singh et al. [41] extend the work of Pavlidis et al. to use shape blending costs to recognize 2-D shapes in general, including a small number of cursive words.

Like shape morphing, image morphing is a graphical technique, but is used to morph one image into another instead of just polygonal shapes. We use principles derived from the work minimization approach to image morphing by Gao and Sederberg [8] in our HR method to improve the warp mesh alignment as described in Section 2.4.5.

Unlike the previous recognition methods that just use DTW cost or shape blending cost as a direct metric of how different words are, we use these methods to align words, but then compute a distinct metric of how different the words are. We describe the metric in Section 2.4.6.

2.4 Methods

For a given pair of word images, I_0 and I_1 , with width/height w_0/h_0 and w_1/h_1 , we create corresponding rectangular meshes, M_0 and M_1 (Figure 2.2). Initial control point spacing for M_0 is $\max(h_0, 4)$, except for the last row and column of control points, which are placed at $y = h_0 - 1$ and $x = w_0 - 1$. The control points of M_1 (the warp mesh) are not spaced evenly, but instead are coarsely aligned by 1-D DP (Section 2.4.3). Morphing (Section 2.4.5) is then used to adjust the control points so that the warped medial axis pixels, A'_0 , align more closely to the medial axis pixels (A_1) of I_1 . We then use D'_0 , the distance map from A'_0 , to compute $C_{0 \rightarrow 1}$ (Section 2.4.6). $C_{0 \rightarrow 1}$ is the cost to match I_0 to I_1 .

Since the cost to match I_0 to I_1 is not necessarily the same as the cost to match I_1 to I_0 , we repeat the steps with I_0 and I_1 swapped to compute $C_{1 \rightarrow 0}$. The total word matching cost, $C(I_0, I_1)$, is the sum:

$$C(I_0, I_1) = C_{0 \rightarrow 1} + C_{1 \rightarrow 0} \quad (2.1)$$

Adding the two costs ensures that $C(I_0, I_1)$ is symmetric for a given pair of images regardless of order. $C(I_0, I_1)$ is the distance metric we use for word comparison. We call it “cost” to avoid confusion with distances in distance maps.

Recognition of a word is performed by computing the word matching cost between the word and each training example and using the label from the training example resulting in the minimum word matching cost.

I_0, I_1 : The two images being compared
 w_0, h_0, w_1, h_1 : Width, height of I_0 and I_1
 M_0, M_1 : Meshes defining the 2-D warp from I_0 to I_1
 $P_{c,r}^0$: Control point (vertex) in M_0 at col c , row r
 $P_{c,r}^1$: Control point (vertex) in M_1 at col c , row r
 A_0, A_1 : Medial Axis pixels of I_0 and I_1
 A'_0 : Pixels of A_0 after being warped (using M_0 to M_1)
 D'_0 : Distance map created from A'_0
 D_1 : Distance map created from A_1
 F_0, F_1 : Feature vectors for DP alignment
 $C_{0 \rightarrow 1}$: Cost of matching I_0 to I_1
 $C_{1 \rightarrow 0}$: Cost of matching I_1 to I_0
 $C(I_0, I_1)$: Total word matching cost between I_0 and I_1

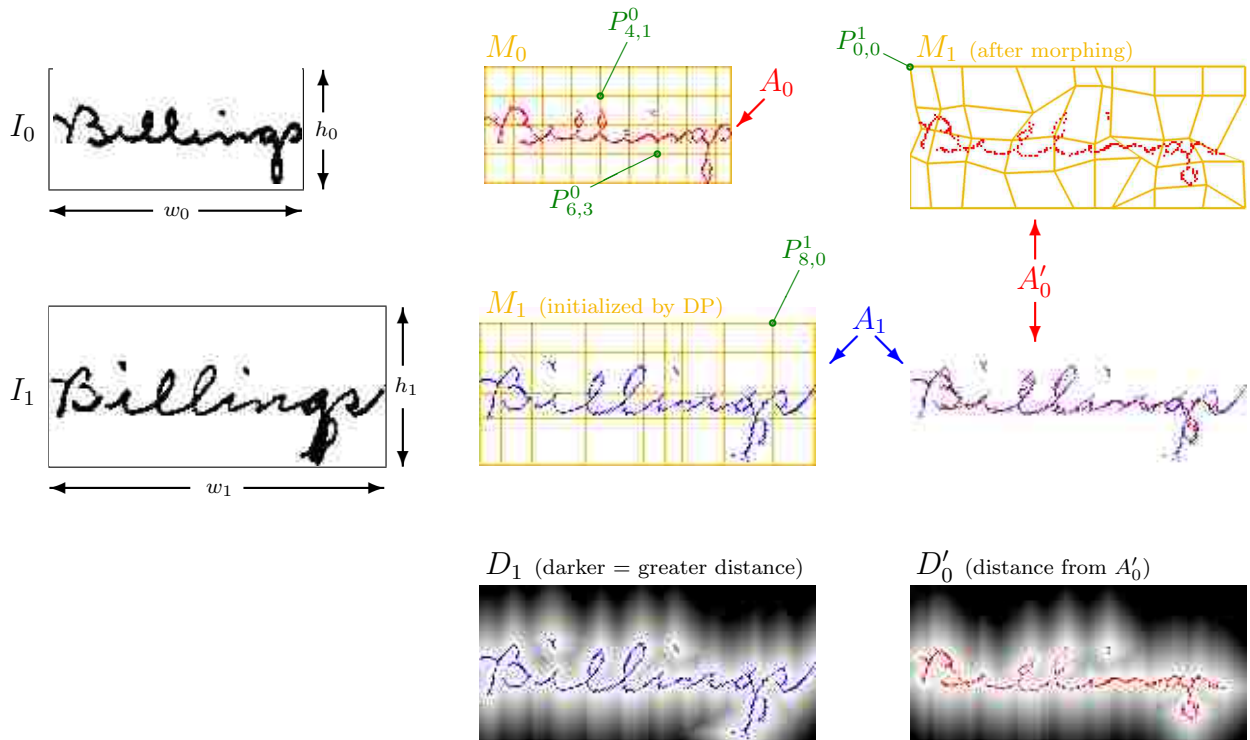


Figure 2.2: Reference key to symbols and notation, with illustrations for clarity.

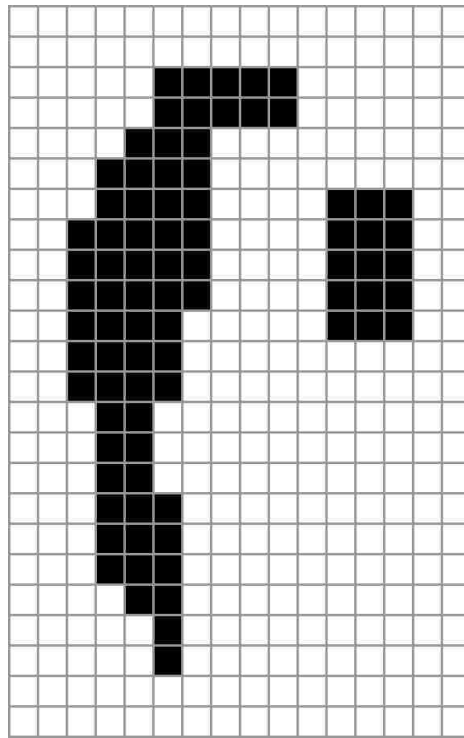
2.4.1 Preprocessing

We preprocess manually-segmented word images by performing background removal, slant correction, crop/pad, and binarization. Background estimation for background removal is computed with a large median kernel as described in [12]. After background removal, a global threshold value for each page is determined for later binarization using a method described in [14]. Slant correction consists of shearing the image horizontally after estimating the slant angle over the central region of each page image. The angle estimation uses ink runlengths accumulated into a histogram based on angle bins. Baseline estimation is used to determine whether to pad the top or bottom of the image. The image is then cropped to the left-most / right-most ink pixel after the slant removal. After all other preprocessing, the word image is binarized using the previously selected threshold.

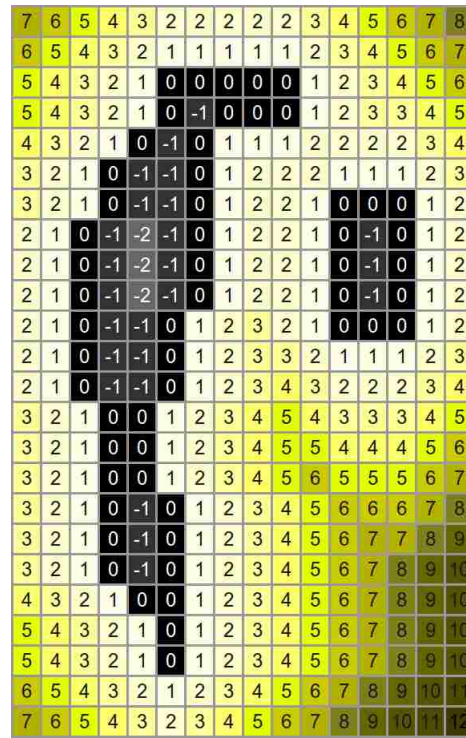
2.4.2 Distance Map and Medial Axis

We compute distance maps for bitonal images (Figure 2.3a) using a forward-backward algorithm very similar to the distance transform introduced by Rosenfeld and Pfaltz in [36]. Each pixel in the resulting distance map (Figure 2.3b) contains the Manhattan distance (in number of pixels) to the nearest edge of an ink component. The greater the distance from ink, the higher the value of the pixel in the distance map. Values within an ink component are zero (on the border with the background) or negative (within the component) – progressively increasing in magnitude as the center of the ink component is approached.

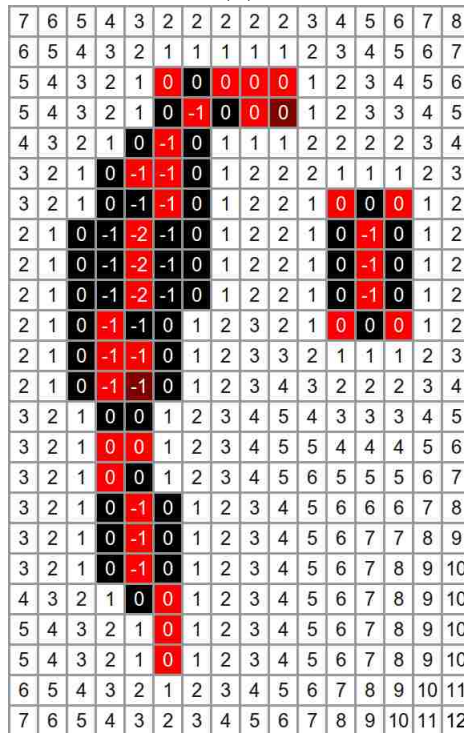
Medial axis pixels are those in the distance map with values less than or equal to zero that do not have any 4-connected neighbors more negative than themselves (Figure 2.3c). We remove from the result any pixels (shown in darker red) for which the North, Northwest, and West neighbors are all also medial axis pixels. This results in our final medial axis (Figure 2.3d).



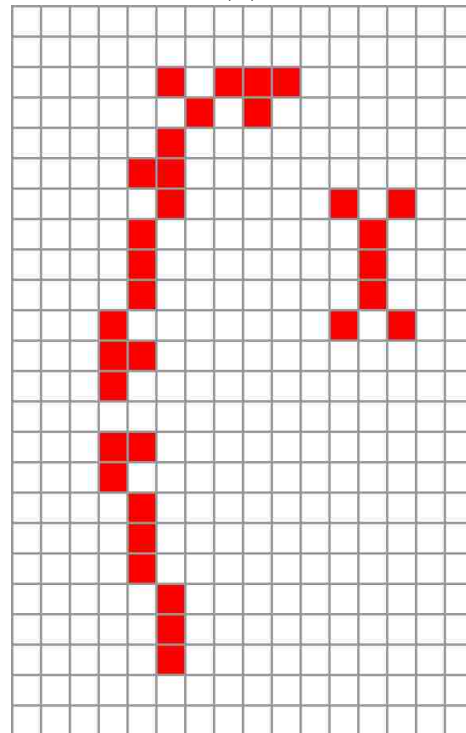
(a)



(b)



(c)



(d)

Figure 2.3: Distance map and medial axis. a) Original– ink pixels are black; b) Distance map; c) Medial axis pixels are red (dark red are removed); d) Final medial axis pixels.

2.4.3 Dynamic Programming for Coarse Mesh Alignment

For horizontal alignment of M_1 , we use the DTW algorithm described in [35]. Feature vectors F_0 and F_1 are computed from the normalized ink profile, upper word profile, lower word profile, and background to ink transition counts of the respective word images, I_0 and I_1 (Figure 2.4). The DTW function to build the DP alignment table (Figure 2.5a) is

$$D(i, j) = \min \left\{ \begin{array}{l} D(i-1, j) \\ D(i, j) \\ D(i, j-1) \end{array} \right\} + d(i, j), \quad (2.2)$$

where $d(i, j)$ is the cost to align $F_0(i)$ with $F_1(j)$, and is defined as

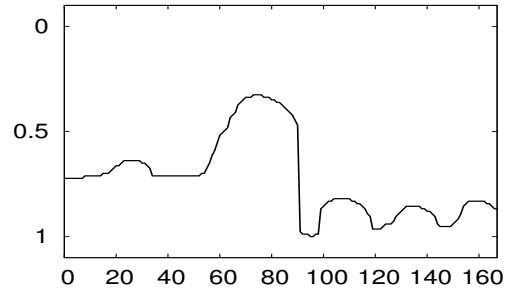
$$d(i, j) = \sum_{k=1}^4 (F_0(i, k) - F_1(j, k))^2, \quad (2.3)$$

where k is the index to access the four features in the vector at the alignment position (profile, upper/lower indentation profile, transition count). We also use the same Sakoe-Chiba band DP constraint with radius 15 as the authors of [35]. The Sakoe-Chiba constraint is a locality constraint that limits how much warping the dynamic programming algorithm permits when aligning the features of one word with those of another. This is accomplished by setting $D(i, j)$ to infinity (or a very high value) for all table positions outside of the Sakoe-Chiba band (the gray area in Figure 2.5a).

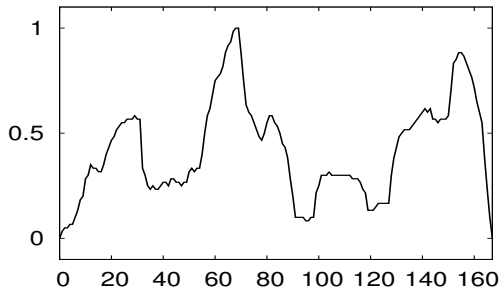
The alignment of F_0 and F_1 is found by following the DP path backward through the DP table when the DTW algorithm is complete. The alignment is used to map x-coordinates of the control points in M_0 to the corresponding x-coordinate to be assigned to the corresponding control point in M_1 . The same is done for y-coordinates of the control points (using the DTW result for vertical alignment) except that we only use a single-dimensional feature vector – just the ink profile of the word images projected onto the vertical axis, normalized to values between 0 and 255 (Figure 2.6). Therefore, the summation in Equation 2.3 is only for $k = 1$ when aligning y-coordinates of the control points.



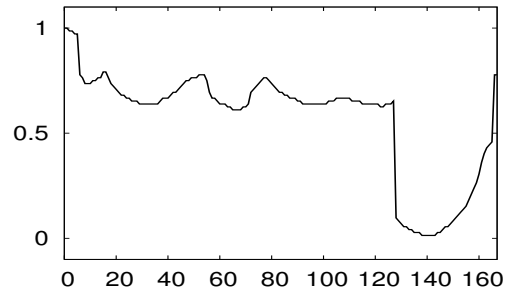
(a)



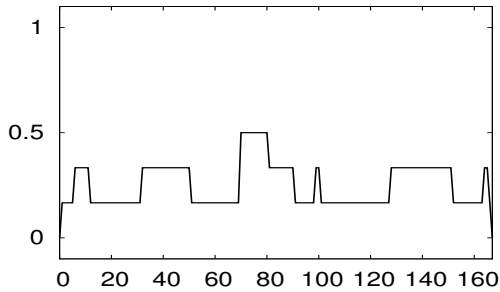
(d)



(b)



(e)



(c)

Figure 2.4: Whole word features. a) Word image. b) Projection profile. c) Background-to-ink transition count. d) Upper indentation profile (inverted for display). e) Lower indentation profile. (Word image from the Washington dataset – Appendix A.2)

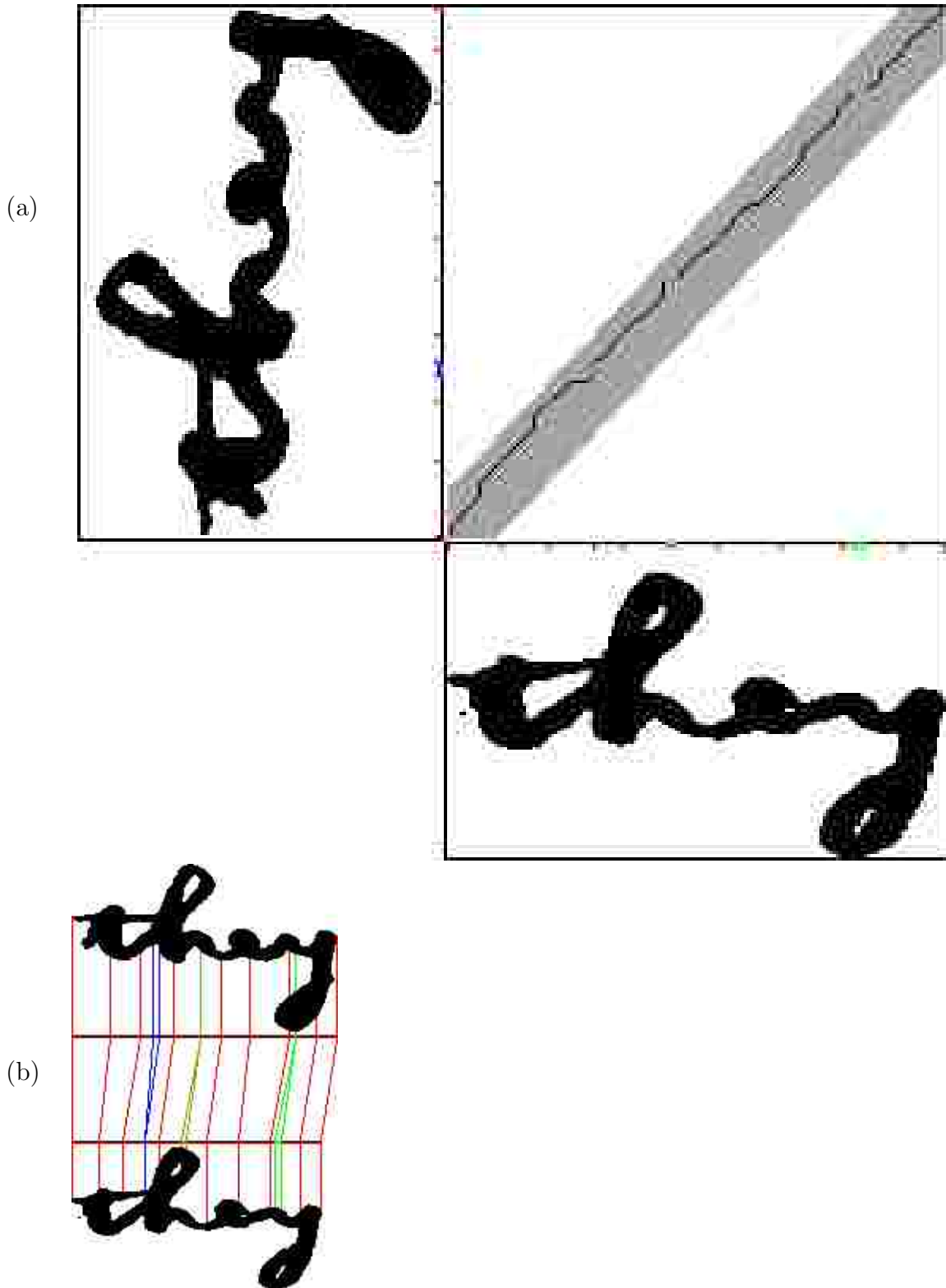


Figure 2.5: DP alignment of two instances of the same word. a) DP table and alignment path with some points along the path (and corresponding marks on edges of table) colored for easy reference (Sakoe-Chiba band is gray); b) DP alignment of two instances of “they” using colored points for reference.

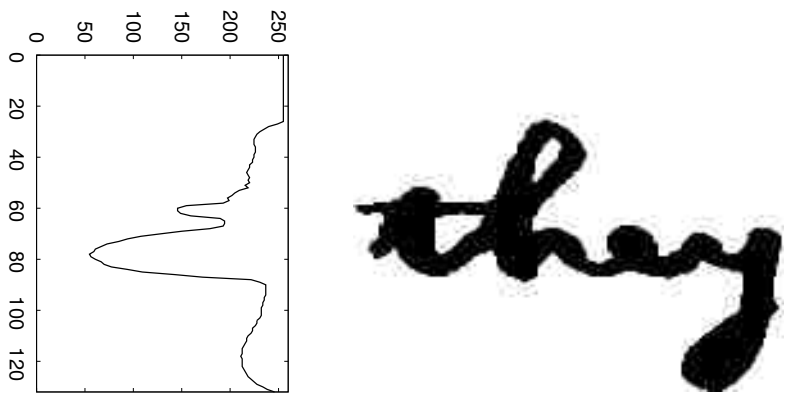


Figure 2.6: Word profile feature for vertical DP alignment. We project the word profile onto the vertical axis. The length of the feature vector is equal to the height of the word image. Values in the feature vector are from 0 to 255.

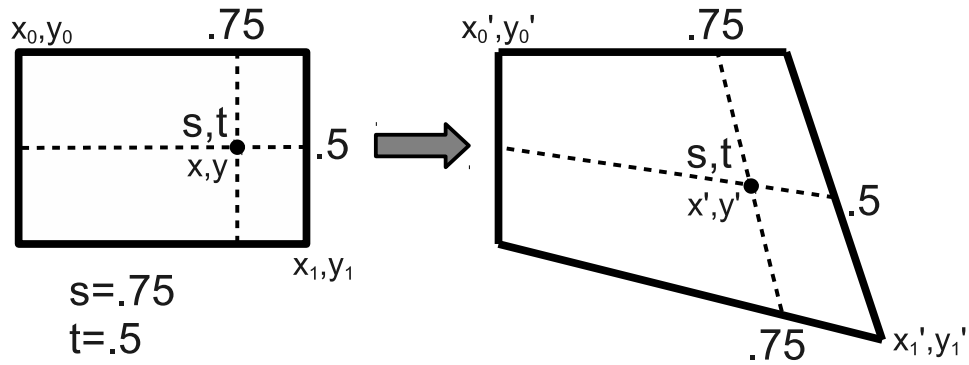


Figure 2.7: Warping from rectangular to non-rectangular mesh quads. The warped position, x', y' , of any point s, t within a quad is easily computed by bilinear interpolation of the point within the warped quad vertices.

2.4.4 Warping Coordinates

Since quads in M_0 are rectangular, the relative coordinate s, t within a quad (s and t having range $[0, 1]$) is easy to calculate for any corresponding image point x, y that is within the quad (Figure 2.7). The values are calculated as $s = (x - x_0)/(x_1 - x_0)$ and $t = (y - y_0)/(y_1 - y_0)$. The warped image coordinate x', y' is then computed by bilinear interpolation of s, t within the vertices of the corresponding quad of the warp mesh, M_1 .

2.4.5 Morphing for Warp Mesh Improvement

In image morphing, a start and end mesh define a warp from one image to another. Interpolating positions and pixel colors at evenly-spaced time slices between the start and end results in a series of images forming a graphical morph from one image to the other. The warp mesh is often defined by manually specifying points of correspondence between the images. The work minimization approach to image morphing by Gao and Sederberg [8] can automatically generate an end mesh. It does so by iteratively *improving* the mesh (adjusting its control points to reduce the overall morph cost according to a work equation), and *refining* the mesh (subdividing it into more detailed quads). The work equation they use for improving the mesh includes costs for work due to angle change, stretching, and pixel color change.

Algorithm 2.1 Morphing algorithm for Word Warping

Inputs:

h_0 // height of image I_0 (to calculate how many refinements to do)
 M_0 // unwarped mesh (together with M_1 , this defines the warp function)
 M_1 // warp mesh (with its control points, P^1 , initialized by DTW algorithm)

Outputs:

```
 $M_1$  // final warp mesh (after morphing)
00 refine_count= 0 // lines 00 – 04 heuristically
01  $m = \max(4, h_0/4)$  // determine refine_count
02 while  $m > 16$  // based on  $h_0$  (image height)
03  $m = m/2$ ;
04 refine_count=refine_count+1
05 for mesh_level=1 to refine_count
06 for imp=1 to improve_count // (we use improve_count=3)
07 // improve:
08 for each  $P_{c,r}^1$  in  $M_1$  // (control point at col  $c$  row  $r$ )
09  $x, y = P_{c,r}^1$ 
10  $min = placement\_cost_{x,y}$  // (Equation 2.4)
11  $XY_{min} = x, y$ 
12 for each  $x, y$  in search area of  $P_{c,r}^1$ 
13 if  $placement\_cost_{x,y} < min$  then
14  $min = placement\_cost_{x,y}$ 
15  $XY_{min} = x, y$ 
16  $P_{c,r}^1 = XY_{min}$  // (update control pt  $c, r$  in  $M_1$ )
17 if mesh_level < refine_count
18 // refine:
19 increase resolution of  $M_0, M_1$  by factor of 2
```

We adapt the automatic morphing algorithm of Gao and Sederberg to the application of aligning handwritten words. Algorithm 2.1 describes how we compute the warp mesh to align the handwritten words.

In lines 00–04 of the algorithm, we use the height, h_0 , of the image to heuristically determine how many times to refine (subdivide) the mesh. For each refinement level of the mesh (lines 05–19), we first *improve* the mesh a few times (lines 06–16) and then *refine* the mesh if more refinement levels of the mesh still are still needed (lines 17–19). We now describe these sections of code in more detail.

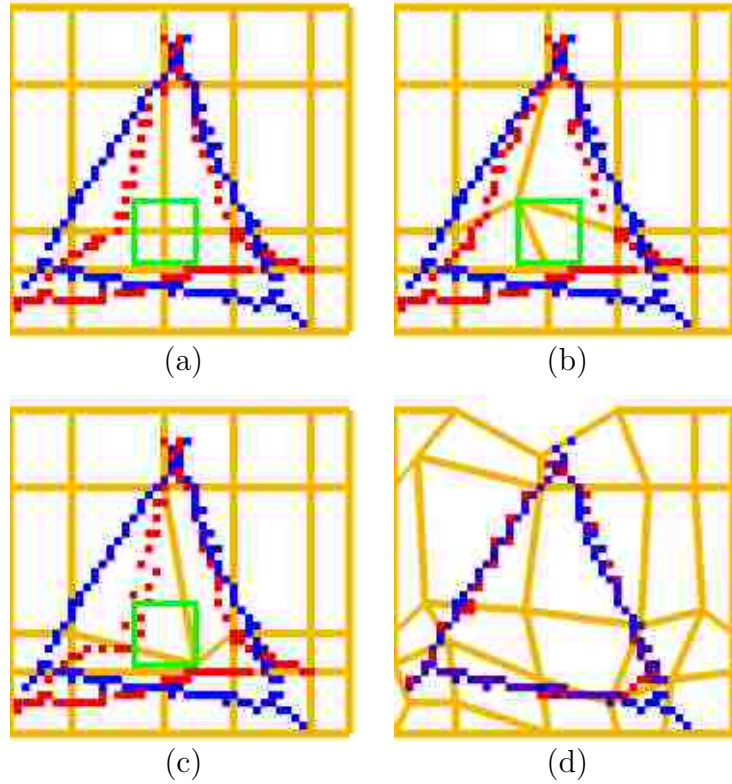


Figure 2.8: *Improve* step of the morphing algorithm. Each control point is moved to every position within a nearby search area and then assigned to the lowest cost position (the position that resulted in the best alignment of the medial axes). We illustrate with control point $P_{2,2}^1$. a) The search area (green) of $P_{2,2}^1$ before the *improve* step; b) Moving $P_{2,2}^1$ to the top-left of its search area pushes the red pixels closer to the blue; c) Moving $P_{2,2}^1$ to the bottom-right pulls red away from blue; d) The improved mesh after choosing the lowest cost position of all control points, and iterating several times.

In the *improve* step (lines 07–16), each control point, $P_{c,r}^1$, is in turn moved to the lowest cost position within its current search area (Figure 2.8). The search area is constrained to a rectangular region surrounding the current position of the control point. The region extends 0.4δ in each direction, where δ is the current control point spacing in M_0 (δ gets halved every time a *refine* occurs). The search area is also constrained by the control points around it. For example, $P_{c,r}^1$ cannot go above any of the 3 control points above it in its 8-neighborhood.

The cost of placing $P_{c,r}^1$ at any given search position x, y within the search area is:

$$placement_cost_{x,y} = d\Delta_{x,y} + \frac{1}{n+1} \sum_{k=1}^n D_1(A'_0[k]), \quad (2.4)$$

where A'_0 are warped using the search position as the position of $P_{c,r}^1$ in M_1 , $n = \|A'_0\|$, $D_1(A'_0[k])$ is the value in D_1 at the position of the k^{th} warped medial axis point in A'_0 , $\Delta_{x,y}$ is the Euclidean distance from x, y to the current position of $P_{c,r}^1$, and d is a constant used to weight the Δ term (we currently use $d = 0.01$). In effect, the cost of placing $P_{c,r}^1$ at this search position is the average distance the medial axis points of I_0 would be from the nearest medial axis pixels of I_1 if we were to place $P_{c,r}^1$ at this search position. The Δ term introduces a small preference for keeping $P_{c,r}^1$ at or near its current position when multiple search positions are otherwise of equally low cost.

We actually do not use the entire set A'_0 of medial axis pixels during cost calculation for search positions. Since only the pixels within the four mesh quads sharing control point $P_{c,r}^1$ as a vertex move when the control point is adjusted, only the costs associated with those points will affect the cost at any given search position for that control point. To speed up processing, we ignore all A'_0 points outside of the four adjacent quads.

The *refine* step (lines 18–19) doubles mesh resolution by adding control points at the midpoints of each quad/edge in M_0 and M_1 (Figure 2.9). Although refinement, itself, does not change the warped position of pixels, it does allow subsequent *improve* steps to work at a finer level of detail.

2.4.6 Word Matching Cost

After M_1 has been aligned using DP and morphing, we compute the warped medial axis, A'_0 , of I_0 (red-shaded pixels in Figure 2.1g). We then compute the distance map, D'_0 , of the warped medial axis A'_0 (Figure 2.1h). Most of the A'_0 pixels should be closely aligned to pixels of A_1 due to morphing. What tells us if the words are actually similar or not is if the

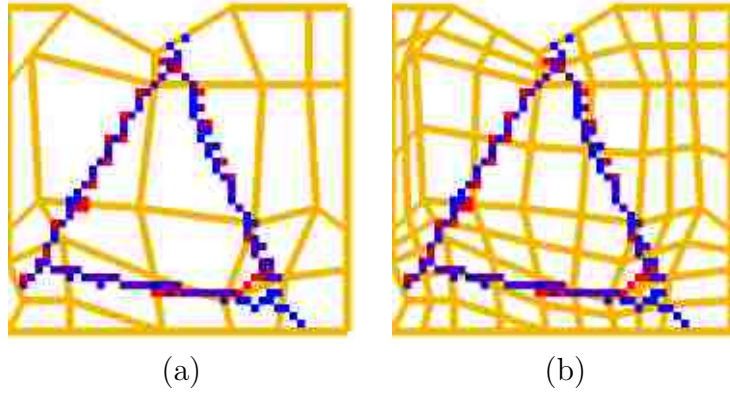


Figure 2.9: *Refine* step of the morphing algorithm. The resolution of the mesh is increased by a factor of two. a) Mesh before *refine* step; b) after *refine* step.

pixels of A_1 (blue-bordered pixels in Figure 2.1h) also align well to pixels of A'_0 , or whether their values in the distance map are high, suggesting that the words are not similar. We compute $C_{0 \rightarrow 1}$, the cost to match I_0 to I_1 , as

$$C_{0 \rightarrow 1} = \frac{1}{\|A_0\| + 1} \sum_{i=1}^{\|A_1\|} D'_0(A_1[i]), \quad (2.5)$$

where $D'_0(A_1[i])$ is the value in D'_0 of the location of the i^{th} medial axis pixel in A_1 .

2.5 Experiments

We perform experiments on two datasets of labeled word images. The first dataset consists of words from a set of 20 pages of George Washington’s manuscripts [21]. The second consists of words from pages of Jennie Leavitt Smith’s diary, downloaded from the “Mormon Missionary Diaries” online collection of the Brigham Young University Harold B. Lee Library, available at <http://www.lib.byu.edu/dlib/mmd/>. We manually segment and label each word to provide ground truth for our experiments (Appendix A).

For each dataset, we select the first 1,000 word images as training examples for which the recognition system is allowed to look at the labels. We use the next 1,000 words (which are not used as training examples) as test data. We compare each test word with the training words and assign it the label from the training word that it most closely matches. This is

done both using our 2-D word warping method and also using just the 1-D Dynamic Time Warping alignment cost [35]. We also record the word warping results when using only coarsely-aligned meshes without morphing.

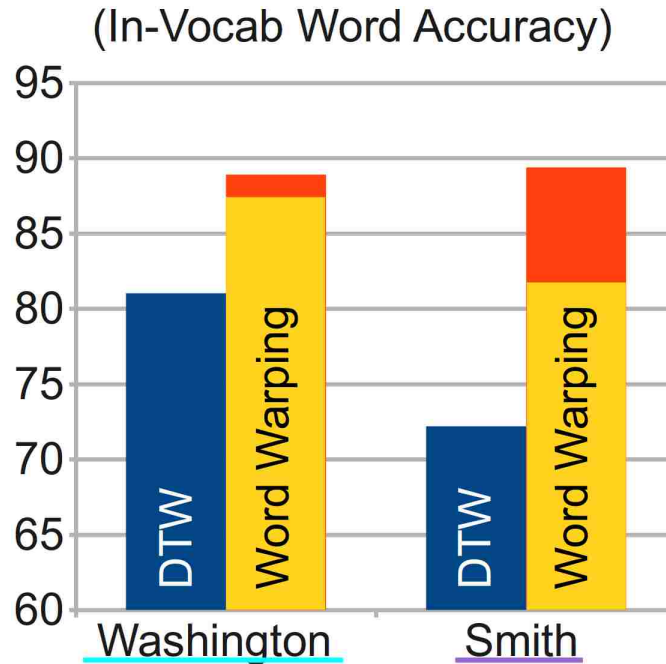
We assess the recognition accuracy of each method by comparing the ground truth labels with the labels assigned by the recognizer. Recognition accuracy is calculated as the number of test words labeled correctly by the recognizer (the number given the same label as its ground truth), divided by the total number of test words. The string comparison between the label and ground truth is case-sensitive.




Since many of the test words are *Out of Vocabulary* (OoV) words, meaning no training examples have the same label as their ground truth, we also report the recognition accuracy with respect to the number of in-vocabulary words (total test words minus the number of OoV test words).

2.6 Results and Discussion

Our word warping method is noticeably more accurate than DTW (the baseline 1-D DP method) on both datasets (Figure 2.10). Even without using morphing to improve the warp mesh, word warping with coarsely-aligned meshes results in an increase in recognition accuracy of 6.41% for in-vocabulary words with the Washington manuscripts dataset and 9.61% with the Smith diary dataset. Recognition is even better when we include the morphing step. For the Washington dataset, in-vocabulary accuracy is 88.90%, an increase of 7.88% from the baseline (DTW). For the Smith dataset, we see a larger improvement of 17.19% to 89.38%.

Morphing only contributes 1.47% ($7.88\% - 6.41\%$) to the accuracy of the Washington dataset, however, it contributes 7.58% ($17.19\% - 9.61\%$) to the accuracy of the Smith dataset. We observe that the the Washington penmanship is exceptionally consistent but there is more variation in the Smith dataset, requiring better alignment in order to recognize words. We are encouraged by this result because it suggests that word warping with morphing is adept



<u>Washington Dataset</u> - 1,000 test words (748 in-vocabulary)		
Method	Total Accuracy (# correct / # possible)	In-Vocab Accuracy (# correct / # possible)
DTW (1-D DP)	60.60% (606 / 1000)	81.02% (606 / 748) 
Word Warping (only coarse aligned)	65.40% (654 / 1000)	87.43% (654 / 748) 
Word Warping (morphing aligned)	66.50% (665 / 1000)	88.90% (665 / 748) 




<u>Smith Dataset</u> - 1,000 test words (791 in-vocabulary)		
Method	Total Accuracy	In-Vocab Accuracy
DTW (1-D DP)	57.10% (571 / 1000)	72.19% (571 / 791) 
Word Warping (only coarse aligned)	64.70% (647 / 1000)	81.80% (647 / 791) 
Word Warping (morphing aligned)	70.70% (707 / 1000)	89.38% (707 / 791) 

Figure 2.10: Experimental Results – Word Recognition Accuracy

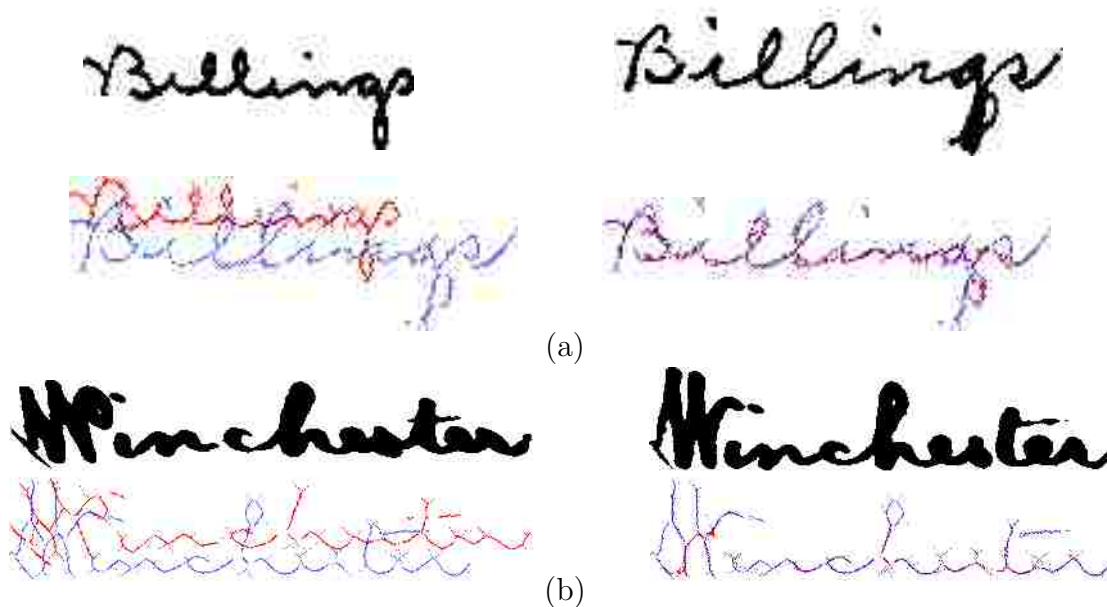


Figure 2.11: Alignment of medial axes. a) Top: two occurrences of the word “Billings” (Smith dataset); Bottom: corresponding medial axes before alignment (left) and after (right). b) “Winchester” (Washington dataset).

at handling local variation and should generalize to datasets with multiple authors. We revisit this hypothesis in Chapter 4, testing our method on a dataset with many authors. This ability to handle variation may even allow us to use synthetically-created training data to improve the OoV recognition accuracy, which we leave for future work. Figure 2.11 shows our medial axis alignment using morphing. As these examples show, our algorithm is often able to align different instances of a word quite well, even when the words do not align well to begin with. Good alignment results in low word matching costs for the words, which is exactly the desired result.

Many of the recognition errors that we see with our method are very minor. For example, some words differ only by the capitalization of the initial character (Figure 2.12a). Others differ only by a single letter, such as “come” vs. “came” and “them” vs. “then” (Figure 2.12b). In Figure 2.12c, we see an example of “Winward” vs. “Winwards” in which the only difference is the final “s”. In all of these cases (and many others), the differences between the mismatched words are very small, and in a few cases it is even difficult for us

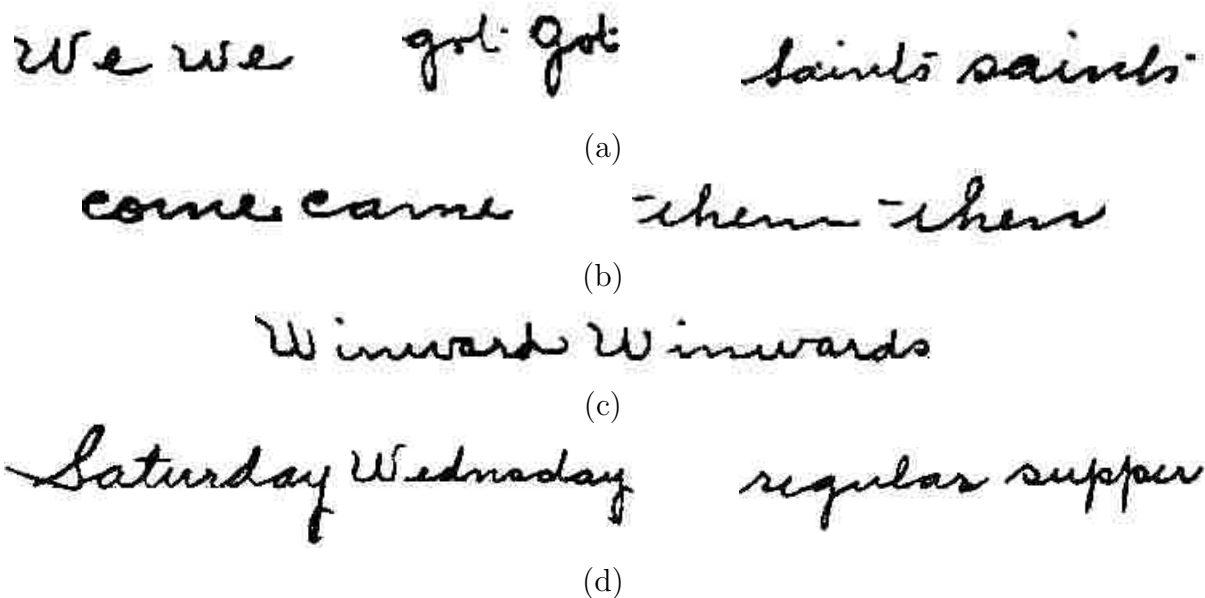


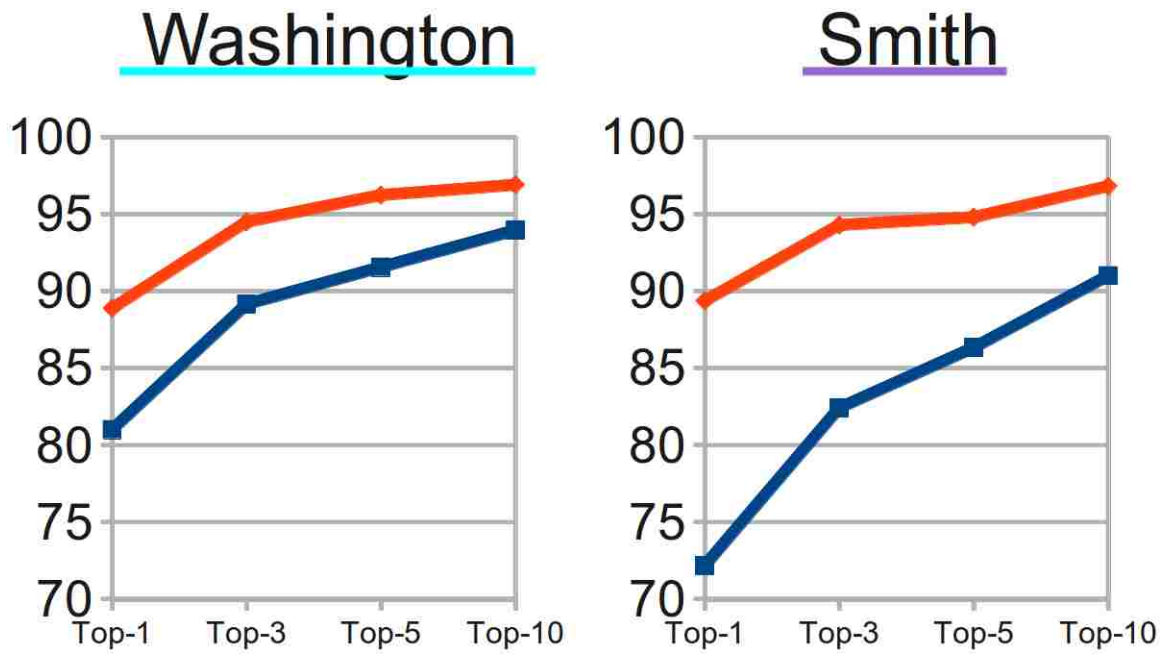
Figure 2.12: Examples of recognition errors (Smith dataset). Test words followed by the erroneous best match. a) Errors only because of capitalization differences; b) Very similar words: “come” vs. “came” and “them” vs. “then”; c) “Winward” vs “Winwards”; d) Some more obvious errors.

(humans) to definitively say which word we are looking at when the word is isolated instead of in context. It is not at all surprising that such similar words are sometimes confused with each other by our automatic HR method. However, some errors are more conspicuous, and the mistakes are more surprising to us (Figure 2.12d). In Chapter 4, we provide significant additional analysis of the recognition errors from both the Washington dataset and Smith dataset. We examine the types of errors that occur, and why they occur. We pay special attention to the more conspicuous types of errors, such as those in Figure 2.12d. We also experiment with some minor modifications to our algorithm and parameters to improve our recognition accuracy.

As a final observation, we find that for many recognition errors the correct match is ranked very near the top (Figure 2.13). In fact, the correct result is ranked in the top 3 matches more than 94% of the time for both datasets (Figure 2.14). This makes us optimistic that our accuracy will improve in the future when we incorporate language models (e.g., word tri-grams) to use surrounding context in selecting the most likely of the top few matches.

Test word word	Best match word (cost)	2nd-best match word (cost)	3rd-best match word (cost)
<i>doctored</i> doctored	<i>lessons</i> lessons (1.990952)	<i>decided</i> decided (2.168316)	<i>doctored</i> doctored (2.271915)
<i>also</i> also	<i>all</i> all (1.960903)	<i>all</i> all (2.173552)	<i>also</i> also (2.197479)
<i>We</i> We	<i>we</i> we (1.355442)	<i>We</i> We (1.676988)	<i>we</i> we (1.856132)
<i>them</i> them	<i>then</i> then (1.642179)	<i>then</i> then (1.931055)	<i>them</i> them (1.953888)
<i>got</i> got	<i>got</i> Got (1.914398)	<i>got</i> got (2.369706)	<i>gave</i> gave (2.522078)
<i>I</i> I	<i>a</i> a (1.391941)	<i>I</i> I (1.706861)	<i>9</i> 9 (1.871795)
<i>practise</i> practise	<i>practised</i> practised (2.270567)	<i>practise</i> practise (2.356809)	<i>practise</i> practise (2.489450)
<i>at</i> at	<i>we</i> we (2.277729)	<i>at</i> at (2.280873)	<i>we</i> we (2.352892)

Figure 2.13: Examples of correct answer (green) in top 3 matches.



<u>Washington Dataset</u> - 1,000 test words (748 in-vocabulary)				
Method	Top-1	Top-3	Top-5	Top-10
<u>DTW</u> (1-D DP)	81.02%	89.17%	91.58%	93.98%
	(606 / 748)	(667 / 748)	(685 / 748)	(703 / 748)
<u>Word Warping</u>	88.90%	94.52%	96.26%	96.93%
	(665 / 748)	(707 / 748)	(720 / 748)	(725 / 748)

<u>Smith Dataset</u> - 1,000 test words (791 in-vocabulary)				
Method	Top-1	Top-3	Top-5	Top-10
<u>DTW</u> (1-D DP)	72.19%	82.43%	86.35%	91.02%
	(571 / 791)	(652 / 791)	(683 / 791)	(720 / 791)
<u>Word Warping</u>	89.38%	94.31%	94.82%	96.84%
	(707 / 791)	(746 / 791)	(750 / 791)	(766 / 791)

Figure 2.14: Correct answer in top- N results

2.7 Conclusion and Future Work

We have presented a 2-D warping method for comparing words to each other for offline handwriting recognition. Our method takes advantage of 2-D warping to get better word matching results. Our early tests on this method are encouraging, showing noticeable improvement over 1-D DP methods. We also find that many of the errors made by our method are very minor, and the correct result is within the top few matching words. This leads us to believe that minor modifications to our algorithm and the incorporation of language models to leverage word context will increase our accuracy in the future. While this paper introduces our novel approach and provides initial results, we anticipate that future work will allow us to build on the groundwork of this paper to improve our method and achieve even better results.

Chapter 3

Offline Signature Verification and Forgery Detection Using a 2-D Geometric Warping Approach

In this chapter we show how our word comparison approach can be applied to signature verification and forgery detection, an application area with the potential to prevent enormous amounts of fraud, including fraudulent financial transactions which can total hundreds of millions of dollars in a year [1, 4, 45]. This chapter is based on the paper “Offline Signature Verification and Forgery Detection Using a 2-D Geometric Warping Approach,” published at the 21st International Conference on Pattern Recognition (ICPR 2012) [16]. This chapter contains some modifications from the published version of the paper. Of particular note are: 1) expanded analysis and discussion of our results, and 2) the addition of Figures 3.1, 3.2, and 3.4.

3.1 Abstract

We present a method of discriminating between authentic and forged signatures using 2-D geometric warping. After an initial coarse-alignment step, we use an automatic morphing correspondence algorithm to compute 2-D geometric warps that align the strokes of a questioned signature with those of known reference examples. We use distance maps to compute a difference metric, and then either accept the signature as genuine or reject it as a forgery depending on how different it is from the reference examples.

Our method achieves equal error rate (EER) accuracies of about 94%–96% on our English dataset of blind forgeries and 87%–91% on casual forgeries (unpracticed imitations).

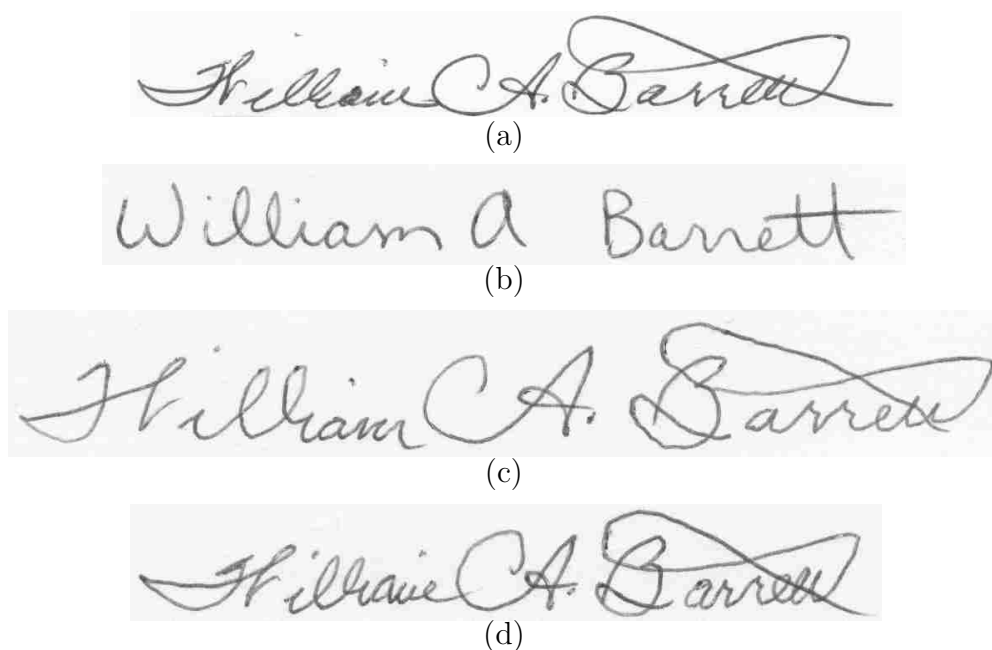


Figure 3.1: Various skill levels of forgeries. a) Genuine signature; b) Blind forgery; c) Casual forgery; d) Skilled forgery.

Further evaluation of our method using the SigComp2011 competition dataset shows that our accuracies for skilled forgeries are comparable to those of several other recent methods. We are particularly encouraged by the performance of our method on the Chinese portion of the dataset, in which our EER accuracy (74%) is better than all but one of the systems that participated in the 2011 competition.

3.2 Introduction

Forged signatures are used for a variety of illicit purposes, including falsifying documents, identify theft, and fraudulent check or credit card transactions. While some are *skilled* forgeries created after practicing, many are *casual* forgeries created by imitating a genuine signature without practice, or even *blind* forgeries made without ever seeing the genuine signature (Figure 3.1). Systematic detection of forgeries using automatic signature verification has the potential to save banks and businesses enormous amounts of money, and save individuals the inconvenience of dealing with the aftermath of their accounts or identities being misused by criminals.

We present a signature verification method based on a 2-D geometric warping approach that we originally developed for whole-word offline handwriting recognition. [15] For handwriting recognition, we used the approach to compute a difference metric between an unknown word image and examples of known (labeled) word images. We used minimum difference to classify the unknown word image and assign it a textual label.

In this paper, we adapt the approach to the task of signature verification. Given a few reference examples known to be written by a particular person, we compare a questioned signature to the reference signatures of that person. The difference metric from [15] is used to quantify how different the questioned signature is from each reference signature. If the questioned signature is similar enough (on average) to the references, it is considered to be genuine. Otherwise, it is rejected as a forgery. We compute the threshold for classification as a function of the variance of the reference signatures, combined with a tuning parameter that allows the system to be biased either against false accepts or false rejects depending on the needs of the user.

We evaluate our method with our own dataset of English signatures, and also with the publicly available dataset that was used in the ICDAR 2011 Signature Verification Competition for Online and Offline Skilled Forgeries (SigComp2011). [23] In addition to genuine signatures, our English dataset has both blind forgeries and casual forgeries. We created this dataset in the absence of any known publicly available dataset of blind and casual forgeries. The SigComp2011 dataset has skilled forgeries in both Dutch and Chinese. We use only the offline portion of that dataset.

Our early results are encouraging (Section 3.6). Our method yields good results for both blind and casual forgeries. It is also comparable to several other methods for Dutch skilled forgeries and outperforms all but one method from the ICDAR 2011 competition for Chinese skilled forgeries. We believe we can improve our accuracy even more with future work.

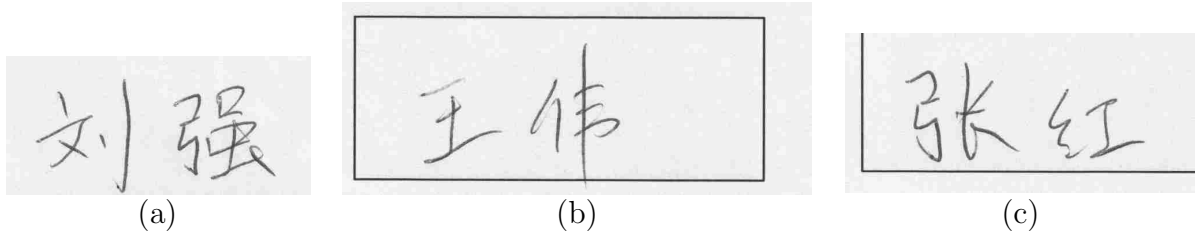


Figure 3.2: Illustration of inconsistent cropping for Chinese signatures (using our own images, since we are not permitted to reproduce images from the dataset). a) Form box properly cropped from image; b) form box not cropped from image; c) form box only partially cropped from image.

3.3 Methods

Besides some minimal preprocessing (Section 3.3.1), our method consists of two main parts:

1. Computing a classification threshold (Section 3.3.3)
2. Classifying questioned signatures as genuine or forgeries (Section 3.3.4)

Central to both of these parts is the difference metric described in Section 3.3.2.

3.3.1 Preprocessing

We preprocess signatures by scaling them down to half their original width and height and then performing binarization using a method we introduced in [14].

For Chinese signatures in the SigComp2011 dataset, we perform additional preprocessing before scaling and binarizing because some of the images have the form box around the signature, while some do not, and others have only portions of the box (Figure 3.2). We reduce noise with a 3x3 median filter, compute horizontal and vertical projection profiles, and then analyze the profiles to detect if and where box lines are present. Box lines are detected at positions where the smoothed profile passes a threshold of 0.85 (empirically chosen for the Chinese dataset) on either side of the ink. We crop the image just inside of detected box lines.

3.3.2 Comparing Signatures by 2-D Warping

To quantify the difference between two signatures, we use the difference metric that we introduced and described in detail in [15]. The metric, referred to as the *word matching cost*, is written as $C(I_0, I_1)$, where I_0 and I_1 are the two word images (signatures) being compared.

We extract the medial axes (centers) of I_0 and I_1 (Figure 3.3a), and coarsely align them using Dynamic Time Warping [35]. A more complete alignment is performed (Figure 3.3c) with our word-morphing correspondence algorithm (details available in [15]):

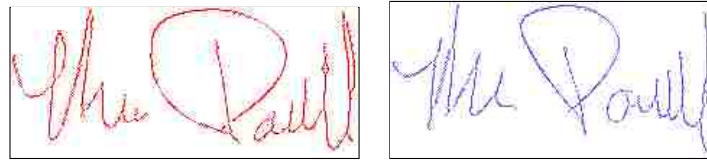
- 1: For each control point (mesh intersection), P :
- 2: For each x, y location around P :
- 3: Warp red medial axis (using P at x, y)
- 4: Measure alignment (use distance map)
- 5: Place control point P at x, y of best-alignment
- 6: Iterate / refine mesh until morph complete

Genuine signatures tend to align better than forgeries (Figure 3.3d). After aligning the medial axes, we use distance maps to compute the “difference,” $C(I_0, I_1)$.

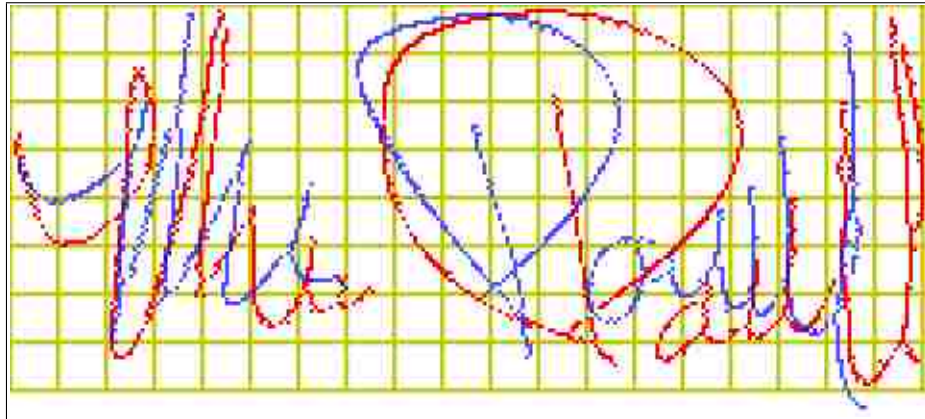
3.3.3 Computing Classification Threshold

The classification threshold, T , is used in Section 3.3.4 to classify signatures as forgeries or genuine. T is computed independently for each person from reference examples of their genuine signature.

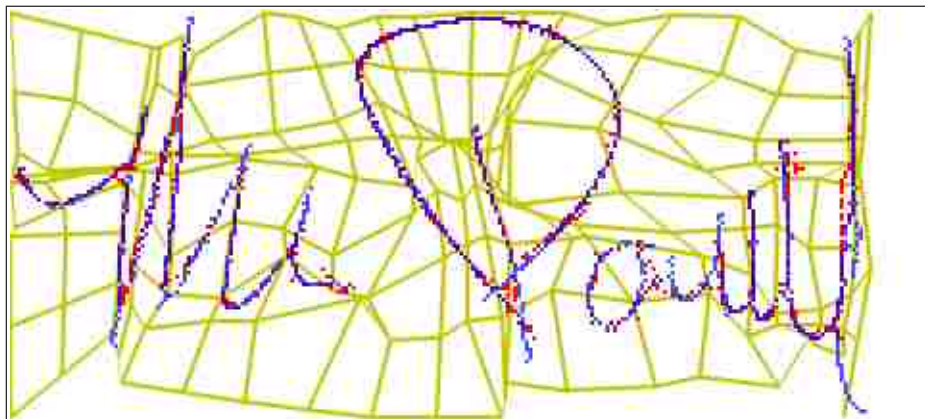
Given a set, R , of reference signatures (with the number of reference signatures being $\|R\|$), each reference signature, R_i , is compared to each of the other reference signatures, R_j , for that same person. The comparison is done by using the difference metric, $C(R_i, R_j)$, described in Section 3.3.2.



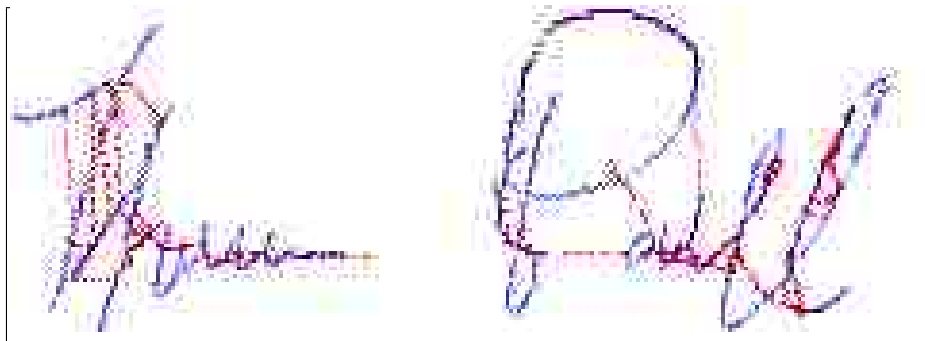
(a)



(b)



(c)



(d)

Figure 3.3: Comparing two signatures. a) Medial axes of two genuine signatures; b) Overlay of both medial axes and unwarped mesh; c) After warping red to align with blue using our word-morphing correspondence algorithm. Better alignment is lower “difference”; d) Warping red to this (blue) forgery gives poor alignment, resulting in a higher “difference” value.

For each R_i , its average difference from the other reference signatures is

$$\overline{C}_i = \frac{1}{\|R\|} \sum_{j=1}^{\|R\|} C(R_i, R_j). \quad (3.1)$$

Once we compute \overline{C}_i for each R_i in R , we then compute the variance, σ^2 , of all of the \overline{C}_i 's. Finally, we compute T as a function of the variance:

$$T = t\sqrt{\sigma^2} = t\sigma, \quad (3.2)$$

where t is a parameter that allows the user to tune the system to be more lenient or strict on what is considered to be a genuine signature. For some applications, the user may want to make the system more lenient to avoid false rejects, while for other applications the user may want to avoid false accepts.

As a refinement step, we check for reference signatures that are very different from the others (those for which $\overline{C}_i > T$). If there are any, we recompute the average differences and the variance while ignoring them. We also ignore them when we compare questioned signatures to the reference signatures for classification (Section 3.3.4).

3.3.4 Accepting and Rejecting Signatures

To classify a questioned signature, Q , as either genuine or as a forgery, we compute the average difference between Q and each reference signature, R_i :

$$\overline{C}_Q = \frac{1}{\|R\|} \sum_{i=1}^{\|R\|} C(Q, R_i), \quad (3.3)$$

where $C(Q, R_i)$ is the difference metric in Section 3.3.2. We then compare \overline{C}_Q to the classification threshold, T , (Section 3.3.3). If $\overline{C}_Q \leq T$, we accept Q as a genuine signature.

Otherwise, Q is rejected as a forgery.

3.4 Datasets

We use two datasets for our experiments (Sections A.4 and A.5 of Appendix A). The first is our own dataset of English signatures. It contains 192 genuine signatures (16 authors, 12 signatures each) and 256 forgeries (8 authors, 16 blind and 16 casual forgeries each). For each author, we collected the 16 genuine signatures in a single session. We collected forgeries from a set of volunteers who did not see the genuine signatures in advance. For the random forgeries, we provided them with typed names to forge. We then allowed them to look at the genuine signatures while they wrote the casual forgeries without any practice.

The other dataset is the offline portion from the ICDAR 2011 Signature Verification Competition for Online and Offline Skilled Forgeries (SigComp2011). The participants were allowed to practice forging these signatures before the forgeries were collected for the dataset. The Chinese test set has 116 genuine reference signatures by 10 authors and 487 questioned signatures (120 genuine, 367 forgeries). The Dutch test set has 648 reference signatures by 54 authors and 1286 questioned (648 genuine, 638 forgeries). [23]

3.5 Experiments

To test the accuracy of our method for blind forgeries, we use our English signature dataset. For each author, we select 6 of the 12 genuine signatures as the author’s reference signatures. We use the other 6 genuine signatures and 8 blind forgeries as questioned signatures for that author. We repeat the test using the other 6 genuine signatures as references and the original 6 reference signatures as questioned signatures. We report the results of both tests (“split 1” and “split 2” of the genuine signatures used for reference, respectively). We test casual forgeries in a similar manner, again using two splits of genuine signatures for reference.

Since accuracy and error rates are dependent upon the classification threshold (Section 3.3.3), we test across a range of values for the system parameter t . For each value of t , we calculate classification accuracy (ACC), false accept rate (FAR), and false reject rate

(FRR) of the system. We use the results to report the value of t that (approximately) results in equal error rate (EER). EER is when FAR is equal to FRR. EER can also be visualized on receiver operator characteristic (ROC) curves as the point on the ROC curve through which the (yellow) diagonal line passes (Figure 3.4).

We evaluate our system for both Dutch and Chinese skilled forgeries using the Sig-Comp2011 dataset. The genuine signatures in the dataset are already designated as either reference or questioned, so we do not test multiple splits. For these skilled forgeries, we compare our EER accuracy to the EER accuracies reported in [23] that were achieved by the systems that participated in the ICDAR 2011 competition.

3.6 Results

ROC curves for each dataset are shown in Figure 3.4. These curves illustrate how the system performs in the trade-off between catching more forgeries versus triggering fewer false alarms. In general, the closer a curve approaches to the top-left corner of an ROC curve, the better the system is considered to perform. It is clear that our system performs best on blind forgeries, followed by casual forgeries, and worst on skilled forgeries, as would be expected. For skilled forgeries, it performs better on Dutch than on Chinese. While it is common in ROC analysis to consider a point near the “knee” of the curve as the best point of the trade-off, the actual choice is very dependent on the particular application. For the purpose of comparison with other methods, we use a value of t that is near EER (visualized by the yellow diagonal), which is also reasonably close to the knee of the curve for these datasets.

We report the EER accuracy of our method in Table 3.1 and Figure 3.5. When the tuning parameter, t , of our system is adjusted approximately for equal error rate (EER), the accuracy of our system is 93.75%–95.98% for blind English forgeries and 86.61%–91.07% on casual English forgeries. Even for skilled forgeries, our system performs quite well. On the Dutch signatures with skilled forgeries, the EER accuracy is 80%, and on the Chinese signatures with skilled forgeries, the EER accuracy of our method is 74%.

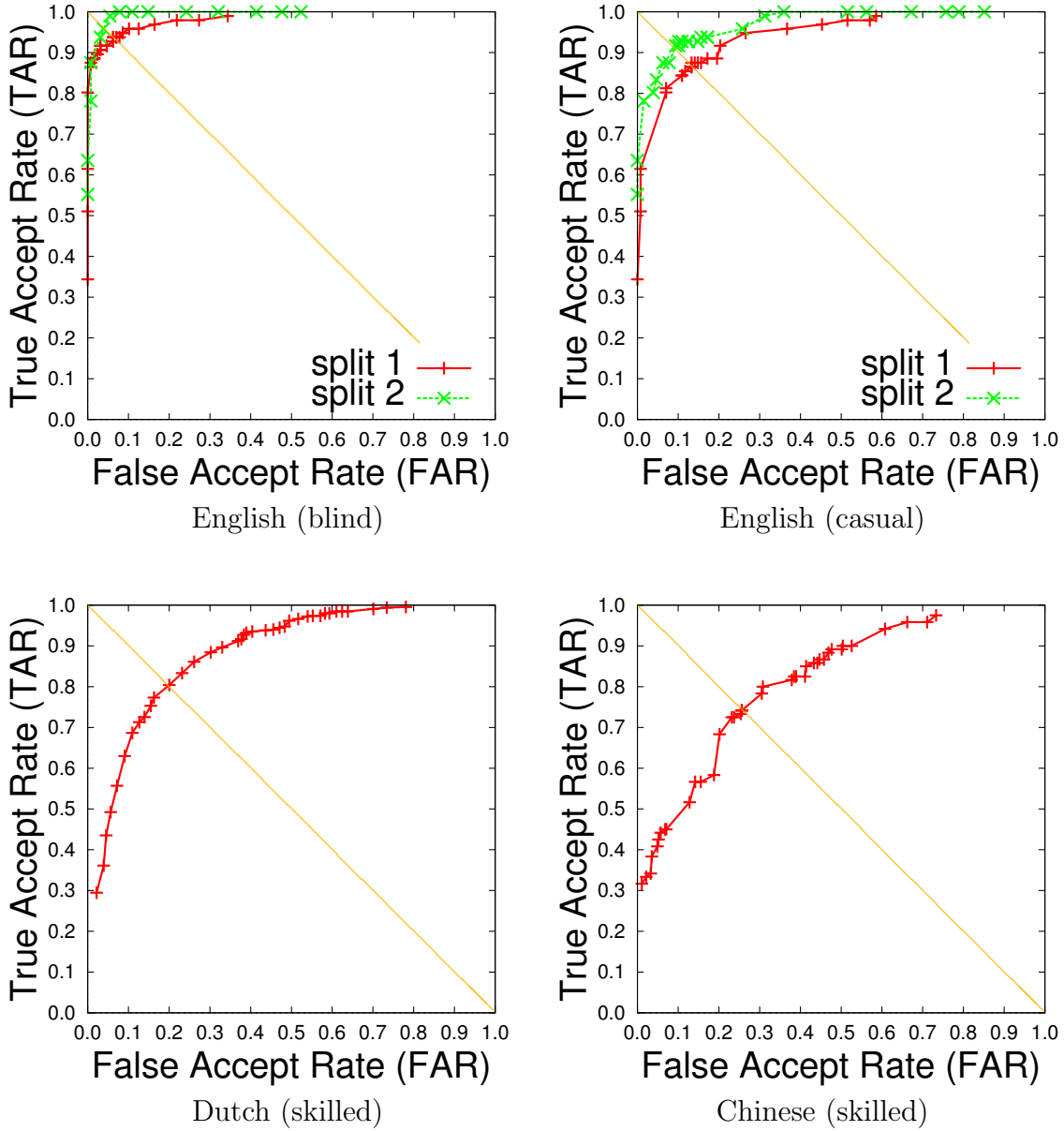


Figure 3.4: ROC curves of our method for datasets with blind, casual, and skilled forgeries. Equal error rate (EER) is where a curve intersects the yellow diagonal.

Table 3.1: Results of our method

Dataset (forgery type)	ACC	FRR	FAR
English (blind) split 1	93.75%	.0625	.0625
English (blind) split 2	95.98%	.0417	.0391
English (casual) split 1	86.61%	.1354	.1328
English (casual) split 2	91.07%	.0833	.0938
Dutch (skilled)	80.19%	.1960	.2003
Chinese (skilled)	74.33%	.2583	.2561

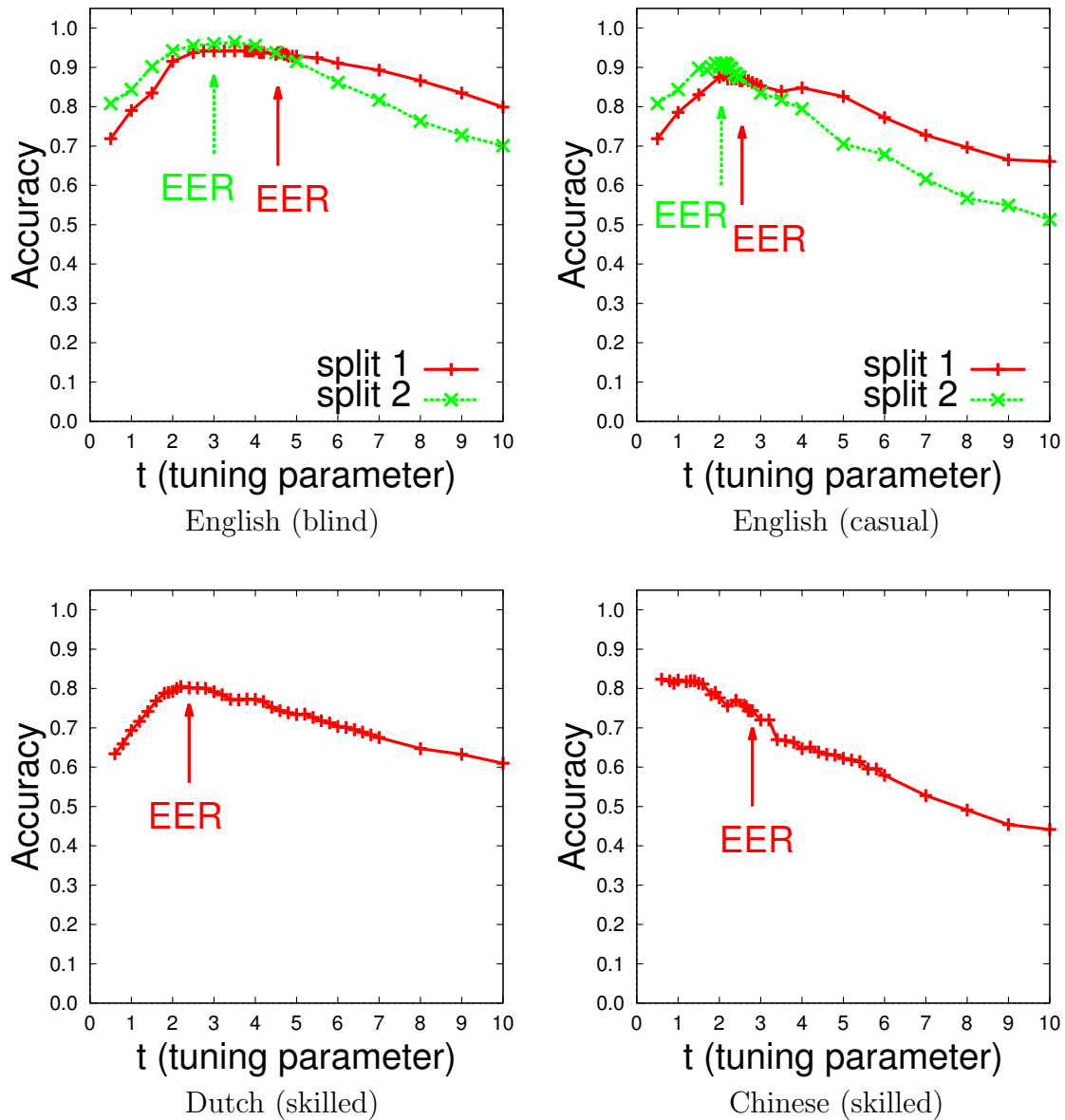


Figure 3.5: Accuracy of our method as t varies. Points at which equal error rates (EER) occur are marked with arrows.

In Table 3.2 and Figure 3.6, we compare our skilled forgery results to the results reported in [23] for the methods that competed in the ICDAR 2011 competition. For Dutch signatures, we see that the EER accuracy of our method is better than half of the systems that were in the competition. For Chinese signatures, only one system is reported to have a higher EER accuracy than ours.

It is important to note that we achieve these results with virtually no language-specific tuning or algorithmic modifications except for necessary preprocessing. The accuracy of our method seems relatively consistent from Dutch to Chinese (a difference in accuracy of only 5.86%). In contrast, some of the other methods seem less adaptable across languages. For example, the method that performed best in SigComp2011 on Dutch signatures (method 6/7), actually performed worst on the Chinese signatures. In fact, the numbers reported in Table 3.2 for method 6/7 are for when the method is specifically optimized for Chinese or Dutch, respectively. Only methods 1 and 2 are more consistent in their accuracy when moving from Dutch to Chinese, with differences of 2.87% and 4.89%, respectively.

3.7 Conclusion

We have presented a method of discriminating between authentic and forged signatures using 2-D geometric warping. In the absence of a known dataset for blind and casual forgeries, we have created one and made it available so that other methods can be compared to ours in the future. Our method performs well on blind and casual forgeries, and even shows promise for skilled forgeries when compared with methods reported in the SigComp2011 competition. Without language-specific tuning, our basic method performs reasonably well on datasets of multiple languages (English, Dutch, Chinese). These early results are very encouraging. We believe that future work will allow us to further improve our method and its accuracy.

Table 3.2: Comparison of our method to methods from SigComp2011 [23]

Method	ACC Dutch	ACC Chinese	Difference
1	82.91%	80.04%	2.87%
2	77.99%	73.10%	4.89%
3	87.80%	72.90%	14.90%
6/7	97.67%	56.06%	41.61%
8	75.84%	62.01%	13.83%
9	71.02%	61.81%	9.21%
Ours	80.19%	74.33%	5.86%

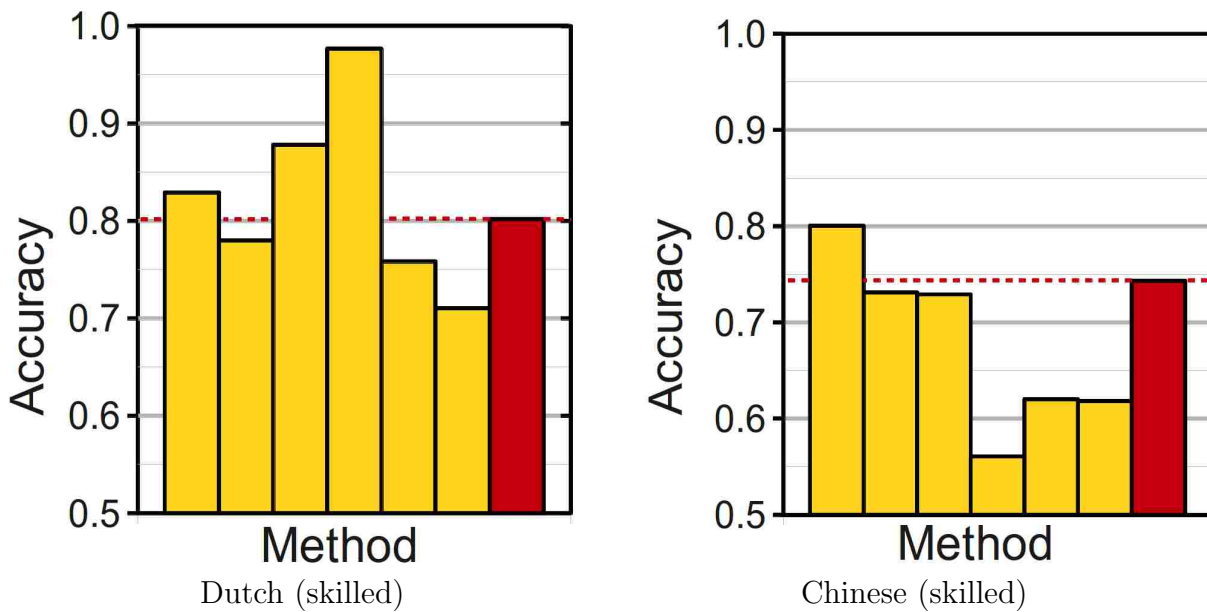


Figure 3.6: Comparison of EER accuracy of our method (red) to the methods from the ICDAR 2011 competition (yellow).

Chapter 4

Additional Analysis, Experiments, and Improvements

In this chapter, we report significant additional analysis of our handwriting recognition method. The analysis includes a detailed look at the types and causes of recognition errors, results after some minor modifications to our method, and results of tuning various system parameters. To aid with comparisons of our method to other methods, we also include experimental results of testing our system's accuracy on a large, publicly available, multi-author dataset. While some of the analysis in this chapter is suitable for publication and will be included in a future journal article submission, the rest of the details augment our understanding of the current strengths and weaknesses of our method and inform our search for ways to improve the method in the future.

4.1 Introduction

In Chapters 2 and 3, we introduce a novel 2-D geometric warping based method of comparing handwritten words, and show that the method is applicable to both offline handwriting recognition and offline signature verification. Those early results are very encouraging and lead us to believe that our method shows significant promise and also the potential to be improved and built upon in the future. However, those chapters include only limited analysis of the method itself, leaving us with several unanswered questions regarding which of our initial implementation decisions and parameter choices can be improved. In addition, our recognition results in Chapter 2 are based on relatively small single-author datasets that are not used to test most other handwriting recognition methods, making it difficult to compare

our method to other methods. In this chapter, we analyze our method and parameter choices in more detail and report results for a larger, more general dataset that is well-known in the handwriting recognition community — the IAM database (IAMDB) [28].

In order to make significant improvements to our method, it is important to understand its current limitations, including what types of recognition errors our method makes and why it makes those errors. In Sections 4.2 and 4.3, we analyze the recognition errors and their causes.

In Section 4.4 through Section 4.11, we report our experimentation with several minor modifications to the recognition algorithm itself and to the system parameters. We find that some of the modifications increase recognition accuracy, including the improved cost metric (Section 4.5), the word length mismatch penalty (Section 4.4.3), and the improved handling of distance map boundaries (Section 4.6). We report our final results after algorithmic improvements and tuned parameters for the Smith and Washington datasets in Section 4.12.

We test our handwriting recognition method on the IAMDB in Section 4.13. This allows us to analyze our results in the context of results for some other HR methods found in the literature and show that our method is competitive with other methods (Section 4.14).

4.2 Analysis of Smith Dataset Recognition Errors

In this section, we augment our analysis from Section 2.6 by looking more carefully at the recognition errors that occur when using our recognition method on the Smith dataset from Chapter 2. We do similar analysis for the Washington dataset in Section 4.3.

In Figure 4.1, we classify the in-vocabulary words that are recognized incorrectly by the type of recognition error that occurs. We see that 9 of the 84 errors (11%) are simply a matter of capitalization (Figure 4.2), and another 29 (35%) differ from the most similar training example only by a single character (Figure 4.3). We subjectively classify the rest of the errors based on how similar the overall shape of the word is to the closest matching training example. Of the errors, 12 (14%) are very similar in shape to the closest

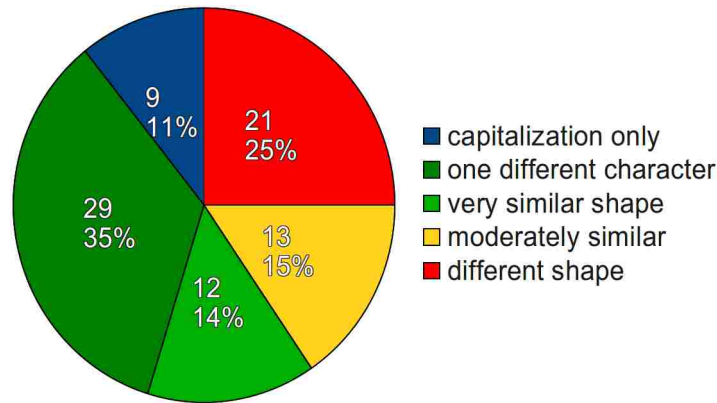


Figure 4.1: Types of recognition errors - Smith dataset.

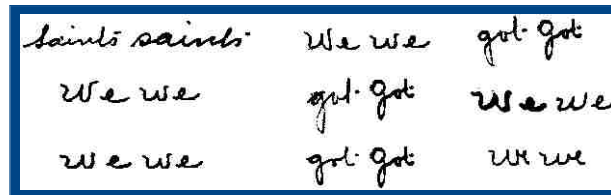


Figure 4.2: Smith dataset errors that only differ by capitalization. For each word pair, the incorrectly recognized word is on the left and the training example it matches is on the right. (These account for 11% of the errors.)

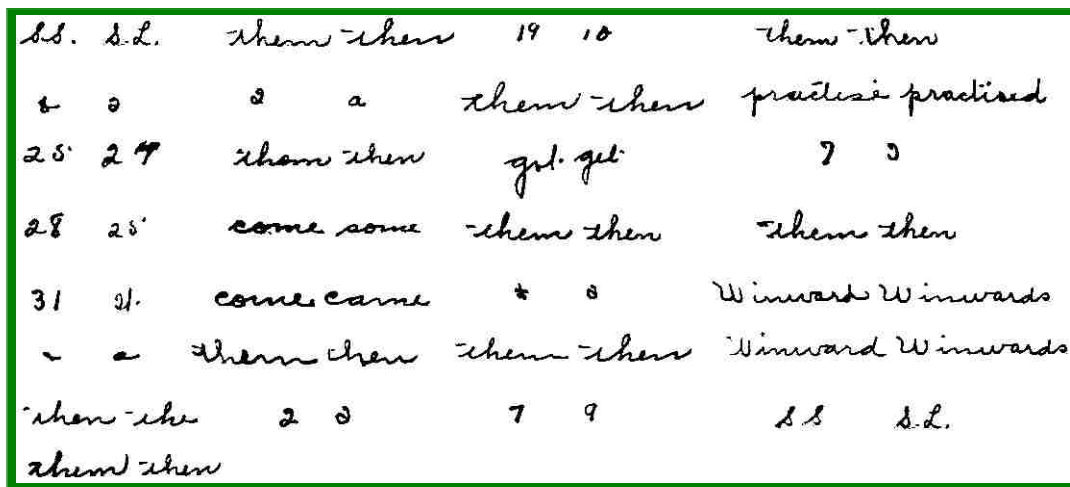


Figure 4.3: Smith errors that only differ by one character. (Account for 35% of the errors)

training example (Figure 4.4), 13 (15%) are moderately similar (Figure 4.5), and 21 (25%) are significantly different (Figure 4.6).

We observe that the incorrect match is often visually more similar to the word being recognized than any training examples of the word itself, so it is not surprising that our method occasionally chooses incorrect training examples. However, when incorrect matches

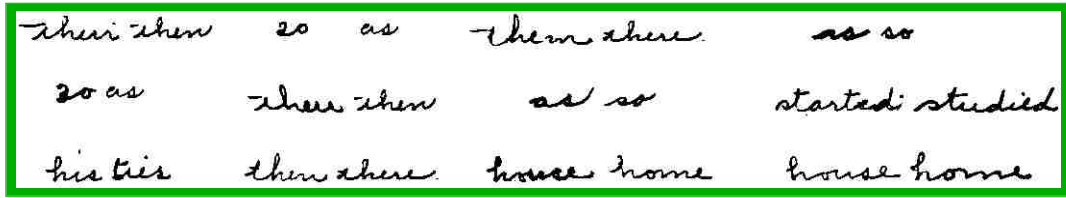


Figure 4.4: Smith errors that have very similar shapes. (Account for 14% of the errors)

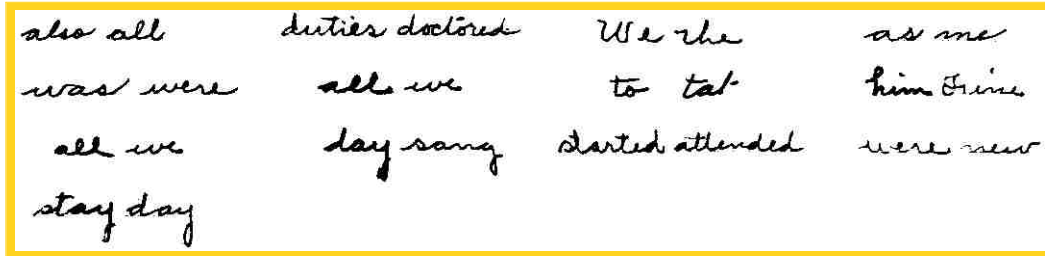


Figure 4.5: Smith errors that have moderately similar shapes. (Account for 15% of errors)

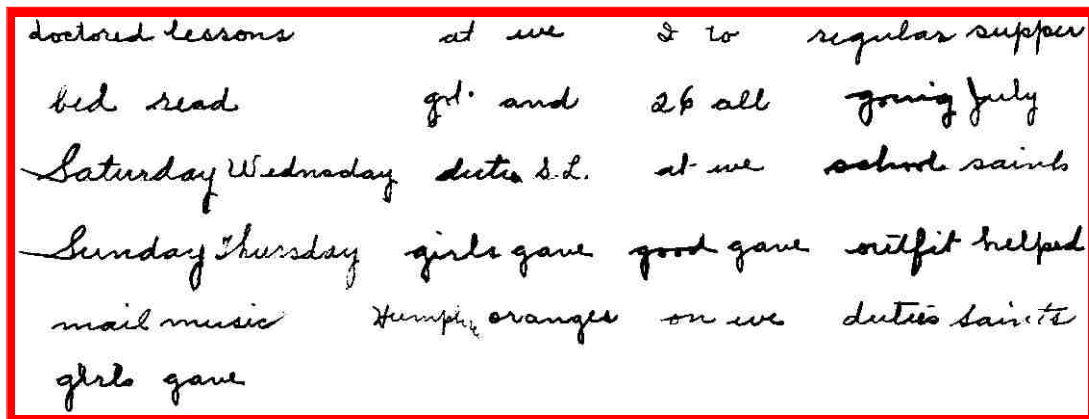


Figure 4.6: Smith errors that have different shapes. (Account for 25% of errors). Even these shapes are not always wildly different (e.g., girls/gave, good/gave, and mail/music).

are significantly different than the word being recognized (such as those in Figure 4.6 that comprise the red category of the pie chart in Figure 4.1), we want to understand why. To this end, we analyze each of those errors more carefully. Details for the causes of each individual error are included in Appendix B. We summarize the main causes of the errors in the following paragraphs.

We find that there are several causes of errors. In some cases the loops of letters are filled in or indistinct, resulting in a medial axis through the center of the ink blob instead of around the loop strokes themselves. An unknown word image with malformed loops may not match well to a training example with well-formed loops (and vice versa). We show

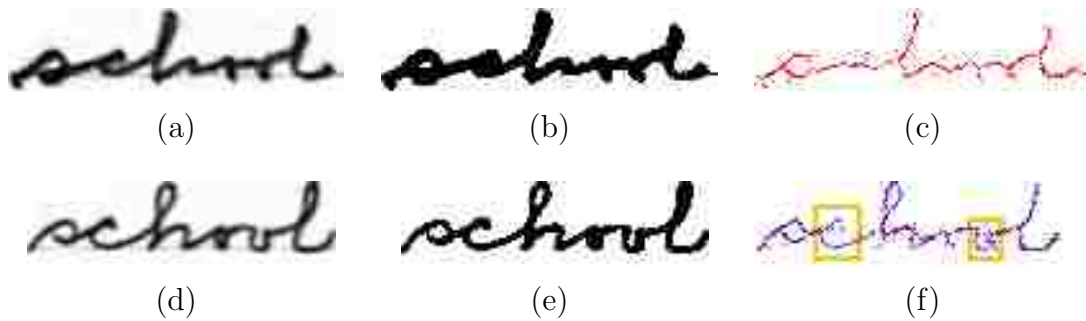


Figure 4.7: Example of filled loops causing a recognition error. a) Original word image “school”; b) loops of “c” and both “o”s are filled in binarized image; c) medial axis through center of filled loops instead of desired loop strokes; d) training image; e) binarized training image; f) filled loops and true loops do not align well after morphing (yellow regions).

one such example in Figure 4.7 (from Appendix B #12). Due to the close proximity of some ink strokes, such as the “c”, the area within the loop is actually a darker shade of gray than the page background and becomes filled in when the image is thresholded for binarization (Figure 4.7b). Better binarization algorithms (which are outside the scope of this dissertation) may help in some such cases. However, both “o”s in Figure 4.7b are also filled in — not due to binarization problems, but because the up-strokes and down-strokes actually retrace the same locations instead of forming a distinct loop. Improved binarization will not help at all in such cases.

Another common cause of recognition errors is when the coarse alignment step of our algorithm (Section 2.4.3) fails to align letters or strokes well enough for the subsequent morphing step to finish aligning them. We show two examples in Figure 4.8 (see Appendix B #13 and Appendix B #6). In Figure 4.8a the coarse (DP) alignment of the letters “Sun” of the red word are too far away from the corresponding letters of the blue word. In Figure 4.8b, the horizontal DP alignment is fine, but the vertical alignment is poor. In both cases, the coarse alignment of the words is too poor for the morphing step of our algorithm to align the strokes in the manner that we would hope. Since the effectiveness of DP warping is influenced somewhat by the value chosen for the Sakoe-Chiba band width constraint (defined in Section 2.4.3), we experiment with various values for this parameter to attempt to



Figure 4.8: Examples of errors caused by poor coarse alignment. a) Horizontal coarse alignment of first three letters of “Sunday” is poor, b) Poor vertical coarse alignment of “got”.



Figure 4.9: Illustration of how $C_{0 \to 1}$ can be low even with poor alignment. All blue pixels are only a distance of 1 from red, even though red is not aligned well with blue.

reduce the number of errors caused by poor coarse alignment. We report these experiments in Section 4.10.

Occasionally, an error occurs because the matching cost ($C_{0 \to 1}$) is low despite the fact that the warped medial axis (A'_0) and the other medial axis (A_1) do not align well. Since $C_{0 \to 1}$ (Equation 2.5) uses only the distance map values corresponding to the A_1 (blue) pixels, if they are all close to the nearest A'_0 (red) pixel then the aggregate distance is low even if there are many pixels of A'_0 (red) that are far from the nearest A_1 (blue) pixel. We illustrate this problem in Figure 4.9. In an attempt to avoid these types of errors, we modify our cost metric in Section 4.5 to include both the distances of red pixels from blue, and blue from red instead of just one or the other. As we describe in that section, we find that our modification does indeed reduce the frequency of this type of recognition error and slightly improves our overall recognition accuracy.

We notice a couple of recognition errors at least partially caused by strange behavior of pixels in the last row or column of the warp mesh (Appendix B #13 and #14). Investigation of these errors reveals minor bugs in our implementation that only affect the last row and column, and only occasionally causing an actual recognition error. We address these errors in Section 4.8.

We observe many recognition errors in which $C_{0 \rightarrow 1}$ is low but $C_{1 \rightarrow 0}$ is high, or vice versa. This means that the morphing alignment of one word to the other works very well, but morphing the other direction does not. Since errors can result from an unlucky failed morph in one direction, we speculate that it might be better to only use the better of the two morphs, $\min(C_{0 \rightarrow 1}, C_{1 \rightarrow 0})$ (the minimum of the two costs), to discriminate between words. Conversely, we recognize that sometimes words that are different actually align quite well when morphing in one direction. For example, one word may collapse onto the other, morphing better than we would have hoped and leaving us with a low cost from the distance map. In those cases, we speculate it may be better to instead only use the worse of the two costs, $\max(C_{0 \rightarrow 1}, C_{1 \rightarrow 0})$, so that we only consider costs in which an incorrect word did not just happen to align well with the other because our morphing was “too good.” In Section 4.11 we experiment using only the minimum or maximum. We find that it is much better in practice to add both costs as we already do ($C_{0 \leftrightarrow 1} = C_{0 \rightarrow 1} + C_{1 \rightarrow 0}$), instead of just using the minimum or maximum.

Some recognition errors occur when the word being recognized is simply shaped differently than the training examples of that word. For example, the handwriting is sloppy or compacted, letters are shaped differently or written with a different slant, or parts of the word are out of place enough that the morphing algorithm is unable to align them. Finding solutions for these types of more difficult errors is left as future work.

As described in the preceding paragraphs, we find many common causes of recognition errors in the Smith dataset. Often, errors are not necessarily due to a single cause, but are instead the result of a multiple causes combined. We also find that in many cases, a

recognition error occurs by a very small margin — the cost to match to the incorrectly chosen training example is almost identical to the cost to match to a correct training example, and the incorrect example just happens to be the slightly better match of the two.

4.3 Analysis of Washington Dataset Recognition Errors

In Figure 4.10, we classify the in-vocabulary words that are recognized incorrectly by the type of recognition error that occurs. There are 83 errors total. Of those, 4 (5%) differ from the most similar training example only by minor punctuation (the Washington dataset labels include some hyphenation, periods, and apostrophes whereas the Smith dataset is scrubbed of all punctuation). Another 7 (8%) differ only by capitalization (Figure 4.11). Another 11 (13%) differ from the most similar training example only by a single character (Figure 4.12). We subjectively classify the rest of the errors based on how similar the overall shape of the word is from the closest matching training example. Of all errors, 24 (29%) are very similar in shape (Figure 4.13), 20 (24%) are moderately similar (Figure 4.14), and 17 (20%) are significantly different (Figure 4.15).

Comparing Figure 4.10 to Figure 4.1, we see that the proportion of minor errors (blue/green) is very similar in both the Washington dataset and the Smith dataset. The proportions of moderate errors (yellow) and more obvious errors (red) are also fairly similar for both authors. This suggests that our method is consistent and predictable in the types of errors it makes, independent of the specific author.

When we look closely at the causes of errors in the Washington dataset (Appendix C), we find that they are very similar to the causes of errors in the Smith dataset described in Section 4.2. However, we do notice that the Washington dataset has many more filled loops than the Smith dataset due to thicker ink strokes.

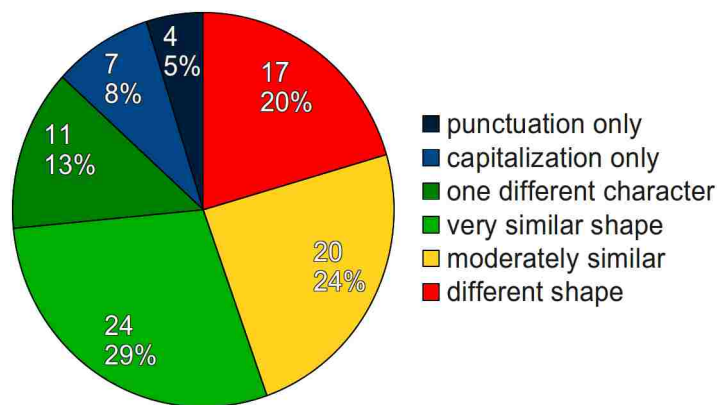


Figure 4.10: Types of recognition errors - Washington dataset.

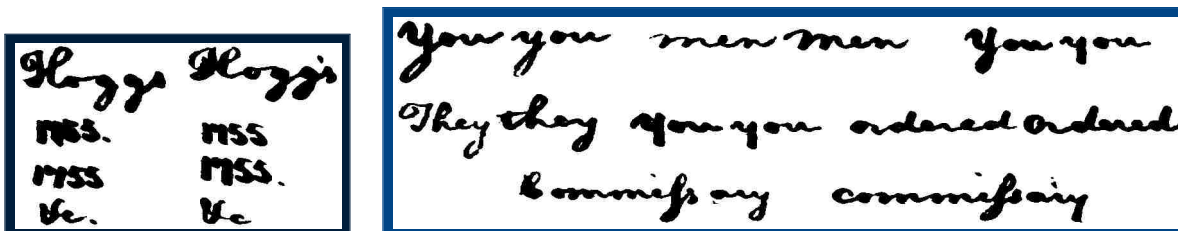


Figure 4.11: Errors that only differ by punctuation (left) or capitalization (right).

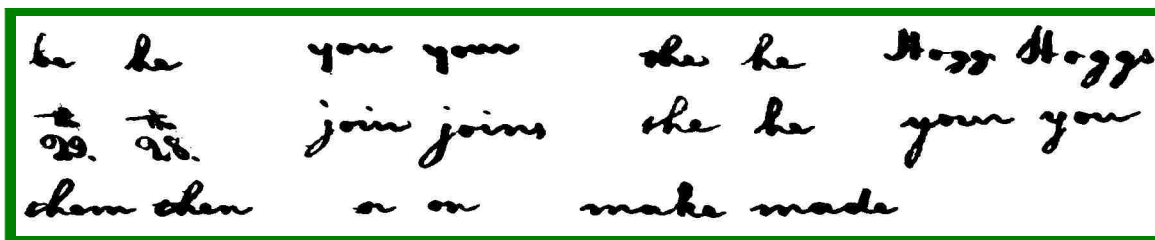


Figure 4.12: Errors that only differ by one character. (Washington dataset)

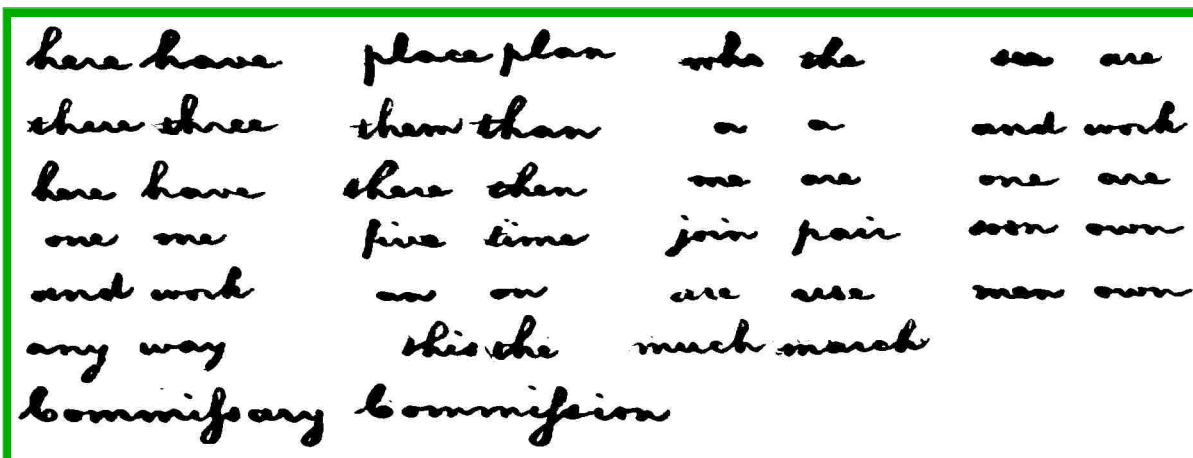


Figure 4.13: Errors that have very similar shapes. (Washington dataset)

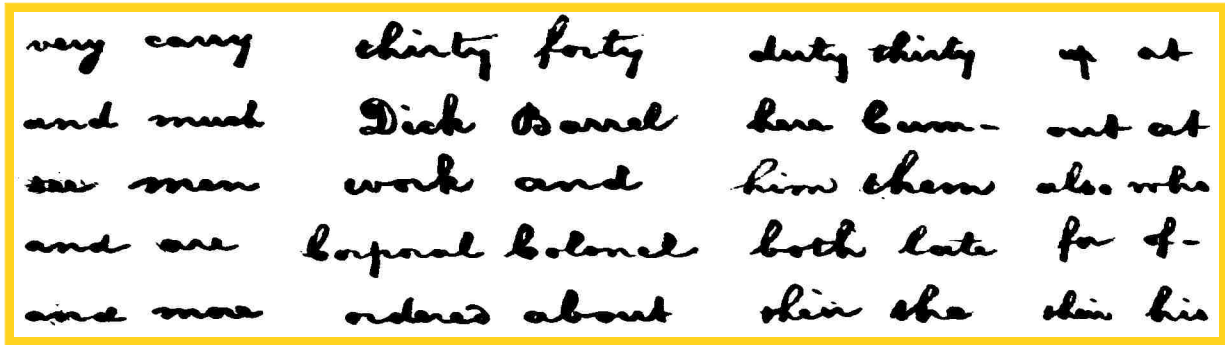


Figure 4.14: Errors that have moderately similar shapes. (Washington dataset)

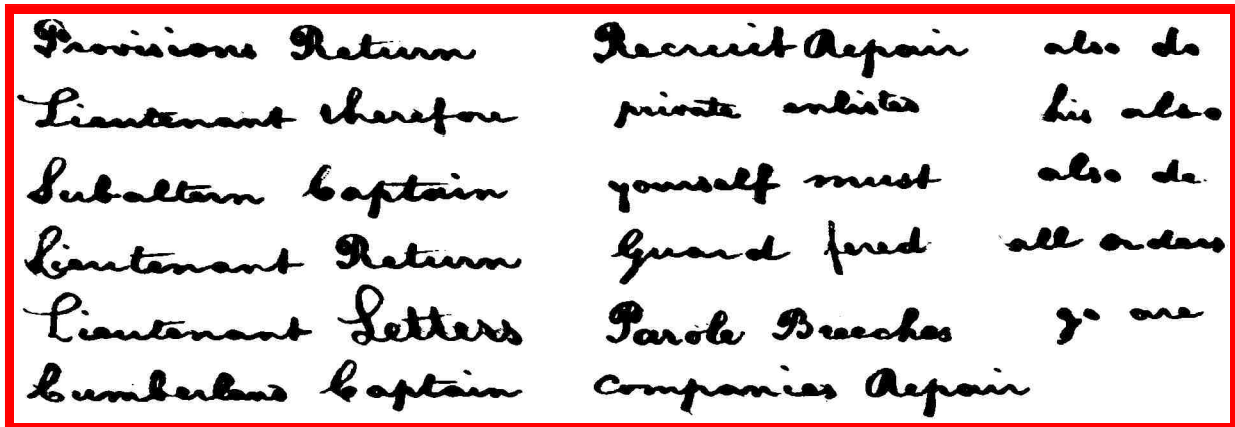


Figure 4.15: Errors that have different shapes. (Washington dataset)

4.4 Incorporation of Morphing Movement Cost into Word Difference Metric

We measure the “difference” between two words using a distance metric, $C(I_0, I_1)$, that we call the *word matching cost* (Section 2.4). This metric is computed using distance maps to determine how far apart the medial axes of two words are after one is warped to the other and vice versa. While this metric gives impressive results, we speculated that our results might be even better if we also include a cost for how much work is required to warp the words to their morphed positions instead of just how similar they are after warping is complete. The intuition behind our speculation is that there may be cases in which two very different words are able to be warped almost exactly into each others’ likenesses, resulting in a very low word matching cost even though there is a significant amount of warping of the words to get them into that final warped position. Incorporating a cost for the warping itself would mitigate those sorts of errors.

However, we also see cases in which adding an additional cost for warping would be harmful instead of beneficial. In many cases, words are very similar in the characteristics that matter (stroke direction, stroke shape, stroke order, etc.) but because of stretched/condensed characters, inconsistent slant between parts of the word, or other variations that are spatially significant but subjectively unimportant, there is significant warping required to match them together. In these cases, it is better to ignore how much work is required to warp the words and focus only on the difference in the final result of the warping.

Without experimentation, it is unclear whether or not it is better in practice to incorporate the warping work into the cost metric as we speculated, and if so, how much weight should be given to the warping work versus the cost calculated from the distance maps. To test our speculation and answer these questions, we incorporate into $C(I_0, I_1)$ a metric of how much warping work is required and test on a range of weights from 0 to 1 (in reasonably small increments) on several datasets.

In general, we find that movement cost alone does not increase accuracy in our experiments, and in fact decreases it. This is true both for handwriting recognition and for signature verification. We report the methods and results of our experiments in Sections 4.4.1 through 4.4.4.

4.4.1 Morphing Movement Cost

In the image morphing correspondence algorithm developed by Gao and Sederberg [8], the morphing work equation includes weighted terms representing the amount of stretching of the warp grid, the changes in angles at the grid vertices, and the change in actual color information within a grid rectangle. When dealing with handwritten ink strokes we are not interested in how much the grid mesh moves or vertex angles change because those metrics are only *indirectly* indicative of the effort involved in aligning one word's medial axis to that of another. What matters is how much the medial axis pixels themselves are moved during the warping operation, not how the grid was changed to make them move.

We define the movement cost, $\mathcal{M}_{0 \rightarrow 1}$, for a warped medial axis as the average Euclidean distance between each warped medial axis pixel and the pixel's position when it is only coarsely-aligned (warped with dynamic programming alignment but not the morphing correspondence algorithm). That is, for medial axis A_0 with coarse (DP) alignment A_0'' and final warped position A_0' , the movement cost is

$$\mathcal{M}_{0 \rightarrow 1} = \frac{1}{\|A_0\|} \sum_{i=1}^{\|A_0\|} |A_0'[i] - A_0''[i]|. \quad (4.1)$$

For the experiments in this section, we modify Equation 2.5 to include $\mathcal{M}_{0 \rightarrow 1}$:

$$C_{0 \rightarrow 1} = (m)\mathcal{M}_{0 \rightarrow 1} + (1 - m)\hat{C}_{0 \rightarrow 1}, \quad (4.2)$$

where $\hat{C}_{0 \rightarrow 1}$ is the original version of Equation 2.5 that calculates costs using only distance maps and m is a weight (from 0 to 1) that specifies how much the movement cost should be used relative to the distance map cost.

4.4.2 Results of Incorporating Movement Cost for Handwriting Recognition

In Figure 4.16, we plot the in-vocabulary accuracy of the Smith diary and Washington manuscript datasets (Section 2.5). For each dataset, we use the first 1000 words as training examples and the following 1000 words as test data (“Split 1”). We vary m , the movement cost weight, in increments of 0.1 from 0 (movement cost not considered) to 1 (only movement cost considered). In areas of interest, we use higher resolution (smaller increments of m). We repeat the process using the second 1000 words as training examples and the first 1000 words as test data (“Split 2”).

We find that incorporating movement cost tends to decrease recognition accuracy instead of increasing it. For the Smith dataset, accuracy is lower for all $m > 0$ than when movement cost is ignored. This is true for both splits of the data. For the Washington dataset, there are some values of m that result in small improvements in accuracy. However,

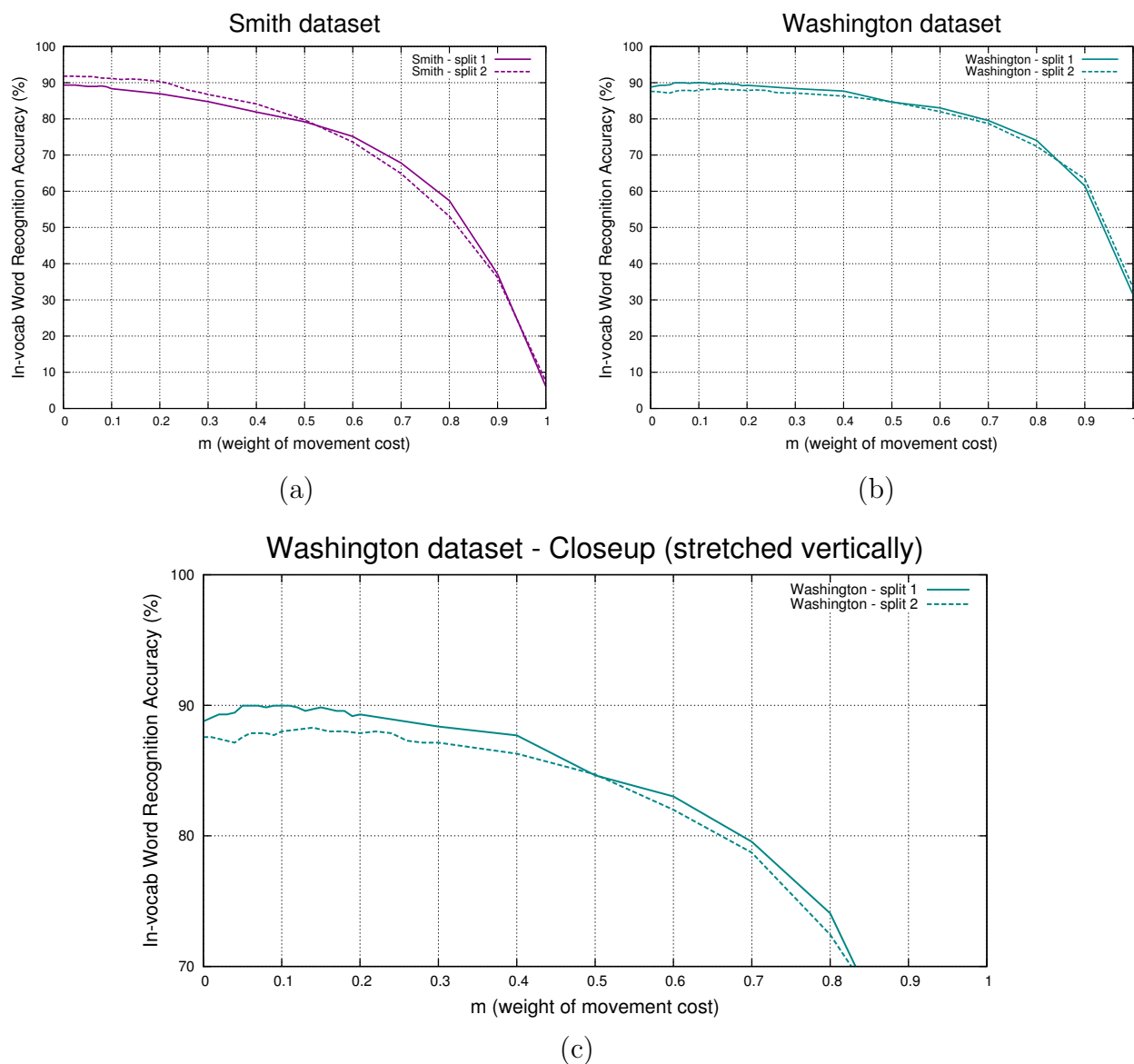


Figure 4.16: Effect of including movement cost in the word difference metric. Overall, as more weight is given to the movement cost, word recognition accuracy decreases. a) Smith dataset accuracy is lower for all $m > 0$. b) Full range of m for Washington dataset. c) Closeup of the Washington dataset graph - slight improvement for some small m values can be seen, but the best m value is inconsistent between splits of the dataset.

the improvements are not consistent, the general trend is still a decrease in accuracy as m increases, and the optimal value of m varies even between two splits of the same dataset. For an unknown dataset, we do not know how to predict whether any non-zero value of m exists that will increase recognition accuracy, and if so, what the best value of m is.

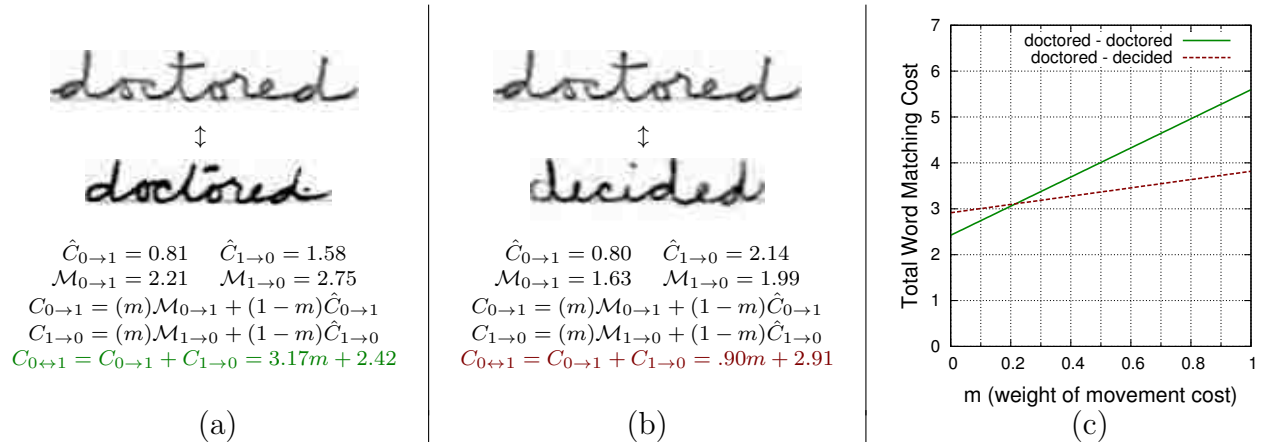


Figure 4.17: Example of the negative effect of movement cost on recognition. Test word “doctored” is incorrectly recognized as “decided” when movement cost is weighted with $m > 0.22$. a) As m varies, $C_{0 \leftrightarrow 1}$ to match to training example of “doctored” is equation of a line (green). b) $C_{0 \leftrightarrow 1}$ to match to “decided” (red). c) “doctored” is the lowest cost match for small values of m , but when m is greater than 0.22, “decided” is the lowest cost match.

It is interesting to observe that movement cost negatively impacts the Smith dataset more than it impacts the Washington dataset, as evidenced by the lower accuracy of the Smith dataset at $m = 1$ and the shorter flat part of the curve before accuracy drops significantly in Figure 4.16a. We notice that there is correlation with the fact that we see more variation in the penmanship of the Smith dataset, whereas the penmanship of the Washington dataset seems more consistent.

We inspect recognition errors introduced by incorporating movement cost into the word difference metric $C_{0 \leftrightarrow 1}$. Figure 4.17 shows an example of a word (“doctored”) that is recognized correctly with the original word difference metric (or $m = 0$), but incorrectly recognized as “decided” when movement cost is heavily-weighted. In fact, any weight greater than 0.22 results in a recognition error because the word matching cost for “decided” will always be lower than the word matching cost of the training instance of “doctored” when $m > 0.22$ (Figure 4.17c).

4.4.3 Adding a Penalty to Movement Cost for Mismatched Word Lengths

If we completely disregard the cost computed from distance maps and use only the movement cost (i.e., set $m = 1.0$), we notice an interesting trend accompanying the very low recognition

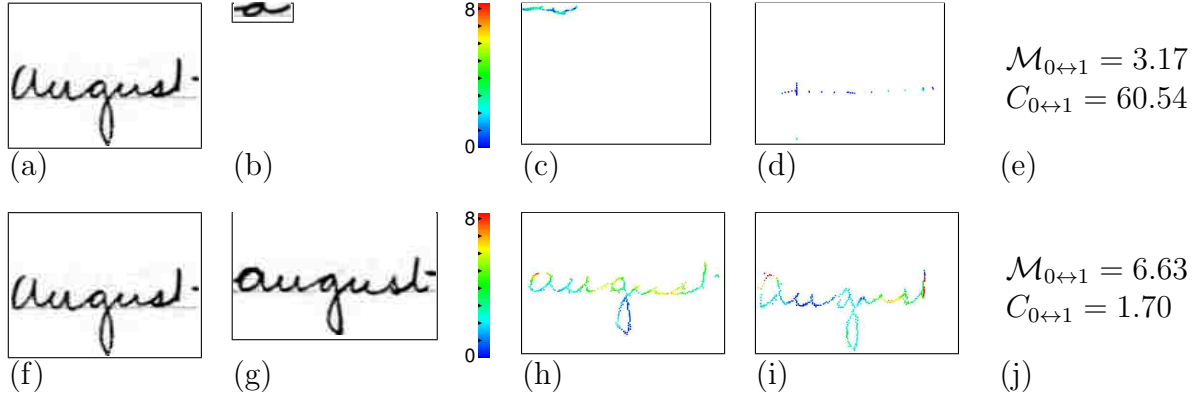


Figure 4.18: Example of how movement cost favors short words. Warping the medial axis of a) to b) results in c), the warped medial axis points color-coded by how far they moved from coarse-aligned positions (blue is less, red is more). Warping b) to a) results in d). e) The total movement cost, $\mathcal{M}_{0 \leftrightarrow 1}$, is low even though $C_{0 \leftrightarrow 1}$ is high. f) through j) show that matching to a second instance of “August” results in higher movement cost, but very low distance map cost.

rate. Almost all words are recognized as very short words such as “a,” “on,” or “the,” even if the actual word is much longer. For example, the word “August” is incorrectly recognized as “a” when $m = 1$ but is correctly recognized as “August” when $m = 0$ (Figure 4.18). Furthermore, when $m = 1$ the top 10 best matches are all relatively short words but when $m = 0$ the top 10 best matches are all instances of the word “August.”

The word “a” does not warp well to “August,” resulting in a very high cost from the distance maps ($C_{0 \leftrightarrow 1}$). Despite that fact, the medial axis pixels move very little from their coarse-aligned positions to arrive at their final warped positions, so $\mathcal{M}_{1 \rightarrow 0}$ is very low.

In order to compensate for the fact that movement cost favors short words in this manner, we introduce another term into Equation 4.2 as a penalty for difference in word-lengths:

$$C_{0 \rightarrow 1} = (m)\mathcal{M}_{0 \rightarrow 1} + (1 - m)\hat{C}_{0 \rightarrow 1} + (p)\frac{w_{long} - w_{short}}{w_{long}}, \quad (4.3)$$

where p is a weighting constant for the new penalty term, and w_{long} and w_{short} are the widths of the longer and shorter word images, respectively. Thus, the penalty is small when two words are close to the same length, and large when one word is much longer than the other.

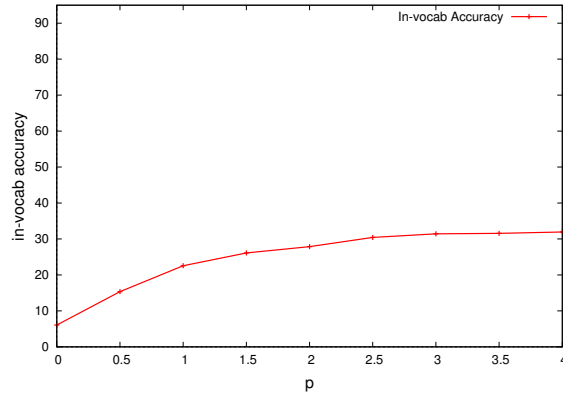


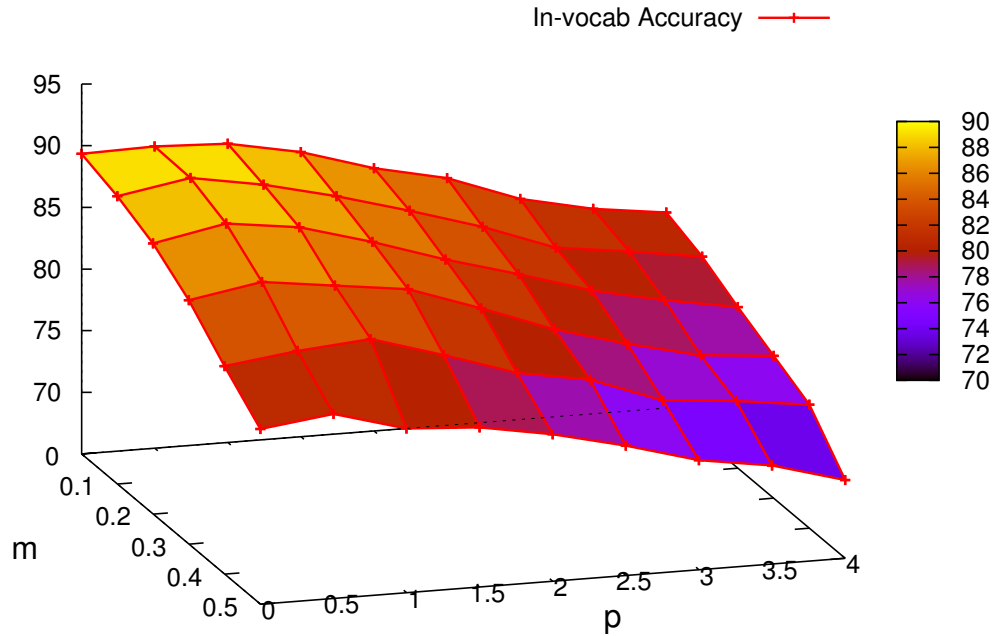
Figure 4.19: Effect of word length mismatch penalty on Smith dataset accuracy when $m = 1$. Penalty term increases accuracy when only movement cost is considered.

Considering only movement cost and the new penalty term ($m = 1.0$ and p varies over a range) for the Smith dataset, we see that the penalty term does increase the recognition accuracy (Figure 4.19). In addition, the results are not consistently skewed toward very short words as is the case without using the penalty term (i.e., when $p=0$). Instead, the top 10 matches tend to be words of a length similar to that of the word being compared.

In Figure 4.20 we report the in-vocabulary word recognition accuracy for the Smith dataset as we vary both m and p . We find that accuracy consistently improves slightly at $p = 0.5$ for all values of m tested (including $m = 0$). This leads us to believe that incorporating the penalty for mismatched word-lengths is useful, and should be explored in more detail in future work. However, we see that even with the penalty term, accuracy still decreases when we incorporate movement cost. This is the case for all $m > 0$. We conclude that it is better not to use the movement cost term at all for handwriting recognition.

4.4.4 Results of Incorporating Movement Cost for Signature Verification

We test incorporating movement cost into signature verification / forgery detection for all of the signature datasets used in Chapter 3. We compute the verification accuracies of each dataset while holding the user-tunable system parameter (Section 3.3.3) constant at $t = 2.5$,



(a)

Smith dataset in-vocabulary recognition accuracy as m and p vary

	0.0	89.35	89.48	89.23	88.09	86.31	85.04	82.89	81.62	80.86
	0.1	88.34	89.35	88.34	86.95	85.30	83.52	81.37	80.61	79.72
m	0.2	86.95	88.09	87.33	85.68	83.78	82.13	80.35	79.09	78.07
	0.3	84.79	85.80	85.04	84.28	82.26	80.10	78.45	77.06	76.55
	0.4	81.88	82.64	83.14	81.37	79.47	78.45	76.30	75.79	75.03
	0.5	79.21	79.97	78.33	77.95	76.93	75.54	73.89	73.00	71.36
		0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
		p								

(b)

Figure 4.20: Smith dataset recognition accuracy as both movement cost and length penalty vary. a) graphical representation. b) numerical values used for the graph.

and computing word differences using Equation 4.2, which includes movement cost weighted by m . We show the resulting signature verification accuracies in Figure 4.21.

While there are some small bumps in the graphs, generally the more weight is given to movement cost instead of cost from the distance maps, the lower the accuracy. This holds true for both splits of the blind forgery dataset, both splits of the casual forgery dataset, and both datasets of skilled forgeries. Interestingly, we see that the more variation and less consistency there is in a dataset's signatures, the more pronounced the downward trend

Signature Datasets

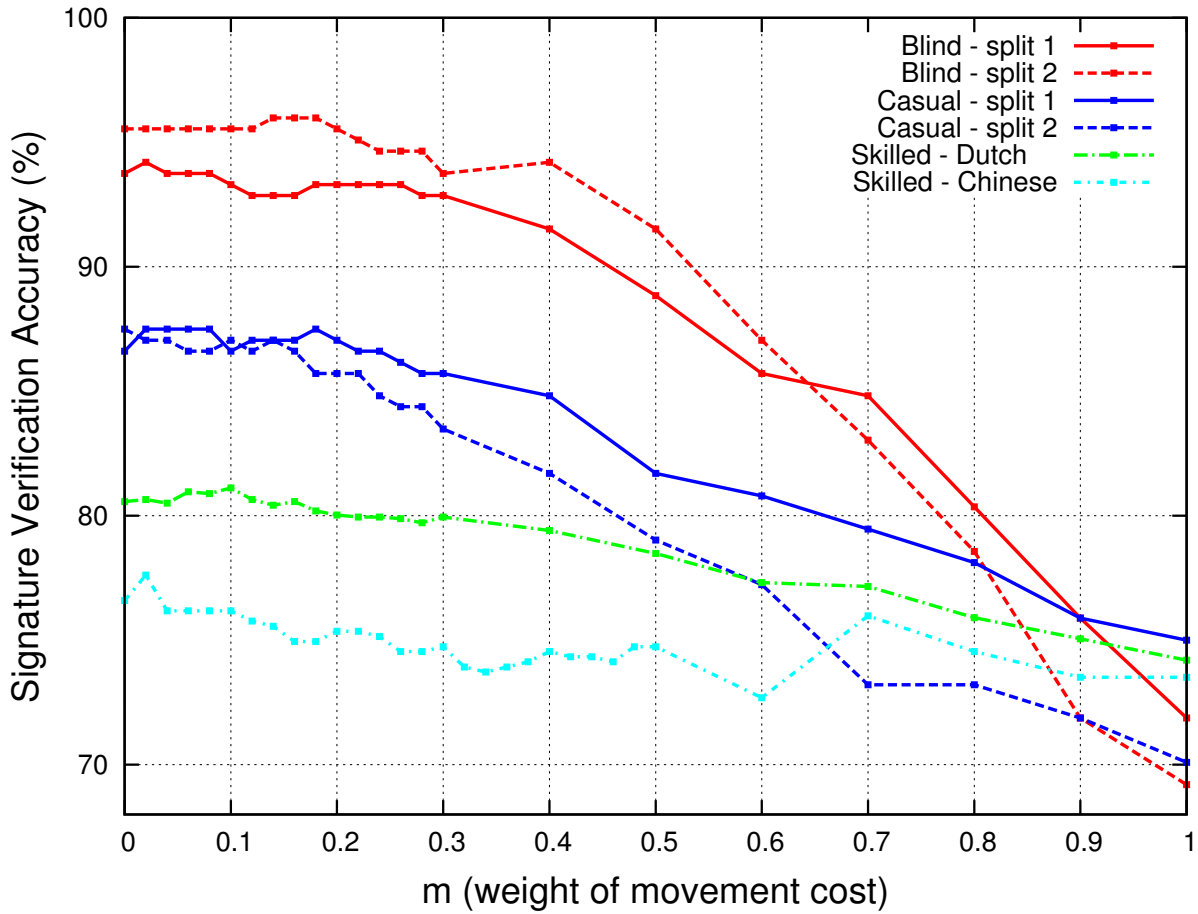


Figure 4.21: Signature verification accuracy for various values of m . There is a general downward trend in accuracy as m increases. The downward trend is more pronounced for unskilled forgeries (with more variation in the handwriting) than skilled forgeries.

is in the corresponding graph as m increases. Incorporating movement cost for the blind forgeries results in an average of more than 20% decrease in accuracy, whereas the decrease is less than 15% for casual forgeries, and less than 5% for skilled forgeries in which there is much less variation in the handwriting. This is reminiscent of the correlation we see in Section 4.4.2 between the amount of variation in handwriting and the amount of negative impact movement cost has on recognition accuracy.

Where small accuracy increases do occur in the graphs, the m values at which they occur are not consistent between datasets, or even between splits of the same dataset where more than one split is tested. On unknown datasets, it would not be possible to predict what value of m would result in a small increase in accuracy, and since the general trend of all of the curves is downward as m increases, we conclude that using $m = 0$ is the best option for signature verification / forgery detection, just like it is for handwriting recognition.

4.5 Improved Cost Metric

As described in Section 4.2 and illustrated in Figure 4.9, our word matching cost metric will have a small value if all of the (blue) medial axis pixels of A_1 are near (red) pixels of A'_0 , even if there are many (red) pixels of A'_0 that are far away from the (blue) pixels of A_1 . In practice, such cases are not very frequent, but we do see them occur occasionally. In a few cases recognition errors result.

In this section, we change the cost metric to include the average distance of the (red) A'_0 pixels to the (blue) A_1 pixels in addition to the average distance of the (blue) A_1 pixels to the (red) A'_0 pixels. This more accurately measures the mutual alignment of both medial axes instead of just one to the other or vice versa. We also eliminate the “+1” smoothing term and explicitly prevent dividing by zero instead. Our modified cost metric equation is:

$$C_{0 \rightarrow 1} = \frac{1}{\|A_0\|} \sum_{i=1}^{\|A_0\|} D_1(A'_0[i]) + \frac{1}{\|A_1\|} \sum_{i=1}^{\|A_1\|} D'_0(A_1[i]). \quad (4.4)$$

We find that the modified cost metric improves our recognition accuracy slightly for both the Smith and Washington datasets. For the Smith dataset, in-vocabulary recognition accuracy improves from 89.38% to 90.0% (a net gain of 5 correctly-recognized words). For the Washington dataset, in-vocabulary accuracy improves from 88.90% to 89.44% (a net gain of 4 correctly-recognized words).

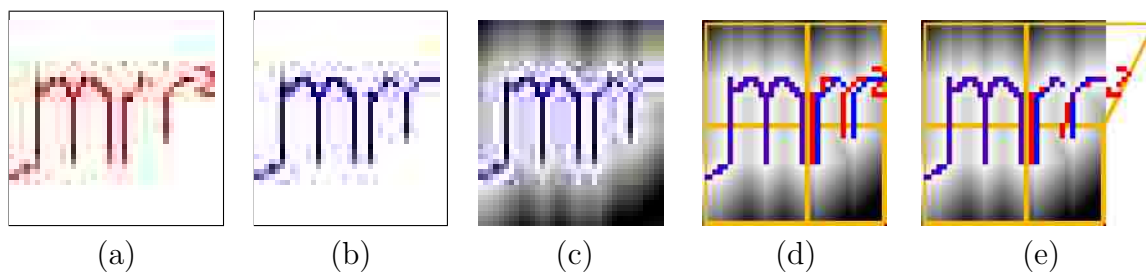


Figure 4.22: Example of warped medial axis pixels outside of distance map D_1 being ignored. a) Medial axis pixels to be warped; b) Medial axis of other word; c) Other word and its distance map, D_1 ; d) Overlay after DP warping; e) As control points are moved, pixels that warp outside of D_1 (such as the red pixels in the top-right corner) are ignored in calculations.

4.6 Improved Handling of Distance Map Boundaries

For the purpose of simplicity in implementation, we ignore any medial axis points that fall outside of distance map boundaries when we calculate $C_{0 \rightarrow 1}$ (Equations 2.5 and 4.4) and $placement_cost_{x,y}$ (Equation 2.4). In Figure 4.22, we illustrate how warped points may fall outside the distance map as control points of warp mesh are moved during morphing. In this section, we test the accuracy when handling those pixels correctly for distance map D_1 instead of ignoring them. Since we cannot extract distance values directly from the distance map if the position of the point is outside of the boundaries, we compensate by adding the nearest boundary value of the distance map to the actual 4-connected (“Manhattan”) distance between that boundary position and the position of the out-of-bounds point.

We find that handling out-of-bounds points correctly for D_1 does improve our recognition accuracy slightly. Combined with the improved cost metric in Section 4.5, this correction brings our in-vocabulary recognition accuracy up to 90.27% for the Smith dataset and 90.78% for the Washington dataset.

4.7 Using $C_{0 \rightarrow 1}$ Instead of Heuristic in $placement_cost_{x,y}$

The $placement_cost_{x,y}$ equation used by our morphing algorithm (Equation 2.4) is intended to allow us to choose a good x, y location to place a control point within the nearby neighborhood of its current location. Ideally, this would be the position within the neighborhood

at which $C_{0 \rightarrow 1}$ is minimized, meaning that the medial axes will be aligned at least as well by choosing to place the control point there as if any other candidate x, y position were to be chosen instead. However, since $C_{0 \rightarrow 1}$ requires us to calculate a distance map based on A'_0 , using $C_{0 \rightarrow 1}$ as $placement_cost_{x,y}$ would require a new distance map to be calculated for each each candidate x, y position after warping A_0 to compute A'_0 for that x, y position. In order to run at a reasonable speed, $placement_cost$ is actually a heuristic estimate of $C_{0 \rightarrow 1}$ that uses only D_1 . Since D_1 does not depend on A'_0 , it is only computed one time, not once for every candidate x, y position of every control point during every iteration.

While a heuristic $placement_cost_{x,y}$ is necessary for our algorithm to run at reasonable speeds on current hardware, we want to know how much our accuracy suffers due to the heuristic. Since hardware performance historically improves very rapidly, it is possible that using the full distance metric will be feasible within a relatively short amount of time on newer hardware. In this section, we test the accuracy of our method with $placement_cost$ changed from Equation 2.4 to

$$placement_cost_{x,y} = d\Delta_{x,y} + \frac{1}{\|A_0\|} \sum_{i=1}^{\|A_0\|} D_1(A'_0[i]) + \frac{1}{\|A_1\|} \sum_{i=1}^{\|A_1\|} D'_0(A_1[i]). \quad (4.5)$$

We find that the in-vocabulary recognition accuracy for the Smith dataset using Equation 4.5 is unchanged from that in Section 4.6, remaining at 90.27%, so our accuracy does not suffer at all by using the heuristic, at least for this dataset.

Computation, however, is much slower when using Equation 4.5. We split the computation over 20 nodes of the m6 cluster in the BYU Fulton Supercomputing Lab. Nodes have hex-core Intel Westmere 2.67 GHz processors, 24GB of 1066 MHz DDR3 RAM, and support 12 threads of execution simultaneously. Running the Smith dataset takes a cumulative walltime total of 327,045 seconds (90.85 hours), or about 4.5 hours of walltime per node on average. For comparison, running on 20 nodes of the supercomputer using the heuristic takes only 1,568 seconds of cumulative walltime (26.13 minutes), or about 1.3 minutes per

node on average. Using the full calculation in Equation 4.5 takes more than 200 times as long as using the heuristic without any increase in recognition accuracy.

When performing a similar test with the full calculation in Equation 4.5 for the Washington dataset, again split over 20 nodes of the supercomputer, many of the jobs do not even finish before the walltime allotment of 30 hrs per node (600 hours total) elapses. Since this is already an unreasonable amount of time — even for hardware over the next several years, we see no value in expending additional computing resources on the supercomputer to complete the test for the Washington dataset, especially since we already see that there is no improvement for the Smith dataset.

4.8 Correct Handling of the Last Row and Column of Warp Mesh

In our analysis of what causes recognition errors (Section 4.2), we observe that occasionally some pixels at the very right or bottom of words do not morph as expected. For example, in Figure 4.23b (from Appendix B #13), we see that the descender of the final “y” collapses during morphing even though both sides of the descender are previously aligned quite well by the coarse (DP) alignment step of the algorithm (Figure 4.23a).

The morphing error is caused by minor implementation errors in how the last row and column of the warp mesh are handled. Although the bug is present in many images, it is only noticeable in rare cases because the improvement step of the morphing algorithm usually compensates enough to fix any resulting problem. However, we correct the bug so that the mesh is refined and morphed properly in the last row and column (Figure 4.23c).

Fixing this minor bug has very little impact on our overall accuracy. On the Smith dataset, the word “girls” (Appendix B #14) is correctly recognized after the bugfix, but the accuracy for the dataset stays the same because a different word is recognized incorrectly. On the Washington dataset, four additional words are correctly recognized after the fix, but seven new errors occur, so the recognition accuracy actually decreases just slightly from 88.9% to 88.5% of in-vocabulary words. For both datasets, the newly recognized words and

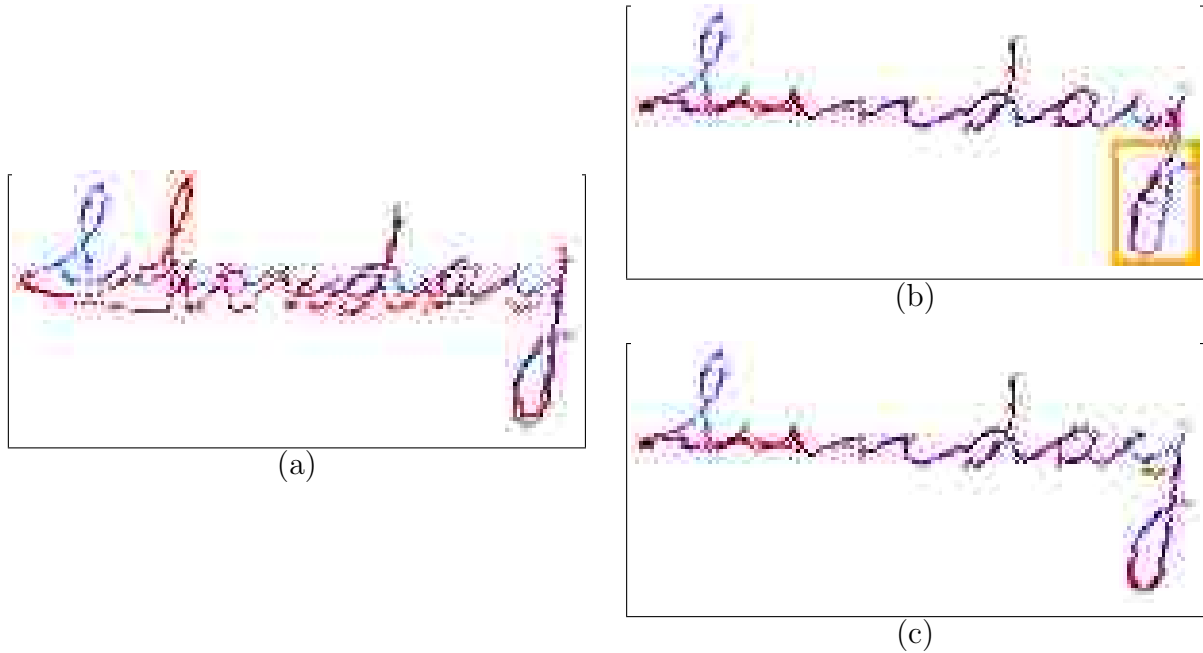


Figure 4.23: Correction of how the last row and column of the mesh are handled, demonstrated for words in Appendix B #13. a) DP coarse alignment. b) Before bug fix, the descender of “y” unexpectedly collapses to the left side of the loop. c) Descender morphs correctly after fix.

the new errors all occur on words for which the cost of matching to an erroneous training example is very close to the cost of matching to the desired training word.

4.9 Analysis of the Effect of Warp Mesh Size

In this section, we look at how the size of quads in the warp mesh affects the accuracy of our method on the Smith and Washington datasets. We look at the effect of both the initial mesh size used when the algorithm starts and the number of times the mesh is refined (subdivided) during the morphing algorithm.

In Section 2.4.5, our morphing algorithm sets the initial width and height of each mesh quad to $h_0/4$, where h_0 is the height of word image I_0 .¹ Our choice of 4 in the denominator is based partially on the intuition that characters (at least in Latin script) can be generally

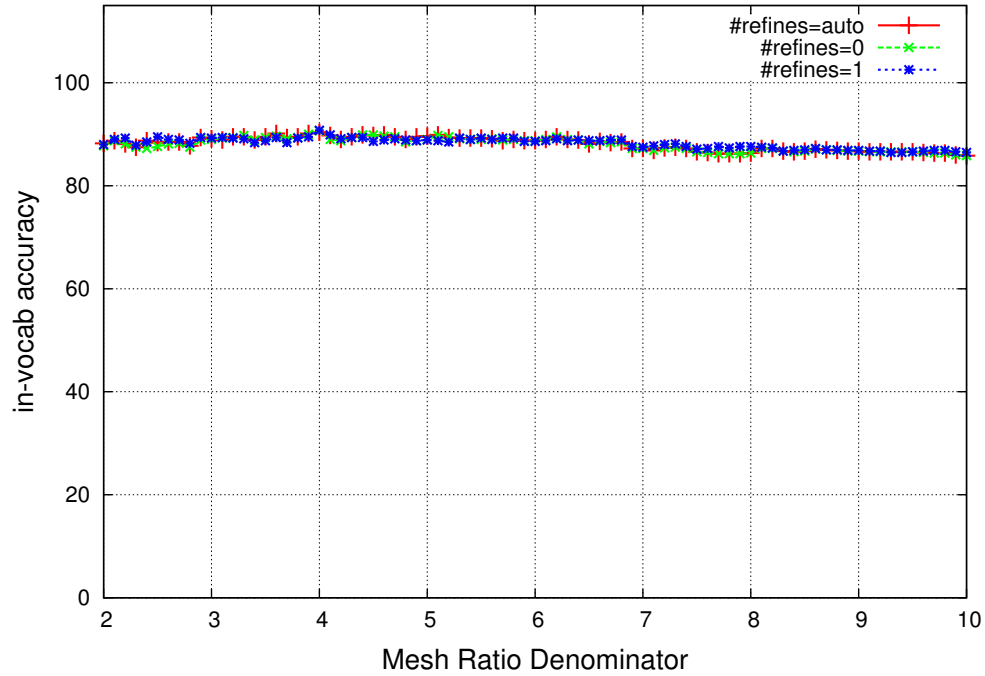
¹While all other quads are the same size, quads in the last row or column of the mesh actually have a height or width less than $h_0/4$ if the width or height of the word image is not an even multiple of $h_0/4$.

divided into three zones — those for the main body of a lower-case character, ascenders (or the top part of capital letters), and descenders — and those three parts of a given letter may need to be stretched, bent, and pulled somewhat independently to align well with the same letter in another word. We choose 4 instead of 3 in the denominator because h_0 for any given image is not necessarily evenly divisible by 3, often resulting in the fourth row being very small and of little use in the morphing. Even when the fourth row of the warp mesh is nearly the same height as the other rows, using four zones instead of three intuitively allows for some additional flexibility in the length of ascenders / descenders, and variation from top to bottom of the main body of the character.

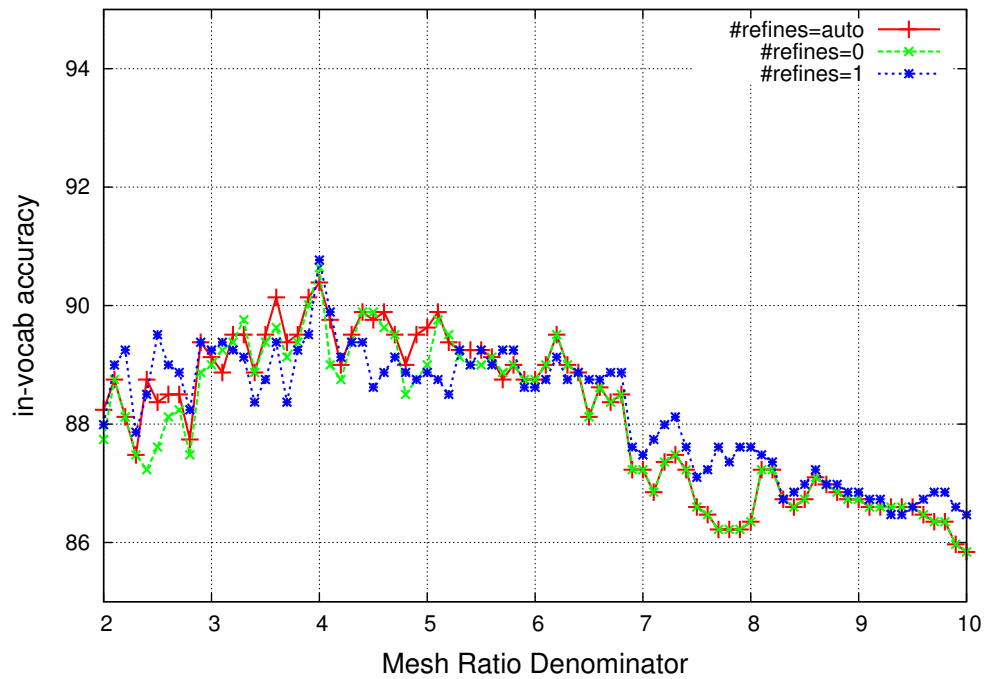
By default, our system automatically selects the number of mesh refinements to perform during the morphing algorithm for a given word. The number of refinements selected is the number of times it takes for the mesh size (width and height of each quad) to be less than 16 pixels. Alternatively, we can explicitly set the number of mesh refinements for every word in an entire dataset.

We test the accuracy of our system over a range of denominator values to see if there is a better value than 4 that we should use. We also test various settings for the number of refinements. Figure 4.24 shows the in-vocabulary recognition accuracy for each value used in the denominator, and for each setting for the number of mesh refinements. We perform similar tests for the Washington dataset and graph the results in Figure 4.25. All tests in this section incorporate the algorithm improvements from Sections 4.5 and 4.6, in addition to a bug fix from Section 4.8.

As is apparent from Figure 4.24a, the initial size of the mesh quads and the number of refinements only have a subtle effect on the overall accuracy for the Smith dataset. Viewing the same graph at a more precise scale (Figure 4.24b) reveals that there is a small peak where the denominator is 4, and a general trend toward lower accuracy as the denominator values move away from 4 in either direction.

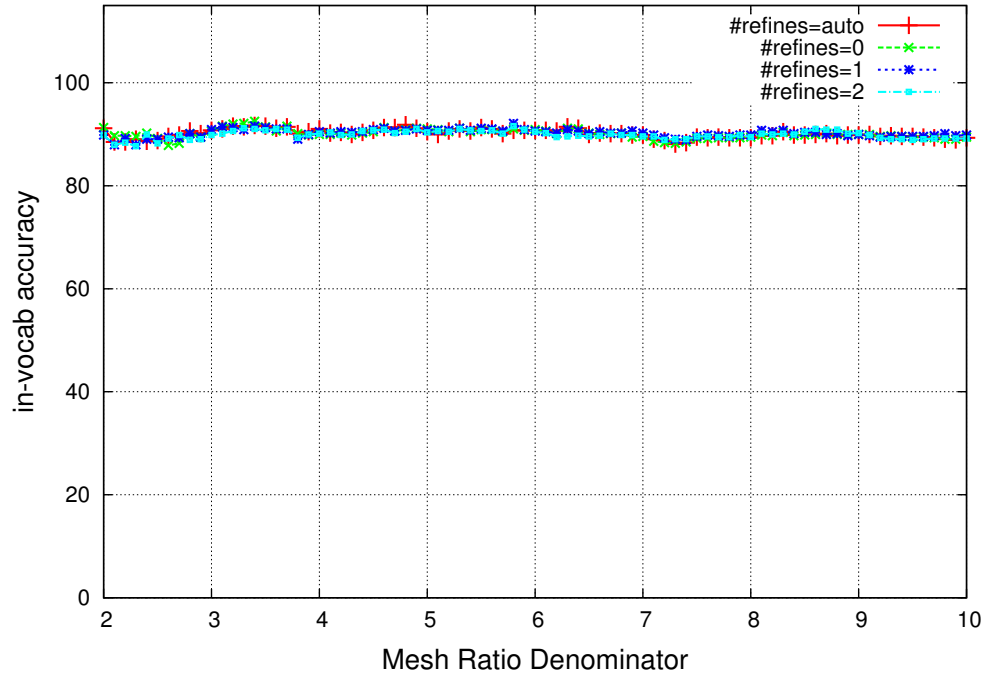


(a)

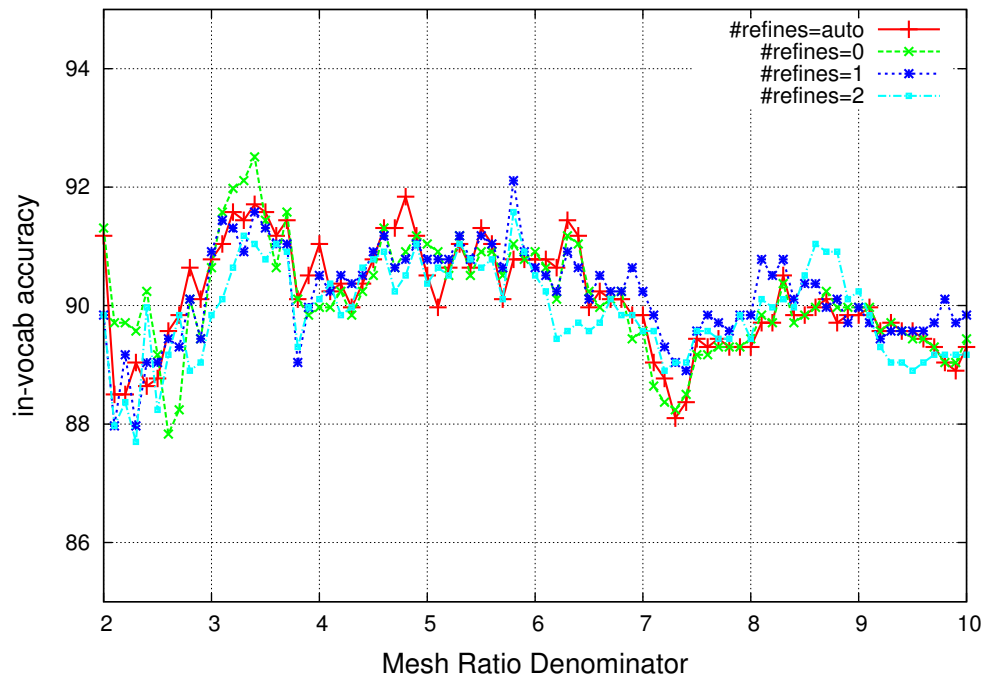


(b)

Figure 4.24: How initial mesh size ratio affects accuracy for Smith dataset. a) Overall effect is subtle. b) Peak at 4.0 with less accuracy elsewhere.



(a)



(b)

Figure 4.25: How initial mesh size ratio affects accuracy for Washington dataset. a) Overall effect is subtle. b) General trend is not clear, but the range 3.0 to 6.4 gives the best accuracy.

In Figure 4.25a, we see that the effect of initial mesh size and number of refinements is also subtle for the Washington dataset. More precise inspection in Figure 4.25b reveals that the peak and general trend of the graph are not as clear for this dataset as they are for the Smith dataset. However, we see that denominator values in the range from 3.0 to 6.4 generally result in higher accuracies than denominator values outside of that range.

For both the Smith dataset and Washington dataset, we see that automatically selecting the number of mesh refinements generally performs at least as well as explicitly setting a specific number of refinements for the peak ranges of denominator values.

Overall, we see that the amount of noise in the graphs of Figures 4.24 and 4.25 is almost as large as the slight difference in accuracy that is made by different settings for the mesh size ratio denominator and the number of refinements. We conclude that our choice of parameters (denominator=4.0 and #refines=auto) is a reasonable default setting for our system. It would be difficult to select better parameters for any given dataset without performing a similar experiment (using a range of parameter settings) on a representative training subset of that dataset.

4.10 Parameter Selection for Improved DP Alignment

In our analysis of recognition errors in Sections 4.2 and 4.3, we find that many recognition errors are at least partially caused by improper coarse (dynamic programming) alignment of the word images. The effectiveness of DP warping is influenced somewhat by the value chosen for the Sakoe-Chiba band width constraint (defined in Section 2.4.3). The Sakoe-Chiba band width constraint limits how much warping the dynamic programming algorithm permits when aligning the features of one word with those of another.

As shown in Figure 4.26, tightening or relaxing the constraint may help or hurt the alignment, depending on the word images being aligned. Using a constraint value of 15 does not allow enough leeway for the first three letters of the red “Sunday” to horizontally align with the blue training example in the horizontal direction (Figure 4.26a). Using a

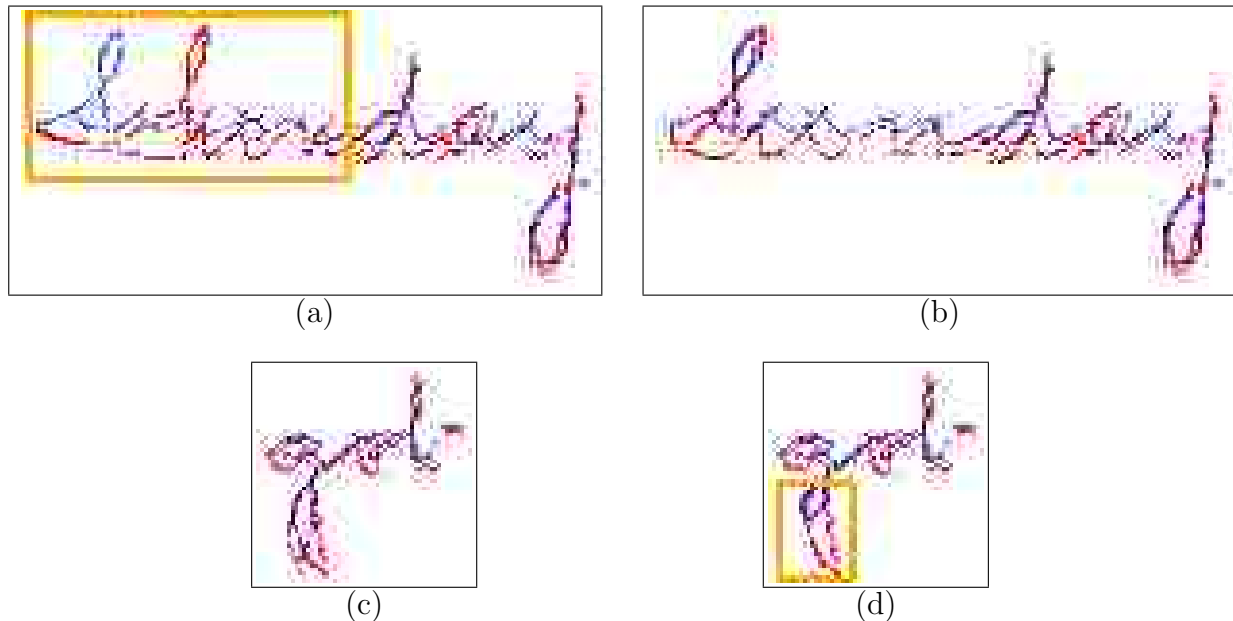
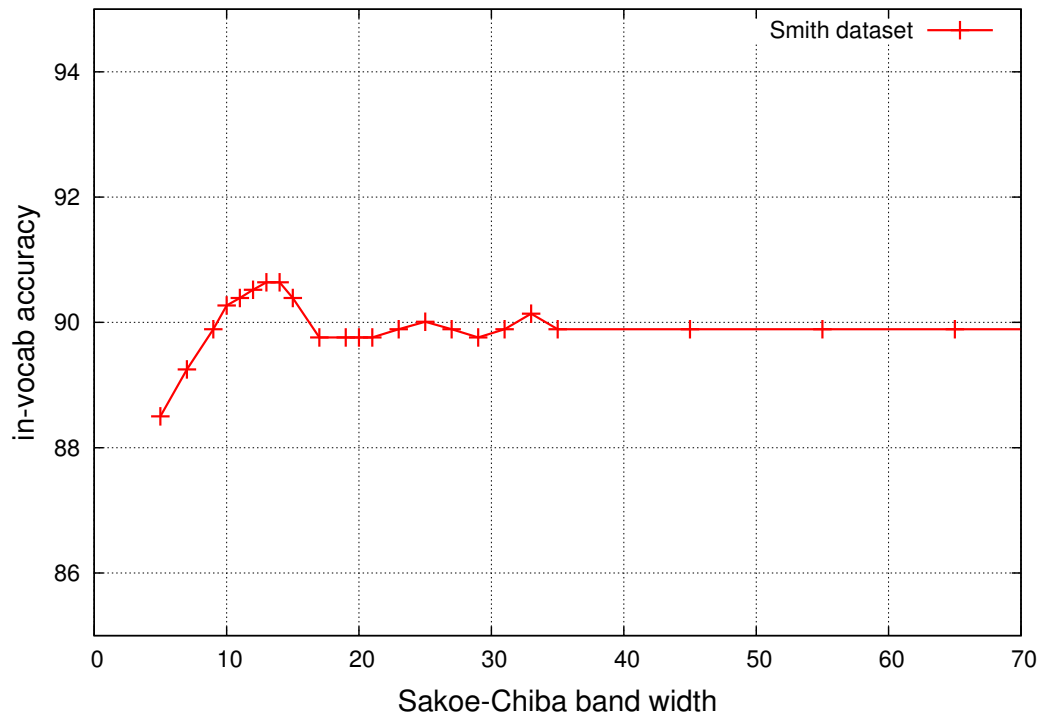


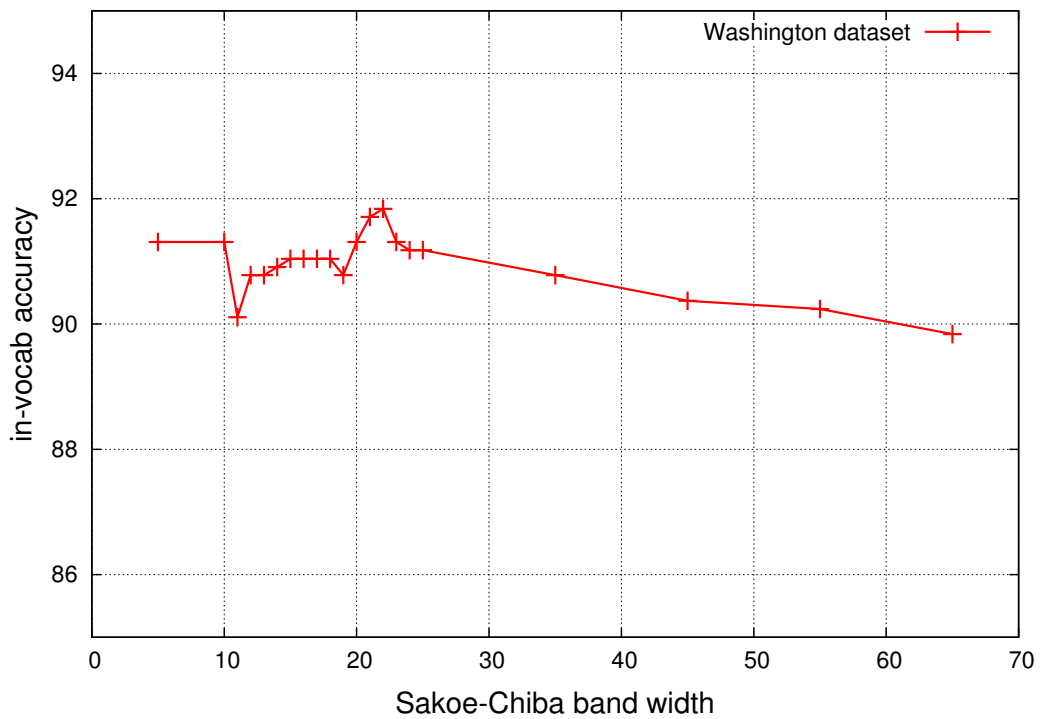
Figure 4.26: Adjustment of Sakoe-Chiba band constraint for Dynamic Programming (coarse alignment) step. a) Poor alignment of first half of word with constraint=15, b) better alignment with constraint=100, c) good alignment with constraint=15, d) descender of “g” aligns poorly with constraint=100.

value of 100 allows proper alignment (Figure 4.26b). The opposite is true for the vertical alignment of the descender of the blue “got” to the red example. Using a value of 15 works well (Figure 4.26c), but relaxing the constraint to 100 allows the descender to collapse too much (Figure 4.26d).

We experiment with various values for the Sakoe-Chiba band width constraint parameter to attempt to reduce the number of errors caused by poor coarse alignment. Our results are shown in Figure 4.27. We find that the best value is 14 for the Smith dataset and 22 for the Washington dataset. Since the Smith images are scaled smaller than the Washington images, it makes sense that the best constraint for the Smith dataset is smaller because less warping is needed at a smaller scale to keep a similar proportionality.



(a)



(b)

Figure 4.27: Accuracy for a range of values of the Sakoe-Chiba DP band width parameter. a) Smith; b) Washington.

4.11 Using $C_{0\leftrightarrow 1} = \max(C_{0\rightarrow 1}, C_{1\rightarrow 0})$ and $C_{0\leftrightarrow 1} = \min(C_{0\rightarrow 1}, C_{1\rightarrow 0})$

In Section 4.2, we observe that there are many cases in which the value of $C_{0\rightarrow 1}$ is low but $C_{1\rightarrow 0}$ is high, or vice versa. In that section, we also speculate that it might be better to use either the maximum of the two or the minimum of the two as our word matching cost, instead of the sum of both $C_{0\rightarrow 1}$ and $C_{1\rightarrow 0}$.

We experiment using both $C_{0\leftrightarrow 1} = \max(C_{0\rightarrow 1}, C_{1\rightarrow 0})$ and $C_{0\leftrightarrow 1} = \min(C_{0\rightarrow 1}, C_{1\rightarrow 0})$. When we use the maximum, we find that our in-vocabulary recognition accuracy decreases from 89.38% to 85.97% for the Smith dataset and from 88.50% to 85.03% for the Washington dataset. When we use the minimum, our system is completely ineffective. Accuracy for the Smith dataset decreases from 89.38% to 0.13%, with only a single word being recognized correctly. We conclude that we should continue to use the sum of both $C_{0\rightarrow 1}$ and $C_{1\rightarrow 0}$ instead of just the maximum or minimum of the two.

4.12 Improved Results for Smith and Washington Datasets

In Sections 4.4 through 4.11, we experiment with various modifications to our algorithm and parameter settings. In this section, we incorporate all of the modifications that we find to be improvements, including the changes in Sections 4.4.3, 4.5, 4.6, and 4.8. Our final cost metric that replaces Equation 2.5 is

$$C_{0\rightarrow 1} = \frac{1}{\|A_0\|} \sum_{i=1}^{\|A_0\|} D_1(A'_0[i]) + \frac{1}{\|A_1\|} \sum_{i=1}^{\|A_1\|} D'_0(A_1[i]) + (p) \frac{w_{long} - w_{short}}{w_{long}}, \quad (4.6)$$

where the first two terms are the improved metric from Equation 4.4 (Section 4.5), and the last term is the term from Equation 4.3 (Section 4.4.3) that penalizes mismatched word lengths.

For the tests in this section, we use $p = 0.1$ for the mismatched word length penalty parameter for both the Smith dataset and Washington dataset. For the Smith dataset, we set the Sakoe-Chiba band width constraint to 14 and the mesh size ratio denominator to 4.0.

Table 4.1: Recognition Accuracy After Improvements and Tuning

Dataset	Previous Accuracy		Improved Accuracy	
	Total (# correct/# possible)	In-Vocabulary (# correct/# possible)	Total (# correct/# possible)	In-Vocabulary (# correct/# possible)
Smith	70.70% (707/1000)	89.38% (707/791)	71.70% (717/1000)	90.64% (717/791)
Washington	66.50% (665/1000)	88.90% (665/748)	68.50% (685/1000)	91.58% (685/748)

For the Washington dataset, we set the Sakoe-Chiba band width constraint to 22 and the mesh size ratio denominator to 4.8. The resulting word recognition accuracies are shown in Table 4.1, along with the previous accuracies from Figure 2.10 in Chapter 2.

After the algorithm and parameter improvements, we see that in-vocabulary recognition accuracy is over 90% for both datasets, and total recognition accuracy (including out-of-vocabulary words) is 68.5%–71.7%. These are small, but noticeable, improvements over the previous accuracies. Since the parameters are optimized based on actual recognition results, these accuracies represent an upper bound on how well our algorithm can perform on these two datasets without either making additional improvements to the algorithm itself or taking additional processing steps (such as incorporating language models).

4.13 IAMDB Dataset Results

The IAM Handwriting Database (IAMDB) is a large, multiple-author dataset of English handwritten text, made available by Marti and Bunke at the University of Bern [28]. The database has examples of handwriting by 657 different writers over 1,539 pages (13,353 text lines) for a total of 115,320 labeled words. We divide the database according to the “Large Writer Independent Text Line Recognition Task” defined in version 3.0 of the IAMDB. The task specifies a training set, two validation sets, and a test set for this task. There are multiple writers for each set, and none of the writers from any set are used for data in any

Table 4.2: Recognition Accuracy on IAMDB Dataset (large, many writers)

IAMDB Dataset	# writers	# words	Total Accuracy (# correct/# possible)	In-Vocab Accuracy (# correct/# possible)
Train	283	46,947	—	—
Validation 1	46	7,894	55.30% (4,365/7,894)	67.14% (4,365/6,501)
Validation 2	43	8,556	56.88% (4,867/8,556)	67.45% (4,867/7,216)
Test	128	17,584	55.53% (9,764/17,584)	65.93% (9,764/14,809)

other set, so the words used for testing are written by completely different people than the words used for training.

To tune our system parameters, we use a small subset of the training and validation sets, but none from the test set. We arbitrarily select 819 correctly segmented words from the full training set as training examples for tuning, and 1106 words (some of which are not correctly segmented) from the two validation sets as test words for tuning. On this tuning subset, we find that our best accuracy occurs when using 15 for the Sakoe-Chiba band width constraint and 3.8 for the mesh size ratio denominator. For the word length mismatch penalty, we use $p = 0.28$.

For our experiments, we use all of the words from correctly segmented text lines of the training set as training examples for our system (46,947 words²). We test each of the validation sets and the test set using the parameters selected from our tuning subset. We include all words in the tests involving the validation sets and test set, including those from incorrectly segmented textlines. Results of our experiments are shown in Table 4.2.

Recognition accuracy on this dataset is significantly lower than on the Smith and Washington datasets. This is expected because instead of recognizing the words of a single writer, we are recognizing words from many different writers, none of whom are the same

²The task specifies 6,161 text lines (53,807 words) for training, but we ignore words from 742 text lines that are segmented incorrectly. We only ignore lines with segmentation errors for training, not for testing.

people used to train the system. In addition, the vocabulary is much larger, requiring more words to be distinguished from each other.

It is important to note that these raw recognition results are obtained without using any sort of language models (word trigrams, for example). Such models have been shown to increase recognition accuracy significantly by leveraging the context of surrounding words, and will undoubtedly increase our accuracy in future work. Our results are also from this one recognition method, whereas some recent work reporting higher accuracy uses multiple recognizers or ensemble methods to improve accuracy. In future work, there is no reason our method could not also be used in consort with other methods to improve overall accuracy. It is also important to note that our results are measured on test sets that include word segmentation errors (about 5.08% of the words if the count in [28] is consistent for the entire dataset). These words will almost certainly be recognized incorrectly by our method, even though the errors are caused by the segmentation algorithm, not our recognition method itself. While this gives us a realistic idea of how our recognizer would perform in an end-to-end system, it should be noted that improvements in segmentation (which are outside the scope of this dissertation) will also improve our accuracy.

In Section 4.14, we show that our raw recognition results are better or at least close to other results for the IAMDB dataset found in the literature, when ignoring the improvements made by language models and ensembles.

4.14 Comparison to Hidden Markov Model (HMM) Approach

Using the results of our method on the IAMDB dataset (Section 4.13), we are able to estimate how accurate our method is compared to the Hidden Markov Model (HMM) approach used by other authors who use the IAMDB dataset. Since our method does not yet incorporate language models and other planned improvements, we attempt to compare our raw results to those of the HMM methods before language models are applied to those methods. Due to differences in experimental setup, metrics, versions of the dataset, and the way the dataset

is used in the various papers, an exact comparison is not possible. However, the indirect comparison gives us a reasonable idea of how our method performs in the context of other methods found in the literature.

Marti and Bunke (IJPAI 2001)

Marti and Bunke present their initial HMM approach to HR in [27], and report their recognition rate when incorporating various statistical language models. When using word bigram language models, they achieve recognition rates as high as 63.39% and as low as 60.05% for different sizes of vocabulary. However, when using only a lexicon without additional language models, they only achieve recognition rates from 40.47% to 51.44%, depending on vocabulary size (higher recognition rates are achieved with smaller vocabularies, as expected).

For comparison, we achieve 55.30% to 56.88% total accuracy (Table 4.2). This is not a direct comparison for two reasons. The first reason is that Marti and Bunke use an earlier version of the IAMDB that has less data overall, and does not use the same breakdown of training / validation / test data. However, the results of our method are quite consistent over multiple mutually-exclusive subsets of the data (validation 1, validation 2, and test are all within 1.6% of each other), so it is probably safe to assume that our accuracy would be fairly consistent on the data used by Marti and Bunke.

The second reason this is not a direct comparison is that Marti and Bunke do not specifically define how they calculate their recognition rate metric. We use $\# \text{ correct} / \# \text{ possible}$, where $\# \text{ possible}$ is the number of pre-segmented word images (including those that have been segmented incorrectly). They appear to be using the number of words in the ground-truth transcription as the $\# \text{ possible}$ instead of the number of word images, in which case they are dividing by a slightly different number than we are. According to [28], 3.62% of the words in the IAMDB are over-segmented and 1.46% are undersegmented, as calculated on a small subset of the database. If those numbers hold for the entire database, there are approximately 2.16% more pre-segmented word images than actual ground-truth

transcription words, meaning our accuracy numbers would compare even more favorably with theirs than our current metric shows, although by a small amount. Using that metric, our total accuracy on the test set would be about 56.75% (9,764/17,205) instead of the 55.53% reported in Table 4.2.

Vinciarelli, Bengio, and Bunke (TPAMI 2004)

Vinciarelli, Bengio, and Bunke [50] report their results with HMM and language models, including word trigrams in addition to unigrams and bigrams. In this paper, a much larger vocabulary is used than in [27] (from 10,000 to 50,000 words instead of about 7,000), and the vocabulary is no longer closed, meaning there are words in the test data that are not in the lexicon or language model. The baseline method without language models achieves between about 29% accuracy (lexicon size = 50,000 words) and 35% accuracy (lexicon size = 10,000). Even with trigrams, their best accuracy is less than 46.5%, much lower than ours.

As with the previous comparison, only an indirect comparison can be made with these results because the authors only use a subset of the IAMDB data for their training, validation, and tests sets, and they do not specify exactly which textlines are used for each. For the sake of comparison, we again assume that our results would be consistent for the subsets of the database used by those authors, since our results are consistent for the test set and both validation sets specified in Version 3.0 of the IAMDB.

Zimmermann, Chappelier, and Bunke (TPAMI 2006)

Zimmermann, Chappelier, and Bunke [53] use an “enhanced and optimized version” of the HMM recognizer used by Marti and Bunke in the IJPAI 2001 paper. [27] Zimmermann et al. report word recognition rates up to 79.4% when using language models and Stochastic Context-Free Grammars (SCFG), and nearly as high — 79.3% — when using the language model but no SCFG. Word recognition rate is defined as $(N - D - S)/N$, where D is the number of deletion errors (when a space between words is missed), S is the number of words

recognized incorrectly, and N is the number of words in the transcription. Thus, word recognition rate is the percentage of words recognized correctly. They also report a metric they call word level accuracy, defined as $(N - D - S - I)/N$, where I is the number of insertion errors (when a single word is erroneously split into two). The word level accuracy is 77.6% with language models and SCFG, and 76.8% with only language models but no SCFG.

When the HMM method is used without any language models or SCFG, their results are much worse. They report a word level accuracy of only 49.1%. Since they report only word level accuracy and do not report recognition rate, we cannot do a direct comparison of our results with those of their HMM approach using no language models. However, if we assume that the difference between accuracy and recognition rate is about 2.5% (as it is when using the language models), then their recognition rate is about 51.6%, lower than our estimated rate of 56.75%. Assuming that the difference is within 2.5% may be conservative, since in the Vinciarelli et al. TPAMI 2004 paper [50] there is greater difference between the metrics (about 15%) when no language model is used than when language models are used (about 4%). Using that additional information, we adjust our estimate to a difference of 9.38% between the metrics ($15\%/4\% = 9.38\%/2.5\%$), meaning the Zimmermann et al. method may achieve a recognition rate of up to about 58.48%, slightly better than our estimated 56.75%. Again, we assume that the results on version 3.0 of the dataset are consistent with those on the version they use. Their test set is only about 22.5% as large as the test set specified in version 3.0 of the IAMDB that we use.

Since several assumptions and estimates must be made to compare our method to that of Zimmermann et al., we cannot be completely confident that comparison is correct, but this at least gives us a reasonable estimate of how our method compares. It seems likely that if our recognition method is not more accurate, then it is at least close.

Bertolami and Bunke (PR 2008)

Bertolami and Bunke [2] achieve accuracies as high as 67.17% using HMM-based ensemble methods and combinations of recognizers. They appear to use the same version of the IAMDB that we use, but they use a different division of data for validation and testing. Their baseline recognizer (a single optimized recognizer) achieves 64.48% accuracy, where the accuracy metric is the same as the word level accuracy metric in the Zimmermann et al. paper in the previous subsection. Unfortunately, they do not report accuracy for their HMM system without language models. However, we can again estimate the accuracy by making some assumptions.

Assuming that language models increase accuracy in this case by the same amount that they do in the Zimmermann paper ($76.8\% - 49.1\% = 27.7\%$), the accuracy of the baseline system without language models would be about 36.78% ($64.48\% - 27.7\%$). If recognition rate is from 2.5% to 9.38% better than word level accuracy, then the recognition rate of Bertolami and Bunke would be from 39.28% to 46.16%, well below our estimated recognition rate of 56.75%. Since we are not completely certain that the assumptions we are making are correct, we cannot positively say that our method is better than their HMM method. However, we are confident that if our recognition rate is not better then it is at least close to theirs.

Analysis of Comparisons

Since using language models and multiple recognizers to improve our accuracy is not currently implemented for our handwriting recognition method, we are unable to perform direct comparisons between our method and HMM methods found in the literature that do use those improvements. However, we are able to make indirect comparisons based on some assumptions of consistency, as described in the previous subsections. We compare our method to the results (or estimates of the results) that those methods achieve without the improvements of language models, ensembles, and multiple recognizers. We find that when ignoring the

improvements, our method seems to be either more accurate or at least almost as accurate as existing methods that report results for the IAMDB. Assuming that improvements such as language models and multiple recognizers will improve our accuracy as much as the accuracy of those methods, we conclude that our method compares favorably with methods in the literature that report results for the IAMDB.

4.15 Conclusion

In this chapter, we have reported significant additional analysis of our handwriting recognition method. We have analyzed the types of recognition errors that occur and the causes of those errors. We have experimented with various modifications to our algorithm and found that some of those modifications improve our recognition results. In particular, we found that the improved word matching cost metric in Section 4.5), the word length mismatch penalty (Section 4.4.3), and the improved handling of distance map boundaries (Section 4.6) are all improvements to our method. We also experimented with different parameter settings to gauge their effect on the accuracy of our method. We find that our final improved results, including tuned parameters, allow us to achieve in-vocabulary recognition rates of greater than 90% for both the Smith and the Washington datasets.

In addition, we have tested our method on the IAMDB, a large, multi-author dataset that is publicly available. Our method achieves in-vocabulary accuracies of over 65% (over 55% including out-of-vocabulary words). We presume that when we add language models and multiple recognizers, we will achieve similar accuracy improvements as other authors. Under that assumption and other assumptions of consistency specified in Section 4.14, we find that our method compares favorably with HMM methods reported in the literature that also report results for the IAMDB data.

Some of the information in this chapter will be included in a future journal submission. The rest has helped us to understand the strengths and weaknesses of our method and will inform our search for ways to improve the method in the future.

Chapter 5

Conclusion and Future Work

In this dissertation, we have presented a novel method of offline handwriting recognition. The method consists of using 1-D dynamic programming for coarse-alignment of word medial axes in both the horizontal and vertical directions, improved medial axis alignment using an automatic word morphing algorithm, and distance maps to compute how different the aligned medial axes are. Our method provides an entirely new approach to handwriting recognition that can be used as a basis for future research and improvement.

We have found that our method (after improvements and tuning) achieves greater than 90% in-vocabulary recognition accuracy (about 70% accuracy when counting errors due to out-of-vocabulary words) on two different pre-segmented single-author datasets. We also find that our method achieves greater than 65% in-vocabulary accuracy (about 55% when including errors due to out-of-vocabulary words) for a large, many-author dataset. These results are all raw recognition rates achieved without using any language models or other postprocessing, which would improve our accuracy significantly. Indirect comparisons indicate that these rates may be higher (or at least similar to) the rates achieved by HMM-based recognizers found recently in the literature, if improvements due only to language models, multiple recognizers, and other postprocessing steps are discounted for those methods. Since the scope of this dissertation does not include preprocessing, segmentation, or postprocessing using language models and combinations of recognizers, inclusion of those steps to create an entire HR system that uses our recognition method is left as future work.

We have also shown that our word comparison method is flexible enough to be applied to at least one other related problem area — that of signature verification and forgery detection. In fact, our method performs competitively with other methods designed specifically for that purpose, and even performs competitively across both Dutch and Chinese with virtually no language-specific optimizations.

Due to its good performance in HR and signature verification combined with the intuition we gained from the additional analysis in Chapter 4, we believe that our HR method will extend well to other related application areas, including word-spotting and general 2-D shape recognition. Since it handles Dutch and Chinese signatures competitively without any language-specific tuning, we also believe that our method may work well for recognition with non-English languages, including some not using Roman/Latin-based scripts such as Chinese and Arabic. We anticipate future work to explore these lines of research.

One limitation of our HR method is that it compares each test word to every training word in order to choose which training example the test word matches best. As training sets become very large (tens of thousands to millions of training examples), our method becomes unreasonably slow on current hardware. Having done some preliminary research with word clustering and hierarchical recognition, we anticipate additional future work along these lines to address the issue of limited scalability using our HR approach.

Another limitation of our method is that it can never correctly recognize out-of-vocabulary words. It can only recognize words that it has seen training examples of, and then only if the training example is similar enough to the test word. If the author of the training example has very different handwriting than the author of the test word, it is unlikely that a word will be recognized correctly even if it is in the vocabulary. As future work, one possible solution is to use synthetic training data — artificially simulated training data — to expand the dataset to include examples of words for which no handwritten training examples exist, or only training examples from very few authors. Training data could be simulated by combining pre-segmented handwritten characters of various authors or by using handwriting

text fonts to create word images for the words in the lexicon that need to be artificially simulated.

In the future, we also consider investigating the possibility of using our morphing-based recognition algorithm as a subword matching strategy (in addition to matching entire words) by performing localized stretching to match parts of words. This may allow discrimination between some of the very minor differences between words that are currently problematic for our method (such as “m” and “n” in “them” vs. “then”).

As future work, we also consider possible improvements to the recognition method, itself. Many of the errors it currently makes are at least partially due to poor coarse alignment, from which the morphing algorithm sometimes cannot recover. There may be a way to use the global distribution of ink of the words being matched to guide the morphing algorithm, instead of just locally perturbing mesh control points. Such a guided approach might prove helpful in preventing strokes from becoming trapped in the wrong part of the word when coarse alignment is incorrect, and may also prevent multiple distinct strokes from collapsing into the same local area. It might even be possible to improve the initial coarse alignment by taking into account the global distribution of ink in the words being aligned.

One final item of future work that should be mentioned is the combination of our whole word HR approach with subword analytic approaches. Whole word methods depend on the overall shape of a word and its strokes, while analytic approaches (such as the HMM-based methods) depend more on local features and their context. We believe that using our 2-D warping based HR method in consort with HMM-based approaches would provide even better accuracy than either approach achieves on its own by taking advantage of both the local features and the entire word.

Appendix A

Datasets

In this appendix, we describe the datasets that we use in the dissertation. When possible, we also show examples of some representative data from the datasets.

A.1 Smith Dataset

The Smith dataset is a single-author dataset consisting of handwritten word images extracted from an online diary of Jane (Jennie) Hill Leavitt Smith. Specifically, the words are extracted from a few sequential pages beginning at page 80 in volume 1 (1916–1917) of Smith’s mission journals, available at the BYU Harold B. Lee Library website in the Mormon Missionary Diaries collection at URL <http://lib.byu.edu/dlib/mmd> (Figure A.1).

We created an interactive “lasso”-style selection tool that allows us to manually trace around individual words in document page images using a stylus on a Wacom tablet (Figure A.2). We manually assign each word its ground truth label (transcription).

Before creating word images from the lasso regions, we remove the background from grayscale versions of each page image using the background removal method of Hutchison and Barrett [12]. A large median filter is used to approximate the background that should be removed, and histogram stretching is performed for contrast enhancement after background removal. We determine a binarization threshold of the background-removed page image using the method we introduced in [14]. We then estimate the average slant of the handwriting on the page using a smoothed histogram of (squared) maximum runlengths of black pixels

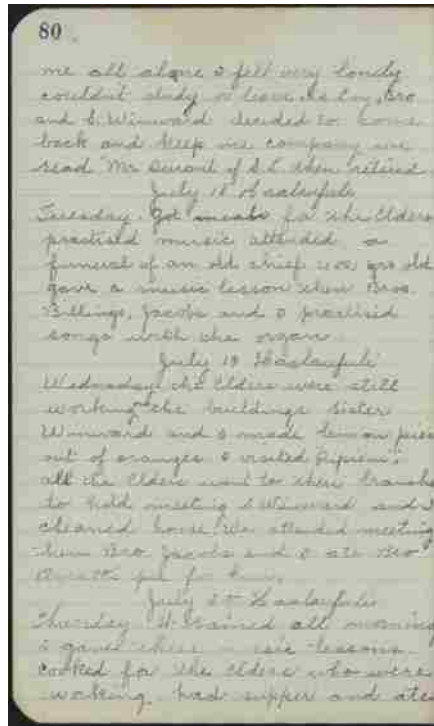


Figure A.1: A page image from the Smith diary.

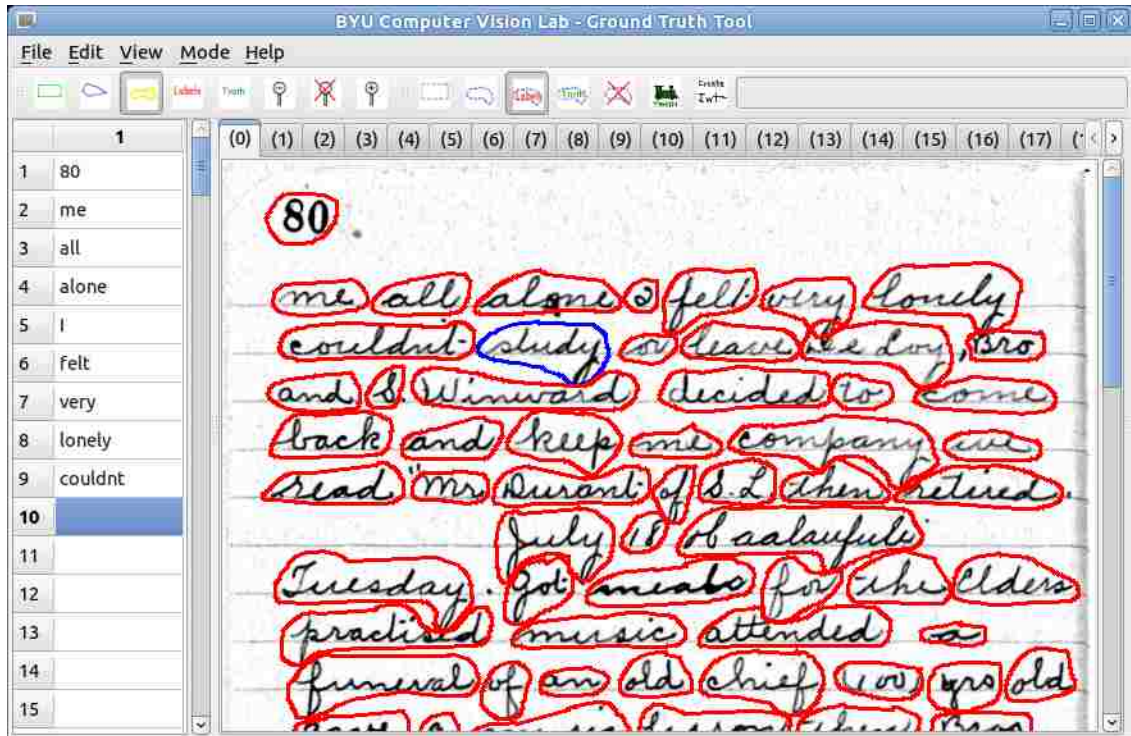


Figure A.2: Ground truthing program for Smith and Washington datasets. We mark words with a “lasso”-style tool and then enter ground truth labels (transcriptions) for each word.

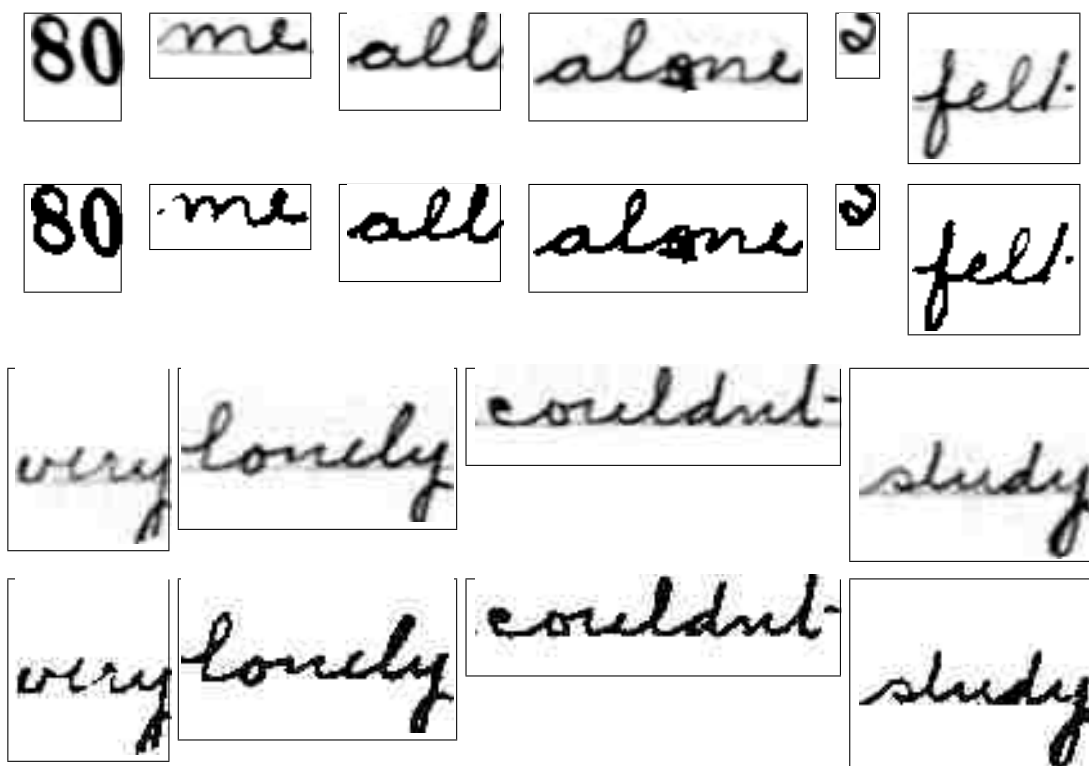


Figure A.3: Example word images from the Smith dataset. Grayscale and binarized versions of the first ten word images are shown.

in each possible direction, across the image. We ignore the image border areas where page edges, noise, and margin lines may introduce error into the slant angle estimate.

To create each word image, we ignore any ink outside of the lasso region for that word. We remove handwriting slant by performing a geometric shear in the horizontal direction, using the page-level estimate of average handwriting slant to determine how much to shear the word. We estimate where the upper and lower baselines of the word are using a profile-based method, and then crop / pad the unslanted word image based on where the baselines are found. The page-level threshold value is used to binarize the word image, and meta data (including the ground truth label) is saved as comments within the image. The first few word images of the dataset are shown in Figure A.3, both in grayscale and thresholded black-and-white format.

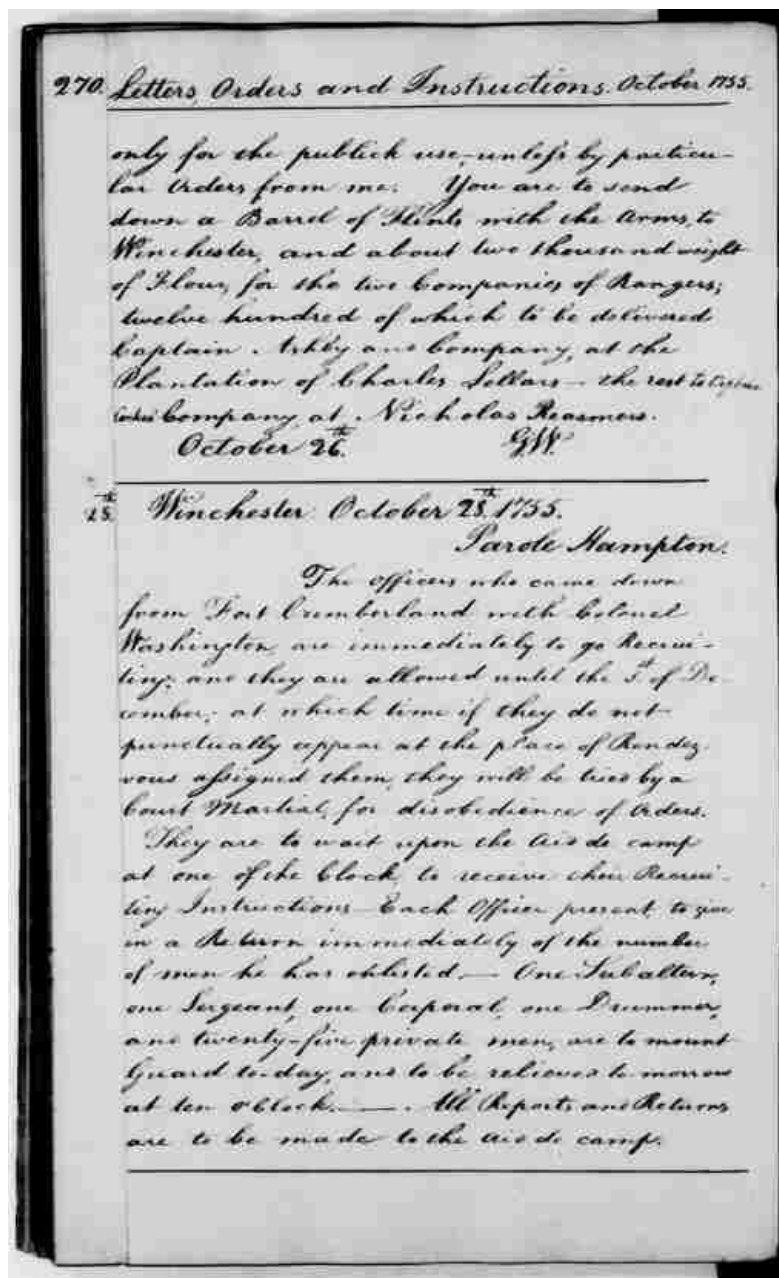


Figure A.4: A page image from the Washington letters [21].

A.2 Washington Dataset

Original page images for the Washington dataset come from a few pages of George Washington's manuscripts, downloaded from <http://ciir.cs.umass.edu/downloads> [21]. An example of one page image is shown in Figure A.4. We create individual word images and ground truth labels using the same tool and preprocessing steps as we do for the Smith dataset.

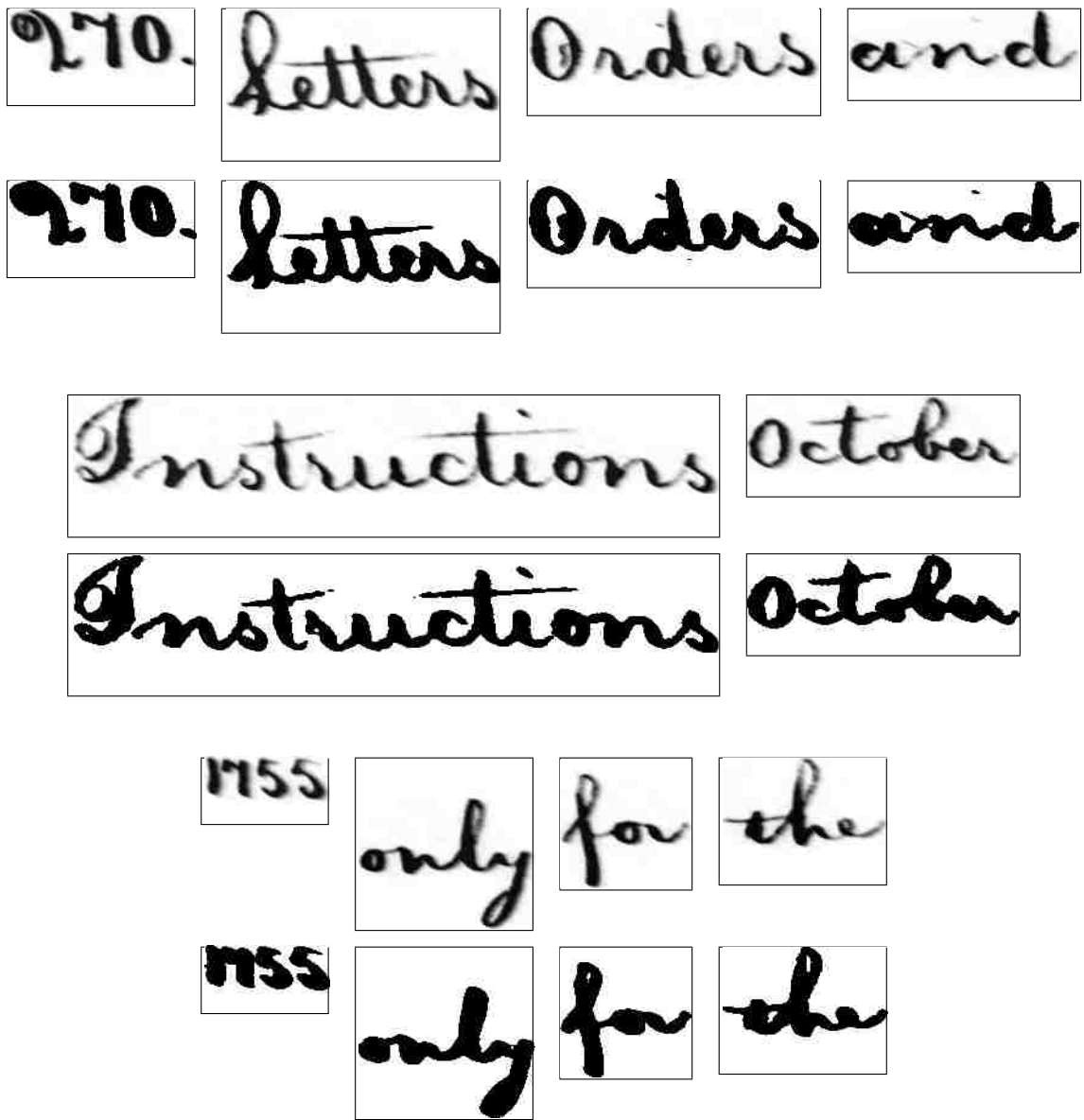


Figure A.5: Example word images from the Washington dataset. Grayscale and binarized versions of the first ten word images are shown.

Grayscale and binarized versions of the first few word images of the dataset are shown in Figure A.5. The slant removal is more noticeable in this dataset than in the Smith dataset because the handwriting is consistently more slanted to begin with than in the Smith dataset.

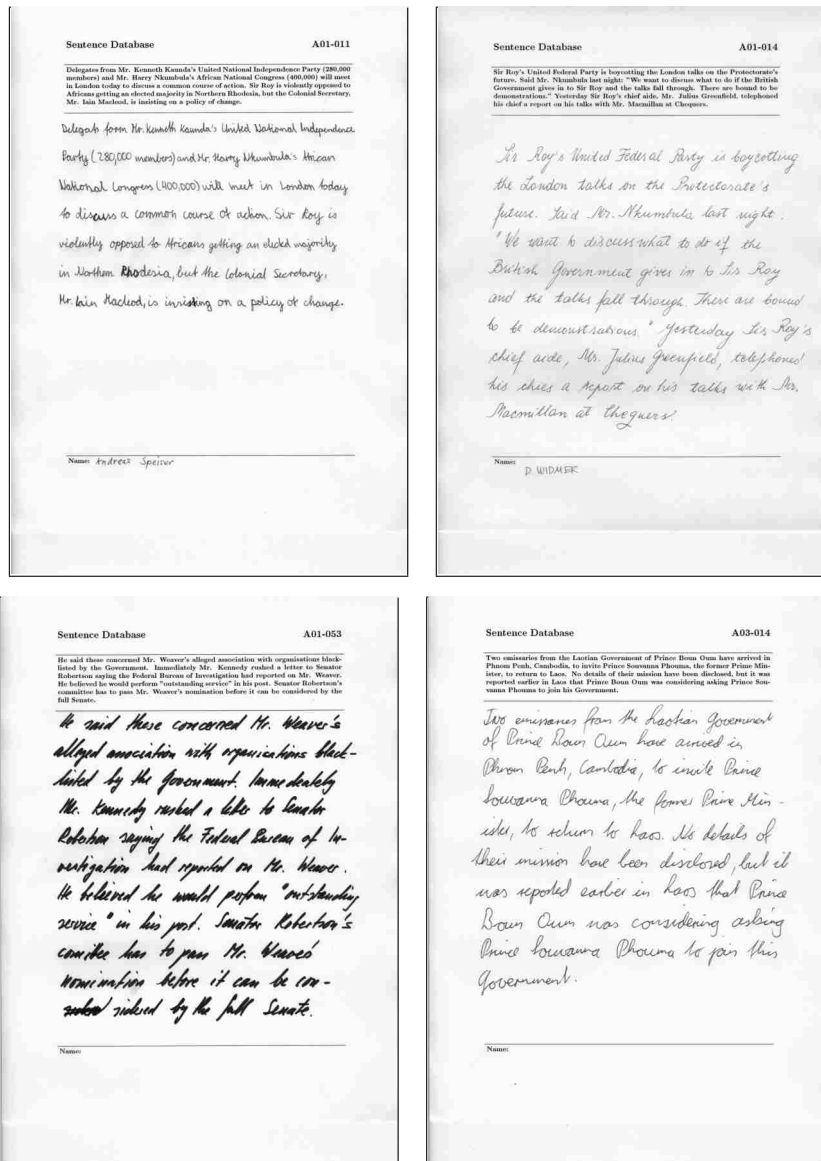


Figure A.6: Example pages from the IAMDB [28], made available by Marti and Bunke at the University of Bern.

A.3 IAM Database (IAMDB version 3.0)

The IAM Handwriting Database (IAMDB) is a large, multiple-author dataset of English handwritten text, made available by Marti and Bunke at the University of Bern [28]. The database has examples of handwriting by 657 different writers over 1,539 pages for a total of 115,320 labeled words. The neatness and penmanship of the writing varies widely. Each page has typed sentences at the top that are handwritten below by the writer (Figure A.6).

The database includes significant metadata, including the ground truth labels of words and the coordinates of bounding boxes for each ink component belonging to any given word. The meta data also contains an annotation for each textline specifying whether or not the entire textline is separated correctly into words.

Instead of using our lasso tool to segment words, we use the list of ink component bounding boxes for each word and set pixels that are not within any of the bounding boxes to the background color (white). We perform background removal, binarization, slant removal, and cropping of the image. For slant removal, we calculate the slant of any given word as a weighted combination of the slant estimates for the entire form, for the textline, and for the individual word. For ground truth labels, we use the ground truth that is already provided with the dataset.

A.4 BYU English Signature Dataset (Blind and Casual Forgeries)

The BYU English Signature Dataset includes 192 genuine signatures (16 authors, 12 signatures each) and 256 forgeries (8 authors, 16 blind and 16 casual forgeries each). The 16 genuine signatures of each author were collected in a single session on a printed form that allowed us to easily separate the ink of the signature from the form lines after scanning the completed form (Figure A.7). One example of each genuine signature is shown in Figure A.8.

The 8 forgers are not the same people as those who provided genuine signatures. Each of the forgers provided 16 blind forgeries (Figure A.9) and 16 casual forgeries (Figure A.10). The blind forgeries were collected first from each forger while he or she could only see a machine-printed version of each name to forge. Then, the forger was given the pages of genuine signatures so he or she could see all of the genuine examples of each signature while creating the casual forgery for that signature. The forger was not allowed to practice the signature in advance since that would constitute a skilled forgery instead of a casual forgery.

Participant Consent and Release Form
Signature Verification Research Dataset

To participate, you must first read and agree to the following statement:

I understand that my signatures will be scanned and included in a digital dataset made available for research purposes. Researchers and others may use the dataset to research, develop, train, and test algorithms and systems for signature verification, signature recognition, forgery detection, writer identification, or similar technological or forensic tasks. The dataset may be used and displayed or printed in academic publications (conference papers, journal articles), presentations, technical reports, research publicity activities, and for other similar uses. The dataset may be made publicly available over the Internet or using any physical, electronic, or other medium. I expressly give consent for my signatures to be used in this dataset and release the makers and distributors of the dataset from responsibility for any potential misuse of my signatures that may result from making the dataset publicly available.

I have read and agree to the statement above. Mum Paull 3-13-12
Signature Date
Printed Name

Please sign your standard signature inside each of the red boxes below (stay within the box)

The left form shows a grid of 12 boxes, each containing a handwritten signature of 'Mum Paull'. The right form shows a grid of 12 boxes, each containing a handwritten signature of 'Kristen Mami'.

Figure A.7: Genuine signature collection forms.



Figure A.8: Examples of genuine English signatures for all 16 authors.

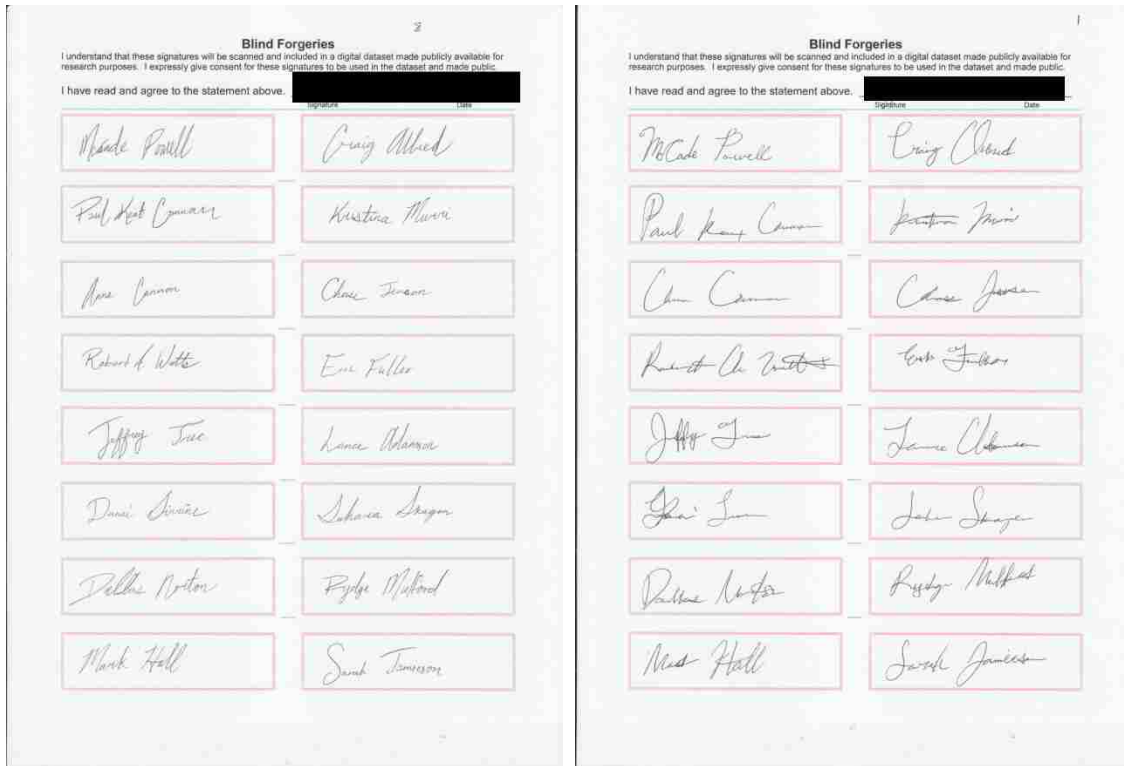


Figure A.9: Blind forgery collection forms for two forgers.

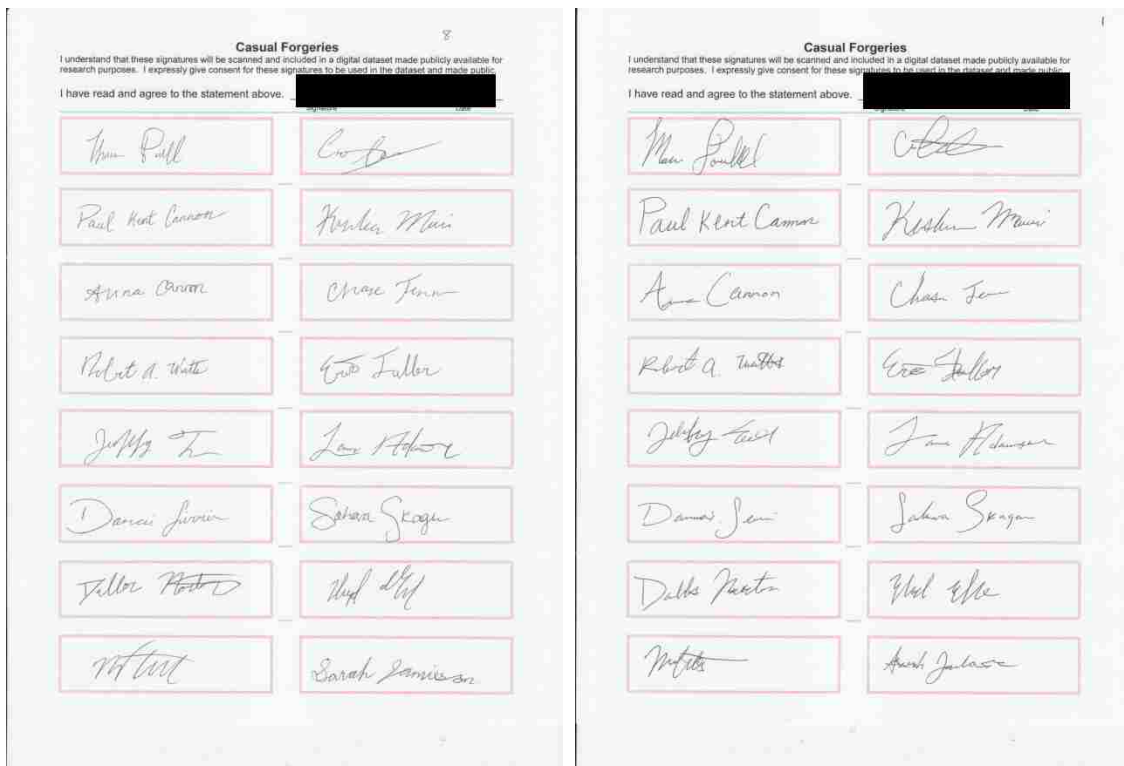


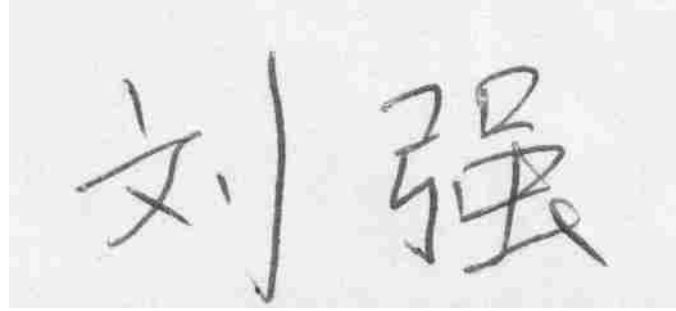
Figure A.10: Casual forgery collection forms for the same two forgers.

A.5 SigComp2011 Dutch and Chinese Signatures (Skilled Forgeries)

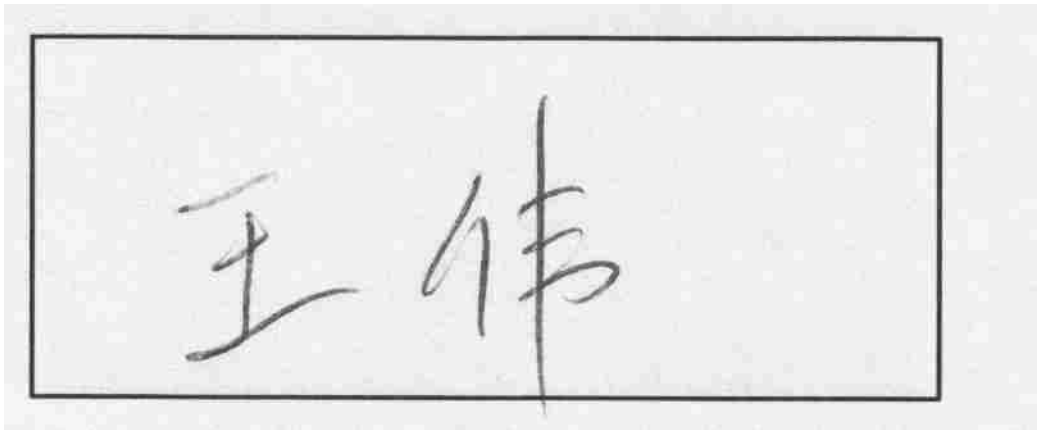
We use the offline portion of the dataset from the ICDAR 2011 Signature Verification Competition for Online and Offline Skilled Forgeries (SigComp2011) [23]. The dataset contains Chinese signatures and Dutch signatures. Forgeries in this dataset are skilled forgeries, meaning not only that the forgers could see the genuine signatures while forging them, but the forgers also had the opportunity to practice forging each signature before forging it for the dataset. The Chinese portion of the dataset has 116 genuine forgeries by 10 authors and 487 questioned signatures (120 genuine, 367 forgeries). The Dutch portion of the dataset has 648 reference signatures by 54 authors and 1286 questioned signatures (648 genuine, 638 forgeries).

Due to a strict license agreement required to use this dataset, we are not allowed to reprint any of the signature images. Dutch signatures seem to be visually fairly similar to English signatures, in general. Like with English signatures, it is often difficult to tell what a name actually is, due to the creative way in which people sign their names. Chinese signatures are visually quite different than English or Dutch signatures, as one might expect. While still stylized by each individual, Chinese characters form the basis of the signatures instead of Roman script.

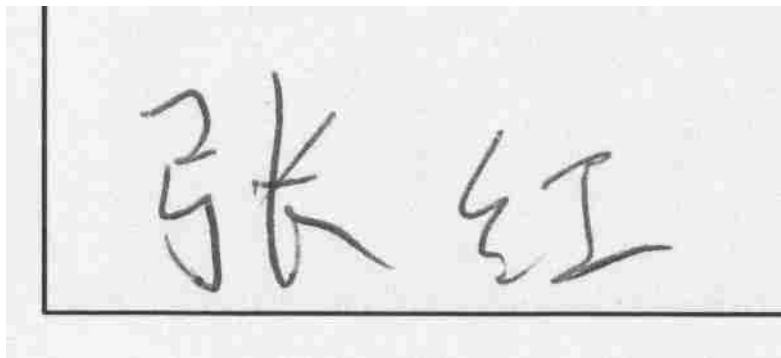
Signature images in this dataset are pre-cropped from the collection forms. The cropping for the Dutch signatures is good, properly extracting the signatures from within the form boxes. However, the cropping for the Chinese portion of the dataset is inconsistent. Sometimes the form box is properly removed, sometimes it is still in the image, and sometimes part of the form box (but not the whole box) is removed from the image. We illustrate the inconsistent cropping of Chinese signatures in Figure A.11, using signatures that are not actually from the SigComp2011 dataset.



(a)



(b)



(c)

Figure A.11: Illustration (with our own images) of inconsistent cropping for Chinese signatures. a) Form box properly cropped from image; b) form box not cropped from image; c) form box only partially cropped from image.

Appendix B

Smith dataset errors with different shapes

In this appendix, we include details for each recognition error in the “different shapes” class of the Smith dataset, which are used in the analysis in Section 4.2. We replicate Figure 4.6 below as Figure B.1. For each word pair in Figure B.1, we show the details for the closest (erroneous) match in the left side of the corresponding details box and the nearest correct match in the right side of the box (i.e., the training word that we would want to match in order to avoid a recognition error). For each pair of word images, we provide the medial axes, ground-truth labels, and word image numbers. We show the coarse-aligned positions of the medial axes and the final morph-aligned positions, warping each direction (aligning from one to the other and then vice versa). Finally, we show the word matching cost for each direction, as well as the total word matching cost. The layout is shown graphically in Figure B.2, followed by the details for each word-pair from Figure B.1 numbered from 1 to 21 in left-to-right, top-to-bottom order.

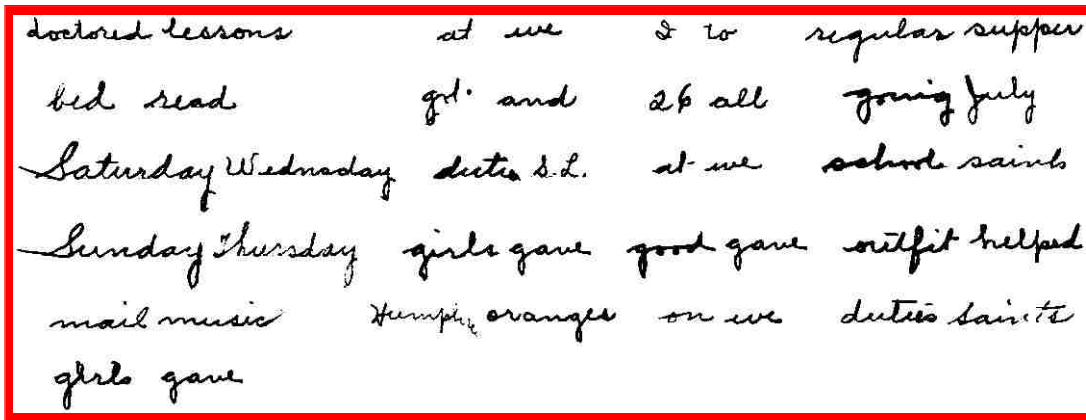
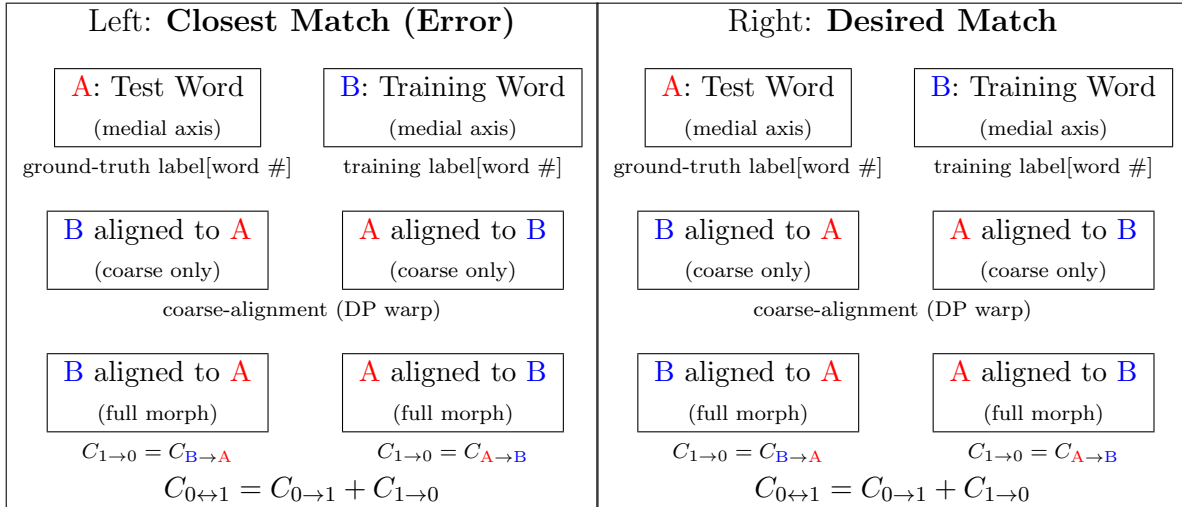
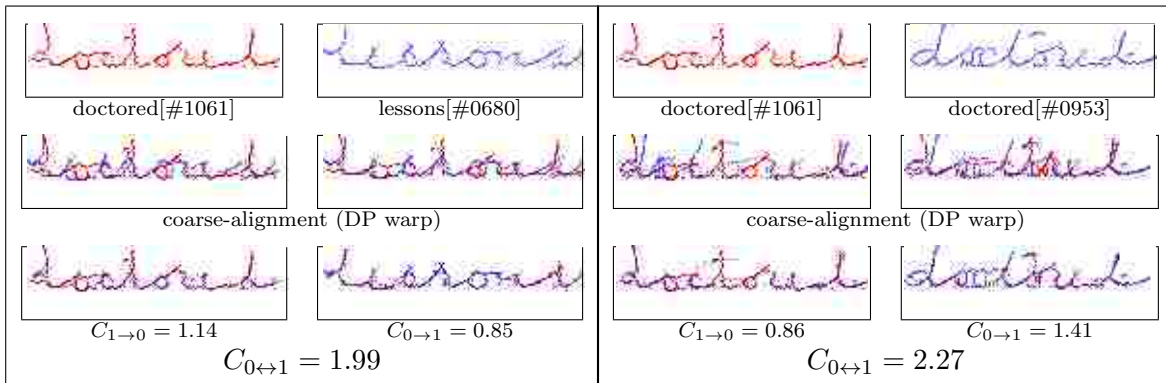


Figure B.1: Errors that have different shapes. (Smith dataset)

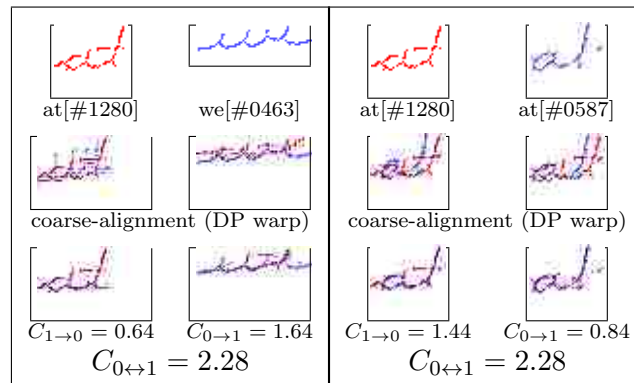


Word pair #: Observations about why the error occurs instead of the desired match.

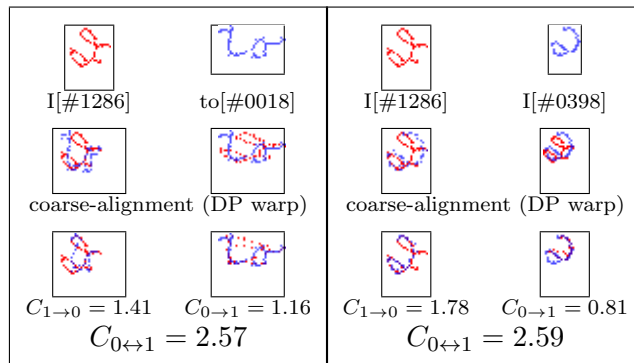
Figure B.2: Layout of data in this Appendix



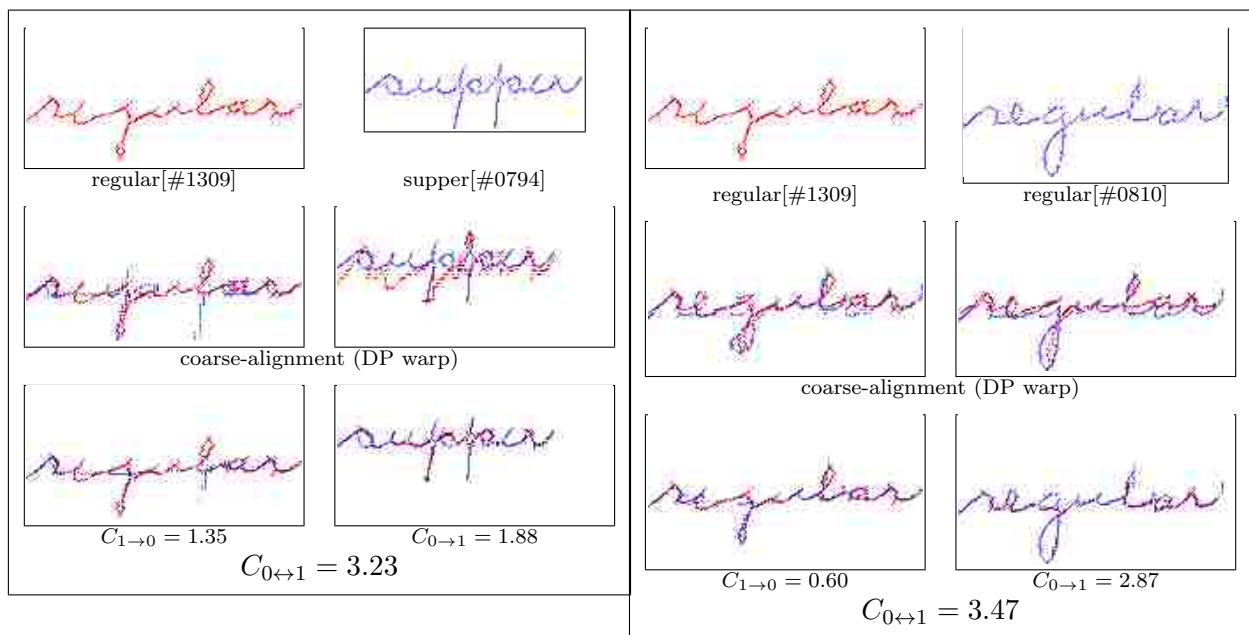
#1: Error in coarse-alignment results in poor morph for the desired match.



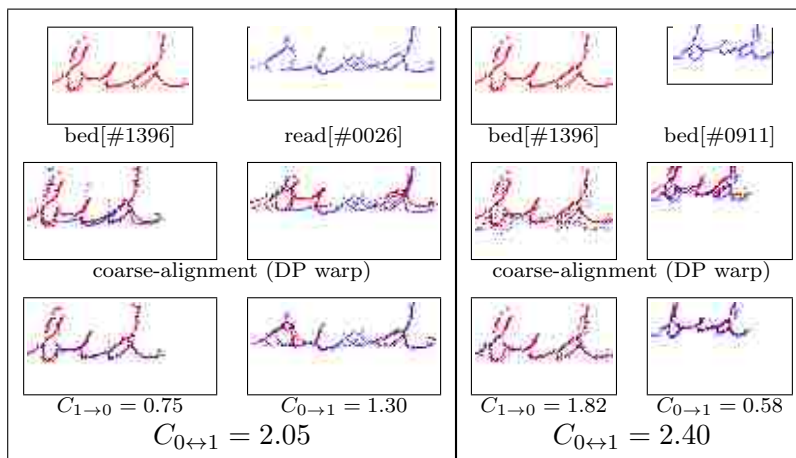
#2: Cost for the desired match is almost identical.



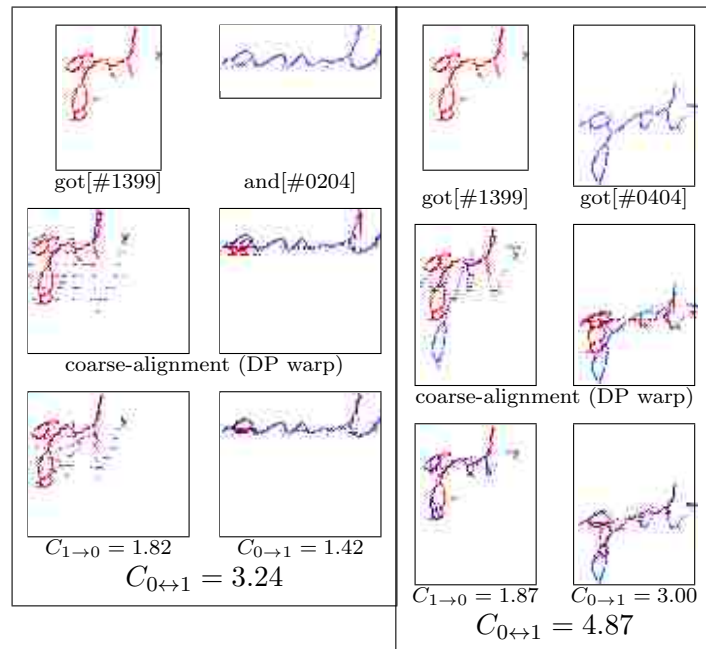
#3: Cost for the desired match is almost identical.



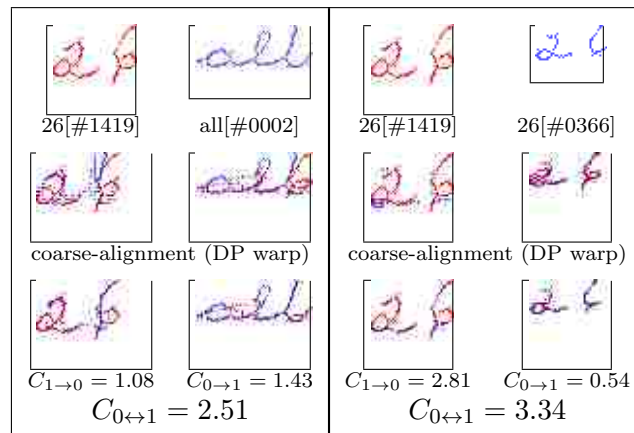
#4: Bad 'e', filled 'g', and collapsed 'l' of #1309 leave loops of #810 far way for $C_{0 \rightarrow 1}$.



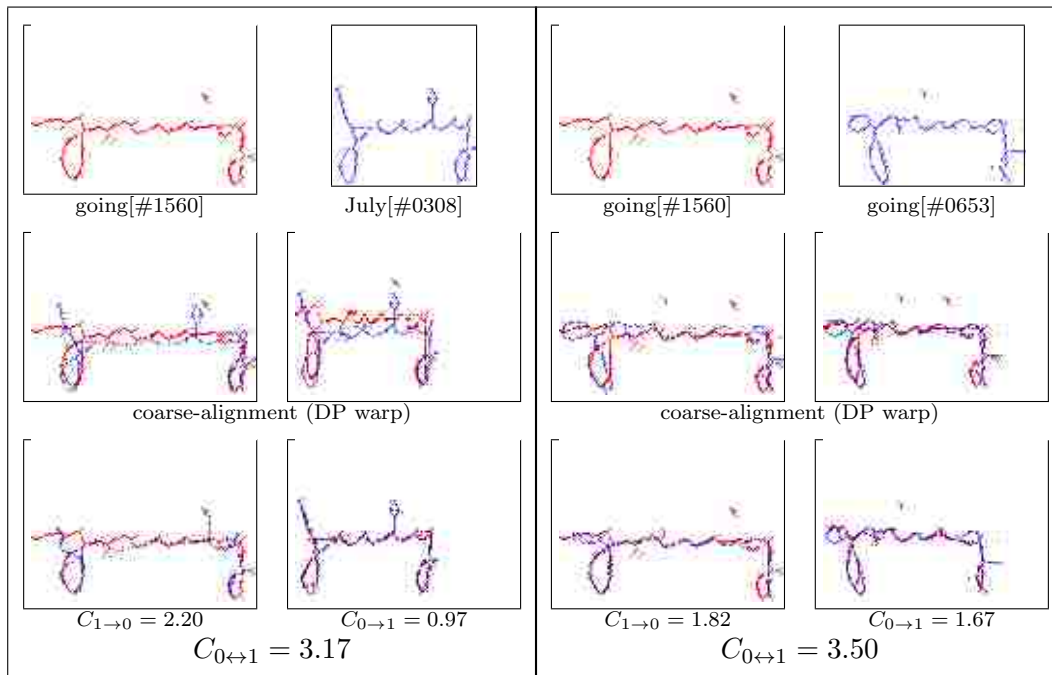
#5: Costs are close, but loop on red 'b' makes the difference in $C_{1 \rightarrow 0}$.



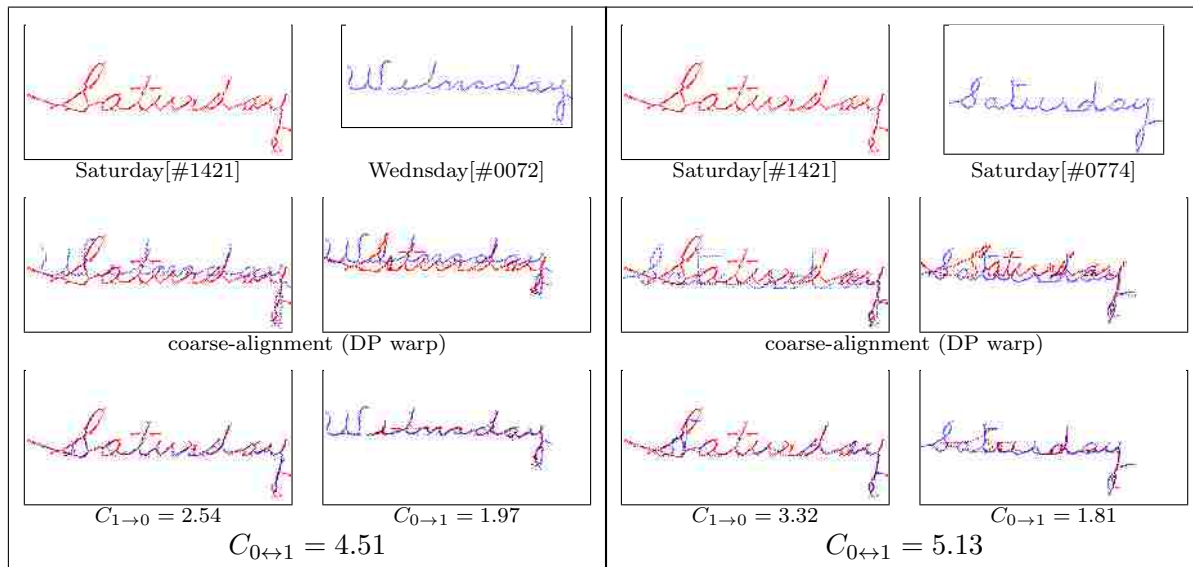
#6: “and” spreads into grid, “got” compresses (left); malformed letters and bad vertical DP.



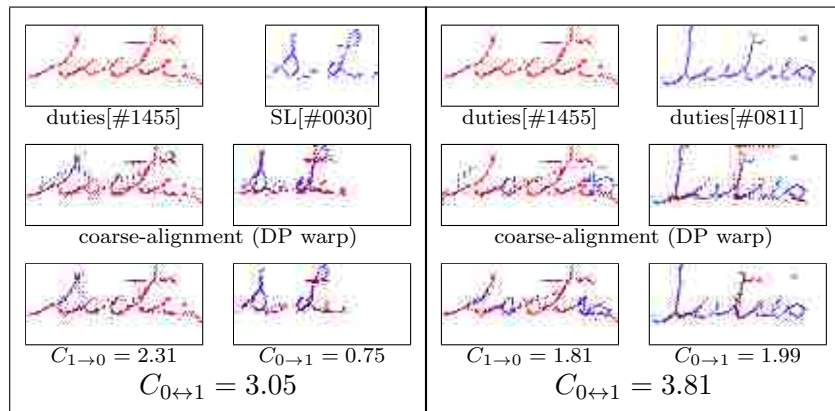
#7: Filled ‘6’ loop and collapsed ‘2’ loop leave strokes far away in $C_{1 \leftarrow 0}$.



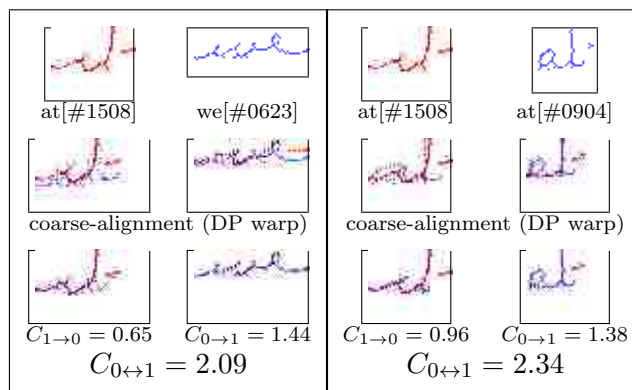
#8: Filled 'g' and 'o', collapsed g, unaligned tittles, final descender loop.



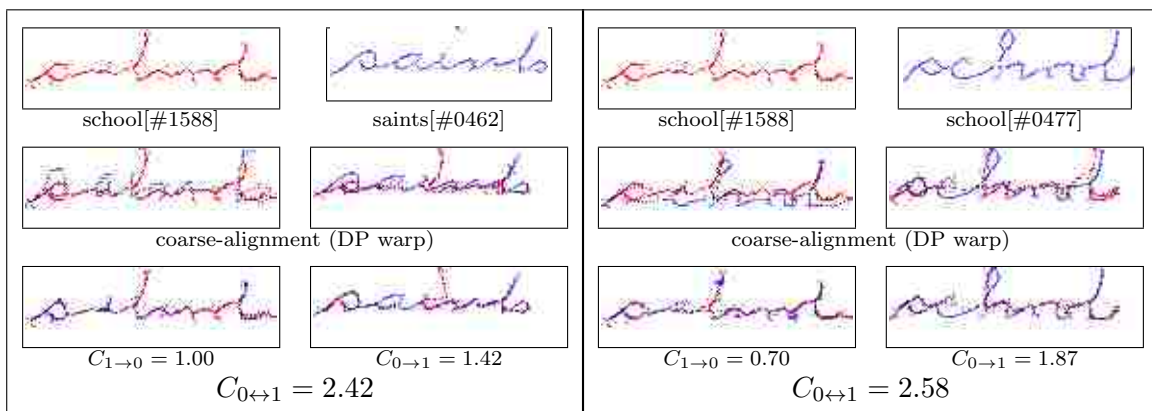
#9: Poor coarse-alignment of word beginning, partially-collapsed loops at end.



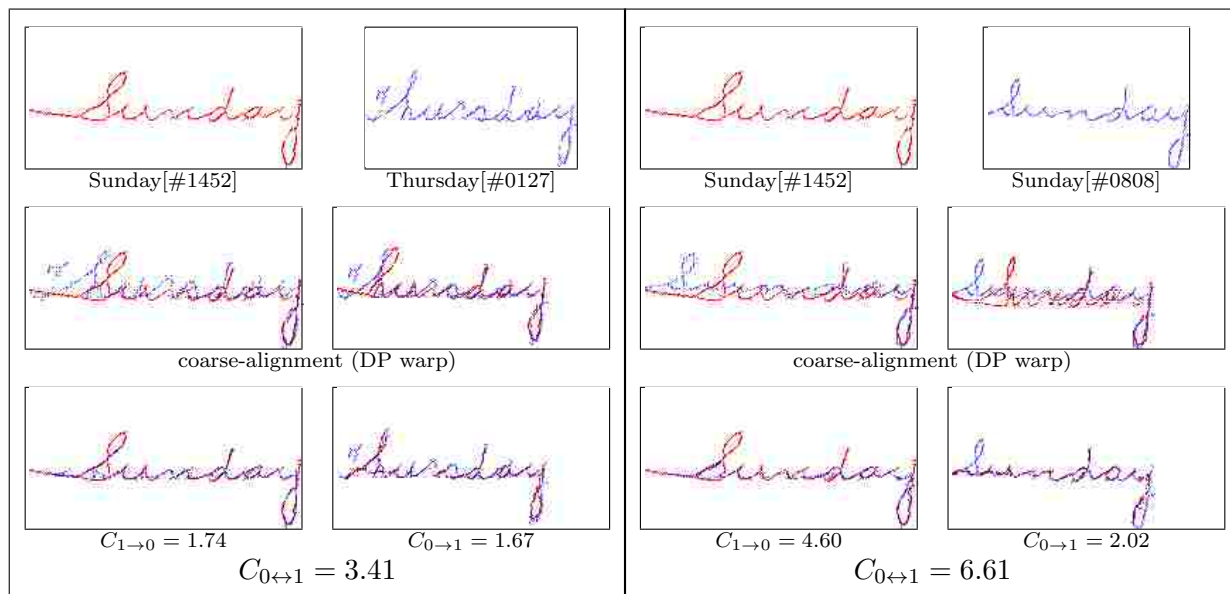
#10: Filled 'd', 'es' run-together/filled, tittle, bad slant mismatch, recovers from bad DP. Stretches/compresses well to S.L.



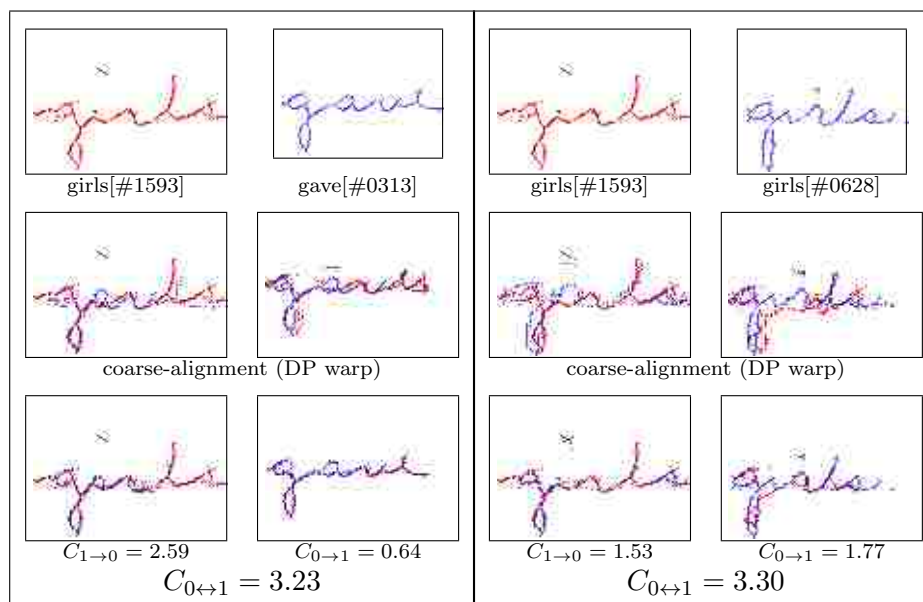
#11: Close costs, but filled loop on 'a' makes the difference.



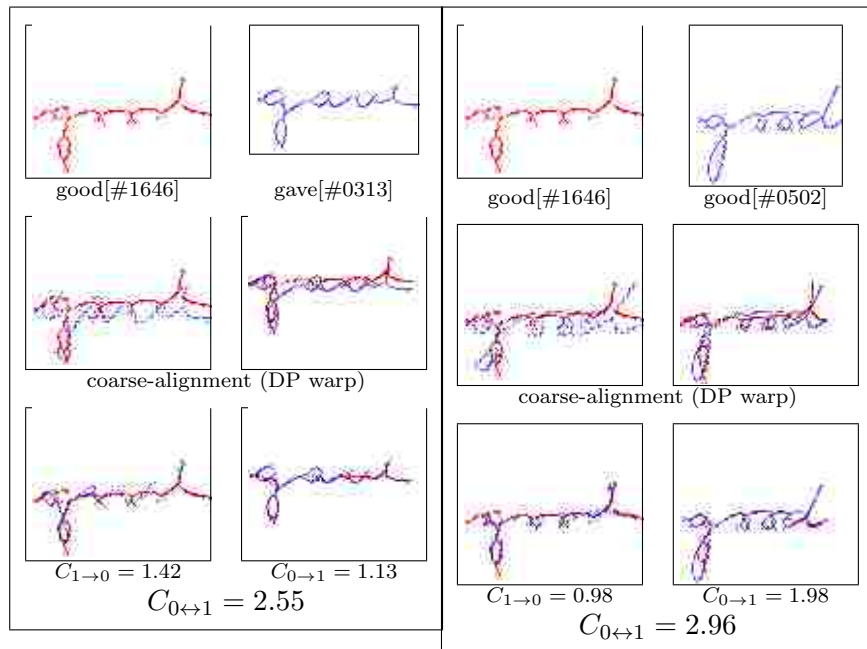
#12: Filled 'c' and two filled 'o's on red, one filled 'o' on blue.



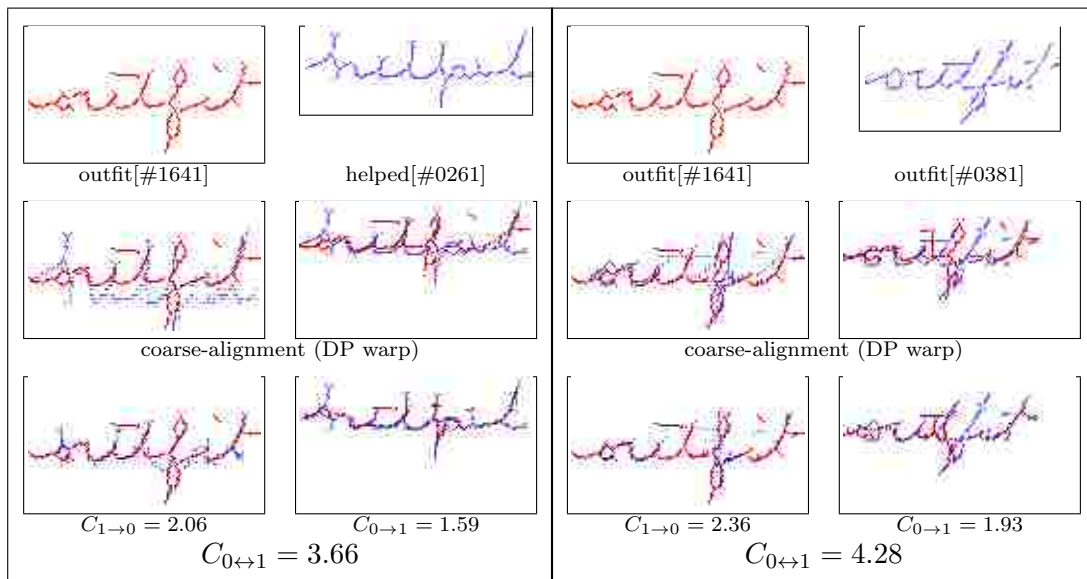
#13: Very poor DP at beginning of word, descender loop collapses due to bug (Section 4.8)



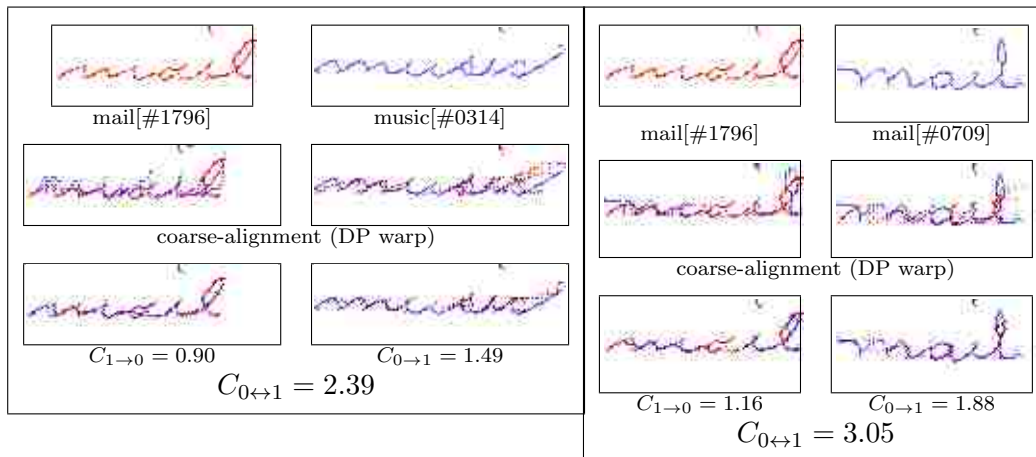
#14: DP fails for 'ir' both directions. 's' jumps at mesh refine due to bug (Section 4.8)



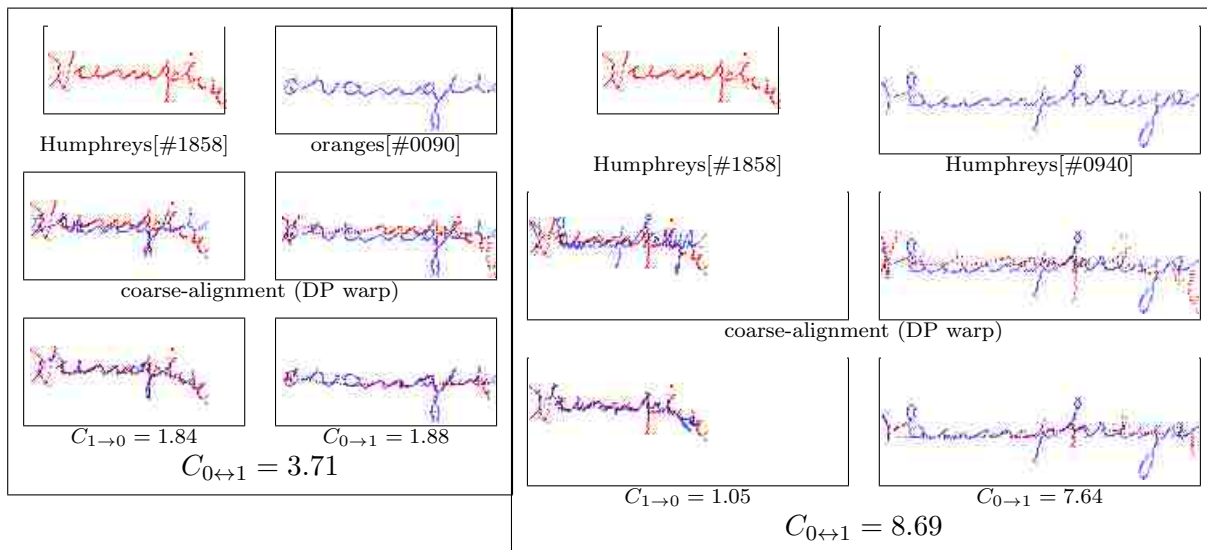
#15: Filled loops on 'o's and 'd'.



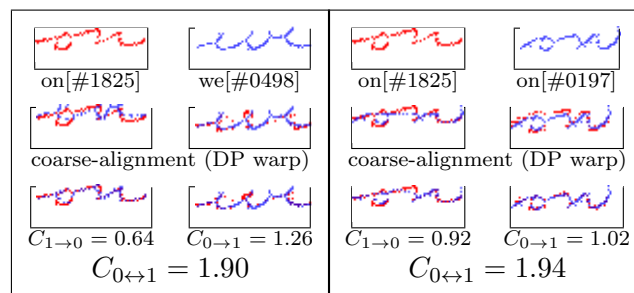
#16: 'f' and 't' morph to each other (slanted ascender on 'f' causes the jump).



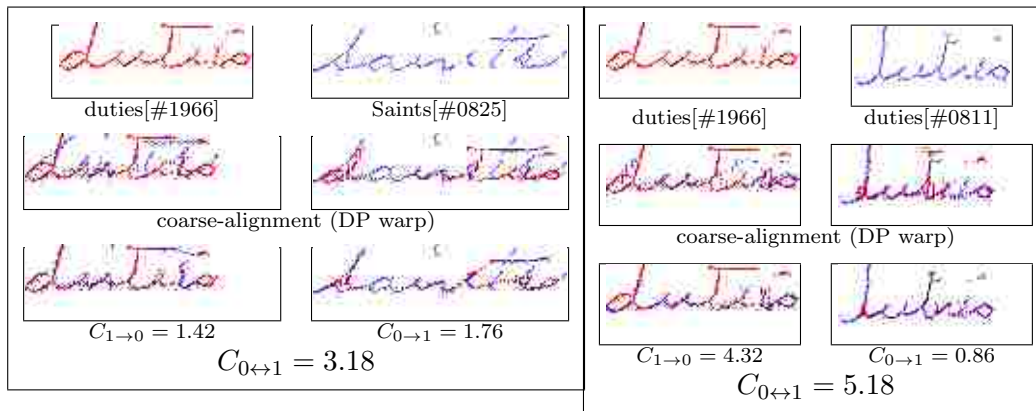
#17: DP too far off.



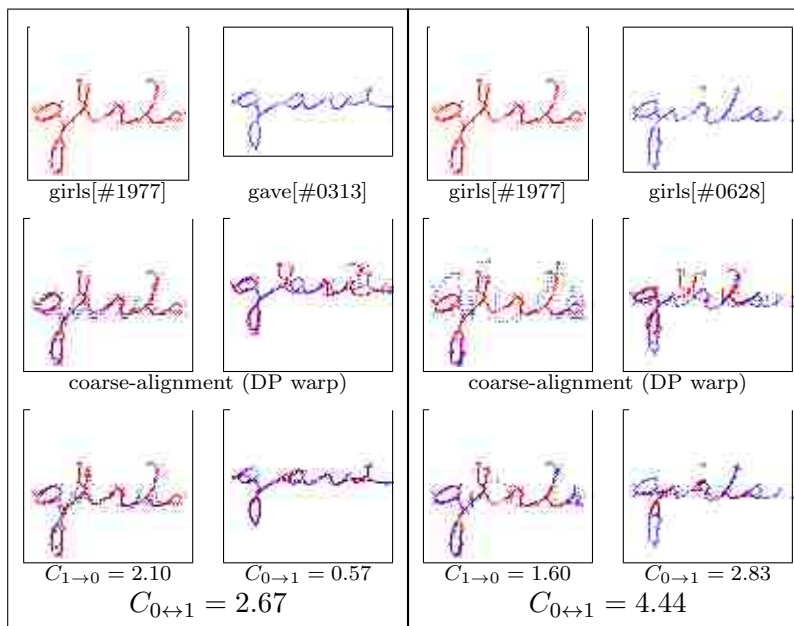
#18: Red word is malformed and very different from desired training example.



#19: Costs are almost equal.



#20: Poor DP.



#21: Poor DP, likely caused by an interfering descender from textline above.

Appendix C

Washington dataset errors with different shapes

In this appendix, we include details for each recognition error in the “different shapes” class of the Smith dataset, which are used in the analysis in Section 4.2. We replicate Figure 4.15 below as Figure C.1. For each word pair in Figure C.1, we show the details for the closest (erroneous) match in the left side of the corresponding details box and the nearest correct match in the right side of the box (i.e., the training word that we would want to match in order to avoid a recognition error). For each pair of word images, we provide the medial axes, ground-truth labels, and word image numbers. We show the coarse-aligned positions of the medial axes and the final morph-aligned positions, warping each direction (aligning from one to the other and then vice versa). Finally, we show the word matching cost for each direction, as well as the total word matching cost. The layout is shown graphically in Figure C.2, followed by the details for each word-pair from Figure C.1 numbered from 1 to 17 in left-to-right, top-to-bottom order.

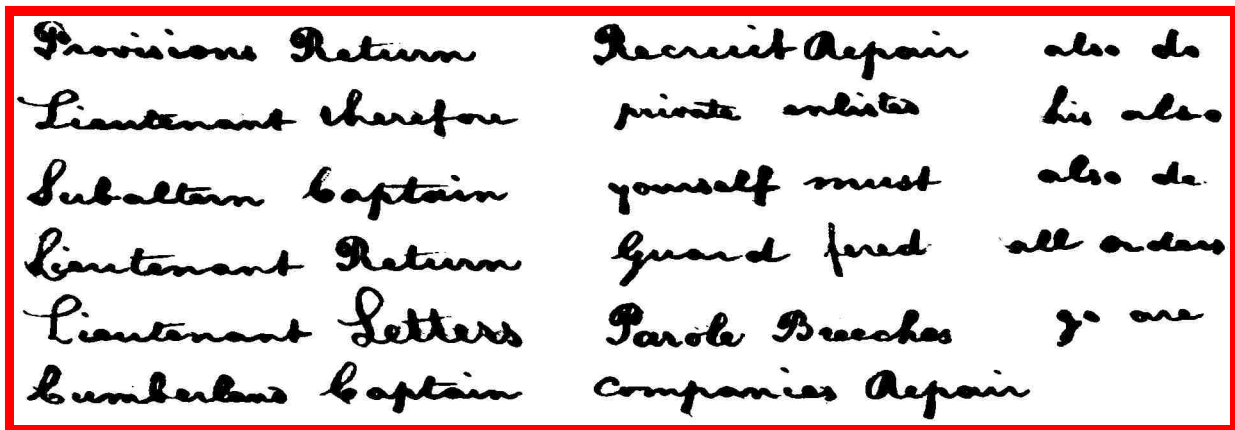


Figure C.1: Errors that have different shapes. (Washington dataset)

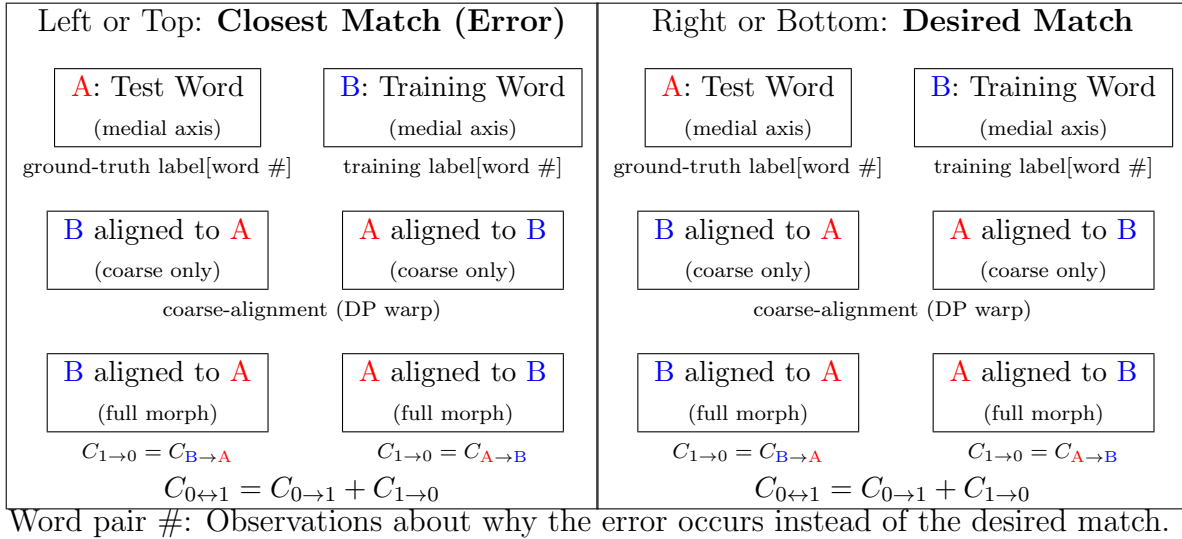
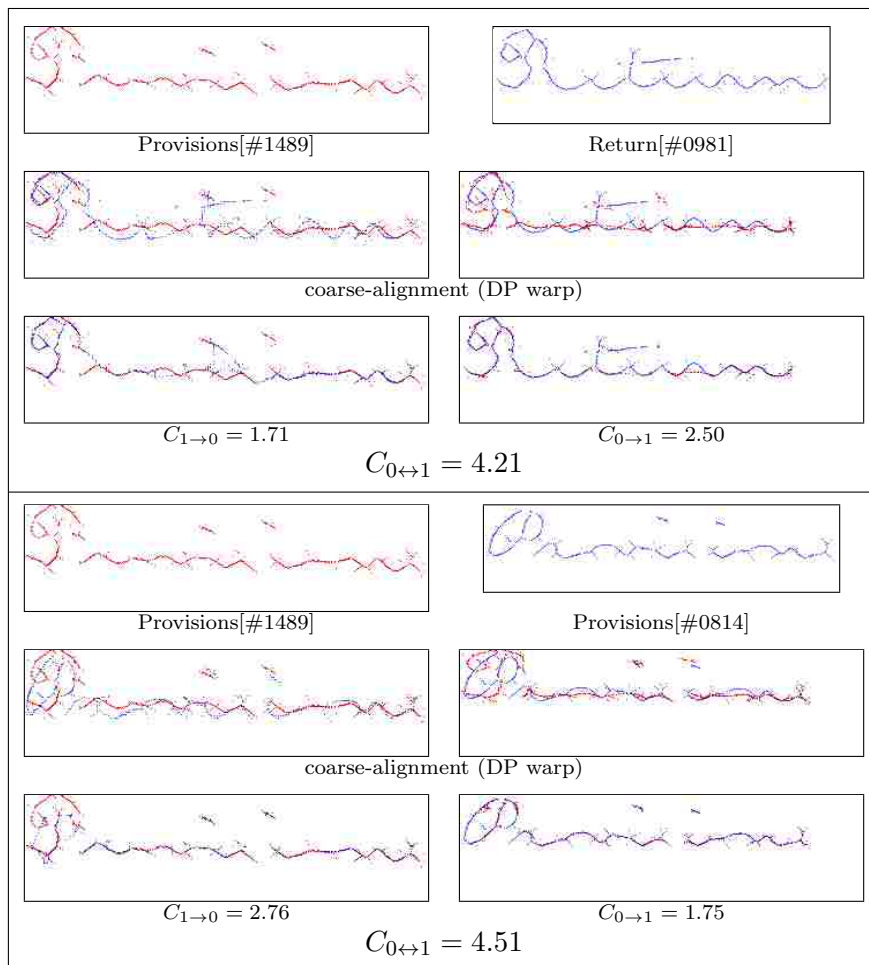
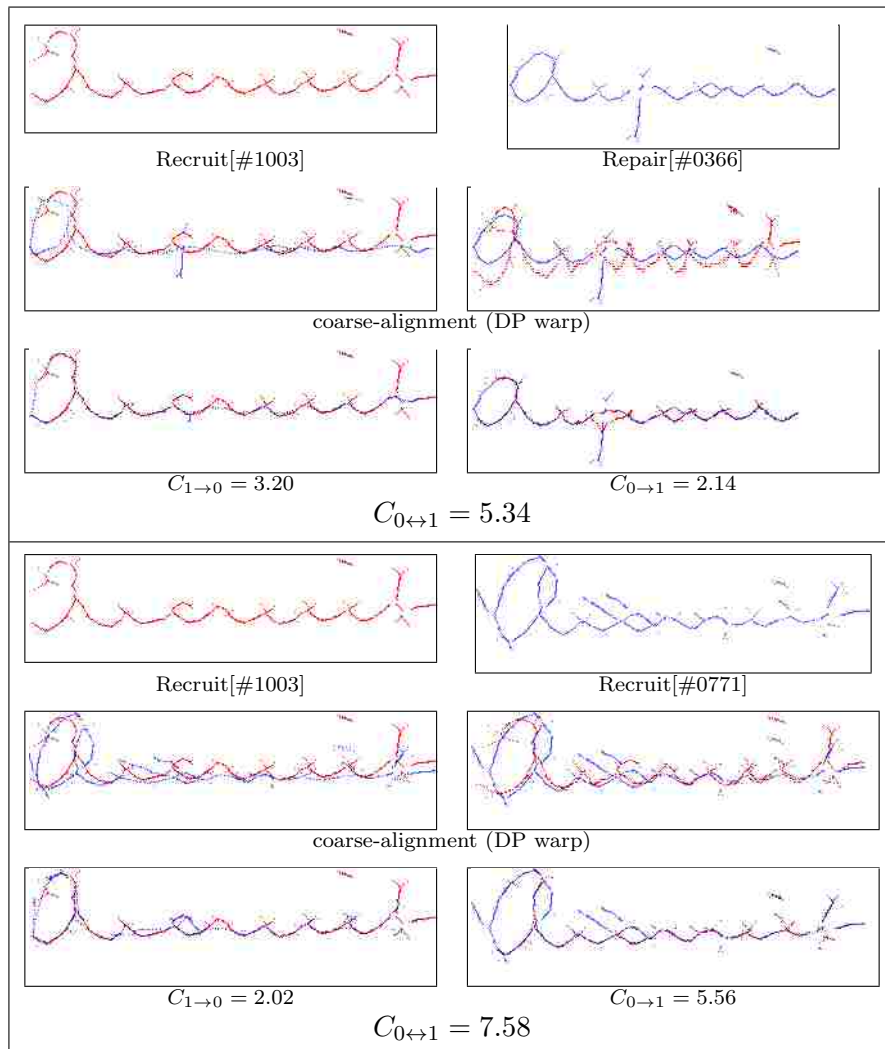


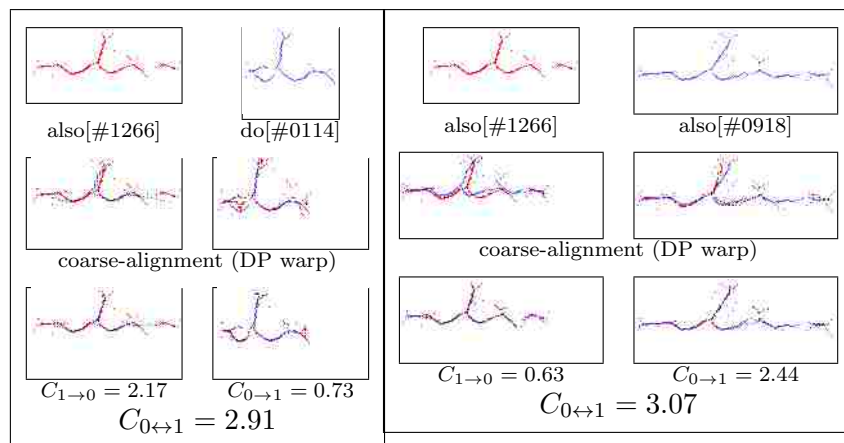
Figure C.2: Layout of data in this Appendix



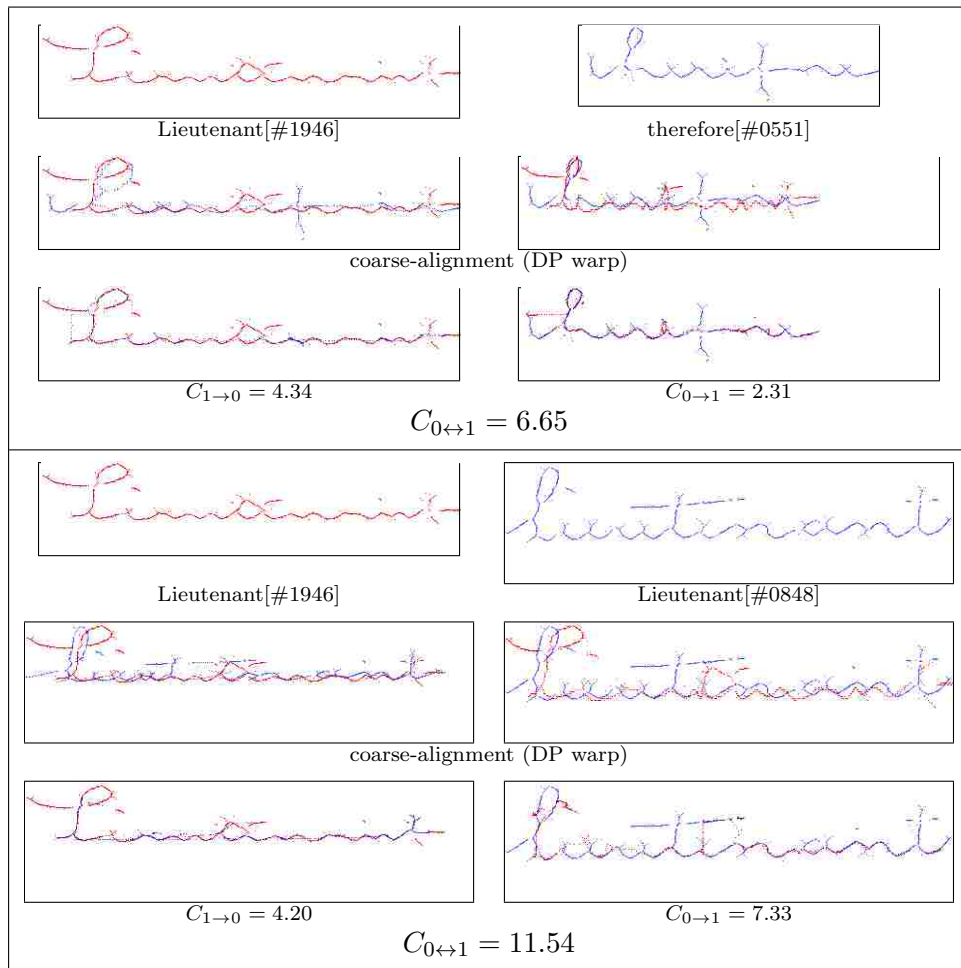
#1: "P" is similar to the "R", loops are filled, costs are close.



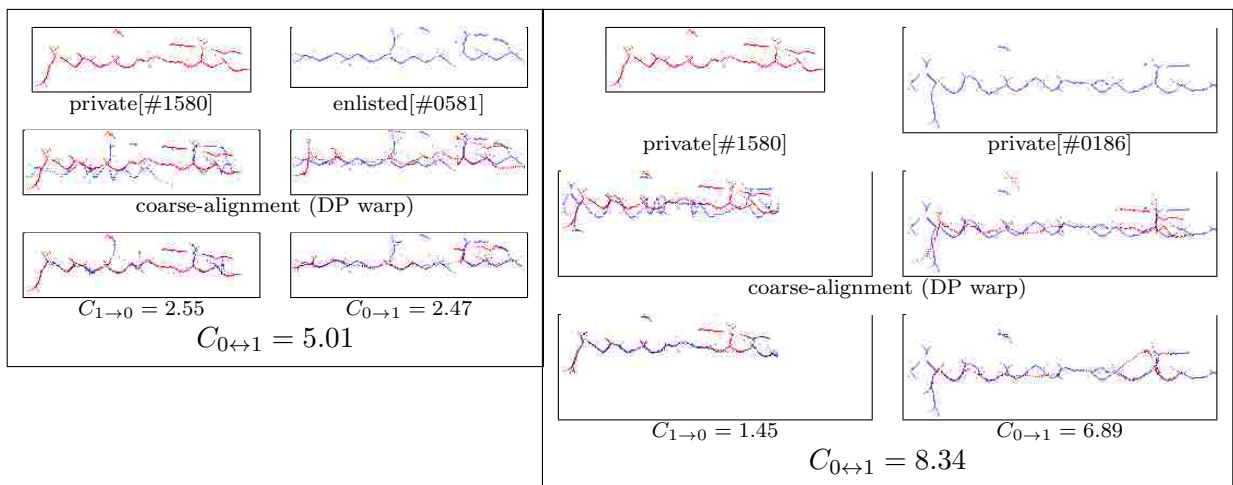
#2: "R"s shaped very differently, loop filled on "e", noise introduced by binarization.



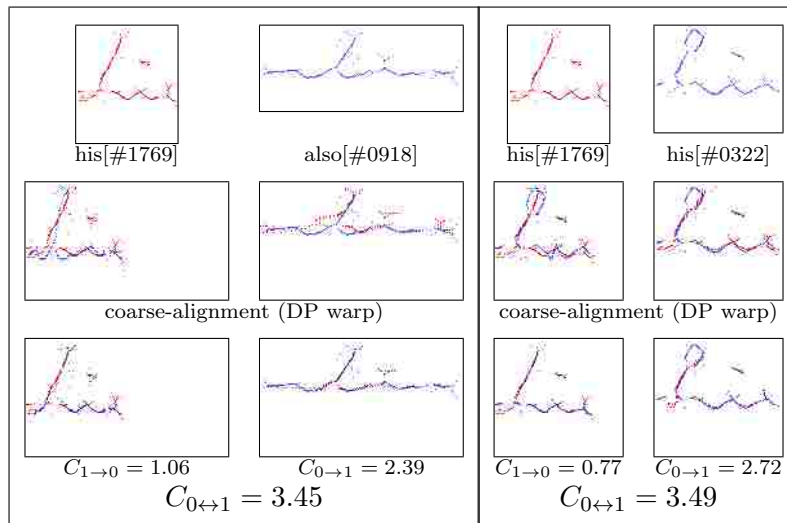
#3: Most loops are filled.



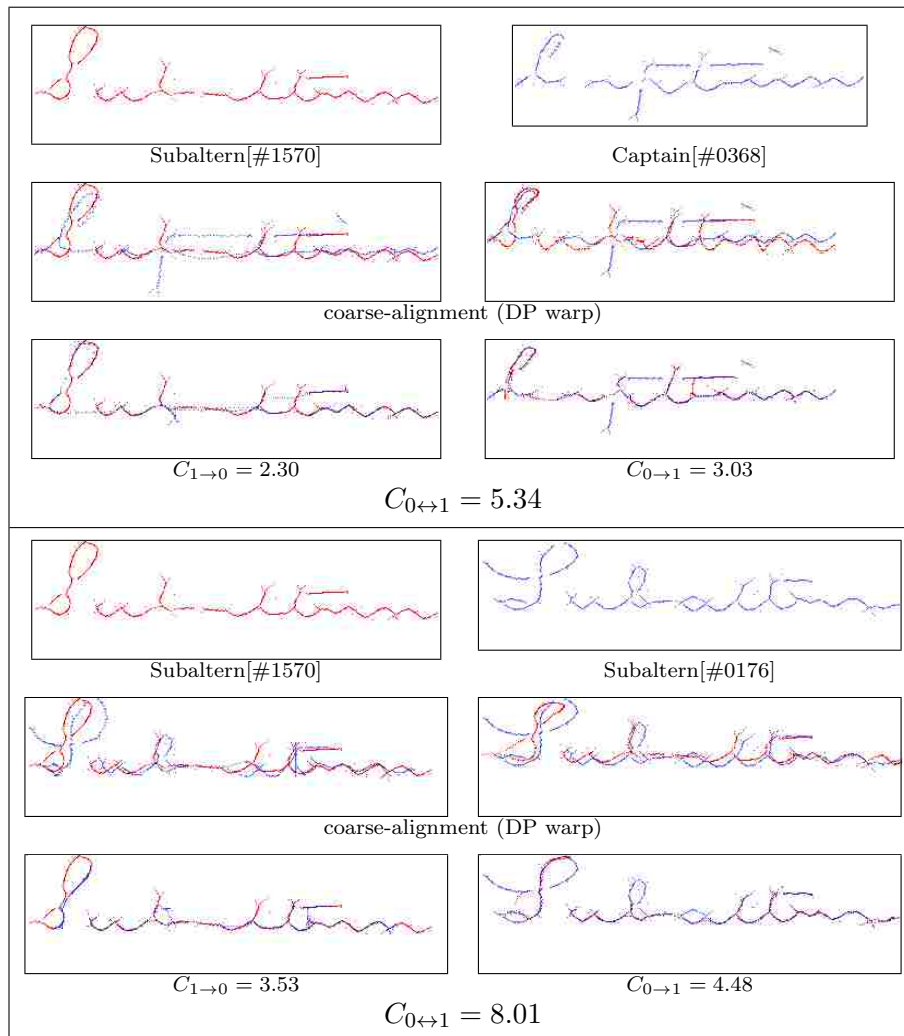
#4: Poor DP alignment, letters shaped differently so they don't morph well, filled loops.



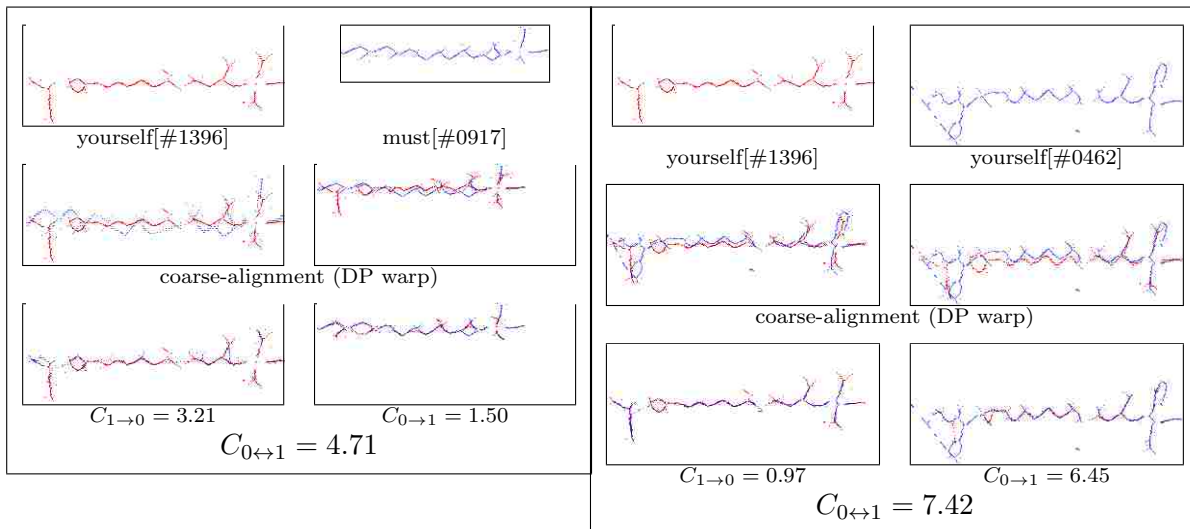
#5: Filled loops on #0581 look more similar than unfilled loops on #0186.



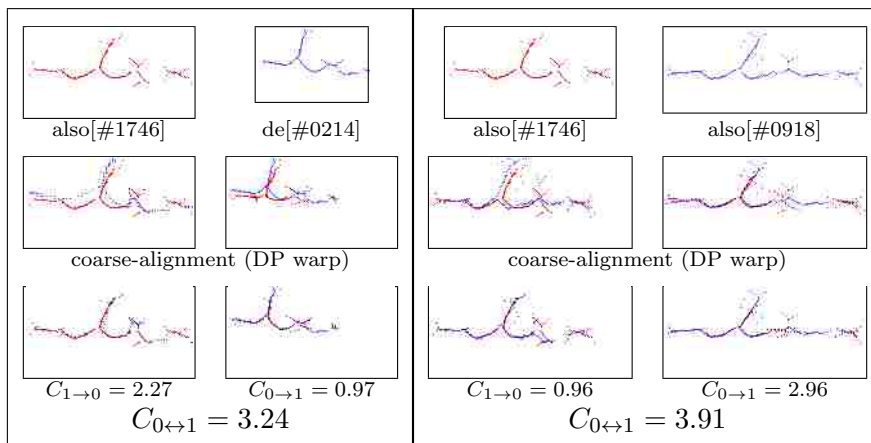
#6: Filled in loops. Costs almost identical.



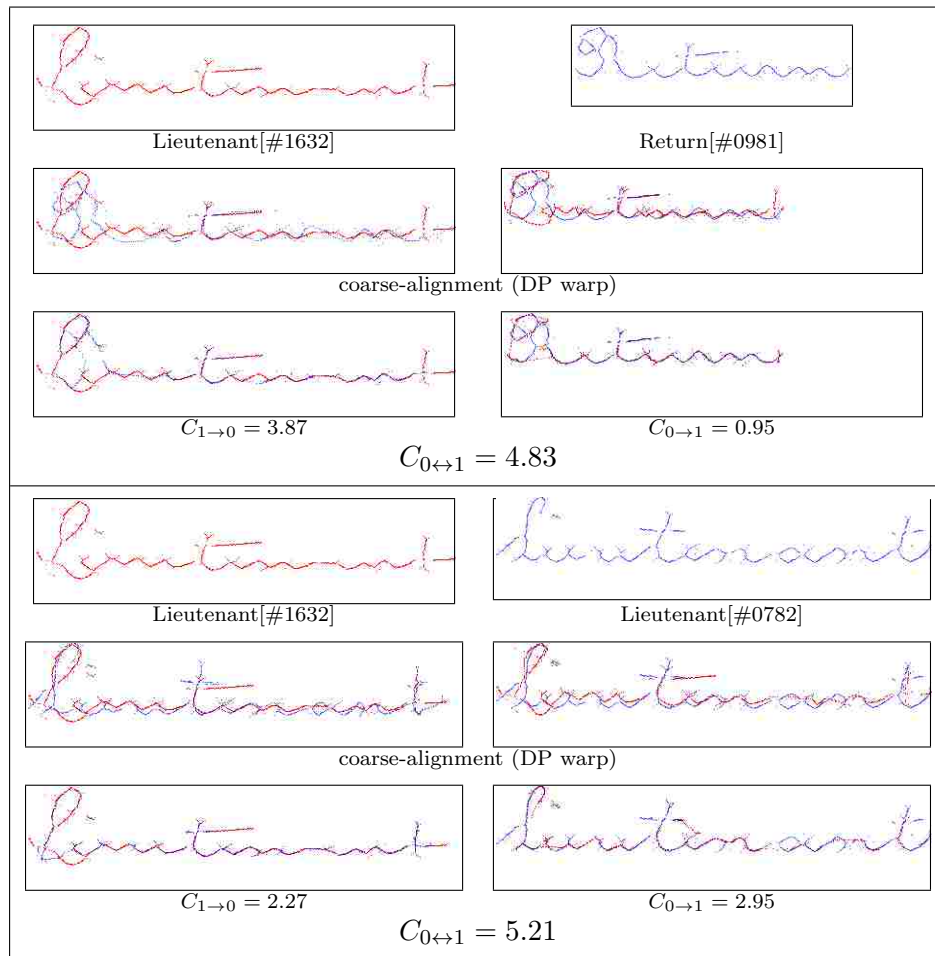
#7: Different loops filled in, "S" shaped differently, poor DP ("l" aligns with "a").



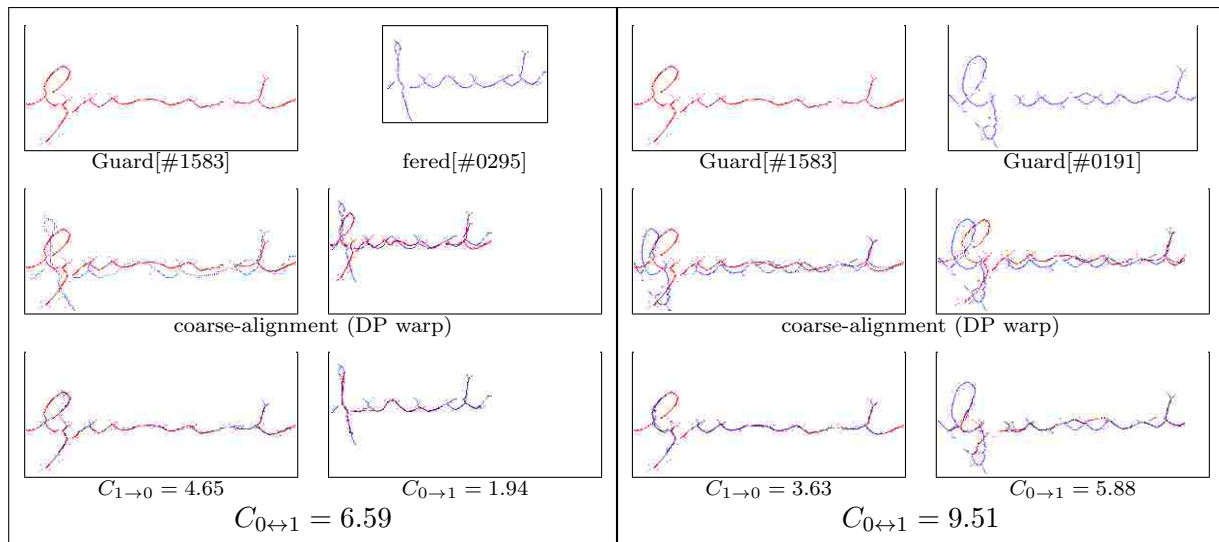
#8: The extra rule line (diagonal stroke) at the left of #0462 causes high cost.



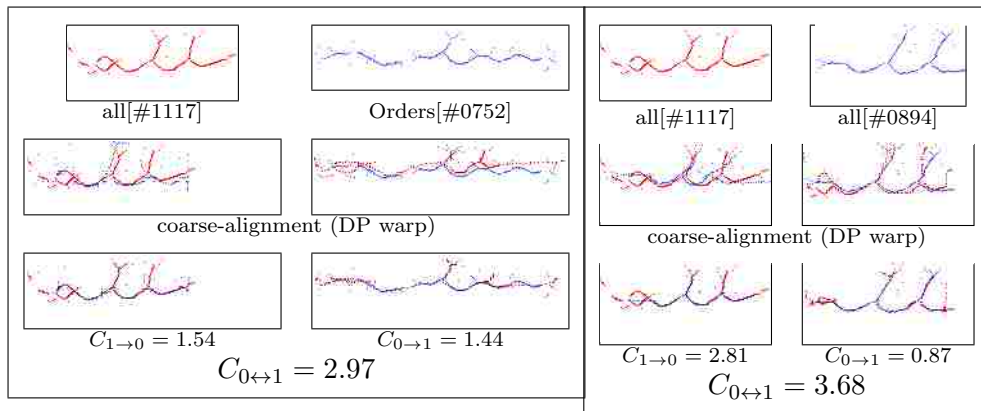
#9: All loops are filled, costs are relatively close.



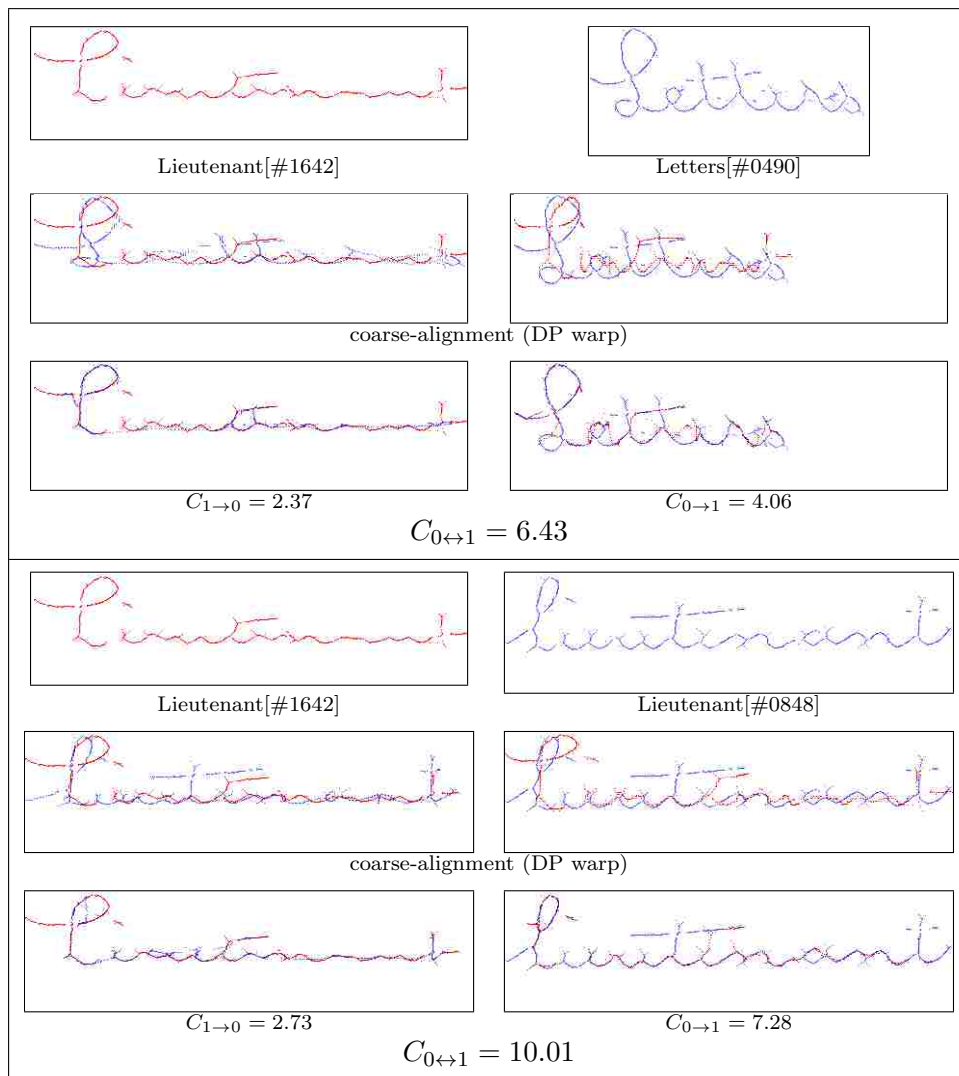
#10: Filled loops are the main problem.



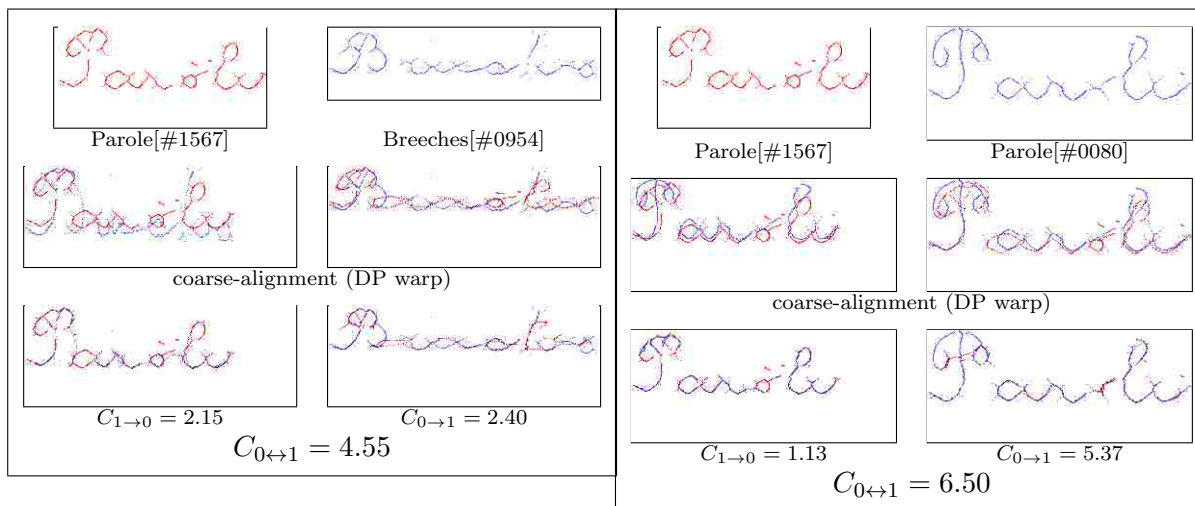
#11: Filled loops, DP alignment of "G" causes ascender to collapse, diagonal rule line.



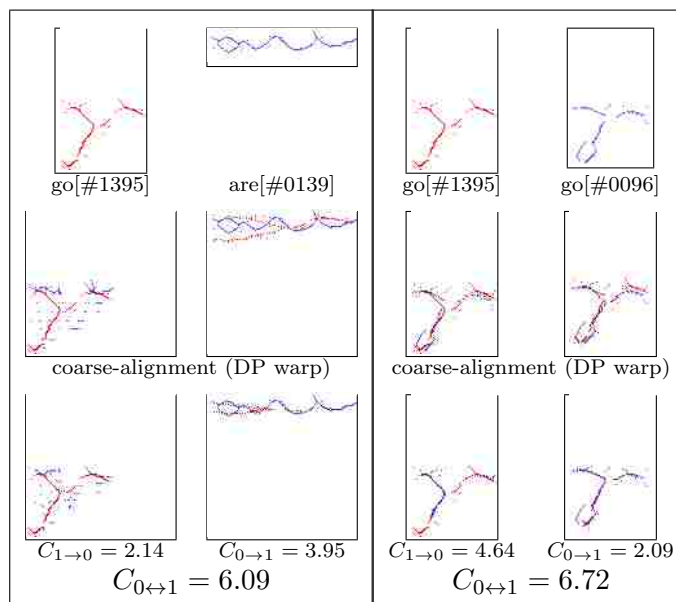
#12: Slants of leading/trailing ligatures don't match well, loop of "a" filled.



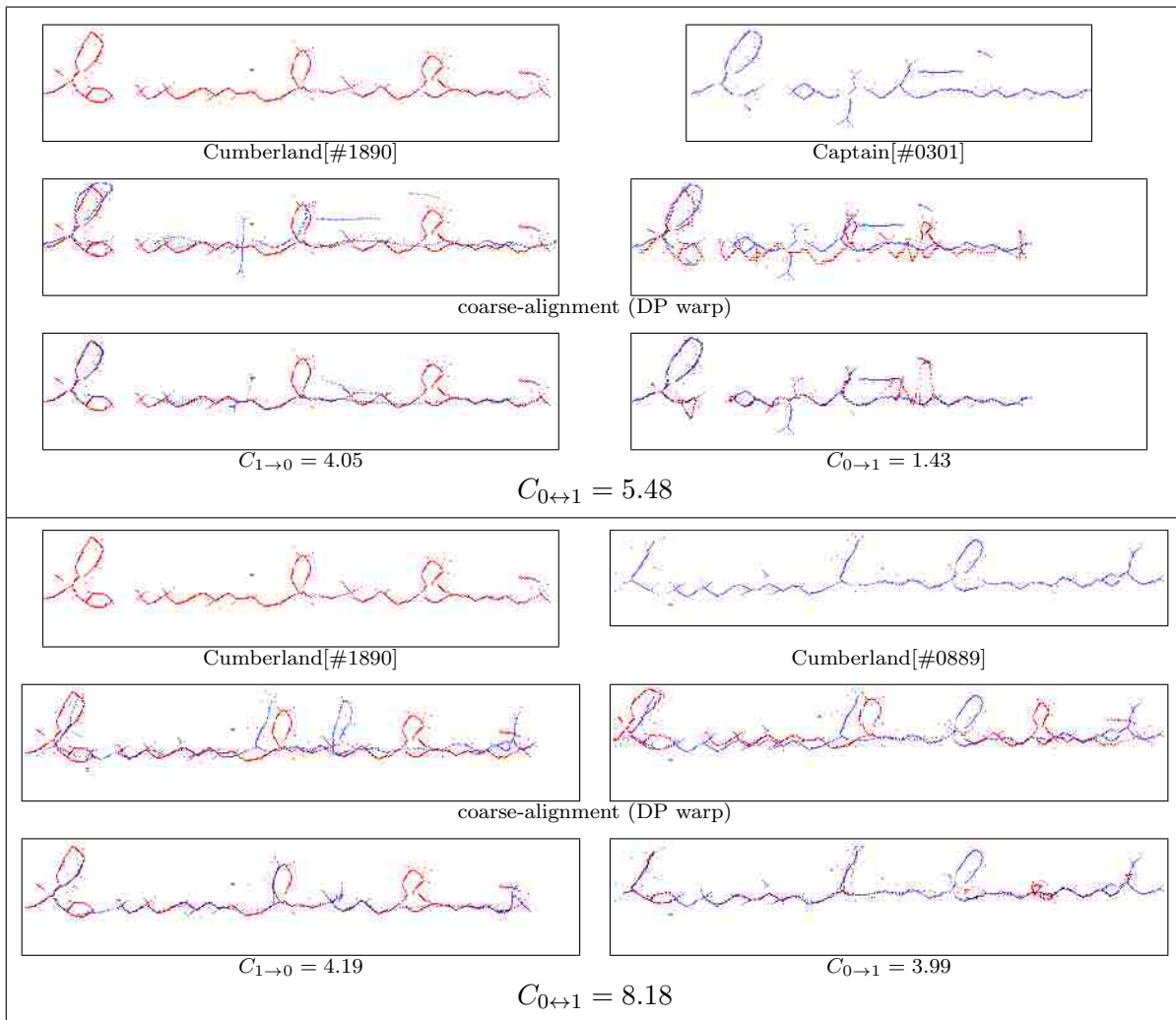
#13: Horizontal DP poor for first "t", vertical DP poor for last "t", filled loops.



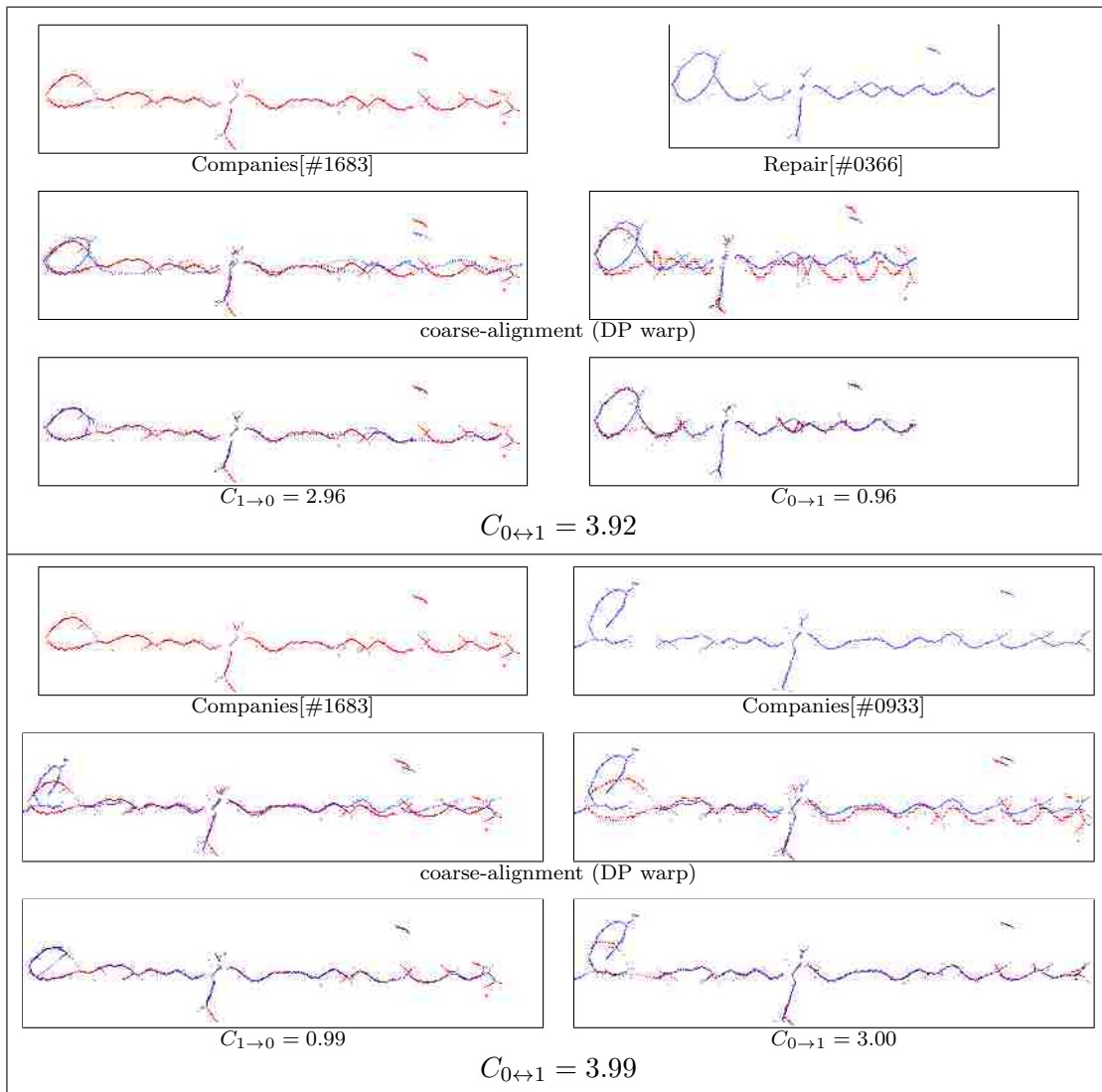
#14: Main problem is top of “P” does not morph well, also loop of “o” filled.



#15: #0139 spreads across #1395 like a grid (DP), “g” descenders don’t align well.



#16: Shapes of first "C" differ, some loops filled, poor DP, misformed "d".



#17: Costs almost identical, loops filled.

Appendix D

Explanation of Code and Documentation

The `documentproject` directory contains both code and documentation related to this dissertation. Code consists of two types: first, a C++ library (the `documentproject` library), and second, application code. Directory organization is as follows:

```
/documentproject - project container directory
/apps_src       - application source code
/bin            - binary executables (compile to here from apps_src)
/doc/index.html - documentation for the library in HTML format
/lib           - library compiles to here
/obj           - object files generated during compilation
/src           - source code for the library
```

The **documentproject library** includes many general-purpose object classes for tasks such as image input/output and general document image processing (thresholding, filtering, etc.). It also includes code more specific to my research and this dissertation, such as the morphing / word warping algorithm code. The two object classes of primary interest for this dissertation are the `DImage` class and the `DMorphInk` class. The library includes documentation in HTML format. The documentation was generated automatically (incorporating special comments within the code) by the open-source `Doxygen` program.

Application code consists of several applications and utility programs that can be compiled individually, most of which rely on the `documentproject` library to compile and run. Each program directory includes a `README.txt` file that describes the program, its purpose, and its functionality. Additionally, running any program without using command-line parameters will cause the program to generate a “usage” message and then exit. The usage message lists any command-line parameters that are expected from the user.

References

- [1] American Bankers Association. 2011 deposit account fraud survey. Summary at URL <http://www.aba.com/Products/Surveys/Pages/2011DepositAccount.aspx> visited 22 Jan. 2013.
- [2] Roman Bertolami and Horst Bunke. Hidden Markov model-based ensemble methods for offline handwritten text line recognition. *Pattern Recognition (PR)*, 41(11):3452–3460, 2008.
- [3] Roman Bertolami, Beat Halter, and Horst Bunke. Combination of multiple handwritten text line recognition systems with a recursive approach. In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 61–65, La Baule, France, Oct. 2006.
- [4] Board of Governors of the Federal Reserve System. Report to the Congress on funds availability schedules and check fraud at depository institutions. Page ii, Oct. 1996, URL <http://www.federalreserve.gov/boarddocs/rptcongress/chkfraud.pdf> visited 22 Jan. 2013.
- [5] Horst Bunke. Recognition of cursive roman handwriting- past, present and future. In *7th International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 448–459, Edinburgh, Scotland, Aug. 2003.
- [6] Francesco Camastra. A SVM-based cursive character recognizer. *Pattern Recognition (PR)*, 40:3721–3727, 2007.
- [7] Myriam Côté, Eric Lecolinet, Mohamed Cheriet, and Ching Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts—the PERCEPTO system. *International Journal on Document Analysis and Recognition (IJ DAR)*, 1:1–15, 1998.
- [8] Peisheng Gao and Thomas W. Sederberg. A work minimization approach to image morphing. *The Visual Computer*, 14:390–400, 1998.

- [9] Basilios Gatos, Ioannis Pratikakis, and Stavros J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition (PR)*, 39:317–327, 2006.
- [10] Basilios Gatos, Ioannis Pratikakis, and Stavros J. Perantonis. Efficient binarization of historical and degraded document images. In *8th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 447–454, Nara, Japan, Sep. 2008.
- [11] Simon Günter and Horst Bunke. Ensembles of classifiers for handwritten word recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 5:224–232, 2003.
- [12] Luke A. D. Hutchison and William A. Barrett. Fourier-Mellin registration of line-delineated tabular document images. *International Journal on Document Analysis and Recognition (IJDAR)*, 8:87–110, Jun. 2006.
- [13] Ergina Kavallieratou, Nikos Fakotakis, and George Kokkinakis. An unconstrained handwriting recognition system. *International Journal on Document Analysis and Recognition (IJDAR)*, 4(1):226–242, 2002.
- [14] Douglas J. Kennard and William A. Barrett. Separating lines of text in free-form handwritten historical documents. In *International Workshop on Document Image Analysis for Libraries (DIAL)*, pages 12–23, Lyon, France, Apr. 2006.
- [15] Douglas J. Kennard, William A. Barrett, and Thomas W. Sederberg. Word warping for offline handwriting recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1349–1353, Beijing, China, Sep. 2011.
- [16] Douglas J. Kennard, William A. Barrett, and Thomas W. Sederberg. Offline signature verification and forgery detection using a 2-d geometric warping approach. In *International Conference on Pattern Recognition (ICPR)*, Tsukuba, Japan, Nov. 2012.
- [17] Gyeonghwan Kim and Venu Govindaraju. A lexico driven approach to handwritten word recognition for real-time applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(4):366–379, Apr. 1997.
- [18] Gyeonghwan Kim, Venu Govindaraju, and Sargur N. Srihari. An architecture for handwritten text recognition systems. *International Journal on Document Analysis and Recognition (IJDAR)*, 2(1):37–44, 1999.
- [19] In-Kwon Kim, Dong-Wook Jung, and Rae-Hong Park. Document image binarization based on topographic analysis using a water flow model. *Pattern Recognition (PR)*, 35:265–277, 2002.

- [20] Alessandro L. Koerich, Robert Sabourin, and Ching Y. Suen. Large vocabulary off-line handwriting recognition: A survey. *Pattern Analysis and Applications (PAA)*, 6:97–121, 2003.
- [21] Victor Lavrenko, Toni M. Rath, and R. Manmatha. Holistic word recognition for handwritten historical documents. In *Proc. of the International Workshop on Document Image Analysis for Libraries (DIAL)*, pages 278–287, Palo Alto, CA, Jan. 2004.
- [22] Nilo Lindgren. Machine recognition of human language, part III—cursive script recognition. *IEEE Spectrum*, 2:104–116, May 1965.
- [23] Marcus Liwicki, Muhammad Imran Malik, C. Elisa van den Heuvel, Xiaohong Chen, Charles Berger, Reinoud Stoel, Michael Blumenstein, and Bryan Found. Signature verification competition for online and offline skilled forgeries (SigComp2011). In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1480–1484, Beijing, China, Sep. 2011.
- [24] Sriganesh Madhvanath and Venu Govindaraju. The role of holistic paradigms in handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(2):149–164, Feb. 2001.
- [25] Uma Mahadevan and Ramesh C. Nagabushnam. Gap metrics for word separation in handwritten lines. In *3rd International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 124–127, Montreal, Canada, Aug. 1995.
- [26] R. Manmatha and Jamie L. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(8):1212–1225, Aug. 2005.
- [27] Urs-Viktor Marti and Horst Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 15(1):65–90, 2001.
- [28] Urs-Viktor Marti and Horst Bunke. The IAM-database: An English sentence database of offline handwriting recognition. *International Journal on Document Analysis and Recognition (IJDAR)*, 5(1):39–46, Jul. 2002.
- [29] Ioannis Pavlidis, Rahul Singh, and Nikolaos P. Papanikolopoulos. On-line handwriting recognition using physics-based shape metamorphosis. *Pattern Recognition (PR)*, 31(11):1589–1600, 1998.

- [30] Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(1):63–84, Jan. 2000.
- [31] Brigitte Plessis, Anne Sicsu, Laurent Heutte, Eric Menu, Eric Lecolinet, Olivier Debon, and Jean-Vincent Moreau. A multi-classifier combination strategy for the recognition of handwritten cursive words. In *2nd International Conference on Document Analysis and Recognition (ICDAR)*, pages 642–645, Tsukuba, Japan, Oct. 1993.
- [32] Yu Qiao, Mikihiro Nishiara, and Makoto Yasuhara. A framework toward restoration of writing order from single-stroked handwriting image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(11):1724–1737, Nov. 2006.
- [33] Yu Qiao and Makoto Yasuhara. Recover writing trajectory from multiple stroked image using bidirectional dynamic search. In *18th International Conference on Pattern Recognition (ICPR)*, pages 970–973, Hong Kong, Aug. 2006.
- [34] Toni M. Rath and R. Manmatha. Features for word spotting in historical manuscripts. In *7th International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 218–222, Edinburgh, Scotland, Aug. 2003.
- [35] Toni M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 521–527, Madison, Wisconsin, Jun. 2003.
- [36] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM (JACM)*, 13(4):471–494, 1966.
- [37] Thomas W. Sederberg and Eugene Greenwood. A physically based approach to 2-D shape blending. *ACM SIGGRAPH Computer Graphics*, 26(2):25–34, Jul. 1992.
- [38] Andrew W Senior and Anthony J. Robinson. An off-line handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 20(3):309–321, Mar. 1998.
- [39] United States Postal Service. Postal facts 2012. URL <http://about.usps.com/future-postal-service/postalfacts-2012.pdf> visited 20 Dec. 2012.
- [40] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging (JEI)*, 13(1):146–165, Jan. 2004.

- [41] Rahul Singh and Nikolaos P. Papanikolopoulos. Planar shape recognition by shape morphing. *Pattern Recognition (PR)*, 33:1683–1699, 2000.
- [42] Sargur N. Srihari. Computer processing of handwriting in documents. Michigan State University, Computer Science Department Distinguished Lecture Series, Presentation slides at URL <http://www.cedar.buffalo.edu/~srihari/talks/MSU.pdf> visited 20 Dec. 2012.
- [43] Sargur N. Srihari and Edward J. Kuebert. Integration of hand-written address interpretation technology into the united states postal service remote computer reader system. In *4th International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 892–896, Ulm, Germany, Aug. 1997.
- [44] Tal Steinherz, Ehud Rivlin, and Nathan Intrator. Offline cursive script word recognition — a survey. *International Journal on Document Analysis and Recognition (IJ DAR)*, 2:90–110, 1999.
- [45] Claudia Swendseid. Payment fraud trends & prevention strategies, Jul. 2012. *National Association of Federal Credit Unions 45th Annual Conference & Exhibition*, URL <http://www.nafcu.org/WorkArea/DownloadAsset.aspx?id=28175> visited 22 Jan. 2013.
- [46] Insup Taylor and M. Martin Taylor. *The Psychology of Reading*. Academic Press, 1983.
- [47] Trails of hope: Overland diaries and letters, 1846–1869. Harold B. Lee Library, Brigham Young University. Online collection available at <http://overlandtrails.lib.byu.edu>.
- [48] Tamás Varga and Horst Bunke. Tree structure for word extraction from handwritten text lines. In *8th International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 352–356, Seoul, Korea, Aug.–Sep. 2005.
- [49] Alessandro Vinciarelli. A survey of off-line cursive word recognition. *Pattern Recognition (PR)*, 35:1433–1446, 2002.
- [50] Alessandro Vinciarelli, Samy Bengio, and Horst Bunke. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(6):709–720, Jun. 2004.
- [51] Chen Yan and Graham Leedham. Decompose-threshold approach to handwriting extraction in degraded historical document images. In *9th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 239–244, Kokubunji, Tokyo, Japan, Oct. 2004.

- [52] Yuanping Zhu, Chunheng Wang, and Ruwei Dai. Document image binarization based on stroke enhancement. In *18th International Conference on Pattern Recognition (ICPR)*, pages 955–958, Hong Kong, Aug. 2006.
- [53] Matthias Zimmermann, Jean-Cédric Chappelier, and Horst Bunke. Offline grammar-based recognition of handwritten sentences. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(5):818–821, May 2006.