2010-08-05

# Text Segmentation of Historical Degraded Handwritten Documents

Oliver Nina
*Brigham Young University - Provo*

# Text Segmentation of Historical Degraded Handwritten Documents

Oliver A. Nina

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Bryan S. Morse, Chair
Parris K. Egbert
Scott Woodfield

Department of Computer Science

Brigham Young University

December 2010

ABSTRACT

Text Segmentation of Historical Degraded Handwritten Documents

Oliver A. Nina

Department of Computer Science

Master of Science

The use of digital images of handwritten historical documents has increased in recent years. This has been possible through the Internet, which allows users to access a vast collection of historical documents and makes historical and data research more attainable. However, the insurmountable number of images available in these digital libraries is cumbersome for a single user to read and process.

Computers could help read these images through methods known as Optical Character Recognition (OCR), which have had significant success for printed materials but only limited success for handwritten ones. Most of these OCR methods work well only when the images have been preprocessed by getting rid of anything in the image that is not text. This preprocessing step is usually known as binarization. The binarization of images of historical documents that have been affected by degradation and that are of poor image quality is difficult and continues to be a focus of research in the field of image processing.

We propose two novel approaches to attempt to solve this problem. One combines recursive Otsu thresholding and selective bilateral filtering to allow automatic binarization and segmentation of handwritten text images. The other adds background normalization and a post-processing step to the algorithm to make it more robust and to work even for images that present bleed-through artifacts. Our results show that these techniques help segment the text in historical documents better than traditional binarization techniques.

Keywords: thresholding, binarization, thresholding of historical documents, text segmentation, binarization algorithm, binarization for OCR.

ACKNOWLEDGMENTS

I would like to thank the one who gives us all, God. Thanks also go to my two advisers Drs. Bill Barrett and Bryan Morse for their patience and their support. I also want to thank my wife for her encouragement and support during starvation times while living on a student budget. I also want to thank my parents that made it possible to get a higher education.

> There is no ingenious mind that has ever invented anything beneficial to the human family but that he obtained it from the one Source, whether he knows or believes it or not. There is only one Source whence men obtain wisdom, that is God, the Fountain of all wisdom; and though men may claim to make their discoveries by their own wisdom, by meditation and reflection, they are indebted to our Father in Heaven for all.
>
> –Brigham Young

## Contents

## List of Figures

# Chapter 1

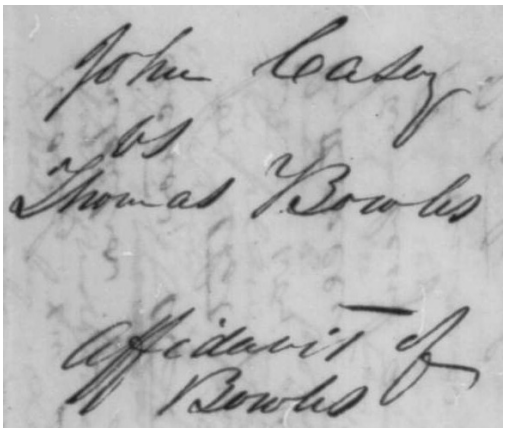
## Introduction

The use of digital images for research in genealogy and history has increased in recent years. These images contain important historical information and are usually available on the Internet in sites such as genealogical websites, digital libraries, or personal websites. Historians and researchers have turned to this new vast collection of digital information because it has become readily available and of low cost. Accessing these new digital resources demands less effort and increases productivity when compared to using physical documents.

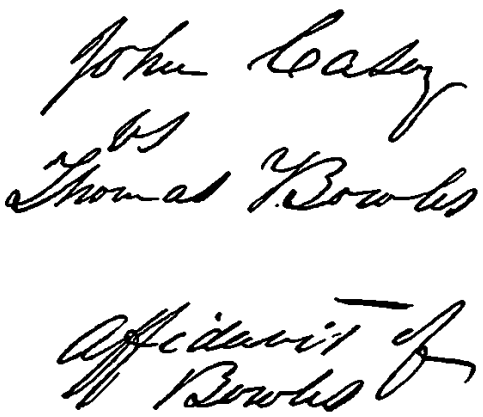
The enormous amount of historical digital images online can make research cumbersome and overwhelming. The Church of Jesus Christ of Latter Day Saints, a primary entity that stores and processes genealogical information, reported in 2006 having scanned 3,500 films and processed 2.2 million images. For genealogical researchers to go through this massive amount of information would be exhausting and lead to a huge amount of processing time even if the images are available online.

One way to alleviate the work of going through countless digital images is by indexing the information found on them. Indexing is a process that uses people or computers to read the information in the images and then digitally stores that information in a database. When researchers are looking for specific data, they can use a database engine or search engine to lookup the information from the indexed images.

Using people to manually index digital text images takes an extensive amount of time. The speed at which a text image can be indexed depends on the speed at which the person indexing the image can read and input the information from it. An automated method could



(a) Original Image



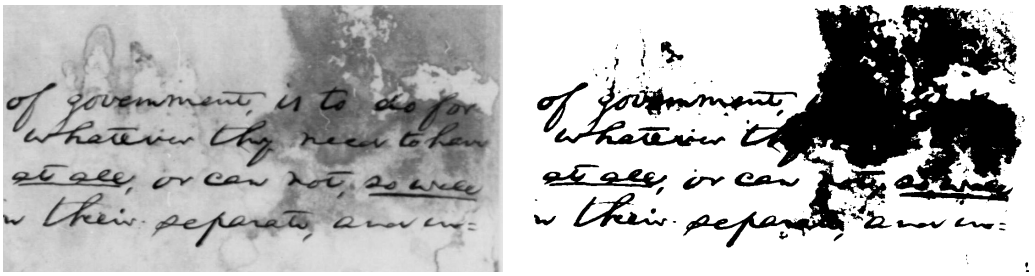
(b) Result of an ideal binarization algorithm

Figure 1.1: Historical document with a noisy background.

improve this speed by several factors. The process to read a text image through a computer is called Optical Character Recognition or OCR, which has had much success for printed documents but little success for handwritten ones.

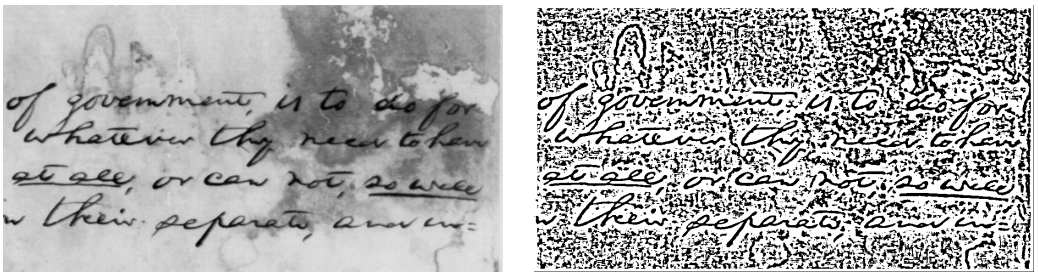
One of the first and very important phases of the OCR process is the *binarization* step. As Zhu puts it "Binarization is always a great challenge in all image process fields, especially in the process of document image whose binarization result can affect the OCR rate directly" [2006]. In order to automatically index historical text images, we need first to separate the text from the rest of the image (background). In other words we need to tell the computer which pixels in the image are text and which should be discarded. We do this by labeling the pixels into foreground (text) and background. Figure 1.1 shows an example of such binarization.

One of the simplest binarization techniques is called *thresholding*, which is a process that studies the statistics of the pixel values in the image and chooses a specific value that delimits or classifies the foreground and background. For instance, in grayscale images, darker values could be classified as text and lighter ones as background. Thresholding is one of the many attempts to separate text from background in grayscale images of historical documents. However, this approach struggles when the strokes on the text are faint and look very much like the background.



(a) Original Image (b) Otsu Thresholding

Figure 1.2: Otsu thresholding



(a) Original Image (b) Niblack Thresholding

Figure 1.3: Niblack thresholding

The poor image quality that hinders the correct binarization of an image is due to what we refer in broad terms as "noise". In this context we will use the term noise to broadly describe any anomaly in the image or the original document that makes the text less legible, such as artifacts from the camera, blobs of ink, degradation of the document, etc. The noise in these type of images is more problematic in parts of the image where the background is darker. This noise is often confused with the text, which makes it difficult for a person or a computer to read it. Figures 1.1, 1.2 and 1.3 show typical examples of this problem.

There are many thresholding algorithms that attempt to separate foreground (text) from background in scanned text documents, but most of them work well only for images where the text and background are clearly defined. Two popular thresholding algorithms such as Otsu [1979] and Niblack [1985] usually fail to properly binarize noisy images as shown in Figures 1.2 and 1.3 respectively. Chapter 2 explains in more detail why these algorithms and other thresholding algorithms fail on images with noisy backgrounds.

This thesis proposes and explores several techniques which include a novel iterative background estimation using median filtering, a novel recursive version of the Otsu [1979] thresholding algorithm for binarization, a novel selective bilateral filter [Tomasi and Manduchi, 1998] for noise removal, and an improved method for background normalization and noise despeckling techniques. The proposed method uses bilateral filtering to help smooth pixels from an image where the approximated background was removed and then applies a recursive Otsu algorithm with selective bilateral filtering to further separate text from noisy backgrounds. In a second variation of the algorithm, instead of doing a background subtraction of the image, we use an improved version of a background normalization and despeckling shown in [Lu et al., 2009a] for dealing with bleed-through artifacts. This thesis demonstrates that the combination of these techniques binarizes the text of historical document images better than traditional methods as shown by its greater accuracy compared to traditional binarization methods (Chapters 4 and 5).

# Chapter 2

# Background

There are many algorithms that attempt to separate foreground (text) from background in scanned text documents. One of these techniques is thresholding. A well known thresholding technique is the Otsu [1979] approach, which we introduce later in this chapter. We also explain in this chapter a technique for noise reduction called bilateral filtering [1998], which we use and adapt in our method.

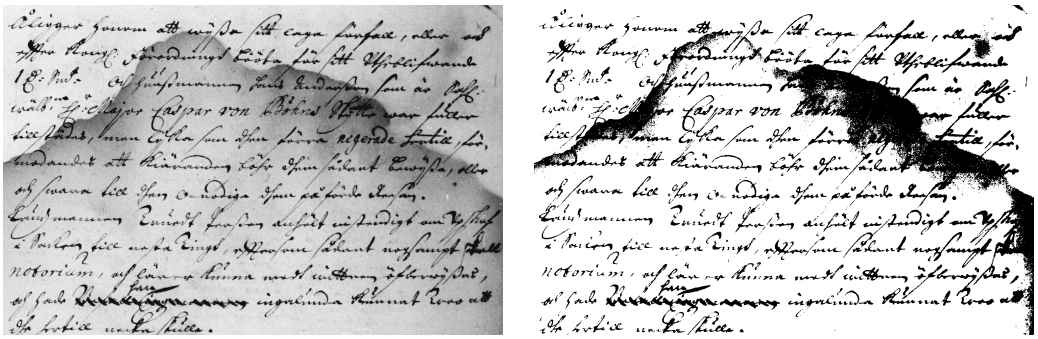
## 2.1 Thresholding

Thresholding algorithms for images are divided into two classes: global and adaptive thresholding. Global thresholding algorithms calculate one general threshold or pixel level value for the whole image and then convert the image to a bilevel image based on this threshold. Given an image $I$ and pixel values in the image represented as $I(x, y)$, the new binary image $I'$ will take new values as follows:

$$I'(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq t \\ 0 & \text{otherwise} \end{cases}$$

where $t$ is the selected global threshold value.

Global thresholding algorithms are usually faster to compute because they use a single threshold based on the global histogram of the gray-value pixels of the image. However, selecting the right threshold for the whole image is usually a challenge. In essence, classifying



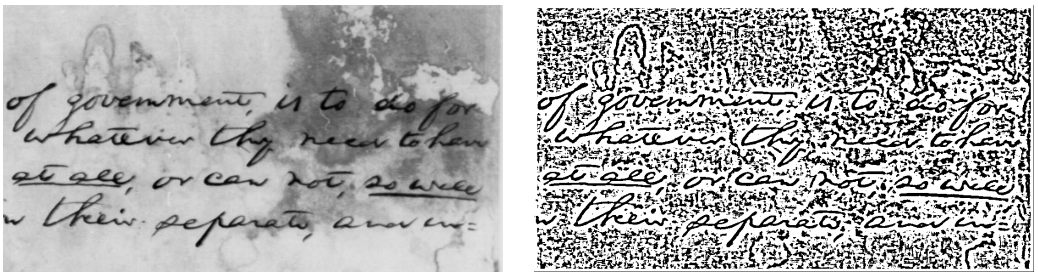
(a) Original Image (b) Otsu Thresholding

Figure 2.1: Demonstration of a global thresholding algorithm

the pixels of the image into 1s and 0s based on a specific threshold is trivial, but coming up with the optimal threshold is not.

One of the drawbacks of global thresholding algorithms is that they cannot differentiate foreground values from noise. This usually happens in text images where faint strokes are similar to the background or where blobs and spots are of similar intensity to the text. Another drawback is that they consider only one small part of the information embedded in the image represented as a histogram. Popular global thresholding techniques include those proposed by Otsu [1979], Kittler and Illingworth [1986], Kapur et al. [1985], Sahoo and Arora [2004] and Yen et al. [1995]. Image 2.1 shows the shortcomings of global thresholding. Notice in this image how the spots caused by degradation of the document confuses global thresholding and are incorrectly classified as text.

The other category of thresholding techniques is adaptive thresholding, which makes use of the fact that local intensities vary throughout images. They use a small window and run this over the entire image. At every pixel in the image a new value is calculated according to the neighboring pixels within the local window. Thus, they use different threshold values for different parts in the image. Although this approach might work better than global thresholding, it can also be very sensitive to noise and thus may not be suitable for text images with noisy backgrounds. Adaptive thresholding techniques include those by Niblack [1985], Sauvola and Pietikinen [2000], White and Rohrer [1983], Yasuda et al.



(a) Original Image (b) Niblack Thresholding

Figure 2.2: Demonstration of a adaptive thresholding algorithm

[1980], Yanowitz and Bruckstein [1988] and Bernsen [1986]. Image 2.2 shows the shortcomings of adaptive thresholding techniques. Notice in this image that the text is thresholded, but with it comes a lot of the noise from the background.

There are other non-thresholding techniques dedicated specifically to segmenting handwritten text images such as those by Shi and Govindaraju [2004], Oh [1995], Gatos et al. [2006], Gupta et al. [2007], Yan and Leedham [2004] and Yang and Yan [2000]. Although some of these techniques show promising results, the way noise gets handled in these techniques is still an ongoing issue.

In a survey of thresholding techniques in 2004, Sezgin and Sankur [2004] identified 40 different thresholding algorithms, which they ranked according to how well they performed in thresholding a set of sample images. They ranked the thresholding algorithms of Kittler and Illingworth [1986] and Sauvola and Pietikinen [2000] as the best performing document binarization algorithms.

In another comparison in 2007 of more current methods, Gupta et al. [2007] makes a comparison of six different methods that show promising results regarding thresholding of old text documents to further process with OCR software. In this paper, they conclude that "the classic Otsu method and Otsu-based methods perform best on average."

More recently, a binarization contest was organized where 35 research groups participated with 45 different thresholding algorithms for type-written and hand-written documents (see [Gatos et al., 2009]). This contest, called Document Image Binarization Contest

or DIBCO, set a more up-to-date benchmark for thresholding methods. In this contest, a recent algorithm developed by Lu and Tan performed the best. This algorithm has been acknowledged by the authors to be very similar to their previously published algorithm in [Lu and Tan, 2007], but it has not yet been published.

## 2.2 Introduction to the Otsu Method

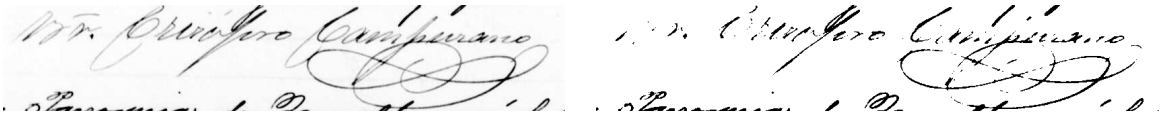
The Otsu algorithm [1979] is a well-known algorithm that determines a global threshold for an image based on its histogram. As described in the classic 1979 paper, Otsu thresholding tries to minimize the within-class variance for each of the two classes in the histogram (foreground pixels and background pixels). But minimizing the within-class variance is the same as maximizing the between-class variance. This makes the computation of the threshold much easier. Calculating the between-class variance $\sigma_B^2(t)$ for a threshold $t$ is expressed as follows:

$$\sigma_B^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$$

where $w_i$ are the weights or number of pixels for the two classes depending on any normalization, $\mu_i$ are the means of the two classes, and $t$ is the value of the potential threshold.

A second step of the algorithm is to look for the maximum between-class variance for all potential values of $t$ and then choose that $t$ as the global threshold.

Although the Otsu technique has proven to be an effective thresholding algorithm, it fails in instances where the foreground and background are similar. Figure 2.3 shows the result of applying Otsu thresholding to a text image where the strokes are weak and blend with the background resulting into a lose of faint strokes in the resulting image. We can also observe in Figure 2.1 how applying Otsu thresholding could introduce false positives into the final result when the background image has a significant amount of noise.



(a) Original image (b) After applying Otsu thresholding

Figure 2.3: The Otsu algorithm.

## 2.3 Introduction to the Bilateral Filter

The bilateral filter has been used before for processing text images. Some of the papers that have used the bilateral filter on text images are [Basavaraj and Samuel, 2007], who use the bilateral filter for offline handwritten character detection and [Nina and Morse, 2009], who use it for interactive smoothing of text images.

The bilateral filter was first introduced by Smith and Brady [1997] who called it "SUSAN". Later, Tomasi and Manduchi [1998] rediscovered it and called it the "bilateral filter", which is its most common name.

The bilateral filter replaces each pixel by a new value that is the weighted average of its neighbors based on two weighting functions: spatial proximity and similarity. These Gaussian weighting functions ($G_\sigma$) are based on proximity to the pixel being processed and similarity in intensity. Considering a gray-level value $I_p$ of a pixel $p$ being processed, the new value $I_p^{'}$ based on bilateral filtering is calculated based on the pixels $q$ in the neighborhood $S$ of $p$ as follows:

$$I_p^{'} = \frac{1}{W_p} \sum_{q \epsilon S} G_{\sigma_s}(||p-q||) G_{\sigma_r}(|I_p - I_q|) I_q$$

where

$$W_p = \sum_{q \epsilon S} G_{\sigma_s}(||p-q||) G_{\sigma_r}(|I_p - I_q|)$$

The parameter $\sigma_s$ defines the standard deviation of the spatial weighting function around a pixel. The parameter $\sigma_r$ controls the standard deviation of the similarity weighting function between a pixel and its neighboring pixels.

One of the advantages of the bilateral filter is that it smoothes only areas where the pixels are similar. This allows it to preserve edges in the image while smoothing everything else.

Since its discovery, the use of the bilateral filter has been limited due to its computational constraints. The original algorithm has a time complexity of $O(n^2)$ where $n$ is the number of pixels in the image. This made it less popular for processing images interactively and automatically. However, the use of the bilateral filter has increased in recent years due to the discovery of new implementations of it. One of these implementations was introduced by Weiss [2006]. His implementation reduced the time complexity of the algorithm to $O(\log n)$. An even faster implementation has been introduced recently in 2008 by Porikli [2008], whose algorithm achieved a constant time complexity of $O(1)$.

These as well as other advances in improving the computation of the bilateral filter have made it more viable to be used in practical applications in image processing. A good example of this is the set of real-time edge-aware image processing tools designed by Chen et al. [2007].

We will discuss in the next chapter the core implementation and functionality of the proposed algorithm.

# Chapter 3

# Proposed Algorithm

Our methods combine both existing and novel techniques to create novel and powerful ways to binarize and segment the text in a handwritten text image. Some of the techniques that are part of our contribution are a novel iterative background estimation using median filtering, a novel recursive version of the Otsu algorithm, a novel selective bilateral filter for noise removal, and improved background normalization and noise despeckling techniques. The rest of this section describes in more detail these techniques and how we combine them.

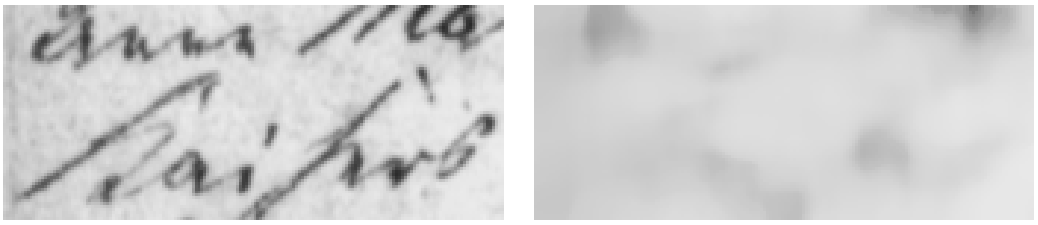
## 3.1 Basic Algorithm

This section explains the basic components for our first algorithm developed originally for this thesis, which we called Algorithm I. In the next section we cover the improvements made to this algorithm, which we call Algorithm II.

### 3.1.1 Background Approximation

The first step is to get rid of as much as possible the noise and degradation that may have been introduced to the image. This technique was first used and described by Hutchison [2003].

To approximate the background we use a median filter to smooth the image. We do this by sliding a small window over the image. The center of the window is the pixel being currently processed. The median filter looks at the pixels inside the kernel and picks the median of these values as the new value for the center of the window. We can approximate the kernel for the median filter by looking at the character size of the text we want to threshold.



(a) Original (b) Background Approximation

Figure 3.1: Background Approximation using a median filter

Usually the bigger the text, the bigger also the kernel to be used in this approximation. In our experiments, we have used a kernel size of $21 \times 21$, which is about half the character size of the text.

This smoothing of the image with a median filter gives an image that looks like the background without the text. This happens because text images have a lower ratio of text pixels to background pixels than other images. This means that the median values of the neighborhoods surrounding the majority of the pixels in the image will be background pixels. This feature of text images allows us to use the median filter to obtain an approximation of the background. Because it is only an approximation, this implies that it is not completely accurate. Therefore further processing is required to eliminate the noise left or introduced in the background. Figure 3.1 shows an example of this background approximation.

### 3.1.2 Background Subtraction

Once we obtain an approximation for the background of the original image, we can remove this background by subtracting the values of the original image from it. We do not perform the subtraction the other way around because we will obtain mostly negative values if we do it that way. We can see in Figure 3.2(a) the image result after subtracting the original image from the background. Since it is easy to see the strokes on the image when the characters are black, we then invert the pixels in the image and convert black pixels to white and so forth. Figure 3.2(b) shows the result of doing this for the example in Figure 3.2(a).

(a) Image with Background Subtracted (b) Inverted Image.

Figure 3.2: Background Subtraction using the original image and estimated background image from Figure 3.1

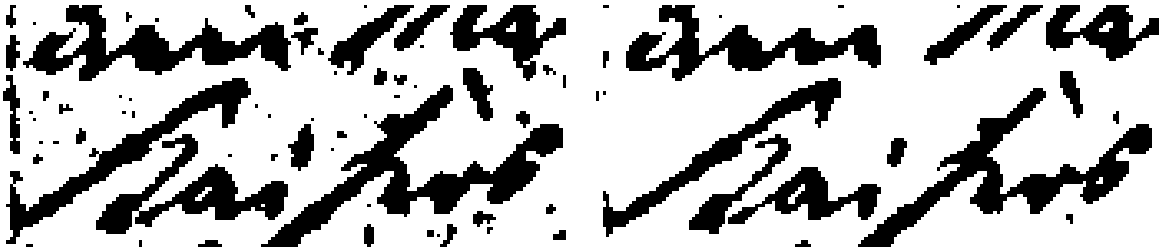
(a) T=248 (b) T=255

Figure 3.3: Image from Figure 3.2(b) manually thresholded at values of 248 and 255.

### 3.1.3 Bilateral Filtering for Noise Removal

Once we subtract the approximation of the background from the original image, we can see that there is still a lot of noise associated with it, which we can more clearly see by manually thresholding the image. Figure 3.3 shows the image from Figure 3.2(b) manually thresholded at values of 248 and 255. We use these values here only in order to better see the noisy pixels that remain in the image despite the background removal. This image shows the noise in the background that remained after the background subtraction.

In order to eliminate the remaining noise in the image, we apply a bilateral filter as described by Tomasi and Manduchi [1998] with parameters $\sigma_s = 10$ and $\sigma_r = 2$. By applying the bilateral filter we can see that we can eliminate much of the noise while still keeping the main strokes of the text. Figure 3.4 shows the background subtracted image with and without the bilateral filter applied to it. Notice that much of the noise has disappeared.



(a) Without bilateral filter (b) With bilateral filter

Figure 3.4: Comparison of the image with and without bilateral filtering, then thresholded at 248.

### 3.1.4 Recursive Otsu Algorithm

In 2007, at the Family History Technology Workshop [Nina and Barrett, 2007], we introduced a recursive version of the Otsu algorithm, which uses the Otsu method to subdivide graylevel ranges for the same image multiple times. This allows us to get different gray-level thresholds for the image and combine them to obtain the final result.

There has been another version of the Otsu algorithm used recursively in [Cheriet et al., 1998]. Some of the differences between our algorithm and the previously mentioned algorithm include the stopping criterion for the iterations, the pre-processing and post-processing of the iterations, and the learning process for the background (which our algorithm does not require).

In our recursive formulation, the first threshold is determined using the original Otsu technique (Section 2.2). But for successive iterations the technique uses only the pixels labeled as background in the previous iteration. This process diminishes the variance of the pixels to be processed but also helps distinguish different levels of the foreground as seen in Figure 3.5. By applying the Otsu technique to parts of the background we can recover many faint strokes that were not recognized on the first pass or iteration. We continue applying Otsu recursively to the rest of the background without considering previously thresholded pixels.

It is also worth to note here that when applying the Otsu threshold to separate the foreground from the background, we could include the threshold value in the foreground

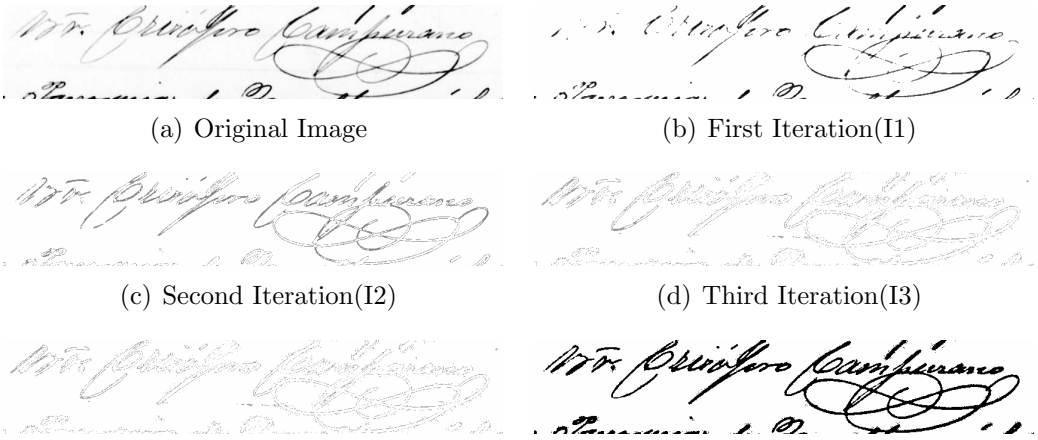
or the background depending on the result we would like to obtain. In our experiments we chose to include the threshold value in the background for Algorithm I and in the foreground for Algorithm II.

The original algorithm published in 2007 had the stopping criterion of considering the threshold of the approximated background as the threshold for stopping the algorithm. If the threshold values from the previous iterations were less than the threshold value for the background, then the algorithm would continue performing more iterations until the threshold exceeds the background threshold.

This stopping criterion worked for most images, but it would not allow extra processing on text pixels that were very faint and look like the background. Hence, we changed the stopping criterion of the algorithm to allow the processing of a greater number of pixels.

In the current version of the algorithm, our stopping criterion assumes that the pixels being added in every iteration to the final result will always be less than the number of pixels obtained in the first threshold. We have empirically noticed that this stopping criterion works well for text images because most of the text is usually recovered on the first pass of the Otsu threshold, and for the subsequent thresholds only faint strokes should be recovered. If the number of pixels being recovered is greater than the number of pixels recovered on the first iteration, then we have reached the background and we should stop.

There are also other stopping criteria we use for this algorithm. We check for the case when between two iterations there are no pixels recovered. This happens usually when there are no more pixels to recover. Beyond a certain threshold we stop doing more iterations because the values are too close to white (255). In our Algorithm I we set this maximum threshold to 249. Also, we use two parameters $d_1$ and $d_2$ to check that the difference between the thresholds of the previous and current iterations is at least greater than $d_1$ and less than $d_2$. This means that we want to add pixels to our solution that are different enough but not too different. For our Algorithm I, we use the values $d_1 = 2$ and $d_2 = 30$, and for Algorithm II we use $d_1 = 2$ and $d_2 = 26$.



(a) Original Image (b) First Iteration(I1)

(c) Second Iteration(I2) (d) Third Iteration(I3)

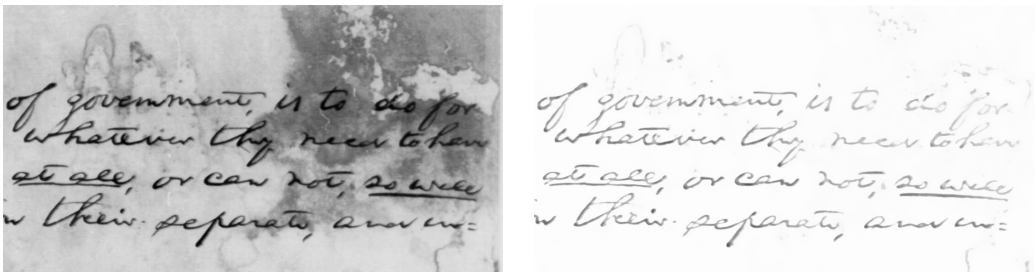
(e) Fourth Iteration(I4) (f) Recursive Otsu(I1+I2+I3+I4)

Figure 3.5: The Original Recursive Otsu algorithm explained in [2007](Compare to Figure 2.3(b)).

### 3.1.5 Selective Bilateral Filtering

Classifying correctly what is background, foreground and noise is the main focus of most binarization algorithms. Now that we have a good approximation of the text and background, we can use a selective variation of the bilateral filter to reduce and alleviate the amount of noise introduced to our final result. The motivation for this is for our selective bilateral filter to act more like a clustering algorithm whose main function is to group every pixel into smaller groups. By grouping (smoothing) all the pixels in the image, noise pixels also get grouped with either the foreground or background groups according to how similar and close to these group pixels they are.

This step is implemented in the following way. We first apply the recursive Otsu algorithm as described in Section 3.1.4 without noise removal to approximate the background and foreground of the image. Based on these approximations, we apply a selective bilateral filter *only between background pixels* using parameters $\sigma_s = 10$ and $\sigma_r = 3$, which means that among the pixels that we believe are background we want to try to group or consolidate most pixels that are far apart by a standard deviation of 10 and that are similar by a standard deviation of 3 in grayscale values. We apply another step of selective bilateral filtering but this time *only between the foreground pixels* using a more conservative filter with parameters



(a) Original Image (b) Selective Bilateral Filtering

Figure 3.6: Result after selective bilateral filtering.

(a) First pass of recursive Otsu (b) Image after noise removal

Figure 3.7: Result after noise removal.

set to $\sigma_s = 2$ and $\sigma_r = 2$. This implies that we do not want to group pixels that are too far apart and too different. Figure 3.6 shows the results after this step.

### 3.1.6 Final Pass of Recursive Otsu Interleaved with Noise Removal

Although the background removal, the recursive Otsu algorithm and the selective bilateral filter attempt to avoid any noise in the image, they still fail to remove small faint speckles that are introduced by iterations after the first one. One way we deal with this is to only add pixels that connect to pixels from the first iteration, or in other words pixels that we know for sure are text. One way we remove noisy pixels from the final result is by searching within the layer being processed for pixels that are connected to pixels denoted as text strokes in the previous iteration. These pixels might not be directly connected to text strokes but might be indirectly connected to text strokes by a number of connected pixels within the layer. In this sense we consider directly and indirectly connected pixels as text strokes. If we can't validate that a pixel being processed is directly or indirectly connected to a text stroke

classified in the previous iteration then we remove it. We also revisit the second threshold at the end of all iterations and again check for connectivity. This is because it is possible that some of the pixels in the second iteration might be now connected because we added new pixels from the third or fourth iterations. This approach resembles that of thresholding with hysteresis in [Canny, 1986]. Figure 3.7 shows the results after applying this step. Only Algorithm I uses this approach as a final step for removing noisy pixels from the result.

### 3.1.7 Algorithm I

Putting the pieces from sections 3.1–3.5 together, we use the following algorithm

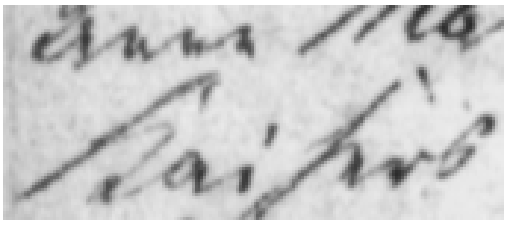
1. Estimate the background (Section 3.1.1).
2. Remove the background (Section 3.1.2).
3. Use a bilateral filter to smooth the resulting image (Section 3.1.3).
4. Apply Recursive Otsu thresholding until it stops (Section 3.1.4).
5. Apply selective bilateral filters to the image from Step 3 using the template from Step 4 (Section 3.1.5).
6. Apply Recursive Otsu thresholding with noise removal until it stops (Section 3.1.6).

## 3.2 Algorithm Improvements

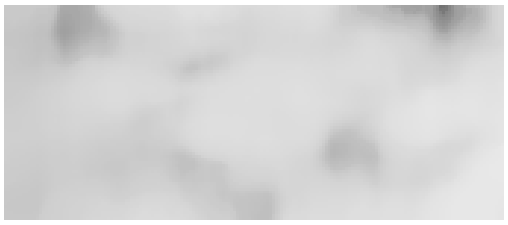
We discuss in this section the improvements we introduced to the algorithm discussed previously in order to achieve better results.

### 3.2.1 Iterative Background Estimation

An alternative way to approximate the background is to do several iterations of the median filter on the same image. This is an effective way to approximate the background of the image because every iteration continues to more accurately resemble the background especially in



(a) Original



(b) Background Approximation

(c) Iterative Background Approximation

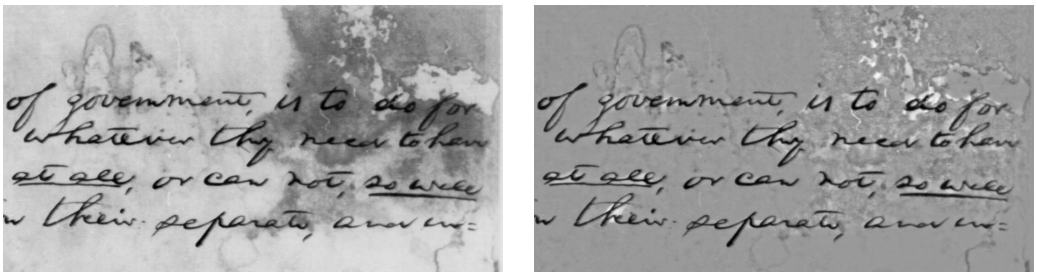
Figure 3.8: Iterative Background Approximation

places where there is a larger concentration of pixel strokes. In our experiments we use three iterations of the 21 $\times$ 21 median filter to better approximate the background.

This technique is better and more accurate than simply increasing the kernel of the median filter because we process fewer numbers of pixels in areas where the kernel visited the image as supposed to larger areas with a larger kernel. This means that with a larger kernel, we have more values to examine within the kernel which turns into less accuracy in choosing the background pixel, whereas with a smaller kernel we have a smaller area and more accuracy.

Lu's algorithm [Lu et al., 2009a] also performs a background estimation that yields results similar to the iterative background estimation presented here. However, their technique uses a polynomial fitting algorithm, which makes it more computationally complex and slower.

Figure 3.8 shows a comparison between single pass background approximation with a median filter and iterative background approximation.



(a) Original Image (b) Compensated Image

Figure 3.9: Result after applying image compensation.

### 3.2.2 Compensation of Contrast Variation

The idea of compensation of contrast variation is introduced in [Lu et al., 2009a]. After obtaining a background approximation for the image, this step makes the background of the original more uniform and even. To do this we use the following formula:

$$\hat{I} = \frac{C}{BG} \times I$$

where $\hat{I}$ is the new image, $C$ is the median intensity of the original image, $BG$ is the estimated background image and $I$ is the original image. The value of the ratio $\frac{C}{BG}$ indicates that wherever the background pixel is darker than the median value, the resulting pixel will be lighter and wherever the background pixel is lighter, the resulting pixel will be darker.

Because this formula uses a ratio that will change the original pixel, we need to be careful in detecting that the pixel does not go beyond the range of [0,255]. Lu does this by applying a cutoff of 255 to the values. So if any pixel goes over 255 then they simply replace such value by 255. Although this works well for most images, we noticed empirically that if we do a normalization (scaling) of the values as opposed to just doing a cutoff we get better results. Therefore in our algorithm for this step we use a normalization of the values instead. Figure 3.9 shows the results of this step.

### 3.2.3 Despeckling

Finally, we use a despeckle algorithm similar to that used in [Lu et al., 2009a]. In their paper, they describe the algorithm by first labeling all segmented text through connected components. Then a difference between each labeled connected component and its corresponding patch within the background image is estimated as follows:

$$Diff(c) = |BC_c - \hat{I}_c|$$

where $\hat{I}_c$ is the average intensity of the connected component and $BG_c$ is the average intensity of the corresponding patch in the background estimated image. The idea here is that generally the difference intensity between the real text and background is much higher than that of noise and background. A threshold is picked by building a histogram and applying the Otsu algorithm.

This approach works well, but we make it even more robust by adding a second parameter to the comparison. We also build a histogram for the size of every connected component and apply Otsu to find a threshold. We noticed that noisy pixels not only are very different in intensity but also in size, so our algorithm removes connected components that are too different in either intensity or size.

### 3.2.4 Algorithm II

Using these ideas we improved our algorithm to make it more robust against noise. We also added improvements to our algorithm to handle bleed-through cases, which is a special case of binarization and is an addition we made to our original thesis proposal. Here is the newer version of the algorithm.

1. Iterative Background Approximation (Section 3.2.1).
2. Compensation of Contrast Variation (Section 3.2.2).
3. Use a bilateral filter to smooth the resulting image (Section 3.1.3).

4. Apply Recursive Otsu until Stop (Section 3.1.4).

5. Apply despeckle algorithm (Section 3.2.3).

# Chapter 4

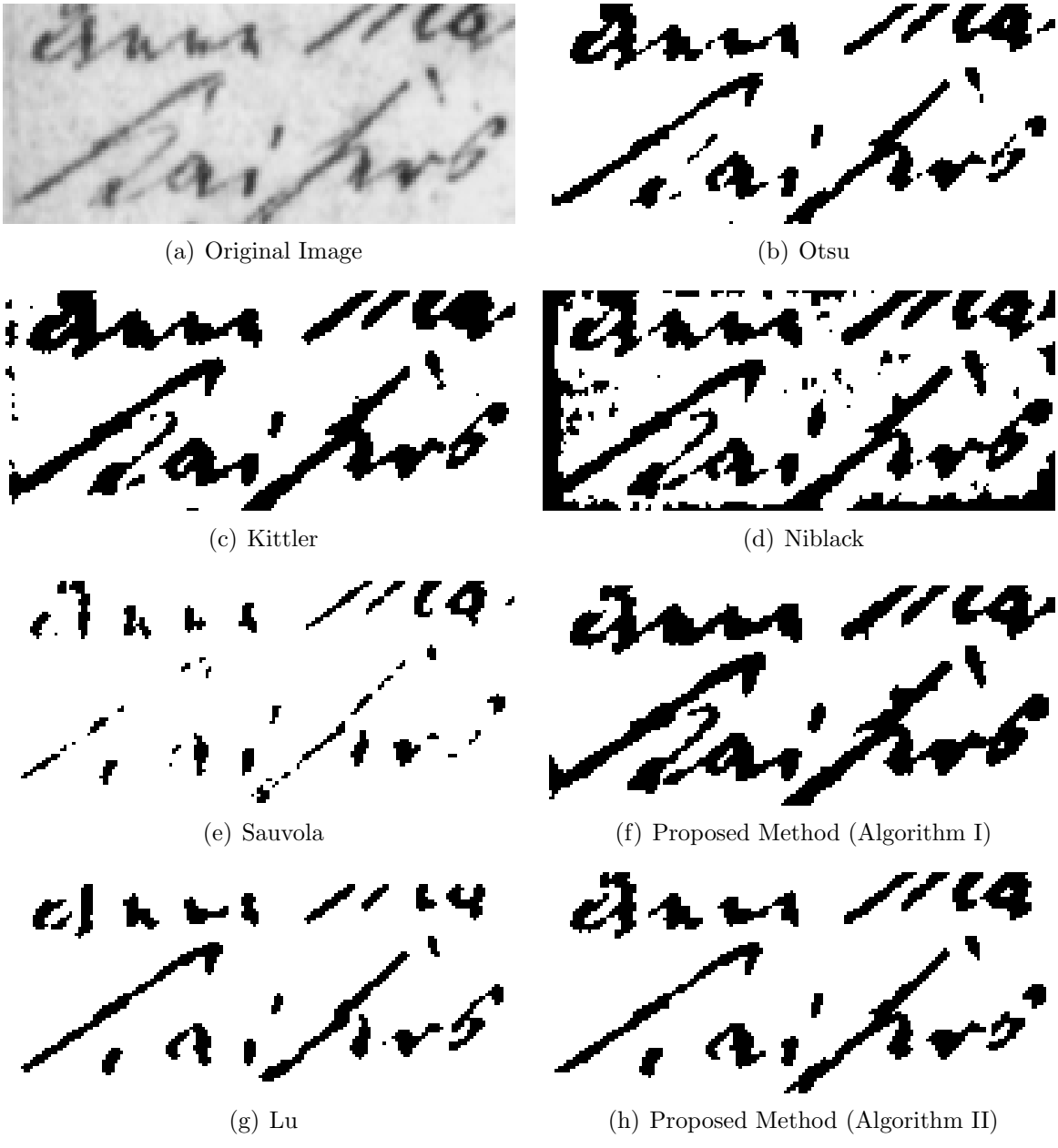
# Results

This chapter presents both qualitative examples of the proposed methods as well as quantitative evaluation of the results. We use the dataset used in the DIBCO 2009 contest for document binarization and explain the metrics used for evaluation [Gatos et al., 2009].

The images shown for Lu's algorithm on the quantitive results are images obtained from the author. We also implemented this algorithm, but the parameters in our implementation differ somewhat from that of their original algorithm.

## 4.1 Qualitative Results

We have used two sets of images to test the performance of our methods. The first set of images is shown in Figures 4.1, 4.2 and 4.3, which present a qualitative comparison of the results of our proposed method with those of the Otsu [1979], Kittler and Illingworth [1986], Niblack [1985], Sauvola and Pietikinen [2000] and Lu et al. [2009a] methods. For this first set of images, visual inspection demonstrates the improved performance of our methods over traditional techniques.



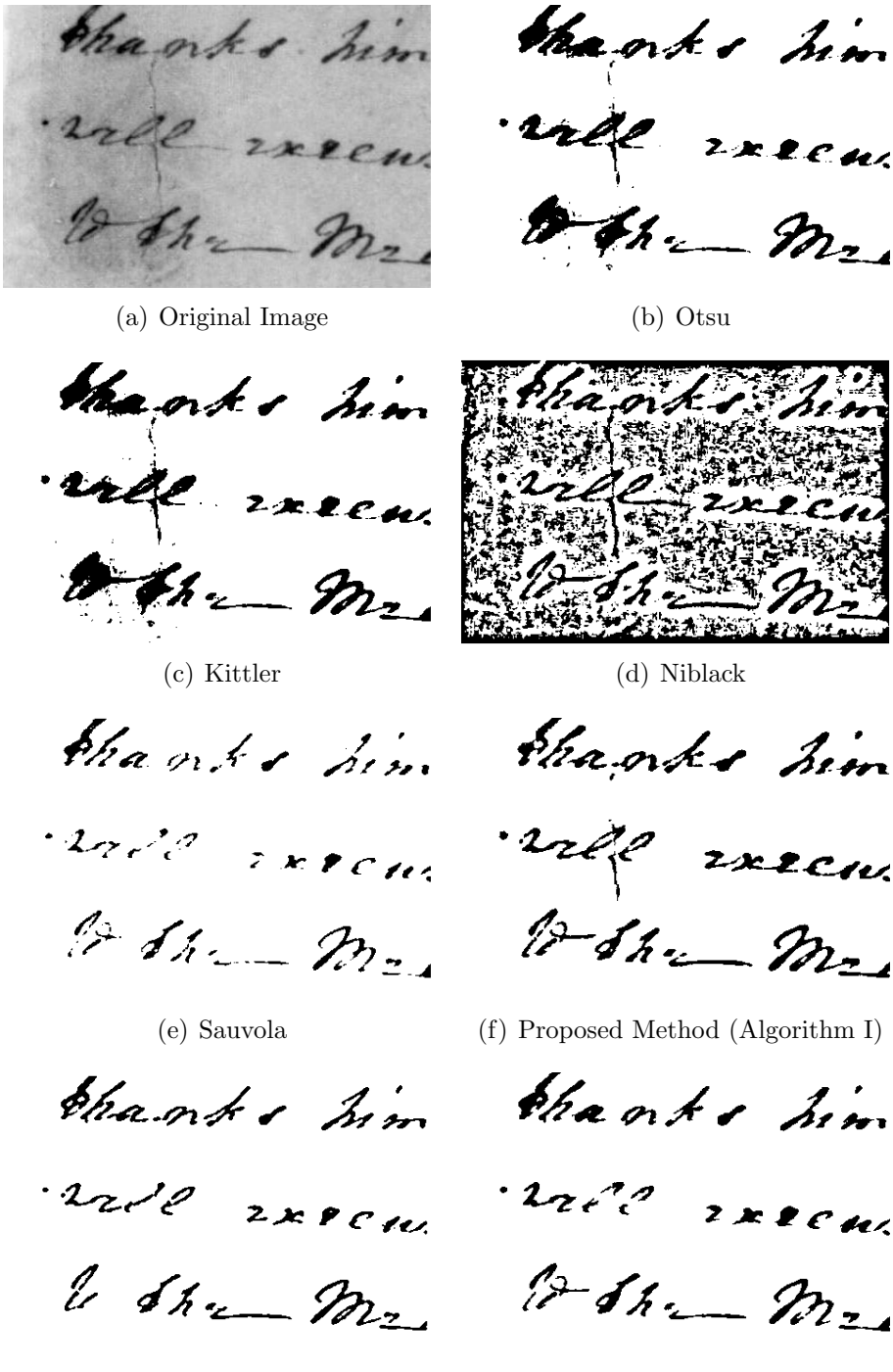
(a) Original Image (b) Otsu

(c) Kittler (d) Niblack

(e) Sauvola (f) Proposed Method (Algorithm I)

(g) Lu (h) Proposed Method (Algorithm II)

Figure 4.1: First comparison of thresholding methods. Notice that our Algorithm I performs best on this test case.



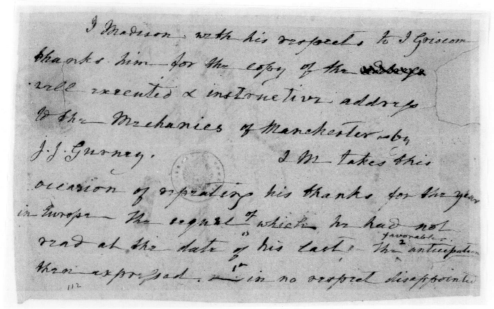
(a) Original Image (b) Otsu

(c) Kittler (d) Niblack

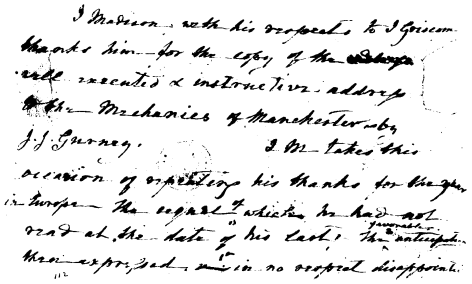
(e) Sauvola (f) Proposed Method (Algorithm I)

(g) Lu (h) Proposed Method (Algorithm II)

Figure 4.2: Second comparison of thresholding methods. The last three algorithms (our two methods and Lu's) have very similar results.

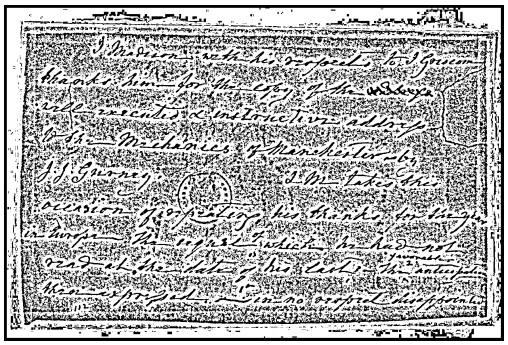


(a) Original Image

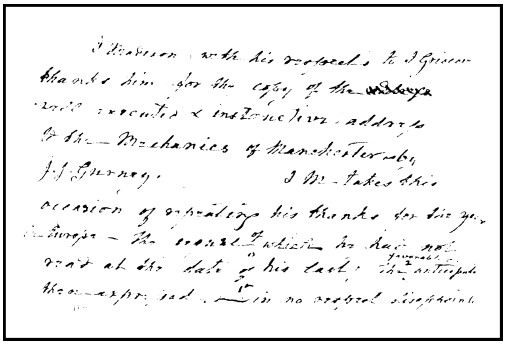


(b) Otsu

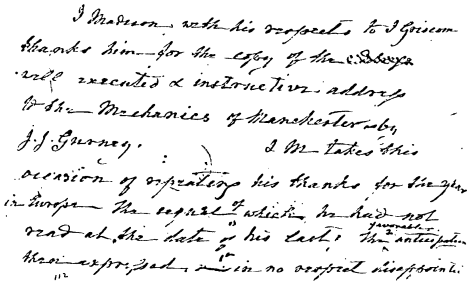
(c) Kittler



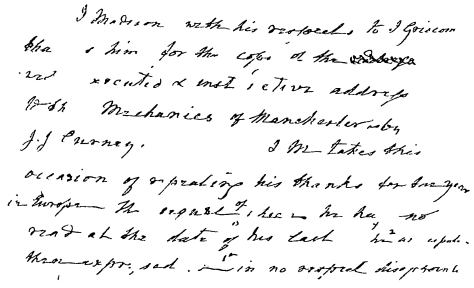
(d) Niblack



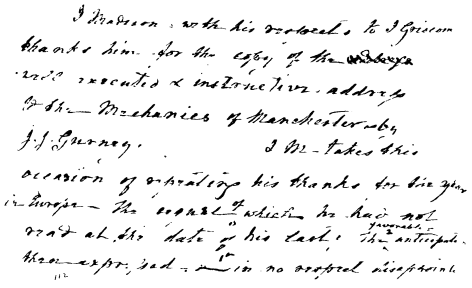
(e) Sauvola



(f) Proposed Method (Algorithm I)

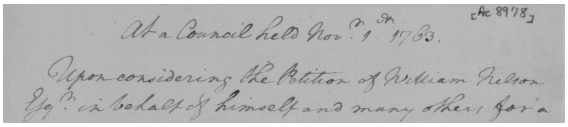


(g) Lu

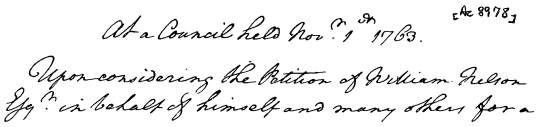


(h) Proposed Method(Algorithm II)

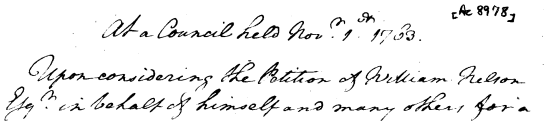
Figure 4.3: Third comparison of thresholding methods. Our Algorithm II performs best by recovering more strokes.



(a) Original Image



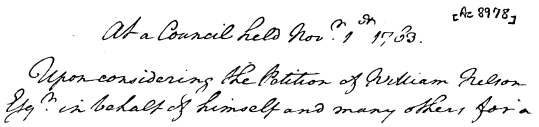
(b) Ground Truth



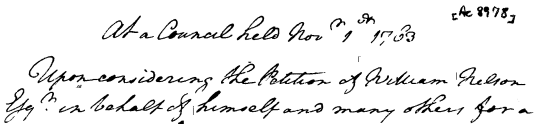
(c) Otsu

(d) Niblack

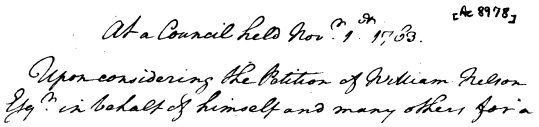
(e) Sauvola



(f) Proposed Method (Algorithm I)

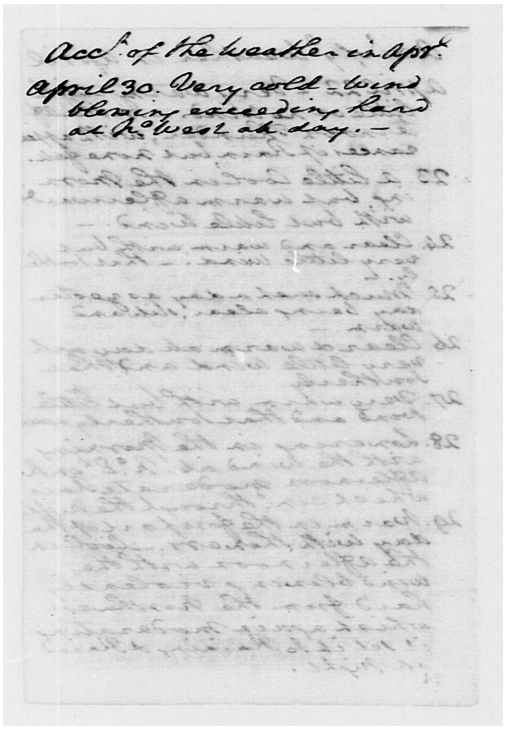


(g) Lu

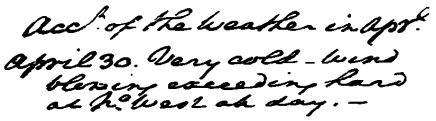


(h) Proposed Method (Algorithm II)

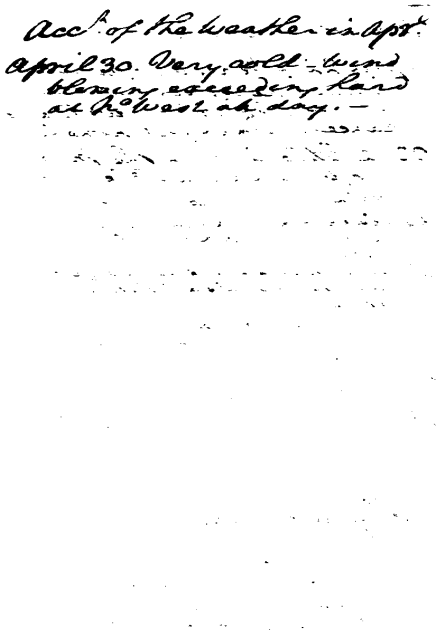
Figure 4.4: Image H01 used in the DIBCO contest with comparison of various binarization methods including the proposed methods.



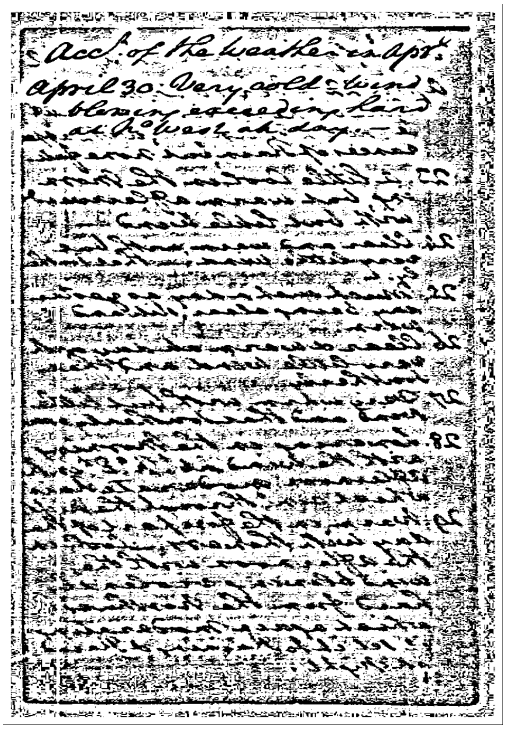
(a) Original Image



(b) Ground Truth

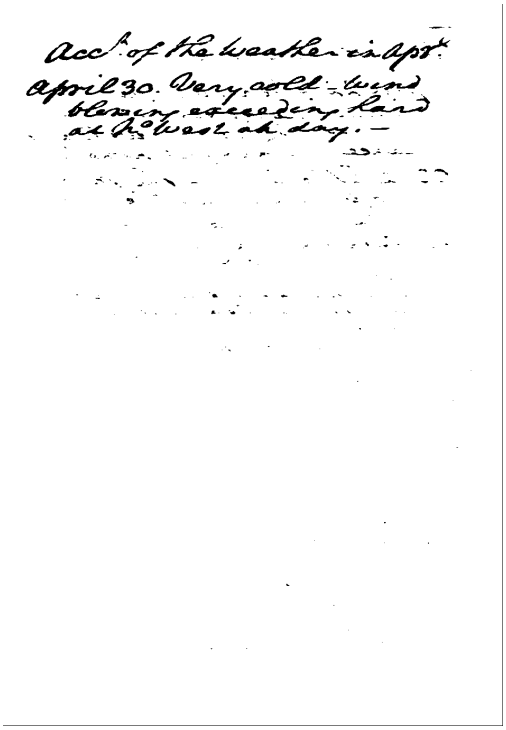


(c) Otsu

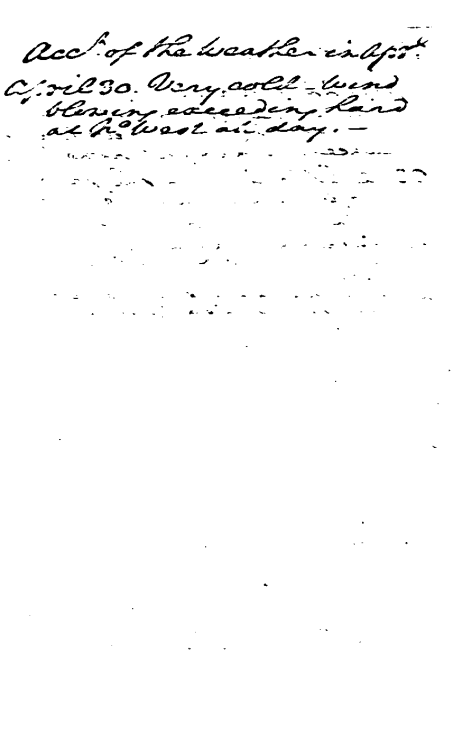


(d) Niblack K=-0.2 R=15

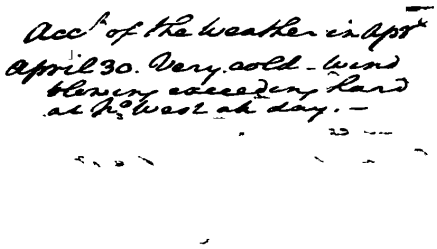
Figure 4.5: Image H02 used in the DIBCO contest with comparison (Part I).



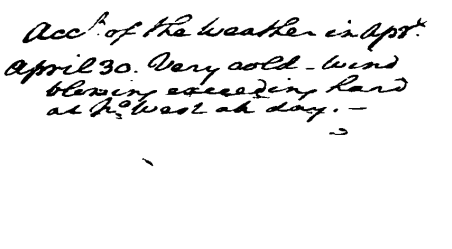
(a) Sauvola Radius=15,K=0.5, R=128



(b) Proposed Method (Algorithm I)

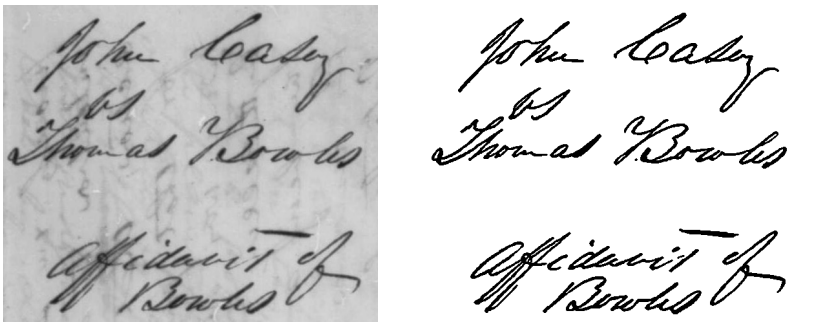


(c) Lu



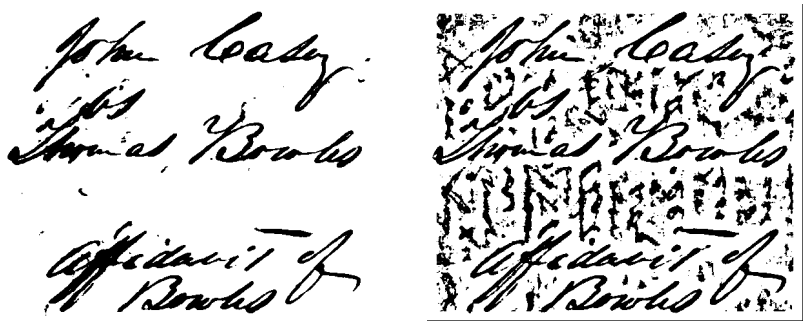
(d) Proposed Method (Algorithm II)

Figure 4.6: Image H02 used in the DIBCO contest with comparison (Part II).



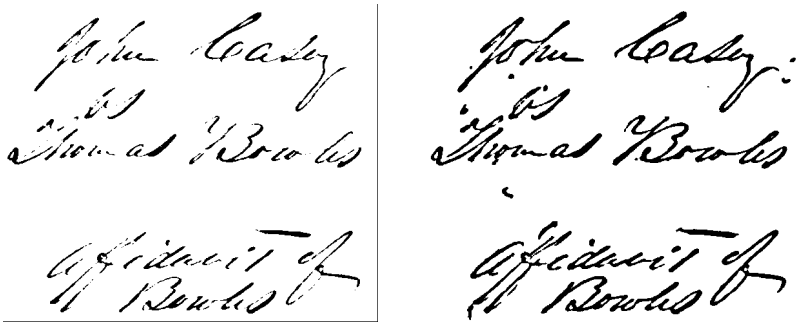
(a) Original Image

(b) Ground Truth



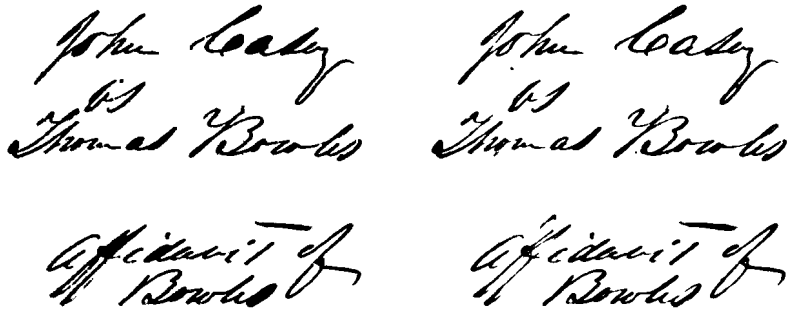
(c) Otsu

(d) Niblack K=-0.2 R=15



(e) Sauvola Radius=15,K=0.5, R=128

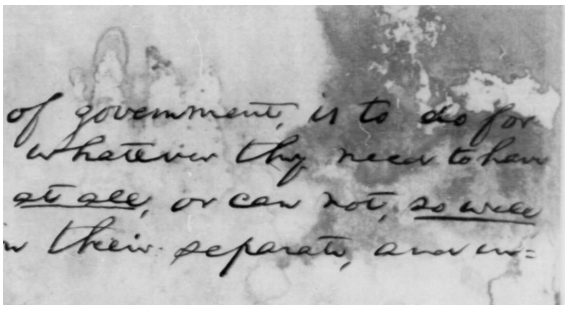
(f) Proposed Method (Algorithm I)



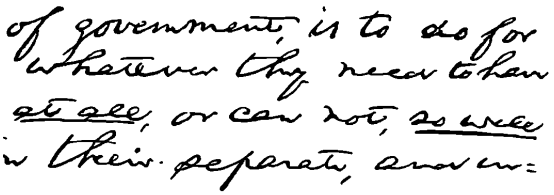
(g) Lu

(h) Proposed Method (Algorithm II)

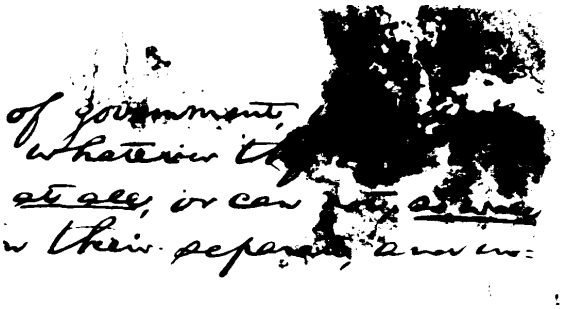
Figure 4.7: Image H03 used in the DIBCO contest with comparison.



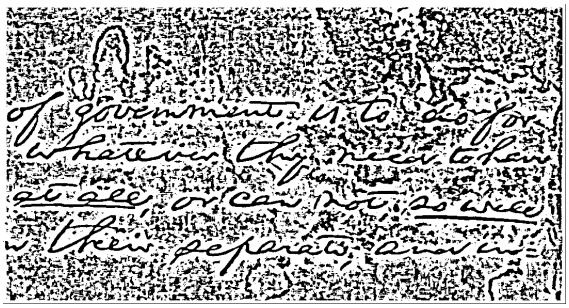
(a) Original Image



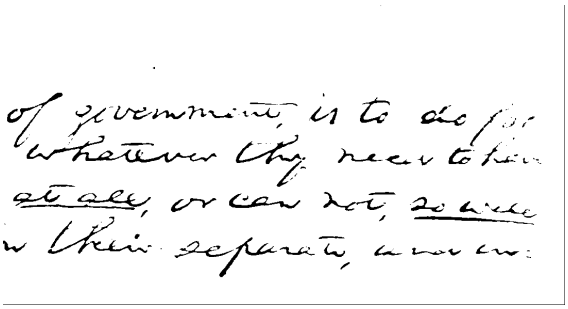
(b) Ground Truth



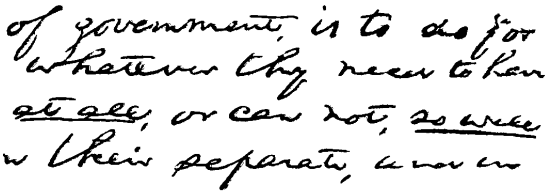
(c) Otsu



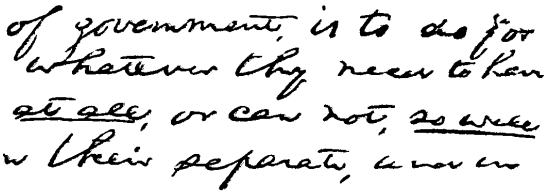
(d) Niblack



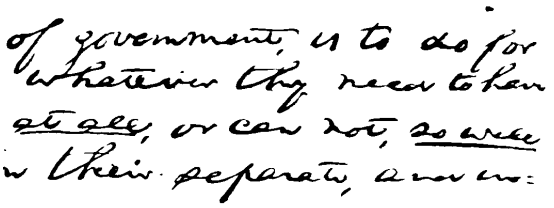
(e) Sauvola



(f) Proposed Method (Algorithm I)

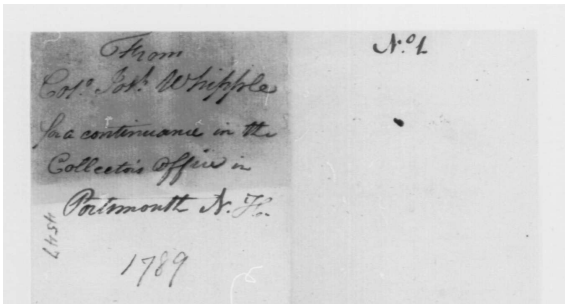


(g) Lu

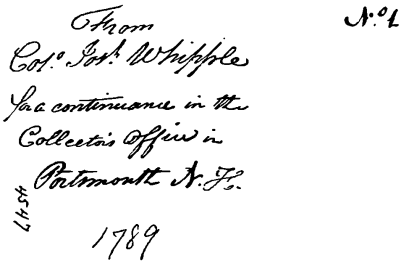


(h) Proposed Method (Algorithm II)

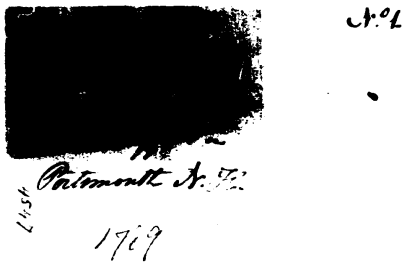
Figure 4.8: Image H04 used in the DIBCO contest with comparison.



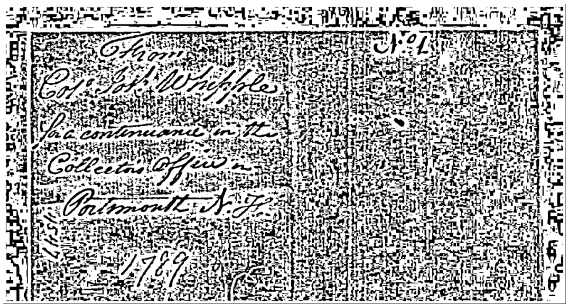
(a) Original Image



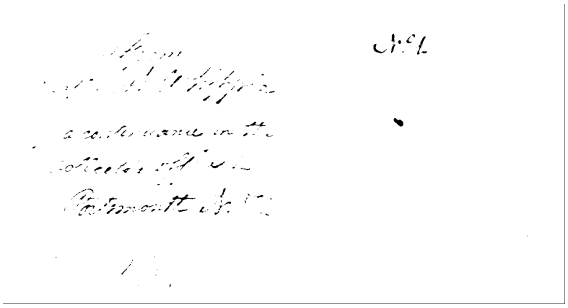
(b) Ground Truth



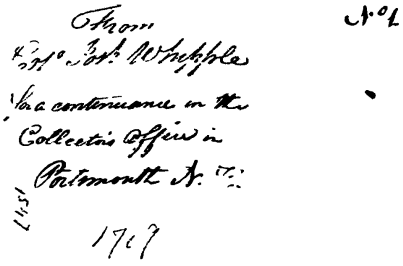
(c) Otsu



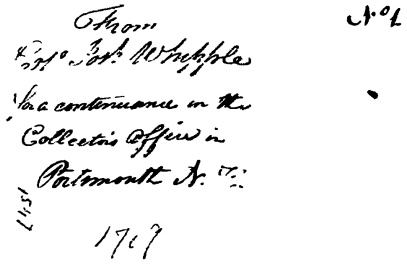
(d) Niblack



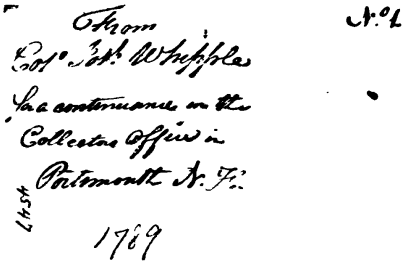
(e) Sauvola



(f) Proposed Method (Algorithm I)



(g) Lu



(h) Proposed Method (Algorithm II)

Figure 4.9: Image H05 used in the DIBCO contest with comparison

## 4.2 Quantitative Evaluation

The second set of images composed of Figures 4.4 through 4.9 are the hadnwritten images used in the DIBCO contest at ICDAR 2009. We also use their ground-truth images to calculate the metrics for the performance of the various algorithms. This section describes and interprets the metrics used for the evaluation of the algorithms.

### 4.2.1 Evaluation Measures

To quantitativly compare the proposed methods to others, we use three of the four metrics used at DIBCO 2009. These metrics are the following: (i) F-Measure; (ii) PSNR and (iii) Negative Rate Metric (NRM). These metrics were chosen because they were the most relevant.

#### F-Measure

F-Measure is a metric used in information retrieval to calculate the accuracy of the results based on ground truth images. In our case, each pixel in the result is compared to its corresponding pixel in the ground truth image. In order to calculate the F-Measure of the results, we first calculate the *precision* and *recall* of the results as follows.

Precision is the number of all correct values retrieved (true positives) as a percentage of all the values retrieved whether they are correct or incorrect values (false positives). In a more formal way, precision takes the following form:

$$Precision = \frac{TP}{TP + FP}$$

where TP and FP stand for True Positives and False Positives respectively.

Recall is the number of pixels labeled correctly in the result divided by all the possible correct pixels corresponding to the ground truth, including those that we did not get right but that should be included in the result (false negatives). More formally, recall is calculated

as follows:

$$Recall = \frac{TP}{TP + FN}$$

where TP, FP and FN stand for True Positives, False Positives and False Negatives respectively.

Using the precision and recall metrics, the F-measure is then calculated as follows:

$$F\text{-}Measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

**PSNR**

Signal-to-noise ratio (PSNR), is a metric that measures how closely one image matches another. PSNR is based on the calculation of the mean square error (MSE) between the result image and the corresponding ground truth image as follows.

The Mean Square Error (MSE) calculates the average squared error per pixel of the estimator or result $I$ compared to the ground truth $I'$. MSE is expressed as follows:

$$MSE = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} \left(I(x,y) - I'(x,y)\right)^2}{MN}$$

where $M$ and $N$ are dimensions of the image. PSNR measures the lack of error or distortion introduced to the signal as compared to the ground truth. Hence, the higher the PSNR value, the more similar the two images are.

$$PSNR = 10 \log \left(\frac{C^2}{MSE}\right)$$

where $C$ is the difference between foreground and background (i.e., in a black and white image $C = 255$).

**Negative Rate Metric (NRM)**

The NRM looks at the mismatch per pixel of the prediction by calculating the ratio of false positives and false negatives retrieved then averaging these values:

$$NRM = \frac{NR_{FN} + NR_{FP}}{2}$$

where

$$NR_{FN} = \frac{N_{FN}}{N_{FN} + N_{TP}}$$

and

$$NR_{FP} = \frac{N_{FP}}{N_{FP} + N_{TN}}$$

$N_{TP}, N_{FP}, N_{TN}, N_{FN}$, stand for number of the True Positives, False Positives, True Negatives and False Negatives respectively. In other words, Lower NRM values imply better prediction.

### 4.2.2 Results

Table 4.1 shows the measurement metrics for all methods used in our comparison. Images 4.4 through 4.9 show the result of our algorithm in the second set of images used at DIBCO.

Table 4.1 shows that Algorithm I and II outperform the traditional methods of Otsu, Niblack, Sauvola and Kittler. The F-measure and NRM of the two methods presented in this thesis also stand very close to that of Lu's algorithm and the PSNR of Algorithm II supasses that of Lu's.

Table 4.2 also shows the performance of our methods against the unpublished state of the art algorithm [Lu et al., 2009a] in an image to image basis. Lu's method shows a sligthly better performance in most images except in image H01 where our Algorithm I outperforms Lu's.

| Methods | F-Measure (%) | PSNR | NRM($\times 10^{-2}$) |
|---|---|---|---|
| Lu | **90.82** | **20.12** | **3.68** |
| Proposed(Algorithm II) | 89.15 | 19.47 | 4.9 |
| Proposed(Algorithm I) | 87.09 | 18.85 | 6.8 |
| Otsu | 66.08 | 13.98 | 7.4 |
| Sauvola | 52.80 | 15.20 | 27.4 |
| Niblack | 29.37 | 5.90 | 17.1 |

Table 4.1: Comparison of thresholding algorithms

| Figures | Lu | Algorithm I | Algorithm II |
|---|---|---|---|
| H01 | 92.62 | **92.72** | 91.03 |
| H02 | **92.48** | 83.11 | 92.00 |
| H03 | **89.95** | 86.80 | 88.16 |
| H04 | **90.84** | 87.37 | 89.53 |
| H05 | **88.24** | 85.43 | 85.01 |
| Overall Avg | **90.82** | 87.09 | 89.15 |

Table 4.2: F-Measure Comparison by image

It is also worth to noting here that through visual inspection the results for Algorithm II look better than Lu's algorithm in some cases (See Figures 4.6 and 4.7) although this isn't reflected in the quantitative results. This is because using metrics like the ones used here might not correlate exactly with how a user could read the results or how good an OCR algorithm could process the text.

# Chapter 5

## Conclusion and Future Work

This thesis has introduced a set of novel techniques that combine a recursive version of the Otsu thresholding technique with selective bilateral filtering and improved versions of background normalization and despeckling to enhance text segmentation of historical handwritten text images.

There are many important conclusions that we can take from this thesis. One is that background subtraction through median filtering is a good way to isolate handwritten text strokes. This is because median filtering attenuates detail. By substracting the background approximation from the the original image, we end up mainly with the text.

Another important conclusion is that adapting the Otsu algorithm in a recursive way is more effective than its original counterpart for these types of images. The way we split the resulting pixels of the image several times helps obtain extra pixels that are usually discarded by the original version of the Otsu algorithm. An important part of our contribution with this thesis is the combination of the recursive way of performing the Otsu algorithm with the stopping criterion of the algorithm.

We can also conclude that the use of the bilateral filter to smooth noisy degraded documents is vital not only for the recursive Otsu technique to work but also in any other case where smoothing of text images is imperative. Bilateral filtering is a technique that has not been used to process text images very much. This technique has shown a lot of potential for processing this type of images. With this thesis, we try to point out that bilateral filtering could greatly help the preprocessing of text images.

## 5.1 Future Work

There is a lot more research that could be done in this area that would help achieve better results for text segmentation. One of the features that would help greatly any algorithm that tries to accomplish text segmentation is better background estimation. Currently, the background estimation in our method is done with a median filter or successively applied median filters. In contrast, Lu's method at DIBCO uses a least-squares fitting algorithm to approximate the background of the image. Although their process has proven to be very effective in estimating the background, it still introduces noise into the result, which they deal with by adding a post-processing step.

Currently, our algorithm works with predefined parameters that we have tuned based on empirical experiments. Although these parameters work well for most images as we have shown in our results, they might not work well with pictures that are very different from the ones we tested our algorithm with, such as images without any text at all or images with extremely low or high resolution. Also, we noticed that our Algorithm II fails on images in which black pixels are part of the background. Our algorithm assumes that black pixels are always part of the foreground, hence if we were to process an image with a black border Algorithm II would fail. For this reason, future work on the algorithm would need to be able to automatically identify special cases and tune its parameters, thus handling any type of image. One way to solve this is by using an approach suggested by [Lu et al., 2009b] where parameter tuning is replaced by user interaction.

In the algorithm proposed, we have used a couple of features to classify a pixel as either background or foreground. Some of the features used are the proximity of the pixels being processed and the similarity of their values being used in the bilateral filter. However, is it possible to add another feature to the algorithm to achieve a better classification? This question also opens the door to a deeper discussion of how the use of other features could lead to a better binarization process.

## References


L. Basavaraj and R. D Sudhaker Samuel. Offline handwritten character detection using image components. *International Conference on Computational Intelligence and Multimedia Applications*, pages 461–465, 2007.

J. Bernsen. Dynamic thresholding of gray level images. *International Conference on Pattern Recognition*, pages 1251–1255, 1986.

J. Canny. A computational approach to edge detection. *IEEE Transactions, Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Transactions on Graphics*, 26(3):103, 2007.

M. Cheriet, J.N. Said, and C.Y. Suen. A recursive thresholding technique for image segmentation. *IEEE Transactions on Image Processing*, 7(6):918–921, 1998.

B. Gatos, I. Pratikakis, and S. J. Perantonis. Adaptive degraded document image binarization. *Pattern Recognition*, 39(3):317–327, 2006.

B. Gatos, K. Ntirogiannis, and I. Pratikakis. Document image binarization contest. *International Conference on Document Analysis and Recognition*, pages 1375–1382, 2009.

M. Gupta, N. Jacobson, and E. Garcia. OCR binarization and image pre-processing for searching historical documents. *Pattern Recognition*, 40(2):389–397, 2007.

L. Hutchison. Fast registration of tabular document images using the Fourier-Mellin transform. Master's thesis, Brigham Young University, 2003.

J. Kapur, P.K. Sahoo, and A.K.C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision Graphics and Image Processing*, 29 (3):273–285, 1985.

J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recogntion*, 19(1):41–47, 1986.

S. Lu and C. L. Tan. Thresholding of badly illuminated document images through photometric correction. *ACM Symposium on Document Engineering*, pages 3–8, 2007.

S. Lu, B. Su, and C. L. Tan. Document image binarization using background estimation and stroke edges. *Unpublished*, 2009a.

Z. Lu, Z. Wu, and M. Brown. Iteractive degraded document binarization. In *Workshop on Applications of Computer Vision (WACV)*, 2009b.

W. Niblack. *An introduction to digital image processing*. Strandberg Publishing Company, Birkeroed, Denmark, Denmark, 1985.

O. Nina and W. Barrett. Thresholding of text documents using recursion of the Otsu algorithm. *Family History Technology Workshop*, 2007.

O. Nina and B. Morse. Interactive smoothing of handwritten text images using a bilateral filter. *Family History Technology Workshop*, 2009.

I. Oh. Document image binarization preserving stroke connectivity. *Pattern Recognition Letters*, 16(7):743–748, 1995.

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transaction on Systems, Man and Cybernetics*, 9:62–66, 1979.

F. Porikli. Constant time O(1) bilateral filtering. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

P.K. Sahoo and G. Arora. A thresholding method based on two-dimensional Renyi's entropy. *Pattern Recognition*, 37(6):1149–1161, 2004.

J. Sauvola and M. Pietikinen. Adaptive document image binarization. *Pattern Recognition*, 33:225–236, 2000.

M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168, 2004.

Z. Shi and V. Govindaraju. Historical document image enhancement using background light intensity normalization. *International Conference on Pattern Recognition*, pages 473–476, 2004.

S. M. Smith and J. M. Brady. SUSAN—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *International Conference on Computer Vision*, pages 839–846, 1998.

B. Weiss. Fast median and bilateral filtering. *ACM Transactions on Graphics*, 25(3):519–526, 2006.

J. M. White and G. D. Rohrer. Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM Journal of Research and Development*, 27:400–411, 1983.

C. Yan and G. Leedham. Decompose-threshold approach to handwriting extraction in degraded historical document images. *International Workshop on Frontiers in Handwriting Recognition*, pages 239–244, 2004.

Y.B. Yang and H. Yan. An adaptive logical method for binarization of degraded document images. *Pattern Recognition*, 33(5):787–807, 2000.

S.D. Yanowitz and A.M. Bruckstein. A new method for image segmentation. *International Conference on Pattern Recognition*, pages 270–275 vol.1, 1988.

Y. Yasuda, M. Dubois, and T.S. Huang. Data compression for check processing machines. *Proceedings of the IEEE*, 68(7):874–885, 1980.

J. Yen, F. Chang, and S. Chang. A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, 4(3):370–378, 1995.

Y. Zhu, C. Wang, and R. Dai. Document image binarization based on stroke enhancement. *International Conference on Pattern Recognition*, pages 955–958, 2006.