A Recursive Otsu Thresholding Method for Scanned Document Binarization

Oliver Nina, Bryan Morse, and William Barrett
Department of Computer Science
Brigham Young University

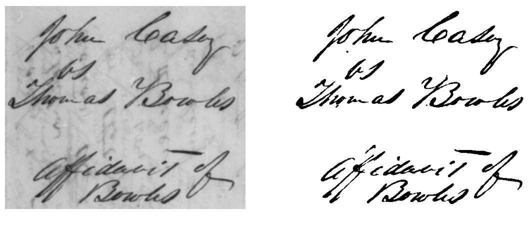
Abstract

The use of digital images of scanned handwritten historical documents has increased in recent years, especially with the online availability of large document collections. However, the sheer number of images in some of these collections makes them cumbersome to manually read and process, making the need for automated processing of increased importance. A key step in the recognition and retrieval of such documents is binarization*, the separation of document text from the page's background. Binarization of images of historical documents that have been affected by degradation or are otherwise of poor image quality is difficult and continues to be a topic of research in the field of image processing. This paper presents a novel approach to this problem, including two primary variations. One combines a recursive extension of Otsu thresholding and selective bilateral filtering to allow automatic binarization and segmentation of handwritten text images. The other also builds on the recursive Otsu method and adds improved background normalization and a post-processing step to the algorithm to make it more robust and to perform adequately even for images that present bleed-through artifacts. Our results show that these techniques segment the text in historical documents comparable to and in some cases better than many state-of-the-art approaches based on their performance as evaluated using the dataset from the recent ICDAR 2009 Document Image Binarization Contest.*

1. Introduction

The use of digital document images for research in genealogy and history has increased in recent years. These images contain important historical information and are often available on the Internet through sources such as genealogical websites, digital libraries, or personal websites. Historians and researchers have turned to these new collections of digital information because of their availability and low cost. Accessing these new digital resources requires less effort and increases access and retrieval when compared to using physical documents.

However, the sheer amount of historical digital images online can make research cumbersome and overwhelming. The Church of Jesus Christ of Latter Day Saints, a leader


(a) Original Image (b) Binarized Result
Figure 1. Historical document with a noisy background.

in digital document and genealogical image storage and retrieval, reported in 2006 having scanned 3,500 microfilms and processed 2.2 million images. For genealogical researchers to search through this massive amount of information by hand would be impractical.

One way to alleviate the work of going through countless digital document 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 or search engine to look up the information from the indexed images.

Using people to manually index digital text images is extremely time-consuming. 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 improve this speed by several factors. Such optical character recognition (OCR) has had much success for printed documents but little for handwritten ones, especially degraded documents.

One of the first and very important phases of the OCR process is *binarization*. According to Zhu [1], "Binarization is always a great challenge in all image process fields, especially in the process of document images whose binarization result can affect the OCR rate directly." In order to automatically index historical text images, we need first to separate the text from the rest of the image (background). Even if later processing is used to provide feedback to the binarization step, an initial binarization is still necessary. Fig. 1 shows an example of such binarization.

One of the simplest binarization techniques is *thresholding*. Global thresholding suffers from the need to find a sin-

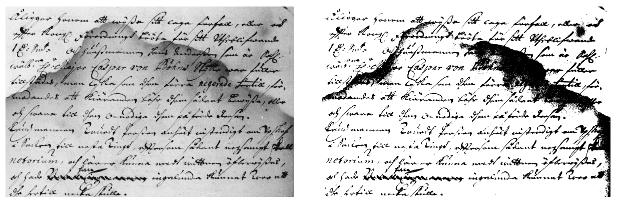

(a) Original Image (b) Binarized (Otsu)

Figure 2. Challenges of image binarization. Degradation, staining, and faint text all combine to make binarization difficult.

gle threshold value that will satisfactorily binarize the handwriting in a document. This becomes particularly problematic in older, historical documents where the writing varies in stroke intensity or where other sources of noise compete with those intensities. (See Fig. 2 for example.) On the other hand, locally adaptive thresholding techniques may be overly sensitive to noise.

In general, poor image quality, such as occurs in historical documents, often hinders the correct binarization of an image because of faded ink, staining, artifacts from the camera, blobs of ink, degradation of the document, bleed-through, other markings on the document, etc. Figs. 1 and 2 show typical examples of this problem. Thresholding methods typically work well only for images where the text and background are clearly defined.

This paper proposes and explores several novel techniques for (a) iterative background estimation using median filtering, (b) a recursive extension to the Otsu [2] thresholding algorithm for binarization, (c) a foreground/background selective bilateral filter [3] for noise removal, and (d) an improved method for background normalization and noise despeckling. 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 technique shown in [4] for dealing with bleed-through artifacts.

2. Related Work

There are many algorithms that attempt to separate foreground (text) from background in scanned text documents, but thresholding in one form or another remains one of the standard workhorses, and many more sophisticated approaches often have thresholding at the heart of the method. As noted in the introduction, thresholding algorithms are divided into two broad classes: global and adaptive.

Global thresholding algorithms are usually faster 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 because it is difficult for the thresholding algorithm to differentiate foreground text from other page darkening. This is most pronounced when faint strokes are similar to the background or where blobs and spots are of similar intensity to the text (Fig. 2). Popular global thresholding techniques include those proposed by [2, 5–8].

Adaptive thresholding attempts to address variations in local intensities throughout the image. This is similar to global thresholding, but a per-pixel threshold is computed based on a local window around each pixel. Thus, different threshold values are used for different parts of the image. Although this approach might work better or be more robust than global thresholding, it can also be very sensitive to image artifacts found in text images. Adaptive thresholding techniques include those from [9–13] and [14].

There are other non-thresholding techniques dedicated specifically to segmenting handwritten text images such as those in [15–20]. Although some of these techniques show promising results, the way degradation and other artifacts are handled still leaves room for improvement in the resulting binarization.

A survey [21] of thresholding techniques in 2004 ranked 40 different thresholding algorithms according to how well they performed in thresholding a set of sample images. In this survey the thresholding algorithms of [5] and [10] were ranked best for performing document binarization. In a 2007 comparison [18] of six different methods for thresholding of text in old documents for subsequent OCR it was concluded that "the classic Otsu method and Otsu-based methods perform best on average."

More recently, a binarization contest was organized in which 35 research groups participated with 43 different thresholding algorithms for type-written and hand-written documents [22]. This Document Image Binarization Contest (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. While this algorithm has not yet been published, the authors claim their results to be very similar to their previously published algorithm in [23].

3. Method: Algorithm 1

Our method adds to existing techniques to create a novel and effective way to binarize historical documents. Novel techniques include an iterative background estimation technique using median filtering, a recursive extension to the Otsu algorithm, a selective bilateral filter for noise removal, and a method for improved background normalization and noise despeckling. It proceeds with the following steps:

1. Background estimation (Section 3.1, Fig. 3b).
2. Background removal (Section 3.1, Fig. 3c).

3. Bilateral filtering (Section 3.2, Fig. 3c).
4. Recursive Otsu thresholding (Section 3.3, Fig. 3d).
5. Selective bilateral filtering using the image from Step 3 and the template from Step 4 (Section 3.4, Fig. 3e).
6. Recursive Otsu thresholding with hysteresis (Section 3.5, Fig. 3f).

This process is illustrated in Fig. 3 for a magnified fragment of an image and described in more detail in the remainder of this section.

3.1. Background Estimation and Removal

The first step is to preprocess the image to get rid of as much as possible the noise and degradation that may have been introduced.

We begin by using a technique first introduced by Hutchison [24] for approximating the background of text documents using a large median filter. This key idea of this method is that because text or other desired portions of the document typically form a small part of each local neighborhood, the median-filtered image generally looks like a smoothed version of the background alone. This assumes, of course, that the size of the window used for the median filter is much larger than the width of the handwriting strokes. Although stroke width in the handwritten documents we are considering tends to stay fairly constant when measured in real-world units, the resolution at which the document was scanned will affect the stroke width in pixels. We have found that for documents of the same scanning resolution a single window size tends to work well across multiple writers. For the results shown here, we use a window size of 21×21, which is about half the character size of the text. For general use this may have to be scaled using the known resolution of the document or by simply prompting the user to indicate a typical character size for the document with which they are working.

Once we obtain an approximation $\bar{I}$ for the background of the original image I, we remove the background by subtracting the values of the original image from it (clipping negative values to 0). For display, we further invert the result to maintain the original polarity of darker text on lighter background, yielding our estimate of the background-removed text $\hat{I}$:

$$\hat{I}(x,y) = 255 - \max([\bar{I}(x,y) - I(x,y)], 0)$$

This process is illustrated in Fig. 3, steps a–c.

3.2. Bilateral Filtering

While the described method for background estimation and removal tends to do a good job of removing intensity variations in the document background due to staining, aging, etc., it does not remove pixel-level noise from the image. In order to eliminate the remaining noise in the image,

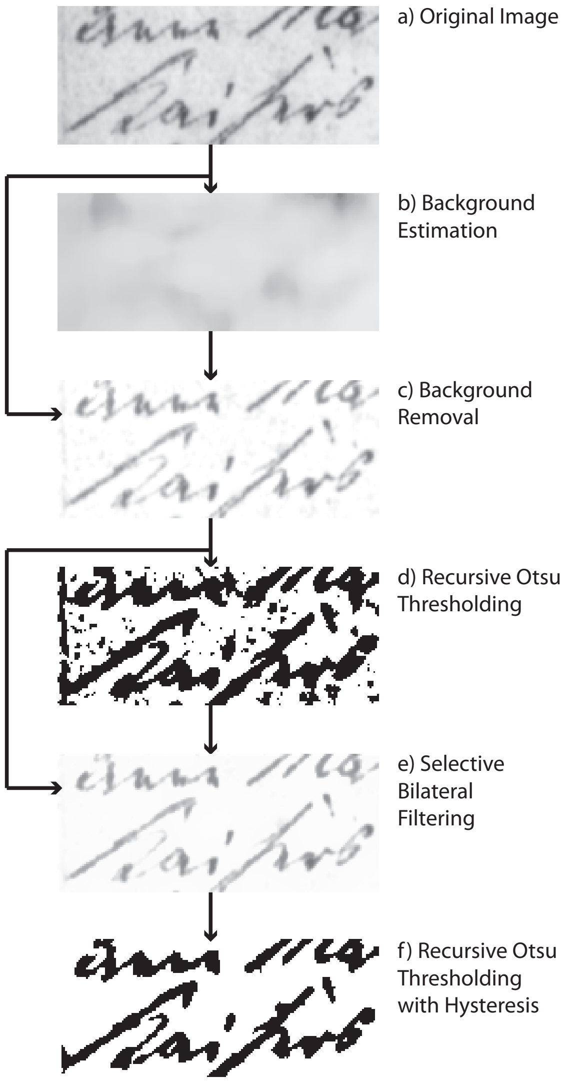
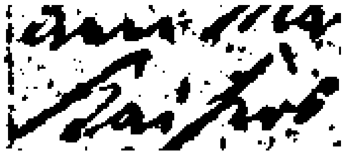

a) Original Image
b) Background Estimation
c) Background Removal
d) Recursive Otsu Thresholding
e) Selective Bilateral Filtering
f) Recursive Otsu Thresholding with Hysteresis

Figure 3. Overview of Algorithm 1. After estimating and removing the background, bilateral filtering is used to further remove noise in the image (a-c). Recursive Otsu thresholding is then used to create an initial binarization of the document (d). This initial estimate is then used to selectively bilateral filter the background-removed image (e), and recursive Otsu thresholding with hysteresis is used to create the final binarization (f).

we user a bilateral filter [3] with parameters $\sigma_s = 10$ and $\sigma_r = 2$. (As with the earlier median filter, the radius parameter σ_r may have to be adjusted depending on the scanning resolution but typically does not have to be adjusted between different writers.)

Because these parameters are fairly conservative, the effect of this filtering can be subtle visually. However, it can have a pronounced effect on the subsequent binarization. The intent here isn't to remove all noise but to reduce it enough that each pixel's value falls on the correct side of a well-chosen threshold. This effect is illustrated in Fig. 4, which shows the background-subtracted image from Fig. 3


(a) Background-subtracted image, thresholded at $T = 248$

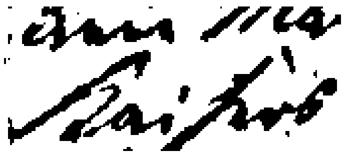

(b) Filtered background-subtracted image, thresholded at $T = 248$

Figure 4. Background-subtracted image from Fig. 3 manually thresholded at $T = 248$, both without (a) and with (b) prior bilateral filtering.

manually thresholded at $T = 248$ both without and with prior bilateral filtering. While this is not the final result, it illustrates the difference that bilateral filtering makes.

3.3. Recursive Otsu Thresholding

Otsu thresholding [2] is a well-known algorithm that determines a global threshold for an image by minimizing the within-class variance for the resulting classes (foreground pixels and background pixels). In practice, this is done by equivalently maximizing the between-class variance $\sigma_B^2(T)$for a given candidate threshold T:

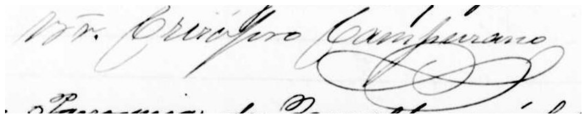
$$\sigma_B^2(T) = w_1(T)\, w_2 T\, [\mu_1(T) - \mu_2(T)]^2$$

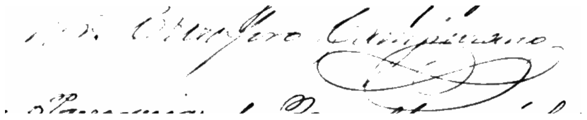
where w_i denotes the number of pixels in each class, μ_i denotes the mean of each class, and T is the value of the potential threshold. Although this must be computed for each candidate threshold T, each of the factors and terms in this computation can be updated efficiently using a histogram and recurrence relations.

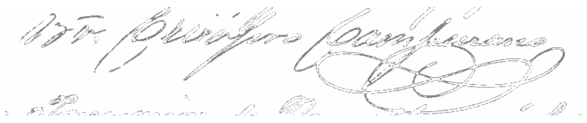
Although the Otsu technique has proven over the years to be an effective thresholding algorithm, it fails in instances where the foreground and background are similar, which occurs most often when the handwritten text is very faint. As a result, faint parts of the text are frequently dropped from the binarized result. To address this, we introduce *recursive Otsu thresholding* as a process for selecting progressively more of the faint text strokes.

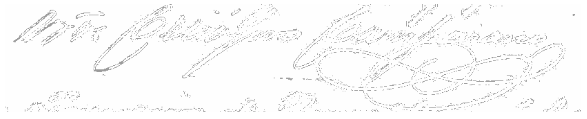
The key idea of recursive Otsu thresholding is to take the set of background pixels as determined by standard Otsu thresholding *and re-threshold these pixels alone* in order to extract more of the text. If we let $\text{Otsu}(a, b)$ denote the Otsu-selected threshold for all pixels in the image between values a and b, we may write recursive Otsu thresholding as

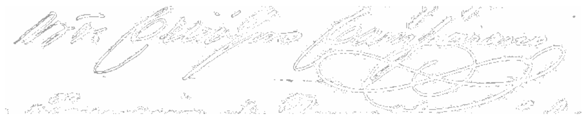
$$\begin{aligned} T_1 &= \text{Otsu}(0, 255) \\ T_k &= \text{Otsu}(T_{k-1}, 255) \end{aligned}$$


(a) Original with background removed


(b) First set of text pixels (S_1)


(c) Second set of text pixels (S_2)


(d) Third set of text pixels (S_3)


(e) Fourth set of text pixels (S_4)

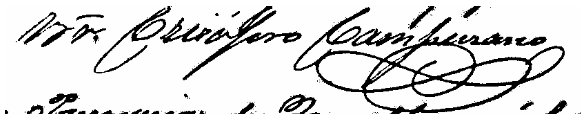

(f) Result ($S_1 + S_2 + S_3 + S_4$)

Figure 5. The Recursive Otsu thresholding algorithm

where T_k is the threshold determined from recursion k.

Of course, this could continue until the resulting set of foreground or background pixels is empty, which is clearly not what we want. We use the following three stopping criteria for the algorithm:

1. Stop if the number of pixels added by the current recursive step exceeds the number of pixels selected by the first threshold T_1, which indicates that large portions of the true background are now being included.
2. Stop if the change in threshold between successive recursive steps is smaller than some small value d_1, which indicates that the algorithm is straining at very subtle detail ($T_k - T_{k-1} < d_1$),
3. Stop if the change in threshold between successive recursive steps is larger than some larger value d_2, which indicates that the algorithm is simply subdividing the actual background pixels. ($T_k - T_{k-1} > d_2$).

Once any one of the stopping criteria are met, we stop the recursion and use the result from the preceding step. For our implementation, we use $d_1 = 2$ and $d_2 = 26$. These values appear to work for most images whose black-to-white range is fairly close to the full dynamic range $[0, 255]$.

The recursive Otsu thresholding process is illustrated in Fig. 5. The set of pixels S_1 selected by the first threshold T_1 is the same as for standard Otsu thresholding (Fig. 5b). The "rim" pixels added by the first recursion (S_2) between T_1 and T_2 include faint portions of the text that may have been dropped when selecting set S_1 (Fig. 5c). This process is repeated for sets S_3 and S_4, which include fainter and fainter "rim" detail (Figs. 5c–d). As seen in this example, the union of these sets (all pixels with value less than T_4, Fig. 5f) is a significantly better binarization than the initial Otsu-selected set S_1 alone.

3.4. Selective Bilateral Filtering

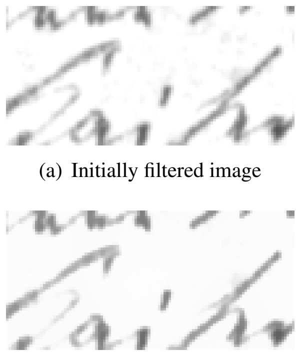
Although recursive Otsu thresholding improves the extraction of faint text, subtle remaining noise in the image can still introduce artifacts in the resulting binarization. However, at this point we can take advantage of the fact that we now have an initial approximation of the text and use it to selectively re-filter the earlier background-subtracted image, *by using the current best-estimate approximation of the text as a mask to blur foreground pixels only with other foreground pixels and background pixels only with other background pixels*.

Specifically, we apply a selective bilateral filter *only between estimated background pixels* using parameters $\sigma_s = 10$ and $\sigma_r = 3$. We apply another selective bilateral filter *only between the estimated foreground pixels* using parameters $\sigma_s = 2$ and $\sigma_r = 2$. Note the more conservative parameters used for the foreground filtering because the text tends to be narrow and have less variance than broader, higher-variance background areas.

The difference between blind and selective bilateral filtering can be seen in Fig. 6, which shows in greater detail the intermediate images from Fig. 3, steps 3 and 5.

3.5. Recursive Otsu Thresholding with Hysteresis

To produce the final binarization, we again employ the recursive Otsu algorithm, but this time we add a more selective element similar in spirit to thresholding with hysteresis [25]. Although the background estimation and removal, the recursive Otsu algorithm, and the selective bilateral filter attempt to avoid effects from noise or artifacts in the image, small faint speckles may still be present in the result. One way we deal with this is to only add pixels on each recursive step if they connect to pixels from the previous step. These pixels need not be immediately adjacent to already-identified text pixels but might be indirectly connected through other pixels within the currently thresholded set. If a candidate pixel is not connected either directly or indirectly to pixels marked as text by the previous step, 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 now be connected because we added new pixels from the third or fourth iterations. The combined noise-removing effects of the selective bilateral filtering and recursive Otsu thresholding with hysteresis steps can be seen in Fig. 3.


(a) Initially filtered image

(b) After text estimation and selective bilateral filtering

Figure 6. Selective bilateral filtering. Background noise is further removed by selectively re-filtering the estimated foreground and background separately. [This noise may best be seen in an electronic copy and is subtle when printed.]

4. Method: Algorithm 2

We have also experimented with an alternative framework, which again centers on recursive Otsu thresholding but with three changes that affect pre- and post-processing:

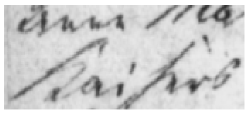
1. Improved background estimation based on iterated median filtering,
2. Alternative background contrast-compensation based on that proposed in [23].
3. Explicit despeckling using extensions to a technique proposed in [23].

These are used in conjunction with some of the steps from Algorithm 1, with recursive Otsu thresholding still as the key component:

1. Iterative background approximation (Section 4.1, Fig. 7).
2. Background removal (Section 3.1).
3. Contrast variation compensation (Section 4.2, Fig. 8).
4. Bilateral filtering (Section 3.2).
5. Recursive Otsu thresholding (Section 3.3).
6. Despeckling (Section 4.3).

4.1. Improved Background Estimation

We have found that iterated median filtering can provide even better background estimation than the single-pass ap-


(a) Original

(b) Background Approximation

(c) Iterative Background Approximation

Figure 7. Iterative Background Approximation

proach of [24]. Specifically, we use three passes of a median filter with a 21×21 window. The effect of this can be seen in Fig. 7. Our experience with this approach suggests that it can provide background estimation comparable to the polynomial-fitting approach of [4, 23] without the computational expense. (Note constant-time implementations of median filtering such as [26].)

4.2. Improved Contrast Compensation

Rather than removing the background through subtraction, [23] proposed a method for contrast compensation based on the ratio between the original image and the background estimate:

$$\hat{I} = \frac{CI}{\bar{I}}$$

where the constant C is the median intensity of the image and $\bar{I}$ again denotes the background estimate for the image.

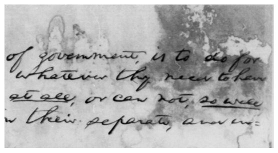
This process might be thought of as implicit background removal through local contrast normalization. We have found that rather than clipping values to the range $[0, 255]$, improved results can be obtained by scaling them to this range prior to thresholding. An example of this contrast compensation is found in Fig. 8.

4.3. Despeckling

Finally, we use a despeckling algorithm similar to that used in [23]. This algorithm consists of 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:

$$\text{Diff}(c) = \left|\bar{I}_c - I_c\right|$$

where I_c is the average intensity of the connected component in the original image and $\bar{I}_c$ is the average intensity of


(a) Original Image

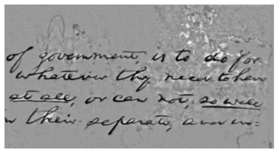

(b) Compensated Image

Figure 8. Result after applying image compensation.

the corresponding patch in the background estimated image. The idea here is that generally the difference in intensity between the real text and background is much higher than that of noise and background. A threshold is picked by building a histogram across all connected components c of $\text{Diff}(c)$ and applying the Otsu algorithm.

This approach works well by exploiting component statistics, but we add a second parameter to the comparison in order to make it even more robust. We also build a histogram for the size of every connected component and use the Otsu method to determine a *size* threshold in addition to the intensity difference threshold. We observed 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.

5. Results and Evaluation

We now compare our results to the methods identified by the surveys in [21] and [18] as well as to the (yet unpublished) method by Lu and Tan that won the recent DIBCO [22] contest.[1]

5.1. Qualitative Examples

Fig. 9 shows the results of our method compared to some of the better traditional thresholding methods (as identified by [21] and [18]) and to Lu's method). Notice that most of the methods fail to pick up the lighter strokes, while Algorithm 1 does a better job picking these up.

Fig. 10, whose original image is from the DIBCO dataset, shows an example in which both Algorithms 1 and 2 perform comparably to Lu's method, and all three clearly outperform traditional thresholding methods. Notice how in this example Lu's method incorrectly fills in the loops of various letters (the "a"s, "d"s, and "o"s) while Algorithms 1 and 2 better preserve the letter shapes.

[1]We would like to thank these authors for providing us with their results through personal correspondence.

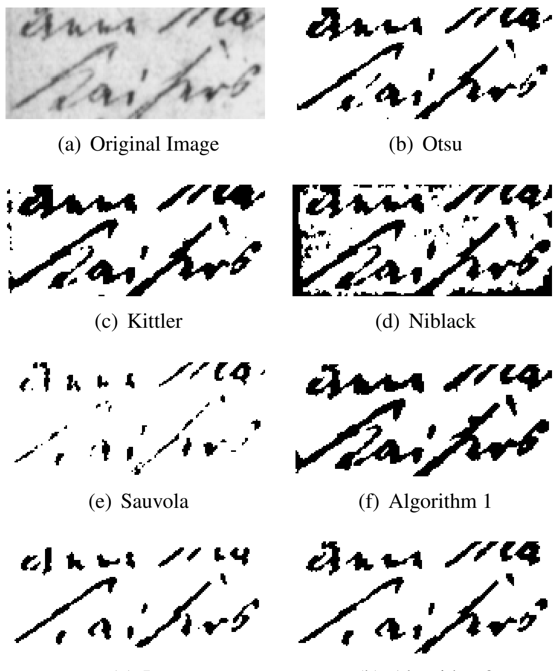


Figure 9. Comparison of thresholding methods. Notice that Algorithm 1 performs best on this test case.

5.2. Quantitative Results

We also compared our methods quantitatively using the metrics used for the DIBCO contest, including F-measure, peak signal-to-noise ratio (PSNR), and negative rate metric (NRM). For more details of these metrics and the procedures of this contest, see [22]. In addition to comparing our results to those published for DIBCO, we also calculated these metrics for a number of traditional thresholding methods for additional comparison.

The results for all three metrics, averaged across the contest's dataset, are found in Table 1. Notice that our proposed methods significantly outperform traditional methods (Otsu, Sauvola, and Niblack) and are comparable to the best-performing method reported for DIBCO (Lu's). Indeed, when inserted into the other rankings from DIBCO, these methods would rank #4 and #5 respectively out of the combined 45 entries.

Table 2 shows the F-measure results for each of the five handwritten images from the DIBCO dataset. Algorithm 1 performs best on H01, though all three methods are comparable. For the other four, Lu's method outperforms the others, but Algorithm 2's performance is usually comparable. (For some of the images with significant bleed-through artifacts, Algorithm 1 sometimes struggled specifically on those artifacts, while Algorithm 2 with its explicit despeckling step typically did better.) Full results for all of these images can be found in [27].

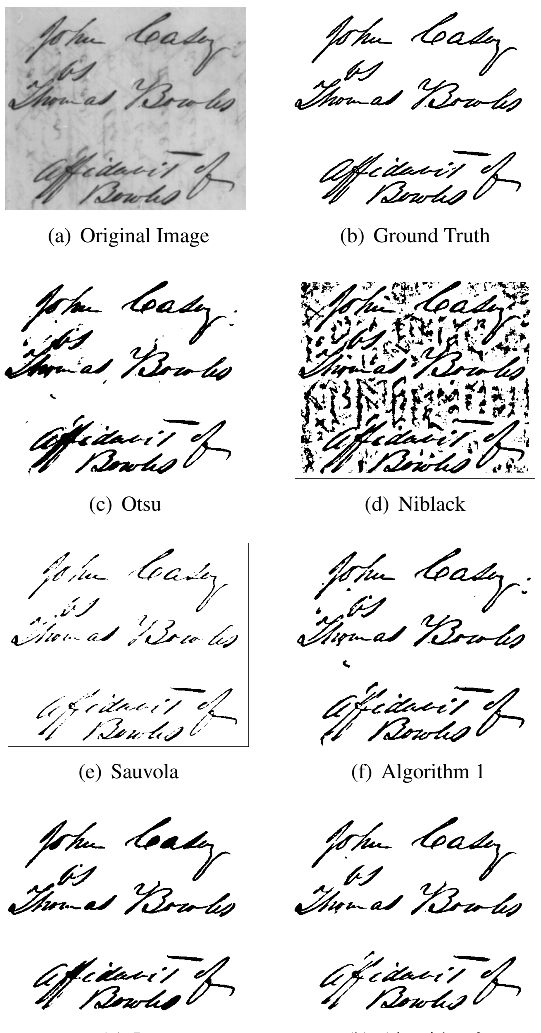


Figure 10. Image H03 from the DIBCO set with comparison.

Methods	F-Measure (%)	PSNR	NRM($\times 10^{-2}$)
Lu	90.82	20.12	3.68
Algorithm 2	89.15	19.47	4.90
Algorithm 1	87.09	18.85	6.80
Otsu	66.08	13.98	7.40
Sauvola	52.80	15.20	27.40
Niblack	29.37	5.90	17.10

Table 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 2. F-Measure Comparison by image

6. Discussion and Conclusion

Unlike machine-printed documents, handwritten historical documents are not inherently bitonal in their make-up. However, they are logically bitonal because the handwriting is logically connected and grouped with characters and words, making binarization the appropriate thing to do for segmentation and subsequent recognition.

In addition to strokes of varying intensity, these documents are also often encumbered with stains, marks and various sources of noise. As a result, binarization with a single threshold is not possible in many cases. Rather, "parsing" the image in intensity space and accumulating the results into a single binarization is necessary to successfully separate the foreground from the background. Multiple thresholded results computed from residual images can be composited into a single successful binarization.

The recursive Otsu algorithm presented in this paper, combined with selective bilateral filtering and background subtraction/compensation provides an effective method for intensity parsing of historical documents. Incrementally thresholded portions of the handwriting are then effectively combined into a single binary result that compares very favorably with existing thresholding algorithms.

Because not all segmentation errors are equivalent, additional analysis needs to be performed to more carefully quantify the accuracy of the this and other binarization algorithms. For example, faint strokes important to the recognition of a character may be more important than additional pixels that are part of a darker stroke that is already well captured. In addition, looped characters such as "o" and "a"s that are filled in may also be more important than other strokes because the filling-in may compromise the subsequent recognition of the character. Losing important stroke connections (or creating false ones) may also take precedence over other strokes that are easier to binarize.

The algorithms presented here perform qualitatively well in terms of capturing the important strokes. However, more task-relative quantitative analysis is needed. Perhaps the most important metric in assessing the accuracy of the binarization is found not in per-pixel retrieval metrics but in the subsequent recognition, a metric that is hard to come by at this stage of handwriting recognition technology.

Because of the need to access and understand both the source and content of handwritten historical documents, additional work in both segmentation and recognition of the associated text will remain an important area of research.

References

[1] Y. Zhu, C. Wang, and R. Dai, "Document image binarization based on stroke enhancement," in *International Conference on Pattern Recognition*, 2006, pp. 955–958.

[2] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Systems, Man and Cybernetics*, vol. 9, pp. 62–66, 1979.

[3] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *International Conf. on Computer Vision*, 1998, p. 839.

[4] S. Lu, B. Su, and C. L. Tan, "Document image binarization using background estimation and stroke edges," 2009, unpublished.

[5] J. Kittler and J. Illingworth, "Minimum error thresholding," *Pattern Recogntion*, vol. 19, no. 1, pp. 41–47, 1986.

[6] J. Kapur, P. Sahoo, and A. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision Graphics and Image Proc.*, vol. 29, no. 3, pp. 273–285, 1985.

[7] P. Sahoo and G. Arora, "A thresholding method based on two-dimensional Renyi's entropy," *Pattern Recognition*, vol. 37, no. 6, pp. 1149–1161, 2004.

[8] J. Yen, F. Chang, and S. Chang, "A new criterion for automatic multilevel thresholding," *IEEE Trans. Image Processing*, vol. 4, no. 3, pp. 370–378, 1995.

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

[10] J. Sauvola and M. Pietaksinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[11] J. M. White and G. D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction," *IBM J. of Res. and Dev.*, vol. 27, pp. 400–411, 1983.

[12] Y. Yasuda, M. Dubois, and T. Huang, "Data compression for check processing machines," *Proceedings of the IEEE*, vol. 68, no. 7, pp. 874–885, 1980.

[13] S. Yanowitz and A. Bruckstein, "A new method for image segmentation," in *International Conference on Pattern Recognition*, vol. 1, 1988, pp. 270–275.

[14] J. Bernsen, "Dynamic thresholding of gray level images," in *International Conference on Pattern Recognition*, 1986, pp. 1251–1255.

[15] Z. Shi and V. Govindaraju, "Historical document image enhancement using background light intensity normalization," in *International Conf. on Pattern Recognition (ICPR)*, 2004, pp. 473–476.

[16] I. Oh, "Document image binarization preserving stroke connectivity," *Pattern Recognition Letters*, vol. 16, no. 7, pp. 743–748, 1995.

[17] B. Gatos, I. Pratikakis, and S. J. Perantonis, "Adaptive degraded document image binarization," *Pattern Recognition*, vol. 39, no. 3, pp. 317–327, 2006.

[18] M. Gupta, N. Jacobson, and E. Garcia, "OCR binarization and image pre-processing for searching historical documents," *Pattern Recognition*, vol. 40, no. 2, pp. 389–397, 2007.

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

[20] Y. Yang and H. Yan, "An adaptive logical method for binarization of degraded document images," *Pattern Recognition*, vol. 33, no. 5, pp. 787–807, 2000.

[21] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 146–168, 2004.

[22] B. Gatos, K. Ntirogiannis, and I. Pratikakis, "Document image binarization contest," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2009, pp. 1375–1382.

[23] S. Lu and C. L. Tan, "Thresholding of badly illuminated document images through photometric correction," in *ACM Symposium on Document Engineering*, 2007, pp. 3–8.

[24] L. A. D. Hutchison and W. A. Barrett, "Fourier–Mellin registration of line-delineated tabular document images," *International J. of Document Analysis and Recognition*, vol. 8, no. 2, pp. 87–110, 2006.

[25] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8(6), pp. 679–698, 1986.

[26] S. Perreault and P. Hebert, "Median filtering in constant time," *IEEE Trans. Image Processing*, vol. 16, no. 9, pp. 2389–2394, Sept. 2007.

[27] O. A. Nina, "Text segmentation of historical degraded handwritten documents," Master's thesis, Brigham Young University, 2010.