# A Comparison of Binarization Methods for Historical Archive Documents

J. He, Q. D. M. Do*, A. C. Downton and J. H. Kim*

*Department of Electronic Systems Engineering, University of Essex, UK. Email: {jhe, acd}@essex.ac.uk*
**Division of Computer Science, KAIST, Kusung-Dong, Yusung-Gu, Daejon, Korea. Email: {quan, jkim}@ai.kaist.ac.kr*

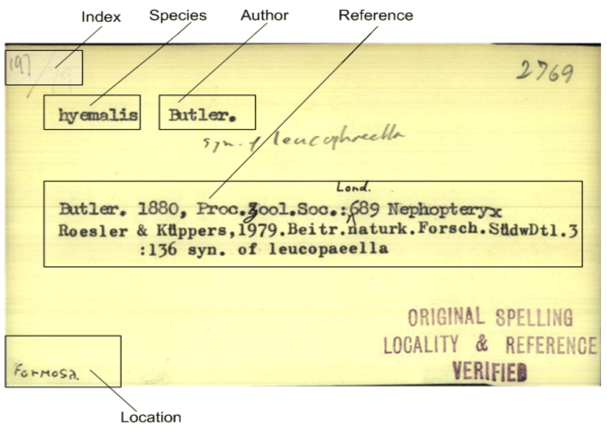
## Abstract

*This paper compares several alternative binarization algorithms for historical archive documents, by evaluating their effect on end-to-end word recognition performance in a complete archive document recognition system utilising a commercial OCR engine. The algorithms evaluated are: global thresholding; Niblack's and Sauvola's algorithms; adaptive versions of Niblack's and Sauvola's algorithms; and Niblack's and Sauvola's algorithms applied to background removed images. We found that, for our archive documents, Niblack's algorithm can achieve better performance than Sauvola's (which has been claimed as an evolution of Niblack's algorithm), and that it also achieved better performance than the internal binarization provided as part of the commercial OCR engine.*

## 1 Introduction

Digital archive construction from historic paper archives is a major document image analysis application of interest both for cultural and scientific purposes. For example, archives at the UK Natural History Museum (NHM) are recorded in card indexes, which contain bibliographical data and other information for one scientific name on each card laid out in a standardised format (Fig. 1) for each index.

We have designed a user-configurable archive document processing system [1] to extract text fields from NHM card images and then feed them into a commercial OCR engine for recognition. The quality of the images however has a significant impact on the OCR performance, since most historical archive document images are of poor quality due to aging and discoloured cards and ink fading. In pursuit of high quality text/background segmentation, we investigated two well-known binarization algorithms (Niblack's and Sauvola's), and also developed two algorithm variants, adaptive Niblack, and adaptive Sauvola. We also investigated an alternative color segmentation algorithm which clusters out the foreground texture from the background. Finally, we compared our 6 algorithm alternatives with the performance achieved using an optimised global threshold and with the internal binarization algorithm included in the commercial OCR engine.



2769

hyemalis Butler.

syn. of leucophaeella

Lond.
Butler. 1880, Proc.Zool.Soc.:689 Nephopteryx
Roesler & Küppers,1979.Beitr.naturk.Forsch.SüdwDtl.3
:136 syn. of leucopaeella

ORIGINAL SPELLING
LOCALITY & REFERENCE
VERIFIED

Formosa.

Figure 1. An index card with multiple hand print and handwriting annotations.

## 2 Binarization Methods

### 2.1 Global Thresholding

The global thresholding algorithm chooses a fixed intensity threshold value $T$ (from 0 to 255). If the intensity value of any pixel of an input image is more than $T$, the pixel is set to white otherwise it is black (see Fig. 2). If the source is a colour image, it first has to be converted to grey level using the standard conversion:

$$Grey = 0.3R + 0.59G + 0.11B \quad (1)$$

where $R$, $G$, and $B$ represent the colours red, green and blue respectively, with values from 0 to 255. All algorithms except colour segmentation (Section 2.6) are applied to grey level images.

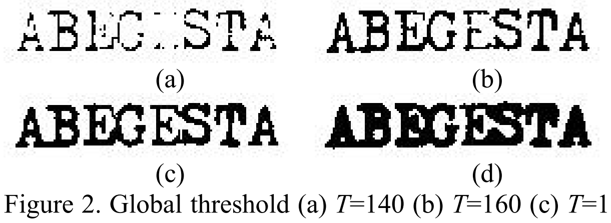


Figure 2. Global threshold (a) $T$=140 (b) $T$=160 (c) $T$=175 (d) $T$=200

Fig.2 (c) shows that global thresholding gives a promising result if a proper threshold can be selected.

However, it is difficult to achieve consistent quality with a fixed threshold while processing a batch of archive images, because both foreground and background colours vary significantly between images.

## 2.2 Niblack's Algorithm

Niblack's algorithm [2] calculates a pixelwise threshold by sliding a rectangular window over the grey-level image. The threshold $T$ is computed by using the mean $m$ and standard deviation $s$, of all the pixels in the window, and is denoted as:

$$T = m + k*s \qquad (2)$$

where $k$ is a constant, which determines how much of the total print object edge is retained, and has a value between 0 and 1. The value of $k$ and the size $SW$ of the sliding window define the quality of binarization. Fig.3 (a), (b) show binarization gives thick and unclear strokes with a small $k$ value, and slim and broken strokes with a large k value, while with a small $SW$ value noise is closer to texture as shown in Fig.3(c). Values for $SW$ and $k$ respectively of 25x25 pixels and 0.6 were heuristically optimal (see Fig 3 (d)).

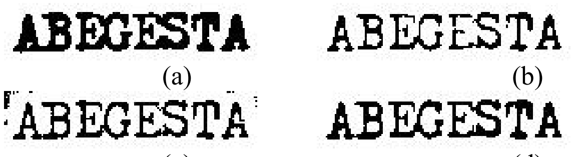


Figure 3. (a) with *SW* 25x25 and *k* 0.1 (b) with *SW* 25x25 and *k* 0.9 (c) with *SW* 11x11 and *k* 0.6 (d) with *SW* 25x25 and *k* 0.6

## 2.3 Adaptive Niblack's Algorithm

In archive document processing, it is difficult to identify suitable $SW$ and $k$ values for all images, as the character size of both frame and stroke may vary image by image. Improper choice of $SW$ and k values results in poor binarization as shown in fig. 3(a),(b),(c). We modified Niblack's algorithm to allow automatically chosen values for $k$ and $SW$, which we called adaptive Niblack's algorithm (see Fig. 4). This comprises five steps:

**Step 1:** Binarize the image $I$ (grey level) by using the global thresholding algorithm to roughly estimate texture area;

**Step 2:** Split the binarized image (texture area) into small blocks $B_i$s by identifying binary connected components which create rectangular areas of continuous connected black pixels (see Fig. 4) ;

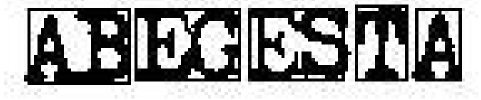


Figure 4. Rectangular blocks created by BCCS

**Step 3:** Calculate the ratio $r$ of number of black pixels against white for each block, and allocate a relevant $k$ (from 0 to 1) value for the block according to its $r$. The higher $r$, the higher the density of black pixels (for example due to thicker strokes), hence a larger $k$ is needed compared with thin strokes.

**Step 4:** Measure the height $h$ of each block. $SW$ is set as $h$x$h$

**Step 5:** Pad $B_i$ with edge pixels (say 3 pixels wide), to ensure enough surrounding pixels are included for $m$ and $s$ calculation. Then use Niblack's algorithm to binarize each block $B_i$ in $I$ using its local $SW$ and $k$ obtained from step 3 and 4, ignoring other background.

The most important strength of this method is that it avoids noise being generated in non-texture areas which can occur when Niblack's algorithm is applied to images with large non-texture areas (see Fig. 5).

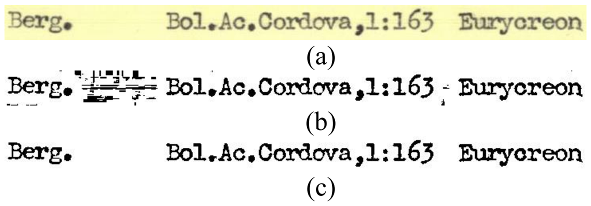

Berg. Bol.Ac.Cordova,1:163 Eurycreon
(a)
Berg. Bol.Ac.Cordova,1:163 Eurycreon
(b)
Berg. Bol.Ac.Cordova,1:163 Eurycreon
(c)

Figure 5. (a) original image (b) Niblack's algorithm (c) adaptive Niblack's algorithm

## 2.4 Sauvola's Algorithm

Sauvola's algorithm [3] is a modification of Niblack's which is claimed to give improved performance on documents in which the background contains light texture, big variations and uneven illumination. In this algorithm, a threshold is computed with the dynamic range of the standard deviation, $R$, using the equation:

$$T = m * (1 + k (s/R -1)) \qquad (3)$$

where $m$ and $s$ are again the mean and standard deviation of the whole window and $k$ is a fixed value.

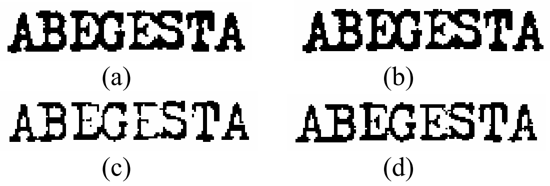


Figure 6: Results of Sauvola's algorithm (a) w=15x15, k=0.05; (b) w=15x15, k=0.12; (c) w=15x15, k=0.25; (d) w=9x9, k=0.12

In our experiments, we found that the value of $R$ has a very small effect on the quality while the values of $k$ and window size affect it significantly. The smaller the value of $k$, the thicker is the binarized stroke, and the more overlap exists between characters. A smaller window size will produce thinner strokes. An optimal combination of $k$

and *SW* will produce a good binary image. In our experiments, we set *R* as 128. Fig. 6 shows some binary images obtained by this algorithm and the effects of the values of *k* and window size on the binary results.

### 2.5 Adaptive Sauvola's Algorithm

With the original Sauvola's algorithm, a good choice of *k* and window size varies significantly from image to image. Parameters should be chosen adaptively for each image and even for each small window. This analysis leads to two proposed improvements to the algorithm. Firstly, the window should be big enough to preserve the thickness of strokes [1]; we chose half of the height of the input characters as the window size. Secondly, the *k* value should be chosen adaptively for each small window. This *k* should be representative of both global (overall image) and local (current window) image characteristics. To do this, we estimated *k'* and *k''*, the local and global values of *k* respectively, from equation (2) by estimating a threshold:

$$k = (T - m)/s \qquad (4)$$

where *T* is a threshold estimated by Niblack's algorithm since it is quite consistent over a range of values of *k* between 0.4 to 0.7 and the window size is big enough to cover a character. Then the value of *k* is defined by this equation:

$$k = (1-\alpha) * k' + \alpha * k'' \qquad (5)$$

where $\alpha$ is a global coefficient which defines how much the global information affects the threshold. The value of $\alpha$ is chosen from our experiments.

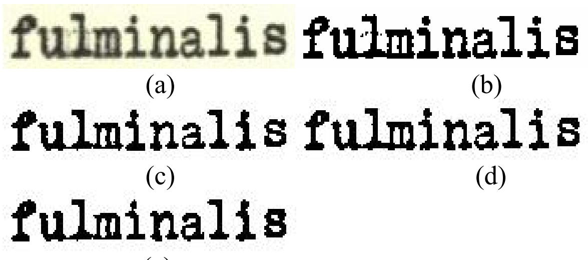


Figure 7: results of adaptive Sauvola's method (a) original images, (b) best Sauvola's result, (c) Adaptive Sauvola's with $\alpha$=0.1 (d) adaptive Niblack's (e) best Niblack's result (In c,d,e, noise in character "l" is removed perfectly);

### 2.6 Color Segmentation

A predefined color map (a collection of clusters) can be used to segment the foreground (useful text) from the background. The foreground is then binarized by using a binarization algorithm. Fig. 8 shows an example of segmented texts in black and red ink respectively. Details of our colour segmentation method have previously been described [4]. However, simply setting all foreground pixels to black in this case produces character images with very thick strokes (see Fig. 8, 1(c) and 2(c)), hence we also applied Niblack's algorithm to the residual foreground colour image (converted to greyscale) to erode the strokes to optimum thickness (as perceived by the OCR).

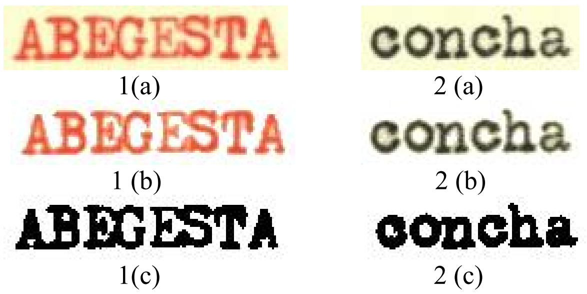


Figure 8: 1(a) and 2(a) are original colour text images; 1(b) and 2(b) are segmented foreground text in red and black ink respectively; 1(c) and 2 (c) are binarized foreground text.

## 3 Evaluation Methods

Evaluation was carried out on a set of 4435 species/genus images extracted randomly from the dataset of 27,578 Pyraloidea archive card images for which a full online database exists [5]. Each image contains only one typewritten word with height typically 23 to 25 pixels, which is either typed in black or red with grey/yellow background (due to age discolouration of the cards). The size of each word image is equal to its contained word size padded with an edge of 10 pixels as shown in Fig. 8 1(a), 2(a). These images were processed by the various binarization methods described in section 2 and fed into a commercial OCR system, Abbyy 6.0, for recognition. The OCR results were then compared with truth data from the NHM database, using word-level comparison, where any character error in a word is counted as an overall word error (i.e. in Table 2, only methods 3.3 and 3.7 give the correct output).

### 3.1 Default OCR – ABBYY

The Abbyy system has its own image pre-processing tool which converts colour images into binary and feeds them on for OCR. In other words, original colour images without any binarization can be directly fed to Abbyy for recognition, as well as pre-binarized images. The results obtained from this method are regarded as a standard reference for comparison with other methods.

### 3.2 Global Thresholding

A preliminary experiment was carried out on an independent small set of 335 images (which are similar to the test set of 4435 images). This determined that a global threshold of 165 achieved the highest recognition rate. Therefore, for the final evaluation, the testset of 4435 images were binarized with a global threshold of 165.

### 3.3 Niblack's Algorithm on Original Images

By carrying out experiments on the same training set of 335 images with *k* values varied from 0.1 to 0.9 in steps of 0.1 and *SW* from 9x9 to 27x27 pixels in steps of 2 pixels (total 7x10 combinations), the highest recognition rate was achieved when *SW* and *k* were respectively chosen as 25x25 pixels and 0.6. The optimum *SW* value corresponds to a "just-fit" character height. Using the optimal *SW* and *k*, the test set of 4435 images were processed using Niblack's algorithm, and then fed into Abbyy for recognition.

### 3.4 Niblack's Algorithm on Background Removed Images

The background was removed from the 4435 image testset using the colour segmentation algorithm described in Section 2.6. Using the same training method as Section 3.3 (training set images also had their background removed), we found values for *SW* and *k* of 25x25 pixels and 0.7 respectively achieved the highest recognition rate. The 4435 testset images were then processed using Niblack's algorithm with optimal *SW* and *k* values.

### 3.5 Sauvola's Algorithm on Original Image

The same method was applied as described in Section 3.3, except we replaced Niblack's algorithm with Sauvola's for binarization. Here also we used the same training method to find the optimal *SW* and *k*, which were 9x9 pixels and 0.09 respectively.

### 3.6 Sauvola's Algorithm on Background Removed Images

The same method was applied as described in Section 3.4, except we replaced Niblack's algorithm with Sauvola's for binarization. Here also we used the same training method to find the optimal *SW* and *k*, which were 13x13 and 0.15 respectively.

### 3.7 Adaptive Niblack's Algorithm

The 4435 image testset was processed using the adaptive Niblack's algorithm described in Section 2.3.

### 3.8 Adaptive Sauvola's Algorithm

The 4435 image testset was processed using the adaptive Sauvola's algorithm with fixed *SW* of 9x9 pixels, global *k'* 0.9 and local *k''* 0.1.

## 4. Results

The recognition results from the eight different evaluation methods are presented in Table 1. Table 2 gives an example of a binarized word image using each binarization method, and the corresponding OCR results. From analysis of the results, the following conclusions can be reached:

1. Unsurprisingly, global threshold binarization (3.1) has the poorest result, performing worse than any of the local thresholding algorithms.
2. Niblack's algorithm (3.3) with optimal parameters performs better than the Abbyy OCR binarization and all Sauvola-related algorithms.
3. Adaptive Niblack's algorithm (3.7) performs only marginally better than the standard Niblack algorithm (3.3). The reason the adaptive algorithm can't give any further improvement is because much of the evaluation data consists of connected characters as shown in Fig. 9. Therefore, the values of *SW* and *k* are mostly chosen based on multi-character binary connected components rather than single character, hence *SW* and *k* are sometimes no more optimal than with the standard Niblack algorithm.
4. Niblack's and Sauvola's algorithms working on background removed image (3.4 and 3.6) don't achieve any overall improvement in recognition, as the colour segmentation algorithm itself has an error rate of 2%, which reduces the end-to-end word recognition rate. However, even if we ignore these errors, the result is still worse than 3.3 and 3.5. This indicates that both Niblack's and Sauvola's algorithm count the background information especially around foreground texture as an important feature for local threshold calculation.
5. The adaptive Sauvola's algorithm (3.8) has better performance than the traditional one (3.5), but still performs less well than either of the Niblack algorithms (3.3 and 3.7) or the Abbyy internal binarization (3.1).

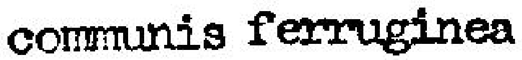

communis ferruginea

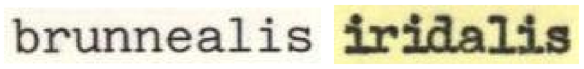
Fig. 9 Tightly connected characters

## 5. Discussion

Previous research [3] has claimed that Sauvola's algorithm has superior performance to Niblack's. Our experimental work with archive documents suggests this is not always true. One of the important advantages of Sauvola's algorithm is that it generates much less noise than Niblack's. For our evaluation data, both Niblack's and adaptive Niblack's algorithm were found to minimize noise to the same low level as Sauvola's algorithm. Under these circumstances, Niblack's algorithm achieves superior performance than Sauvola's when both must have fixed parameters but deal with varying background images, as is shown by comparison of the evaluations in 3.5 and 3.6 with those in 3.3 and 3.4. This phenomenon indicates that Sauvola's algorithm is more sensitive to the change of background than Niblack's, and is hence more difficult to tune to varying backgrounds over a complete dataset. The backgrounds of our evaluation data vary significantly image by image (see Fig. 10). In Sauvola's evaluation data [3], the background of images didn't appear to vary so much. Although our new adaptive version of Sauvola's algorithm can improve the performance, it is still worse than Niblack's. However, Sauvola's algorithm has other advantages: it costs less computationally, as a smaller *SW* is used than Niblack's algorithm. Our conclusion is that the choice of algorithm is really case-dependent.

Table 1. Evaluation results (4435 word images)

| Methods | Algorithm | Correct | Rate(%) |
|---|---|---|---|
| 3.1 | ABBYY | 3720 | 83.9 |
| 3.2 | Global | 3409 | 76.9 |
| 3.3 | Niblack | 3780 | 85.2 |
| 3.4 | Niblack-BG | 3701 | 83.4 |
| 3.5 | Sauvola | 3658 | 82.5 |
| 3.6 | Sauvola-BG | 3578 | 80.7 |
| 3.7 | Ad-Niblack | 3781 | 85.3 |
| 3.8 | Ad-Sauvola | 3701 | 83.4 |



(a) (b)

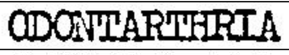
Fig.10 (a) light background (b) darker background

## 6 Conclusion

This paper has presented a comparison of several binarization algorithms by measuring their end-to-end word recognition performance on archive document word images. We drew several conclusions. First, for this archive word image dataset, Niblack's algorithm and our new adaptive Niblack's algorithm performed better than the internal binarization included in the commercial OCR engine. Furthermore, contrary to previously reported results, under some circumstances (e.g. archive documents with significantly varying background), Niblack's algorithm appears to perform better than Sauvola's,

Our proposed refinements to existing algorithms, adaptive Niblack's and adaptive Sauvola's, both achieved slightly better performance than the originals. As both Niblack's and Sauvola's algorithm need to utilise some information hidden in the background for local threshold calculation, removing the background from images before these two algorithms are applied doesn't bring any benefit in better binarization. Finally, as might be expected, all the local thresholding algorithms we tested have superior performance to a global thresholding algorithm.

Table 2. Example of Evaluation Result

| Method | Image | OCR Text |
|---|---|---|
| 3.1 | | ODCNTABTHRIA |
| 3.2 | | ODCNtABTHBIA |
| 3.3 | | **ODONTARTHRIA** |
| 3.4 | | GDCHTARTHBIA |
| 3.5 |  | ODQHTARTHRIA |
| 3.6 | | GDOmAKEBKEA |
| 3.7 | | **ODONTARTHRIA** |
| 3.8 | | ODQHTARTHRIA |

## References

[1] J.He, A. C.Downton, 'User-Assisted OCR Enhancement for Digital Archive Construction', submitted to ICDAR 2005.

[2] W.Niblack, An Introduction to Digital Image Processing. Prentice Hall, Englewood Cliffs, 1986.

[3] J.Sauvola,T,Seppanen, S.Haapakoski and M.Pietikainen, Adaptive Document Binarization, ICDAR'97 4th Int. Conf. On Document Analysis and Recognition, Ulm, Germany, August 1997, pp.147-152.

[4] J.He, A. C. Downton, 'Colour Map Classification for Archive Docuements', 6th International Workshop, Document Analysis Systems, DAS 2004, pp.241-251.

[5] G. Beccaloni, M. Scoble, G. Robinson and B. Pitkin, The Global Lepidoptera Names Index, http://www.nhm.ac.uk/entomology/lepindex.