Binarising Camera Images for OCR

Mauritius Seeger and Christopher Dance
Xerox Research Centre Europe
61 Regent Street, Cambridge
CB2 1AB, United Kingdom
seeger@xrce.xerox.com, dance@xrce.xerox.com

Abstract

In this paper we describe a new binarisation method designed specifically for OCR of low quality camera images: Background Surface Thresholding or BST. This method is robust to lighting variations and produces images with very little noise and consistent stroke width. BST computes a "surface" of background intensities at every point in the image and performs adaptive thresholding based on this result. The surface is estimated by identifying regions of low-resolution text and interpolating neighbouring background intensities into these regions. The final threshold is a combination of this surface and a global offset. According to our evaluation BST produces considerably fewer OCR errors than Niblack's local average method while also being more runtime efficient.

1. Introduction

Our research has been motivated by the convenience of using digital video cameras as opposed to conventional scanning devices. Cameras occupy little space on a user's desk, provide excellent feedback for alignment, capture instantly and allow documents to be scanned face up. However, since cameras acquire images under less constrained conditions than devices specifically designed for high-quality document capture, they can introduce severe image variations and degradations. This makes it especially hard to obtain reliable OCR result from these images.

Hence our aim was to design a binarisation algorithm specifically for OCR of camera images. In order to yield acceptable error rates in conjunction with off-the-shelf OCR software, this method must perform well in the presence of degradations such as low-resolution, lighting variations, blur, noise and compression artefacts. Specifically, it should work robustly with images at a resolution of 120-140 dpi in any lighting condition and with a minimal computational overhead. In engineering our method we therefore decided to measure and design for two criteria which are directly related to the usability of our camera scanning system: OCR error rates and runtime efficiency.

We have found that global thresholding methods typically designed for images acquired on flatbed scanners are unsuitable for camera images [3, 4, 8], mainly due to the presence of lighting variations and blur. Although local adaptive algorithms yield considerably better results [6],we have found that local average methods such as Niblack's method [2], which is often quoted as one of the best adaptive algorithms, tend to break down in the presence of large homogeneous areas and hence require post-processing [7]. Yanowitz and Bruckstein's method [9], which has been shown [6] to perform almost as well as Niblack's method, derives a threshold surface by extracting and interpolating from areas identified as character boundaries. However, it also requires a post processing step and is not particularly runtime efficient due to its iterative interpolation scheme. Furthermore, this method results in "noisy" threshold values since pixels lying on character boundaries have particularly variable grey values [5].

This paper presents a simple but novel adaptive threshold algorithm that achieves considerably better OCR performance than Niblack's method, while being more runtime efficient. This method is called background surface thresholding or BST. As the name suggests, this algorithm determines the background intensity at every pixel in order to derive a suitable threshold surface. In the following section we present an overview of BST. We then give a more detailed description of the algorithm, and finally present the results of a comparison between BST and Niblack's method.

2. BST Method

BST can conceptually be divided into the following parts:

- Labelling of text areas at low-resolution

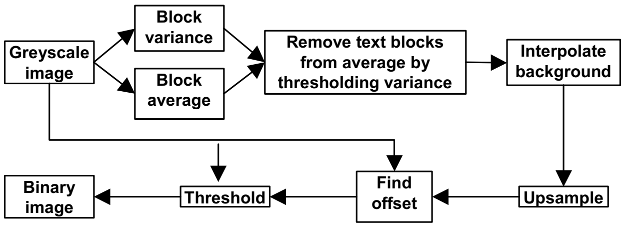

Figure 1. Outline of the BST algorithm

- Estimation of background intensity in text areas by interpolation
- Performing thresholding at the background plus some offset.

The initial segmentation between fore- and background relies on the assumption that page illumination is slowly varying. More specifically, that the scale of background variations is larger than that of foreground variations, i.e. transitions such as character edges. This is mostly observed in practice: the variance of grey levels in a small neighbourhood of pixels is larger in areas containing text, than in background regions. Hence using a measure of variance and a suitable threshold for this, our algorithm is able to distinguish between fore- and background.

In the next stage, the background in areas containing text is estimated by a linear interpolation of surrounding background intensities. To obtain good results, it is important to avoid filling foreground regions with incorrectly labelled background. Hence our priority was to conservatively label blocks as foreground even if they contain little or no text, since given the nature of the slowly varying illumination, background estimation is much more robust to mislabelling of this kind.

The image is binarised in a third stage. Pixels are labelled as fore- or background given a threshold which is the sum of the background surface and a global offset. This offset is proportional to the average distance between the fore- and background surface.

Figure 1 outlines the main computational steps of BST. Examples of intermediate results are shown in figure 2.

2.1. Text Labelling at Low Resolution

Based on the assumption that pixels in a window containing text have a higher variance than background regions, we have designed a variance test to label pixels as foreground. We compute a mean, $M(x,y)$, and variance, $V(x,y)$ image, as shown in figure 2, using adjacent w by w pixel blocks (where $w = 11$). An assumption here is that w is larger that the character stroke width, since we cannot expect a higher variance in homogeneous foreground regions, than

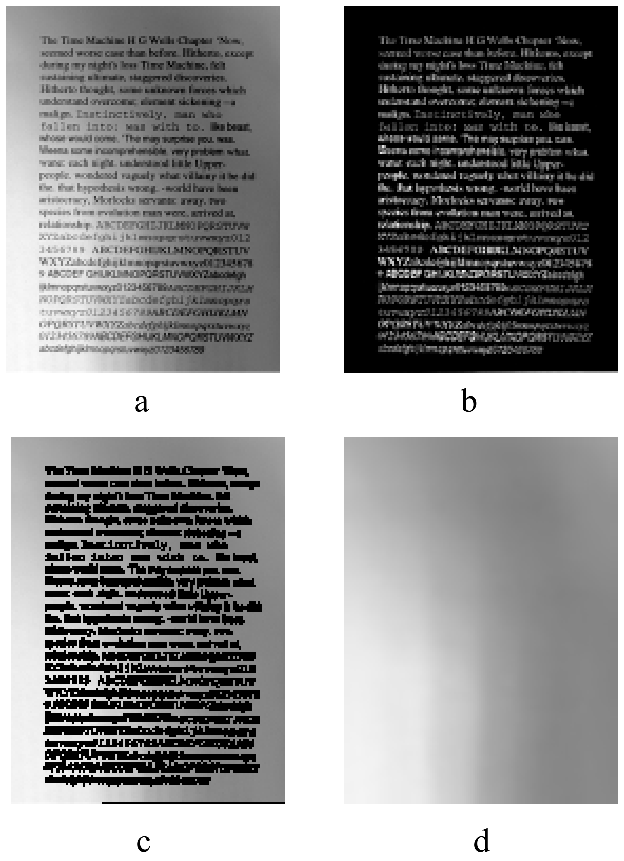

Figure 2. Intermediate results: (a) block average; (b) block variance (high variance shown by bright areas); (c) block average with high variance areas removed; (d) missing areas interpolated to yield a continuous background estimate. Original input: 640×480; size after pre-processing: 1440×1920; size of intermediate results: 131×175

in background areas. On the other hand, if the block width is too large spaces between text lines may not be detected correctly or background regions with rapid lighting variations may be classified as foreground. For 12 pt text at 120 dpi, good results are obtained for block sizes of between 7 and 19, and results are given here for $w = 11$.

Areas of text are initially identified by thresholding the variance image at the local average variance, $V_{mean}(x,y)$, computed using an r by r window of block variances, where $r = 23$. We have found this method to be more robust than attempting to detect text regions from average intensity information.

At this stage we also estimate a global measure of the variance due to noise in background regions. This step is crucial in making the method robust to images acquired un-

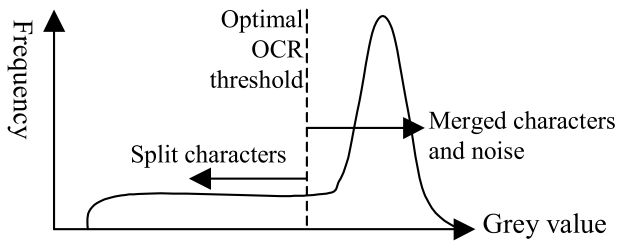

Figure 3. Idealised histogram of a camera image of text

der different conditions of lighting, contrast, camera noise and blur. Using an initial guess of $V_{noise} = 16$, we refine the background variance in a first pass using the following variance threshold surface:

$$T_v(x,y) = hV_{mean}(x,y) + V_{noise}, \tag{1}$$

where $h = 0.3$ yields the best OCR results according to our experiments, and the method remains robust in the range $h = 0.2 - 0.4$. By averaging the variance of all pixels for which $V(x,y) < T_v(x,y)$, we obtain a refined estimate of V_{noise}. In a second pass using equation 1 and the updated V_{noise}, we "remove" foreground pixels from the block-average image, $M(x,y)$, as shown in figure 2 c.

2.2. Interpolation

The background intensity for the "removed" pixels is estimated by interpolating from neighbouring pixels. This yields a continuous background surface, $B(x,y)$, as shown in figure 2 d. For efficiency we perform a 1-D linear interpolate between the closest two neighbouring points along rows and columns separately and combine the results later. In cases where missing points are not situated between two known background values, the interpolate is set equal to the nearest background value. Row and column results are combined by selecting values from the most accurate interpolate. This accuracy is measured using the distance in pixels between the closest know background value and the position of the interpolated pixel. For each pixel we select the interpolate that minimises this distance. The combined results are then smoothed by using a box blur of size 5 to reduce artefacts created during interpolation and then bilinearly upsampled by a factor of $w = 11$ to match the resolution of the original image.

2.3. Thresholding

In a final step we threshold the original grey level image, $I(x,y)$. Figure 3 illustrates an idealised histogram typical of a camera image of text in the absence of lighting variations. Usually, it is not bimodal for resolutions below 200 dpi, even though bimodality is assumed by many global thresholding methods [4]. The best choice of threshold is a tradeoff of the following risks:

- Split characters if the threshold is too low
- Merged characters and noise if the threshold is too high.

Typically we find that OCR engines such as ScanSoft TextBridge are most sensitive to split characters and background noise. Hence, as shown by figure 3, the best choice of threshold is somewhere just below the background peak. In typical camera images of documents, most of the lighting variation experienced is of a diffuse nature, and hence observed pixel values g might be described as a product between an incident illuminant E and a suitably normalised "underlying" image f, with additive sensor noise n,

$$g = Ef + n. \tag{2}$$

This formula suggests that the threshold be selected as some multiple of the background surface. However, our experience is that thresholding at an offset from the background gives better performance in practice. This might be because the threshold is largely influenced by the noise n whose variance does not change much with the illuminant. Use of an offset also appears more robust in situations where an unknown gamma correction has been applied to the image by a camera.

The threshold surface is a weighted sum of the previously determined background, $B(x,y)$ and a global offset, d, which is the average distance between the foreground and background:

$$d = \frac{\sum_{(x,y)\in S}(I(x,y) - B(x,y))}{|S|}, \tag{3}$$

where

$$S = \{(x,y)|B(x,y) > I(x,y)\}.$$

The threshold surface is then given by:

$$T(x,y) = B(x,y) - qd. \tag{4}$$

The factor, q, determines the thickness of character strokes and thus also the amount by which characters are either merged or split. We present results for $q = 1.5$, and find that these results remain stable in the range $q = 1.4 - 1.6$.

3. Pre-processing

We have been able to dramatically improve OCR results by pre-processing images before binarisation [5]. Images

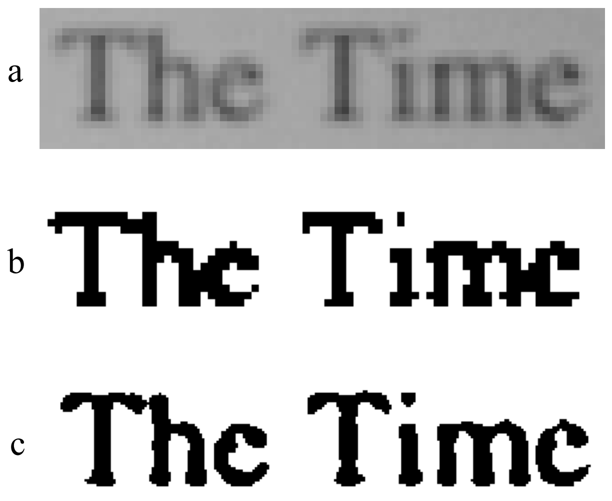

Figure 4. Quality improvement of text images gained by pre-processing (a) original image (b) BST without pre-processing (c) BST after pre-processing

are deblurred using a simple sharpening or high frequency boost filter [1] and upsampled bicubically by a factor of three. This enhances the high frequencies and allows what we call binary super-resolution [5]: trading of grey scale intensity resolution for spatial resolution. Figure 4 illustrates the advantage gained from this type of pre-processing.

4. Results

We have compared the performance of BST with Niblack's local average thresholding method [2]. The Niblack method operates on the following threshold surface:

$$T(x,y) = M(x,y) + k\sqrt{V(x,y)}, \tag{5}$$

where $M(x,y)$ and $V(x,y)$ are the local mean and variance respectively computed using a moving window of size w.

A previous comparison of binarisation methods by Trier and Jain [6] concluded that Niblack is the best performer when the goal is character recognition. However, as shown in figure 5, we found that Niblack produces noisy results when used with the recommended window size of $w = 15$ ($w = 45$ for our 3x up-sampled images) and inconsistent stroke width when used with larger windows, causing some characters to merge and others to split. More importantly, an evaluation of OCR performance shows that Niblack binarised images produce significantly higher character error rates. We have also found that the runtime of an efficient implementation of Niblack is roughly twice that of BST. The runtime excluding pre-processing on a 700MHz PC is 0.5 seconds for BST v.s. 1.25 seconds for Niblack, when binarising a 1440$\times$1920 image. Pre-processing (3$\times$ upsampling and sharpening of a 640$\times$480 image) can be completed in 0.5 seconds.

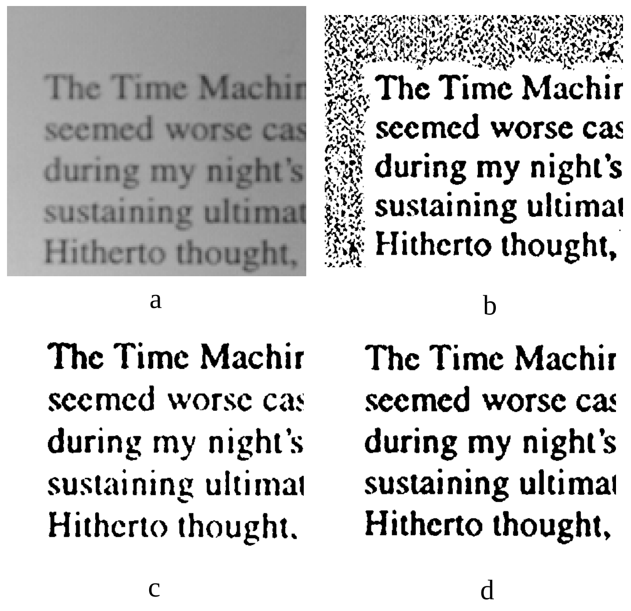

Figure 5. BST vs. Niblack (a) Original grey level image of 12 pt Times New Roman at 120 dpi; (b) Niblack *w*=45, *k*=-0.4; (c) *w*=200, *k*=-1; (d) BST

We compared the OCR performance of BST and Niblack using 17 images of A6 portions of magazine articles, newspaper articles and office documents, with 10-12 pt text acquired with a Philips Vesta Pro video-conferencing camera at 110-130 dpi. ScanSoft TextBridge was employed for OCR and the character error rates were computed as the Levenstein (string edit) distance between the output and the manually derived ground truth.

To achieve a fair comparison, we fine-tuned the performance of Niblack's method by choosing parameters w and k and the parameters of our pre-processing step to minimise the average OCR error rate. As shown in figure 6, Niblack performs significantly better with a large window and in the best case achieves an average character error rate of 3.1% compared to 2.3% for BST. The error rates observed were highly variable from image to image, hence the significance of the average error rate is questionable here. Indeed, in two cases the Niblack method produced more than 10% errors. Nonetheless for all but one of the images, the error rate for BST was less than that for the Niblack method.

Error rates with Niblack were especially high for small windows ($w < 200$) due to the large amount of noise in background regions. For larger windows ($w > 200$)

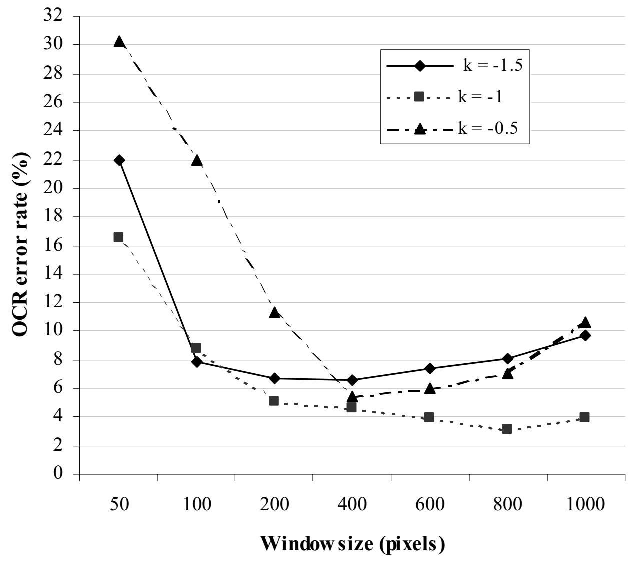

Figure 6. Character error rates achieved with Niblack's method for different window sizes *w*, and variance gains *k*. Best performance: 3.1% for *w*=800, *k*=-1

the noise is reduced, but Niblack becomes less robust to lighting variations, since it then effectively behaves like a global threshold, and also less robust to large changes in the amount of foreground in the window.

5. Conclusion

Our comparison suggest that BST performs better binarisation of camera images for OCR than Niblack's method. In addition, the BST implementation can be more runtime efficient. We feel that the poor OCR performance of Niblack is in particular due to noisy background regions produced when using small windows and inconsistent stroke width when using larger windows. Since BST uses a threshold that is globally offset from the background, it is much less susceptible to misclassification of large homogeneous regions. Also, since threshold levels in BST are not locally related to neighbouring features it produces characters with more consistent stroke width.

References

[1] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1989.

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

[3] L. O'Gorman. Binarization and multithresholding of document images using connectivity. *CVGIP: Graphical Models and Image Processing*, 56(6):494–506, 1994.

[4] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41:223–260, 1988.

[5] M. J. Taylor and C. R. Dance. Enhancement of document images from cameras. In *SPIE Conference on Document Recognition V*, volume 3305, pages 230–241. SPIE, September 1998.

[6] Ø. D. Trier and A. K. Jain. Goal-directed evaluation of binarization methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1191–1201, 1995.

[7] J. S. Valverde and R. Grigat. Optimal binarisation of technical document images. In *ICIP 2000 Proceedings*, pages 985–988. IEEE, September 2000.

[8] H. Yan. Unified formulation of a class of image thresholding techniques. *Pattern Recognition*, 29(12):2025–2032, 1996.

[9] S. D. Yanowitz and A. M. Bruckstein. A new method for image segmentation. *Computer Vision, Graphics and Image Processing*, 46(1):82–95, 1989.