Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition 39 (2006) 89–101

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Efficient computation of adaptive threshold surfaces for image binarization

Ilya Blayvas*, Alfred Bruckstein, Ron Kimmel

CS Department, Technion, Haifa 32000, Israel

Received 30 December 2004; received in revised form 11 August 2005; accepted 11 August 2005

Abstract

The problem of binarization of gray level images, acquired under non-uniform illumination is reconsidered. Yanowitz and Bruckstein proposed to use for image binarization an adaptive threshold surface, determined by interpolation of the image gray levels at points where the image gradient is high. The rationale is that high image gradient indicates probable object edges, and there the image values are between the object and the background gray levels. The threshold surface was determined by successive over-relaxation as the solution of the Laplace equation. This work proposes a different method to determine an adaptive threshold surface. In this new method, inspired by multiresolution approximation, the threshold surface is constructed with considerably lower computational complexity and is smooth, yielding faster image binarizations and often better noise robustness.

MSC: 68T45

Keywords: Computer vision; Image binarization; Threshold surface; Bernsen; Eikvil–Taxt–Moen; Niblack; Yanowitz–Bruckstein

1. Introduction

Let us consider the problem of separating objects from their background in a gray level image $I(x, y)$, where objects appear lighter (or darker) than the background. This can be done by constructing a threshold surface $T(x, y)$, and constructing a binarized image $B(x, y)$ by comparing the value of the image $I(x, y)$ with $T(x, y)$ at every pixel, via

$$B(x, y) = \begin{cases} 1 & \text{if } I(x, y) > T(x, y), \\ 0 & \text{if } I(x, y) \leqslant T(x, y). \end{cases} \tag{1}$$

It is clear that a fixed value of the threshold surface $T(x, y) = const$ cannot yield satisfactory binarization results for images obtained under non-uniform illumination and/or with a non-uniform background.

Chow and Kaneko in Ref. [1] were among the first researchers to suggest using adaptive threshold surfaces for binarization. In their method the image was divided into overlapping cells, and sub-histograms of gray levels in each cell were calculated. Sub-histograms judged to be bimodal were used to determine local threshold values for the corresponding cell centers, and the local thresholds were interpolated over the entire image to yield a threshold surface $T(x, y)$. This was certainly an improvement over fixed thresholding, since this method utilized some local information. However, the local information was implicitly blurred to the size of the cell, and this, obviously, could not be decreased too much.

Yanowitz and Bruckstein made a step forward in Ref. [2] by suggesting to construct a threshold surface by interpolating the image gray levels at points where the image gradient is high. Indeed, high image gradients indicate probable object edges, where the image gray levels are between the object and the background levels. The threshold surface was required to interpolate the image gray levels at all support points and to satisfy the Laplace equation at non-edge pixels. The surface was determined by a successive over-relaxation method (SOR) [2,3].

* Corresponding author. Tel.: +972 545 404 938; fax: +972 482 939 00.
E-mail address: blayvas@cs.technion.ac.il (I. Blayvas).

Trier and Taxt conducted a performance evaluation of 15 binarization methods by comparing the performance of OCR system with respective binarization method as the first step [4]. The Yanowitz–Bruckstein (YB) method produced the best results with the Trier–Taxt method just slightly behind. After the addition of a ghost-elimination step from Yanowitz and Bruckstein method, the methods of Niblack [5], Eikvil–Taxt–Moen [6] and Bernsen [7] performed slightly better.

As will be shown later, the last three methods are not scale-invariant, and their performance is optimal only for some specific object sizes or requires parameter tuning. The Yanowitz–Bruckstein method is scale invariant, however the computational complexity of successive over-relaxation method is expensive: $O(N^3)$ for an $N \times N$ image and the resulting binarization process is slow, especially for large images. Furthermore, the threshold surface tends to have sharp extremum at the support points, and this can degrade the binarization performance.

We here follow the approach of Yanowitz and Bruckstein and use image values at the support high gradient points to construct a threshold surface. However, we define a new threshold surface via a method inspired by multiresolution representation [8]. The new threshold surface is constructed as a sum of functions, formed by scaling and shifting of a given original function. This new threshold surface can be stored in two ways: as an array of coefficients a_{ljk}, or as a conventional threshold surface $T(x, y)$ which is obtained as a sum of scaled and shifted source functions, multiplied by appropriate coefficients a_{ljk}.

The threshold surface coefficients a_{ljk} are determined in $O(P \log(N))$ time, where P is the number of support points and N^2 is the image size. These coefficients can then be used to construct the threshold surface $T(x, y)$ over the entire image area N^2 in $O(N^2 \log(N))$ time or to construct the threshold surface over smaller region of the image of M^2 size in only $O(M^2 \log(N))$ time. Furthermore, the adaptive threshold surface can be made smooth over all the image domain.

The rest of this paper is organized as follows: Section 2 reviews the best performing methods according to Trier and Taxt evaluation [4], Niblack [5], Eikvil–Taxt–Moen [6], Bernstein [7], and Yanowitz–Bruckstein [2]. Section 3 describes a proposed new method to construct a threshold surface. Section 4 describes the implementation of the surface computation. Section 5 presents some experimental results, comparing the speed and binarization performance of the proposed method with the methods of Niblack and Yanowitz–Bruckstein. Finally Section 6 summarizes this work with some concluding remarks.

2. Review of binarization methods

2.1. Niblack's method

The idea of this method is to set the threshold at each pixel, based on the local mean and local standard deviation. The threshold at pixel (x, y) is calculated as

$$T(x, y) = m(x, y) + k \cdot s(x, y), \tag{2}$$

where $m(x, y)$ and $s(x, y)$ are the sample mean and standard deviation values, respectively, in a local neighborhood of (x, y). The size of the neighborhood should be small enough to reflect the local illumination level and large enough to include both objects and the background. Trier and Taxt recommend to take 15×15 neighborhood and $k = -0.2$.

2.2. Eikvil–Taxt–Moen's method

The pixels inside a small window S are thresholded on the basis of clustering of the pixels inside a larger concentric window L, S and L are sliding across the image in steps, equal to the size of S [4,6]. For all the pixels inside L, Otsu's threshold T [9] is calculated to divide the pixels into two classes. If the two estimated class means $\hat{\mu}_1$ and $\hat{\mu}_2$ are further apart than a pre-defined limit l,

$$\|\hat{\mu}_1 - \hat{\mu}_2\| \geqslant \ell, \tag{3}$$

then the pixels inside S are binarized using the threshold T. Otherwise, all the pixels inside S are prescribed to the class with the closest updated mean value. Trier and Taxt recommend $S = 3 \times 3$, $L = 15 \times 15$ and $\ell = 15$.

2.3. Bernsen's method

For each pixel (x, y), the threshold $T(x, y) = (Z_{low} + Z_{high})/2$ is used, where Z_{low} and Z_{high} are the lowest and highest gray level pixel values in a square $r \times r$ neighborhood centered at (x, y). If the contrast measure $C(x, y) = Z_{high} - Z_{low} < \ell$, then the neighborhood consists of only one class, that is assumed to be a background. Trier and Taxt recommend $r = 15$ and $\ell = 15$.

2.4. Yanowitz–Bruckstein's method

The essential steps YB binarization method [2] are the following:

(1) Find the *support points* $\{p_i\}$ of the image $I(x, y)$, where the image gradient is higher than some threshold value G_{th},

$$p_i = \{x_i, y_i\} : |\nabla I(x_i, y_i)| > G_{th}. \tag{4}$$

(2) Find the threshold surface $T(x, y)$ that equals to the image value at the support points and satisfies the Laplace equation at the rest of the image points:

$$\begin{aligned} &T(p_i) = I(p_i), \\ &\nabla^2 T(x, y) = 0 \quad \text{if } \{x, y\} \in \Omega \backslash \{p_i\}. \end{aligned} \tag{5}$$

Here Ω is the set of all the image points. The solution of Eq. (5) is found by a relaxation method.

(3) Determine the binarized image $B(x, y)$ according to Eq. (1).

These three steps are a simplification of the original method, made in order to discuss the essential steps without being lost in the details. The original method also included the following steps. A smoothing of the image before Step 1. The one-dimensional relaxation along the image boundary between the Steps 1 and 2 in order to use the obtained values as the Dirichlet boundary conditions for Step 2. Discarding of 'ghost' objects after Step 3, determined as the objects in the binarized image with relatively small gradients along the edge.

The smoothing of original image and discarding of ghost objects were omitted here, while the one-dimensional relaxation along the boundary and use of the result as Dirichlet boundary condition was substituted by the use of Neumann boundary conditions in Step 3.

The SOR starts with an approximate solution $t(x, y)$, and numerical iterations take it to the unique solution $T(x, y)$ of the Laplace equation [2].

2.5. Analysis of the binarization methods

In order to make a goal-oriented evaluation of the binarization methods Trier and Taxt built an experimental character recognition module. The binarization methods were applied to a hand-written hydrographic maps. Elliptic Fourier descriptors were extracted from the contour curve of the figures to form 12-dimensional features. Then the extracted features were fed into the quadratic classifier [10], assuming multivariate Gaussian distributions for each of the ten digit classes.

According to the evaluation by Trier and Taxt, the modified methods of Niblack, Eikvil–Taxt–Moen, Bernsen, and the Yanowitz–Bruckstein's method were ranked, respectively, to places $1, 2, 3$ and 4. Obviously, this evaluation procedure could serve a good indicator for the performance of the binarization methods not only for the applications of recognition of hydrographic maps but also for other character recognition applications. However, the authors note that the generalization of the results to other application domains is not straightforward.

In the following paragraphs, we show that the methods of Niblack, Eikvil–Taxt–Moen, and Bernsen are scale dependent, and will not work properly if the object sizes or the scale of the illumination uniformity vary significantly along the image. The threshold surface, constructed in the Yanowitz–Bruckstein method does not have explicit scale dependency. However, we shall show that the properties of this surface shade a doubt on its optimality for image binarization.

In the Niblack's method, (2) $T(x, y)=m(x, y)+k\cdot s(x, y)$ defines the threshold T inside the square of a fixed size, typically 15×15. Every such region is separated into an object and a background. Consider a completely white region, say, at the blank region of the page. The pixels will have some mean m and standard deviation s. Whatever the intensity distribution of the pixels, some pixels will necessarily fall below the threshold T defined by Eq. (2). Therefore, in every image region of size 15×15 some pixels will be classified as objects and some as a background. This will be a misclassification for the images having regions of blank or objects of size 15×15 or larger. The recommended value $k = -0.2$ can be considered as an incorporation of the prior knowledge and reflects the fact that more bright background than the dark objects is expected.

In the Eikvil–Taxt–Moen's method the problem of single-class regions is treated somewhat better, since the condition $\|\hat{\mu}_1 - \hat{\mu}_2\| \geqslant \ell$, in Eq. (3) detects the cases of a single class in a region.

However, the existence of a 'magic' size $L = 15 \times 15$ makes the method scale dependent. Obviously, this scale is about the best compromise, at least for the case studied by Trier and Taxt, however it can be too small for cases when the objects are large and too large for the cases when the illumination changes too fast along the image.

Bernsen's method is also scale dependent, as can be shown by applying similar arguments.

In the Yanowitz–Bruckstein's method there is no explicit scale factor, and therefore this method is more appropriate for the general cases. However, the price of constructing the threshold surface that depends on the entire image is high computational complexity. Really, every method that is limited to a fixed square size will scale linearly with the size of the image $t = O(N^2)$. In the relaxation solution each iteration requires $O(N^2)$ operations for N^2 grid points and there should be $O(N)$ iterations to converge to a solution, therefore the method complexity is $O(N^3)$ [2]. The solution of Eq. (5) can be found in just a $O(N^2)$ time using multigrid methods [11]. However, it will become clear from the following paragraph that not only the speed of computation but also the properties of the threshold surface can be improved.

The general form of the solution of Eq. (5) in the continuum limit is

$$\phi(x, y)=\psi(x, y)-\sum_{i=1}^{P} q_i \log\left(\sqrt{(x-x_i)^2+(y-y_i)^2}\right), \tag{6}$$

where $\psi(x, y)$ is smooth and bounded function [12]. This solution has singularities at the support points. In the case of a problem discretized on a finite grid, the iterative solution of Eq. (5) will be finite, yet, it will have sharp extrema at the support points. These sharp extrema and especially the hanging 'valleys' between them can cause the unwanted 'ghost' objects in the binarized image. These

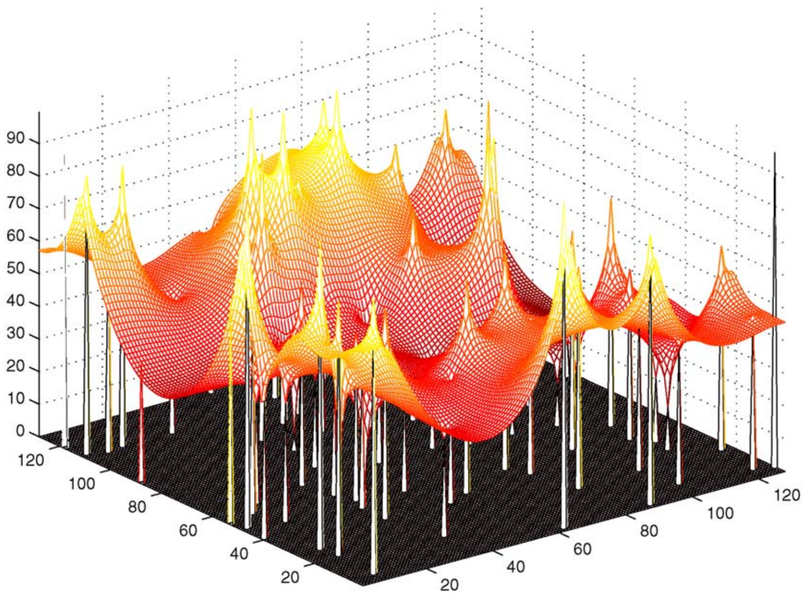

Fig. 1. Solution of the Laplace equation by the over-relaxation method.

ghost objects where eliminated in Ref. [2], however, it is preferable to get rid of them already by a careful construction of the threshold surface. To illustrate the sharp extremas at the support points and the hanging 'valleys' in between, Fig. 1 shows a surface computed by SOR for 100 support points with random values in the range of 0–100. The support points were randomly scattered over a 128×128 grid.

Ideally, a good threshold surface should indicate the local illumination level, which is usually a smooth function of the coordinates. Moreover, the value of an image at a support point probably indicates the local illumination level in its vicinity and there is no reason that it will be a local extrema. Hence, what actually happens to the threshold surface obtained by SOR solution of the Laplace equation is not what we would expect a good adaptive threshold surface to be. Therefore, it would be better not to put an interpolation constraint on the threshold surface, but to construct it as a smooth approximation of the support points thus making it robust to noisy outliers among the support points. The next section describes a new efficient way to construct such a threshold surface.

3. The new threshold surface

We propose to construct and represent the threshold surface as a sum of functions, obtained by scaling and shifting of a single source function, similar to what is done in wavelets or multiresolution representations [13]. In multiresolution representation [8] the coefficients are calculated on the basis of an original signal that is known a priori. In our case the complete threshold surface is not known in advance, but only its approximate values at the support points: $T(p_i) = I(p_i) \equiv v_i$. Here $p_i = \{x_i, y_i\}$ and $v_i = I(x_i, y_i)$ denote the ith support point and its value. This section presents an efficient way to construct surfaces that interpolate and approximate image values at the support points $I(p_i)$. First an interpolation algorithm is presented. However, the interpolation surface obtained is discontinuous and cannot serve as a good threshold surface. Therefore, a small modification to the interpolation algorithm is presented, that results in a continuous and smooth approximation surface.

Let us consider a unit step source function, given by

$$G_{000}(x, y) = \begin{cases} 1 & \text{if } (x, y) \in \Omega(I), \\ 0 & \text{if } (x, y) \notin \Omega(I). \end{cases} \tag{7}$$

Here $\Omega(I) = [0, 1]^2$ denotes the set of all the image points. All the other functions we shall use are generated by downscaling of this source function and shifting the downscaled functions around the image plane and thus cover only part of the image:

$$G_{ljk}(x, y) = G_{000}\left(x2^l - j, y2^l - k\right), \tag{8}$$

where $l = 0, \ldots, \log_2(N)$ is a scale factor and $j, k \in \left\{0, \ldots, 2^l - 1\right\}$ are spatial shifts.

The threshold surface will be given by

$$T(x, y) = \sum_{l=0}^{\log_2(N)} \sum_{j,k=0}^{2^l-1} a_{ljk} G_{ljk}(x, y). \tag{9}$$

3.1. Interpolation algorithm

Let us introduce an algorithm to calculate the decomposition coefficients a_{ijk} in order to obtain an interpolating surface $T(x, y)$ by Eq. (9), passing exactly through all the support points $T(p_i) = I(p_i)$.

The algorithm runs as follows:

(1) The decomposition coefficient a_{000} is set equal to the average of all the support points

$$a_{000} = \langle v_i^{(0)} \rangle = \frac{1}{P_{000}} \sum_{i=1}^{P_{000}} v_i^{(0)}. \tag{10}$$

Here the first zero in index 000 refers to the 0th resolution level, the following 00 refer to the only possible spatial position at this level. The support points $\{p_i\}_{i=1}^{P_{000}}$ are defined by Eq. (4) and P_{000} is the total number of support points. After step 1 every support point $v_i^{(0)}$ is already approximated by the average a_{000}, so it remains only to interpolate the difference between the value of every support point and the average.

(2) The values of the support points are updated as follows:

$$v_i^{(1)} = v_i^{(0)} - a_{000}. \tag{11}$$

The quantities $v_i^{(1)}$ will be referred to as the *first-order residuals*.

(3) The image is divided into four cells, with corresponding indexes {*jk*} relating to the spatial position of the cell: {00, 01, 10, 11}. The average of the updated support points $v_i^{(1)}$ of each cell *jk* is calculated to yield the appropriate decomposition coefficient a_{1jk}:

$$a_{1jk} = \frac{1}{P_{1jk}} \sum_{p_i \in S_{1jk}} v_i^{(1)}. \tag{12}$$

Here $p_i \in S_{1jk}$ denotes a support point p_i that belongs to the cell at the 1st resolution level, situated at the (j, k) spatial position. P_{1jk} denotes the number of support points in this cell.

(4) After step 3 the values of support points in each cell *jk* are approximated by $a_{000} + a_{1jk}$, so their values are updated to be

$$v_i^{(2)} = v_i - a_{000} - a_{1jk} = v_i^{(1)} - a_{1jk}. \tag{13}$$

(5) Steps 3 and 4 are repeated for successive resolution levels. At every resolution level $(l - 1)$ each of the 4^{l-1} cells of this level is divided into four cells to yield 4^l cells at the resolution level l. The coefficients a_{ljk} of the cells at level l at (j, k) spatial position are set to be equal to the average of the residual values of the support points, belonging to this cell:

$$a_{ljk} = \frac{1}{P_{ljk}} \sum_{p_i \in S_{ljk}} v_i^{(l)}. \tag{14}$$

Here $p_i \in S_{ljk}$ denotes a support point p_i that belongs to the cell at level l, placed at (j, k) spatial position. P_{ljk} denotes the number of support points in this cell. After calculation of the coefficients a_{ljk}, the values of the support points are updated:

$$v_i^{(l+1)} = v_i^{(l)} - a_{ljk}. \tag{15}$$

(6) The procedure ends at the highest resolution level L ($L = \log_2(N)$ for $N \times N$ image), when the size of the cell equals to one pixel. At this step there is at most one support point in every cell *jk*, with a residual value $v_i^{(L)}$. The coefficient a_{Ljk} is set to $a_{Ljk} = v_i^{(L)}$.

The threshold surface, constructed in accordance with Eqs. (7)–(9) with the coefficients a_{ljk} obtained by the algorithm as described in steps 1–6, will be an interpolation surface of the support points $\{p_i, I(p_i)\}$, i.e. it will pass through every support point. This can be proved by the following argument:

Consider some arbitrary support point p_i. The value of the threshold surface at this point will be

$$T(p_i) = \sum_{l=0}^{L} a_{lj_lk_l}. \tag{16}$$

Where the j_lk_l chooses at every level l the cell that contains the p_i.

On the other hand the residual value $v_i^{(L+1)}$ of the support point p_i equals to (step 6):

$$v_i^{(L+1)} = v_i^{(0)} - a_{000} - a_{1j_1k_1} - \cdots - a_{Lj_Lk_L} = 0, \tag{17}$$

which can be rewritten as

$$v_i^{(0)} \equiv I(p_i) = a_{000} + a_{1j_1k_1} + \cdots + a_{Lj_Lk_L}. \tag{18}$$

From Eqs. (16) and (18) it follows that for an arbitrary support point p_i, $T(p_i) = I(p_i)$.

Fig. 2 shows the interpolation surface, obtained by our method for the same set of support points that was used for the over-relaxation solution, shown in Fig. 1.

3.2. Approximating source function

The method presented in the previous section yields a surface that interpolates the support points. However, the obtained interpolation surface is discontinuous. In order to obtain an n-continuously differentiable approximation surface, the source function (7) must be substituted by n-times

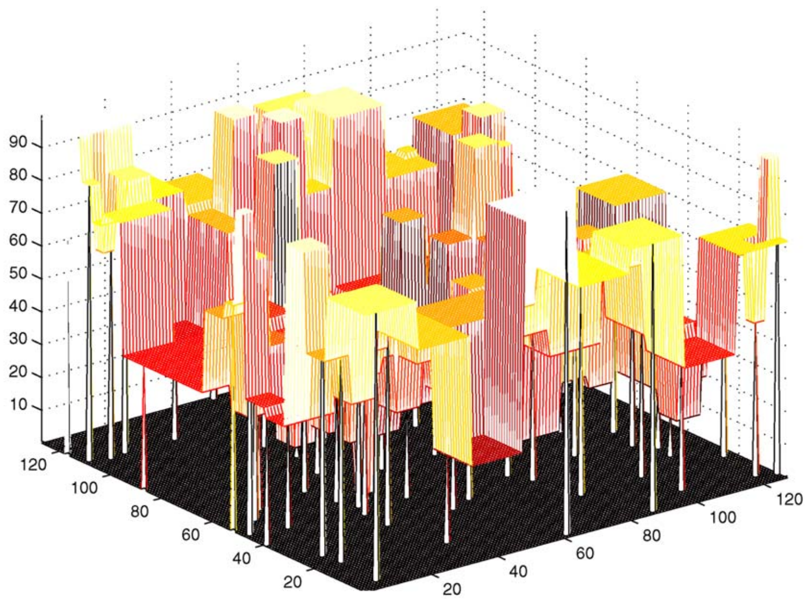

Fig. 2. Interpolating surface, obtained by a new interpolation method.

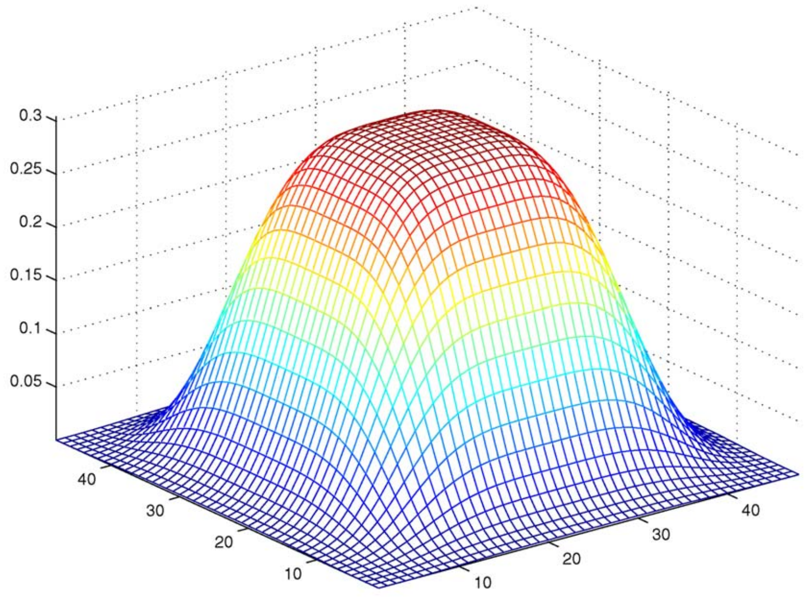

Fig. 3. The source function, given by Eq. (19).

continuously differentiable function vanishing together with n first derivatives at the boundary of its support.

In the practical case of finite grid it is enough to consider a source function having a value and derivatives small enough at the boundary. However, there are three additional requirements from the source function: (approximation) it should have value close to 1 in the domain of its cell; (normalization) the integral of the source function over its support must be equal to the image area; (smoothness) it should decrease gracefully towards the boundary of its support. The first two requirements are necessary in order to build the threshold surface really approximating the support points and the third one in order to have it practically smooth.

As a compromise between these contradicting requirements we chose a source function with support $[-1, 2] \times [-1, 2]$, extending over the image area $[0, 1] \times [0, 1]$. Therefore the threshold surface (9) is constructed with scaled functions, overlapping at each resolution level. It was found

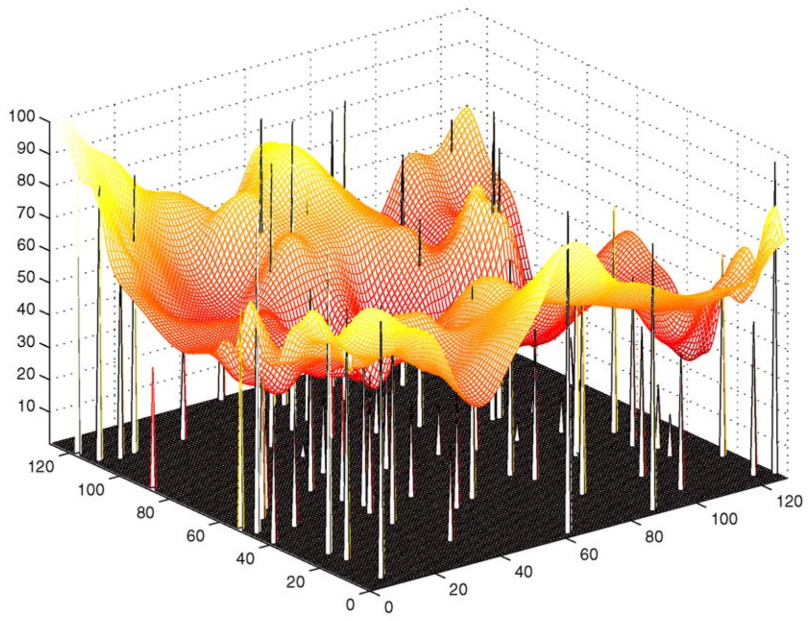

Fig. 4. An approximating surface, obtained with source function (19).

empirically that source (19) gave a good performance:

$$G_{000}(x,y) = \begin{cases} \dfrac{e^{-(x-1/2)^4-(y-1/2)^4}}{\int_{-1}^{2}\int_{-1}^{2} e^{-(x-1/2)^4-(y-1/2)^4}} & \text{if } \{x,y\} \in [-1,2]^2, \\ 0 & \text{if } \{x,y\} \notin [-1,2]^2. \end{cases} \quad (19)$$

The point $\{x, y\}=\{\frac{1}{2}, \frac{1}{2}\}$ is the center of the image, spanning over $[0, 1] \times [0, 1]$. Fig. 3 shows the source function (19). The support points that will determine the decomposition coefficients lie in the central cell $[0, 1] \times [0, 1]$, where the source function (19) is practically flat. Eight periphery cells will overlap neighboring functions thus making the threshold surface smooth.

Fig. 4 shows the smooth threshold surface, constructed with the source function (19) for the same set of support points that was used to construct the interpolated surfaces of Figs. 1 and 2. Figs. 1, 2, and 4 show the support points by vertical spikes. Some of the support points of Fig. 4 are lying apart from the threshold surface. This is due to the fact that support points have random values for demonstration purposes and therefore the approximating surface passes far from some of the support points. In real cases, the neighboring support points usually have similar values and the approximation surface will be close to them.

The new threshold surface is smooth. It does not necessarily pass exactly through the support points, however, this is an advantage rather than disadvantage, because if several neighboring support points have substantially different and ‘noisy’ values this indicates either that the threshold surface is under-sampled by the support points or that there is some error or noise in their values. In both cases there is not enough information at the support points about the threshold surface and the best thing to do is probably to set the threshold surface somewhere in between, as done by the proposed approximation algorithm.

4. Implementation

The algorithm described in the previous section was implemented in Matlab. The subsections below describe the data structures and then the algorithm implementation.

4.1. Data structures

The basic data structures are two arrays:

The first array is called *coeffs* (Table 1), it stores the decomposition coefficients of the cells a_{ijk} in the first row and the number of support points P_{ljk} of these cells in the second. a_{ljk} denotes the decomposition coefficient of the cell *ljk*, which is situated at the (j, k) spatial position at the level l of the resolution. P_{ljk} stores the number of support points in this cell.

First column of *coeffs* stores the single coefficient of the lowest level a_{000} and the total number of support points $P \equiv P_{000}$, following are four columns of coefficients of the first level $(a_{100}, \ldots, a_{111})$ and number of points in each of these cells $(P_{100}, \ldots, P_{111})$, etc.

Table 1
Array *coeffs*

a_{000}	a_{100}	a_{101}	...
P_{000}	P_{100}	P_{101}	...

Contains decomposition coefficients a_{ljk} and number of support points P_{ljk} in the cell *ljk*.

Table 2
Array *pointarr*

p_1	p_2	...	p_P
...	...	...	...
$p_{\log_2(N)1}$	$p_{\log_2(N)2}$	...	$p_{\log_2(N)p}$

Column i contains the indices of the cells containing p_i.

Every support point belongs to one and only one cell lj_lk_l at every resolution level *l*. There are $\log_2(N)$ different resolution levels, starting from single cell of size $N \times N$ at level 0 to N^2 cells of size 1×1 at level $\log_2(N)$.

The second array, called *pointarr* (Table 2), has *P* columns and $1 + \log_2(N)$ rows. Every column of *pointarr* contains the residual value of the support point $p_i^{(l)}$ in the first row, and the indices ind_{il} in other rows. These indexes refer to the cells which contain p_i at every level *l*: *coeffs*[:, ind_{il}]=[a_{ljk}; P_{ljk}].

Fig. 5 shows an example of a point, which belongs to $cell_{000}$ at level 0 (as every point does), $cell_{100}$ at level 1, to cell $cell_{211}$ at level 2, etc. This point will contribute in the construction of the threshold function only through the coefficients $a_{000}, a_{100}, a_{211}, \ldots$. These coefficients are stored in the first row, columns $1, 2, 11 \ldots$ of array *coeffs* (Table 1). Therefore the column of *pointsarr*, corresponding to this point will have values $1, 2, 11, \ldots$ in its second, third, fourth ... rows.

4.2. Algorithm description

(1) Array *points* (Table 2) is created and gradually filled. Every column *i* of this table contains value of the point p_i in the first row. For every point p_i a calculation is performed to determine to which cell $lj_{il}k_{il}$ it belongs at each level l, $l = 0, \ldots, \log_2(N)$. The positions of these cells in the array *coeffs* (Table 1) are filled into rows $2, \ldots, N$ of ith column of *pointarr*, and simultaneously, for every encountered cell the counter of the points belonging to this cell is increased in the array *coeffs*. This requires $P\log_2(N)$ calculations of the cell index and $P\log_2(N)$ increments of the point counters (because each of *P* support points entered into $\log_2(N)$ cells).
(2) The coefficients a_{ljk} in the array *coeffs* are calculated. a_{000} is set to be an average value of all points (10). After this the value of every point in *points* is updated: average value is subtracted from it (11).
(3) Step 2 is repeated for a higher level: Every point contributes its current value to the cell it belongs to, this value is divided by the number of points which belong to the cell. After all the points of a given level have contributed their residual values to the cells, their values are updated: from each point belonging to $cell_{ljk}$ the value of a_{ljk} is subtracted.
(4) The threshold surface is built based on the *coeffs* and the basis function (19). This requires $O\left(N^2\log_2(N)\right)$ operations.

So an approximation surface for *P* support points scattered over N^2 grid is determined as a set of coefficients using $O\left(P\log_2(N)\right)$ operations and built explicitly using $O\left(N^2\log_2(N)\right)$ operations.

In the reconstruction phase, the virtual coefficients beyond the image boundary were created to effectively maintain Newman boundary conditions.

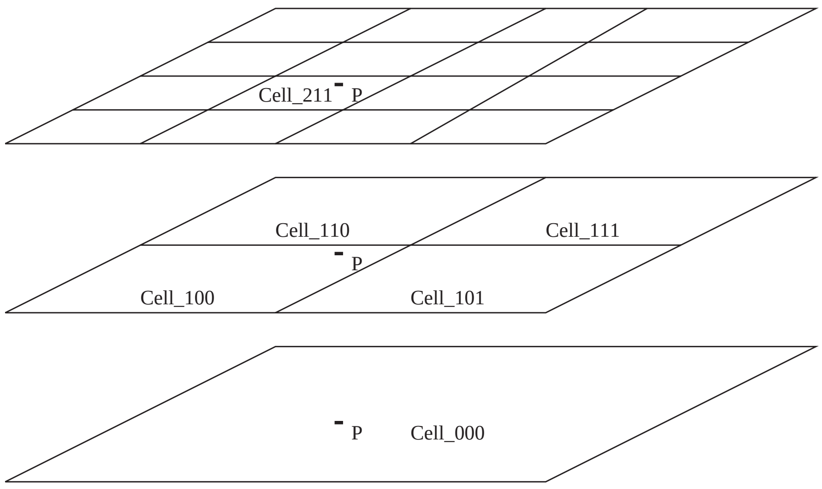

Fig. 5. Cell hierarchy.

5. Experimental results

The three methods, YB with adaptive threshold surface obtained by SOR and the new one with adaptive threshold surface obtained by multiresolution approximation and the Niblack's method were compared for speed and quality of binarization. The programs were implemented in MATLAB and ran on an IBM-Thinkpad-570 platform with 128 MB RAM and a Pentium-II 366 MHz processor.

Four artificial black–white images were generated by simulating non-uniform illumination of the black and white pattern. This allowed to give a quantitative measure of the error of the binarization method. The error was calculated as a normalized L^2 distance between the binarized and the original B/W image. The post-processing step of ghost-elimination was omitted for all the methods. Figs. 6–21show the gray level images and the B/W images, reconstructed by the three binarization methods. Table 3 presents the runtimes and the binarization errors for each method. In

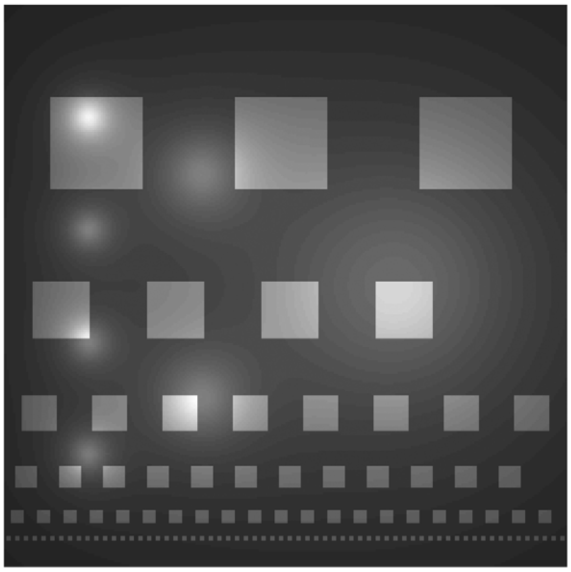
Fig. 6. Gray level image of ‘Squares’.

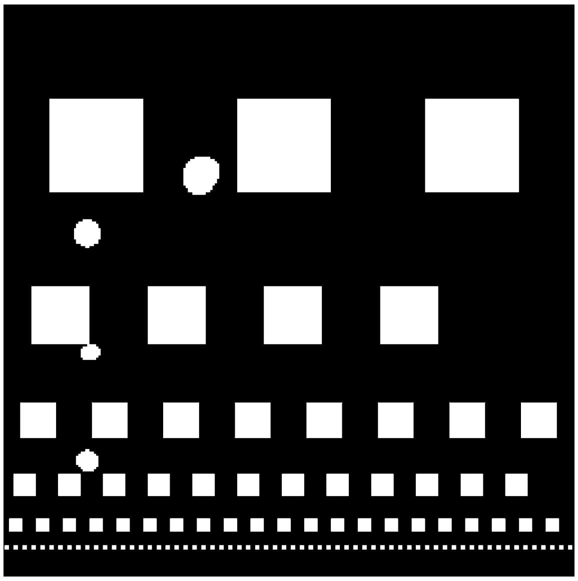
Fig. 8. ‘Squares’ binarized with MA method.

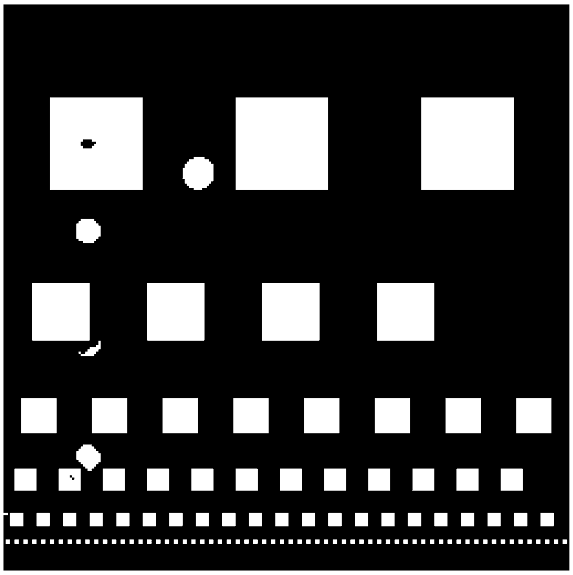
Fig. 7. ‘Squares’ binarized with YB method.

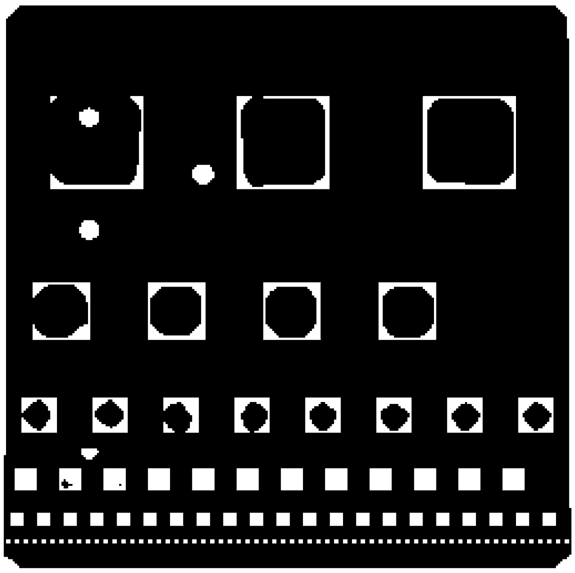
Fig. 9. ‘Squares’ binarized with Niblack method.

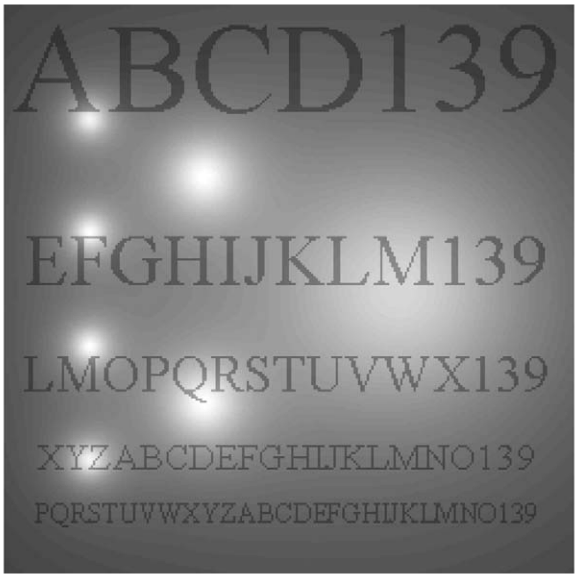

Fig. 10. Gray level image of ‘Text’.

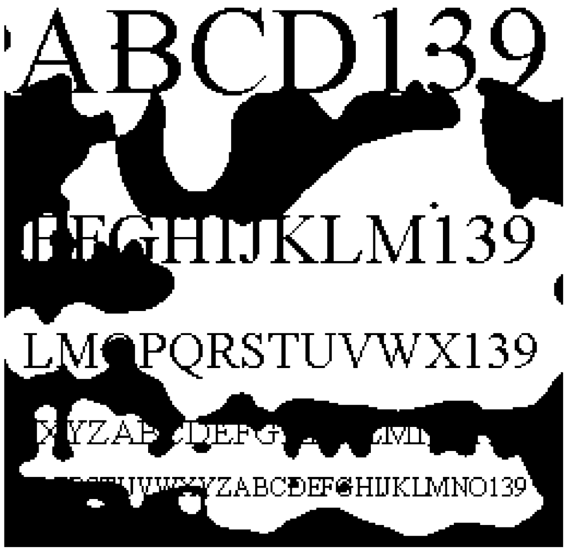

Fig. 12. ‘Text’ binarized with MA method.

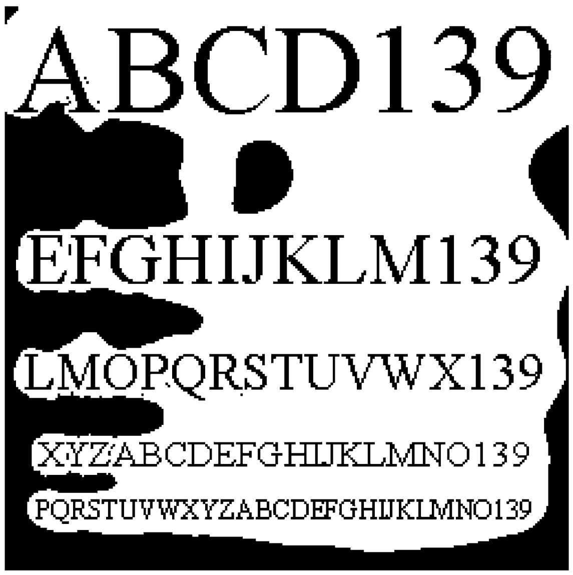

Fig. 11. ‘Text’ binarized with YB method.

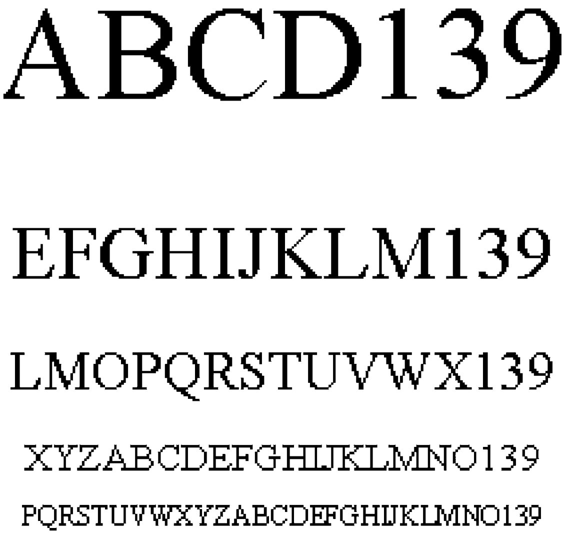

Fig. 13. ‘Text’ binarized with Niblack method.

the Niblack binarization method the value of k from Eq. (2) was chosen to minimize the error, independently for each image. It was equal $+0.8$, -0.6, $+0.05$ and -0.2, respectively for the ‘Squares’, ‘Text’, ‘Rectangles’, and the ‘Stars’ test patterns.

For the ‘Squares’ test patterns the YB method gave the best results. The ‘Text’ test pattern reveals the definite superiority of the Niblack method for this important class of images. The ‘Rectangles’ test pattern was created by addition of 1% salt and pepper noise to the gray level image of a geometric series of rectangles. For this pattern the proposed method gave the best results. Finally, for the ‘stars’ test pattern the proposed method gave the best results again. The YB method had difficulties in the large regions without support points near the boundary, while the Niblack method gave perfect binarization for the objects of specific scale, and produced less impressive results for larger objects, as predicted in Section 2.5.

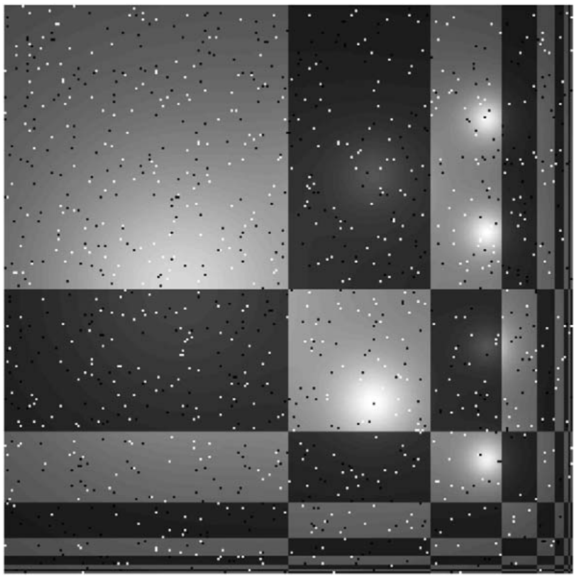
Fig. 14. Gray level image of 'Rectangles'.

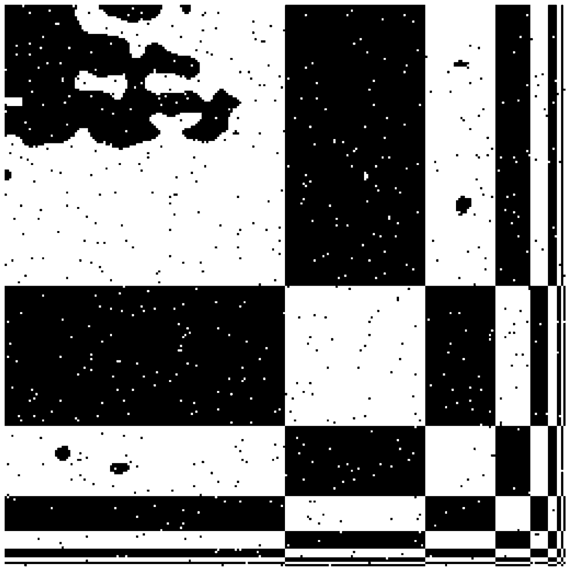
Fig. 16. 'Rectangles' binarized with MA method.

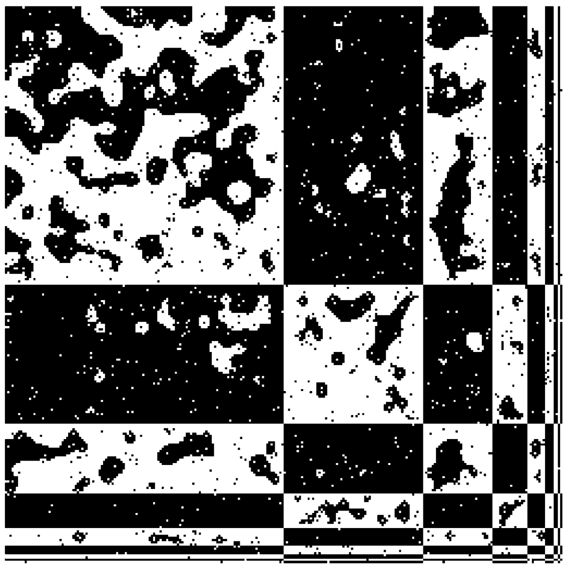
Fig. 15. 'Rectangles' binarized with YB method.

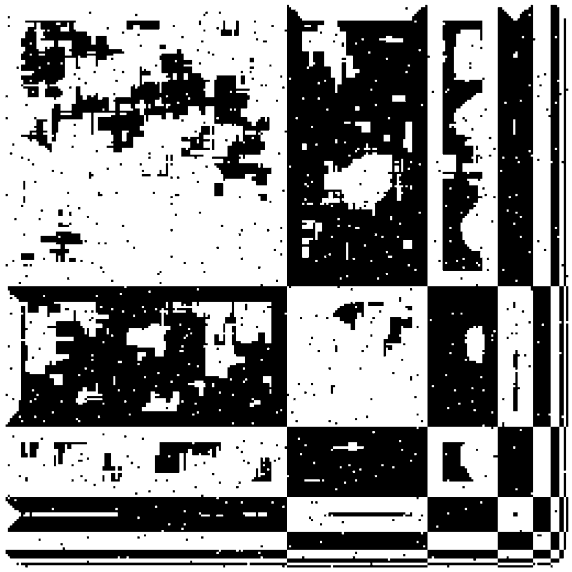
Fig. 17. 'Rectangles' binarized with Niblack method.

6. Concluding remarks

In this work we proposed a new way to construct a threshold surface in order to improve the Yanowitz–Bruckstein binarization method. The new threshold surface is constructed with considerably lower computational complexity and hence in much shorter time even for small images. The new method allows even more gain in speed in region-of-interest processing scenarios. The new threshold surface can be made smooth and by the nature of its construction should be similar to the local illumination level. These qualities allowed to expect a better visual performance of the binarization process. A binarization with the new threshold surface was compared to Yanowitz–Bruckstein and the Niblack methods on the set artificial images, with 4 representative cases presented and discussed here. Considering the experimental results it is apparent that there is no clear winner. The Niblack method was the best for the Text binarization.

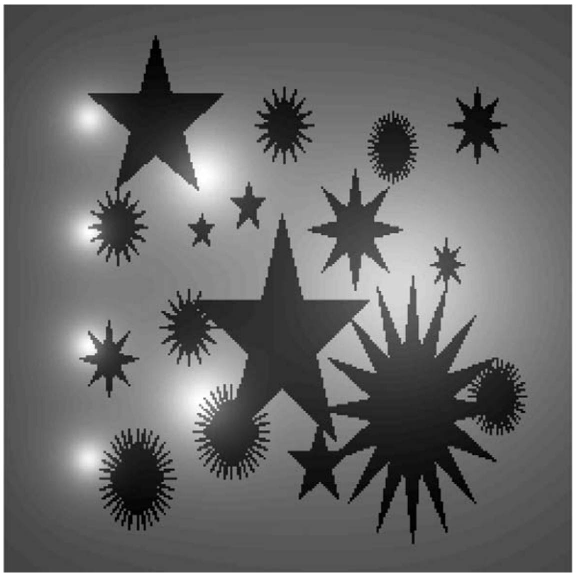
Fig. 18. Gray level image of ‘Stars’.

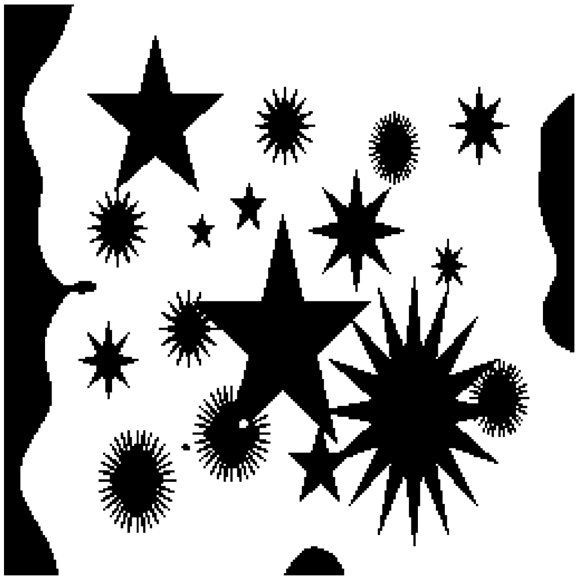
Fig. 20. ‘Stars’ binarized with MA method.

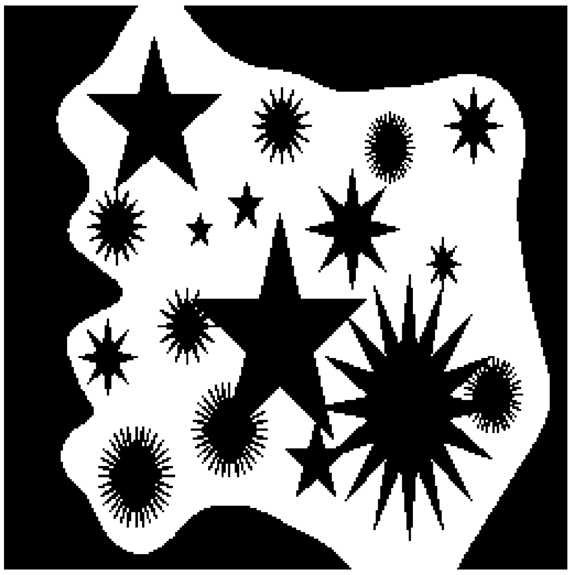
Fig. 19. ‘Stars’ binarized with YB method.

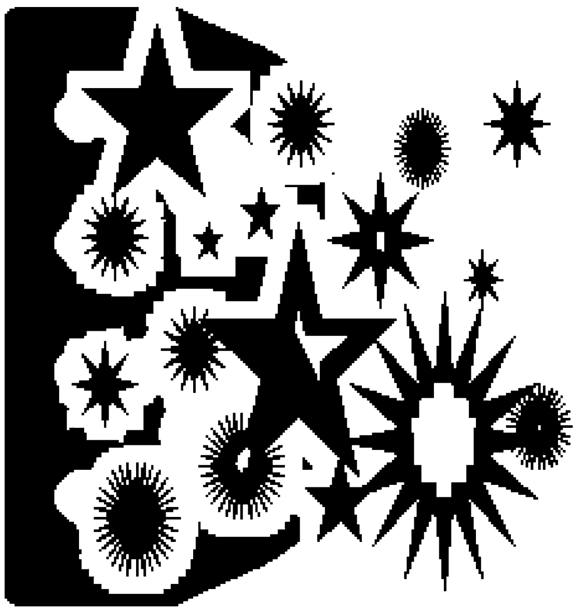
Fig. 21. ‘Stars’ binarized with Niblack method.

Table 3
Comparison of the speeds and performance of the Yanowitz–Bruckstein (YB) and multiresolution approximation (MA) and Niblack binarization methods

Test image	‘Squares’		‘Text’		‘Rectangles’		‘Stars’	
	Runtime	Error	Runtime	Error	Runtime	Error	Runtime	Error
SOR	165.2	**0.0063**	160.2	0.212	160.7	0.191	161	0.312
MA	9.7	0.0072	11.4	0.312	15.9	**0.078**	13.4	**0.096**
Niblack	0.95	0.128	0.96	**0.000**	0.98	0.152	0.98	0.1745

The performance of the best method is printed in bold.

Another advantage of the method is its speed and simplicity of implementation. However, Niblack method is scale dependent, and that made it inferior to Yanowitz–Bruckstein and our methods on the non-text images, where objects of different scales appeared. For these images our method was comparable or better than the Yanowitz–Bruckstein method, while having a significant speed advantage.

References

[1] C.K. Chow, T. Kaneko, Automatic boundary detection of the left-ventricle from cineangiograms, Comput. Biomed. 5 (1972) 388–410.
[2] S.D. Yanowitz, A.M. Bruckstein, A new method for image segmentation, Comput. Vision Graphics Image Process. 46 (1989) 82–95.
[3] V.R. Southwell, Relaxation Methods in Theoretical Physics, Oxford University Press, Oxford, 1946.
[4] D. Trier, T. Taxt, Evaluation of binarization methods for document images, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 312–315.
[5] W. Niblack, An Introduction to Digitall Image Processing, Prentice-Hall, Englewood Cliffs, NJ, 1986.
[6] L. Eikvil, T. Taxt, K. Moen, A fast adaptive method for binarization of document images, in: Proceedings of the First International Conference on Document Analysis and Recognition, Saint-Malo, France, 1991, pp. 435–443.
[7] J. Bernsen, Dynamic thresholding of grey-level images, in: Proceedings of the Eighth International Conference on Pattern Recognition, Paris, France, 1986, pp. 1251–1255.
[8] S.G. Mallat, A Wavelet Tour of Signal Processing, Academic Press, New York, 1999.
[9] N. Otsu, A threshold selection method from grey-level histograms, IEEE Trans. Systems Man Cybernet. 9 (1) (1979) 62–66.
[10] D. Stork, R. Duda, P. Hart, Pattern Classification, Wiley, New York, 2000.
[11] W.L. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, PA, 1987.
[12] R. Courant, D. Hilbert, Methods of Mathematical Physics, Interscience Publishers, 1953.
[13] I. Blayvas, A.M. Bruckstein, R. Kimmel, Efficient computation of adaptive threshold surface for binarization, in: Proceedings of the CVPR, Hawaii, USA, 2001, pp. 737–742.

About the Author—ILYA BLAYVAS received his B.Sc. from the faculty of aerophysics and space research of Moscow Institute of Physics and Technology (MIPT) in 1992, and his M.Sc. (with honours) in laser physics from Ben Gurion University in 1996. In 1996–1999 he designed Monolitic Microwave Integrated Circuits at Israely Armament Development Authority, and in 2000–2001 he performed an electro-optical characterization of CMOS image sensors at Tower Semiconductor ltd. From 2001 he is doing his Ph.D. research in Computer Vision at CS Department of Technion.

About the Author—ALFRED M. BRUCKSTEIN received the B.Sc. (honors) and M.Sc. in electrical engineering from the Technion, Israel Institute of Technology, Haifa, and his Ph.D. in electrical engineering from Stanford University, Stanford, CA, in 1977, 1980, and 1984, respectively. Since 1985, he has been a Faculty Member at the Technion, Israel Institute of Technology, where he currently a full Professor, holding the Ollendorff Chair in Science. During the summers from 1986 to 1995 and from 1998 to 2000 he was a Visiting Scientist at Bell Laboratories, and in 2001-2002 a visiting chaired professor at Tsing-Hua Univarsity in Beijing, China. He served on the editorial boards of Pattern Recognition, Imaging Systems and Technology, Circuits Systems, and Signal Processing. He also served as a member of program committees of 20 conferences. His research interests are in Image and Signal processing, Computer Vision, Computer Graphics, Pattern Recognition, Robotics (especially Ant Robotics), Applied Geometry, estimation theory and inverse scattering, and neuronal encoding process modeling. Prof. Bruckstein is a member of SIAM, AMS, and MM and is presently the dean of the Technion Graduate school. He was awarded the Rothschild Fellowship for Ph.D. Studies at Stanford, Taub Award, Theeman Grant for a scientific tour of Australian Universities, and the Hershel Rich Technion Innovation Award twice.

About the Author—RON KIMMEL received his B.Sc. (with honors) in computer engineering in 1986, the M.S. degree in 1993 in electrical engineering, and the D.Sc. degree in 1995 from the Technion–Israel Institute of Technology. During the years 1986–1991 he served as an R&D officer in the Israeli Air Force.
During the years 1995–1998 he has been a postdoctoral fellow at the Computer Science Division of Berkeley Labs, and the Mathematics Department, University of California, Berkeley. Since 1998, he has been a faculty member of the Computer Science Department at the Technion, Israel, where he is currently an associate professor. He is now a visiting Professor at the Computer Science Department, Stanford University, and working with MediGuide Inc. His research interests are in computational methods and their applications in: Differential geometry, numerical analysis, image processing and analysis, and computer graphics.
He was awarded the Hershel Rich Technion innovation award (twice), the Henry Taub Prize for excellence in research, Alon Fellowship, the HTI Postdoctoral Fellowship, and the Wolf, Gutwirth, Ollendorff, and Jury fellowships.
He has been a consultant of HP research Lab in image processing and analysis during the years 1998–2000, and to Net2Wireless/Jigami research group during 2000–2001. He is on the advisory board of MediGuide (biomedical imaging 2002–2005), and has been on various program and organizing committees of conferences, workshops, and editorial boards of image processing and analysis journals, like International Journal of Computer Vision, and IEEE Trans. on Image Processing.
He is the author of 'Numerical Geometry of Images' published by Springer, November 2003.