PERGAMON

Pattern Recognition 32 (1999) 811–824

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

Block decomposition and segmentation for fast Hough transform evaluation

Stavros J. Perantonis[a,*], Basilios Gatos[a,b], Nikos Papamarkos[b]

[a] *Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", 153 10 Aghia Paraskevi, Greece*
[b] *Electric Circuits Analysis Laboratory, Department of Electrical and Computer Engineering, Demokritus University of Thrace, 67100 Xanthi, Greece*

Received 16 October 1997; in revised form 1 June 1998

Abstract

The decomposition of binary images using rectangular blocks of foreground pixels as primitives is considered. Based on this type of decomposition, a fast method for evaluating the Hough transform is introduced. A complexity analysis of the proposed block Hough transform algorithm sets constraints on the complexity of algorithms used for block decomposition, so that the total decomposition and Hough transform application time is much less than the time consumed by the usual point Hough transform. Using this analysis, we propose two algorithms for the decomposition and segmentation of binary images into rectangular blocks. A combination of these methods leads to significant acceleration in the identification of linear features, which is demonstrated in various image processing experiments.

Keywords: Fast Hough transform; Image decomposition; Segmentation; Skew detection

1. Introduction

The description of a digital image in terms of simple geometrical shapes is a well established methodology that often proves useful for effective image segmentation [1]. Moreover, such a description can also contribute towards effective and fast implementation of image processing algorithms. Well-known examples are the description of images in terms of polygonal shapes [2], and shape description using morphological operations [3]. Both methods can be used to speed up image processing operations, like moment calculations [2].

If the description of an image in terms of primitive geometrical shapes is a priori known, then the total processing time will be taken by the image processing algorithm. However, if this description is not a priori available, suitable algorithms must be proposed in order to decompose the image into the required shapes. In this case, the total processing time will be the sum of the processing times taken by the decomposition and image processing algorithms. Clearly, the design of the decomposition algorithm must take into account the computational complexity of the image processing algorithm, so that the speedup of the latter is not cancelled by long processing times spent in the decomposition stage.

The Hough transform (HT) [4] has emerged in recent decades as a powerful method for many image processing

* Corresponding author.

0031-3203/99/$ – See front matter
PII: S0031-3203(98)00125-3

and pattern recognition applications. However, a major drawback of its implementation in large images is its relatively low speed. To overcome this drawback, the development of fast variants of the original HT has attracted much attention in the literature. To this end, different techniques have been proposed, including the exploitation of gradient information from edge images [5–7], parallel implementation [8–12], hierarchical schemes [13–16], and application-dependent fast algorithms [17].

In this paper, we propose that the description of binary images using rectangular blocks can be useful for speeding up the implementation of the HT algorithm. More specifically, we propose a block Hough transform (BHT) method that takes advantage of the rectangular block decomposition of a binary image and achieves fast evaluation of the HT field by analytically calculating the contribution to cells in the Hough accumulator array of a whole rectangular block rather than of each individual pixel.

We derive formulas for the computational complexity of the proposed modified HT implementation and thus find the minimum size of the rectangular blocks, for which BHT is faster than the usual point Hough transform (PHT). Moreover, using our computational complexity analysis, we set bounds on the computational complexity of the algorithms that can be used to decompose images into rectangular blocks, so that the total process (decomposition and BHT) is faster than PHT. Two such algorithms are introduced, respectively for the description of a binary image using rectangular blocks of foreground pixels, and for its segmentation. In the first algorithm, the image is represented as the union of rectangular blocks of foreground pixels, while in the second algorithm the image is segmented using shapes consisting of assemblies of adjacent rectangular blocks.

In the experimental section, we study four different image processing applications, where identification of prevalent linear directions is needed. We compute the total time required for decomposition and BHT implementation and compare it to the time consumed by PHT. Significant acceleration is observed, especially in applications where prevalent linear features cannot be captured correctly using edge detection, so that the whole image should preferably be used.

2. Block Hough transform algorithm

Consider a binary image $I(x, y)$, $x = 1, 2, \ldots, x_{max}$, $y = 1, 2, \ldots, y_{max}$, defined as follows:

$$I(x, y) = \begin{cases} 1 & \text{for foreground pixel,} \\ 0 & \text{for background pixel.} \end{cases} \tag{1}$$

We are interested in the detection of straight lines parametrized by $r = x\cos\theta + y\sin\theta$ $(0 < \theta \leqslant \pi)$ (see Fig. 1). Let $\Delta\theta$ and Δr be the angular and radial resolution, respectively.

We first form a decomposition of the image into rectangular blocks of foreground pixels. Next, we examine each of the rectangular blocks of pixels and evaluate the contribution of its pixels to each cell in the accumulator array of the HT space. Consider a block $\mathscr{R}$ whose opposite vertices are located at at (k_1, l_1) and (k_2, l_2), where $k_2 \geqslant k_1$ and $l_2 \geqslant l_1$. Clearly, $(k_2 - k_1 + 1)$ $(l_2 - l_1 + 1)$ pixels lie in the interior or on the perimeter of $\mathscr{R}$. Consider a cell in the accumulator array $A(r_1, \theta)$ that corresponds to all straight lines determined by the parameters θ and $-1/2 + r_1 < r < 1/2 + r_1$, with r_1 integer. Clearly, the number of points P in $\mathscr{R}$ contributing to this cell can be approximated by the area occupied by the intersection of the rectangle $\mathscr{R}$ and of the strip in the xy plane restricted between the straight lines $(r_1 - 1/2, \theta)$ and $(r_1 + 1/2, \theta)$ (small discrepancies are due to the discrete image grid). The calculation of P can be done analytically as follows: For a given θ, we first consider the area $E(r_0)$ of the region defined by the intersection of $\mathscr{R}$ and the semi-plane $r < r_0$ whose boundary is the line (r_0, θ). Let $X_i, Y_i, i = 1, \ldots, 4$, be the coordinates of the four corners $Q_i, i = 1, \ldots, 4$, of $\mathscr{R}$ in the order in which they are encountered as the family of parallel straight lines (r_0, θ) sweeps the plane moving towards increasing r_0. Moreover, let $q_i = X_i\cos\theta + Y_i\sin\theta$. As illustrated in the appendix, straightforward geometrical calculations lead to the following formulas:

$$E(r_0) = \begin{cases} 0, & \text{if } r_0 < q_1, \\ \dfrac{(r_0 - q_1)^2}{2|\cos\theta\sin\theta|}, & \text{if } q_1 \leqslant r_0 < q_2, \\ \dfrac{(q_2 - q_1)^2}{2|\cos\theta\sin\theta|} + (r_0 - q_2)L, & \text{if } q_2 \leqslant r_0 < q_3, \\ (k_2 - k_1 + 1)\ (l_2 - l_1 + 1) - \dfrac{(q_4 - r_0)^2}{2|\cos\theta\sin\theta|} & \text{if } q_3 \leqslant r_0 < q_4, \\ (k_2 - k_1 + 1)\ (l_2 - l_1 + 1) & \text{if } r_0 \geqslant q_4 \end{cases} \tag{2}$$

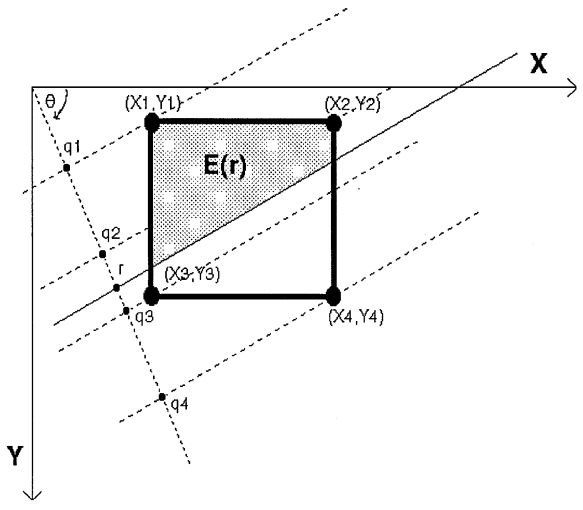


Fig. 1. Evaluation of rectangular block contribution to Hough transform space.

for $\theta \neq k\pi/2$, k integer, and

$$E(r_0) = \begin{cases} 0 & \text{if } r_0 < q_1, \\ (r_0 - q_1)L & \text{if } q_1 \leqslant r_0 < q_3, \\ (q_3 - q_1)L & \text{if } r_0 \geqslant q_3 \end{cases} \tag{3}$$

for $\theta = k\pi/2$, k being an integer.

L depends on whether Q_2 has the same x or y coordinate with Q_1:

$$L = \begin{cases} \dfrac{l_2 - l_1 + 1}{|\cos\theta|} & \text{if } X_2 = X_1, \\ \dfrac{k_2 - k_1 + 1}{|\sin\theta|} & \text{if } Y_2 = Y_1. \end{cases} \tag{4}$$

Finally, P can be readily computed as

$$P = E(r_1 + 1/2) - E(r_1 - 1/2). \tag{5}$$

As a result of the above considerations, our BHT algorithm proceeds as follows:

Given the rectangular block decomposition of a binary image $I(x, y)$, initialize all accumulator array cells $A(r, \theta)$. For all blocks in the image and every angle θ:

Step 1: Compute and sort the quantities $q_i = X_i \cos\theta + Y_i \sin\theta$.

Step 2: Compute $E(1/2 + r_1)$ from Eqs. (2)–(4) for all integers r_1 between $[q_1]$ and $[q_4] + 1$, where $[q]$ denotes the integer part of q.

Step 3: Compute P for all integers r_1 between $[q_1]$ and $[q_4]$ using Eq. (5) and, respectively, add their contributions to the accumulator array cells $A(r_1, \theta)$.

3. Complexity analysis

Let R be a rectangular block of foreground pixels with dimensions X and Y. The purpose of this section is to derive a formula for the ratio of T_{BHT} (CPU time needed to compute the contribution to the HT accumulator array of R using the BHT) over T_{PHT} (the CPU time needed to compute the same contribution using the PHT). It will turn out that this ratio depends on the dimensions of the rectangle. We will thus be able to isolate those rectangles, for which it is beneficial to use the BHT instead of the PHT.

Starting with the PHT, let A_p be the CPU time needed to compute the contribution of one foreground pixel. This is equal to the CPU time needed to complete the two multiplications and one addition of $x\cos\theta + y\sin\theta$. For the whole rectangle and for each bin in the θ space, the time needed is equal to XYA_p. Therefore, the total CPU time for all angles can be approximated by

$$T_{PHT} = \frac{XY\pi A_p}{\Delta\theta} \tag{6}$$

Next, we consider the BHT. For every angle θ, the quantities of Eqs. (2) or (3) have to be computed for every relevant r_0. Computations independent of r_0 have to take place before the r_0 loop is entered. These include evaluation and sorting of the q_i as well as calculation of the r_0 independent parts of Eqs. (2) and (3). Therefore, T_{BHT} can be obtained as the sum of the "overhead" time T_0 needed to compute quantities independent of r_0 and of the time T_r needed for computations within the r_0 loop.

Let A_v denote the total "overhead" time needed for a particular value of θ. Clearly, A_v does not depend on the dimensions of the particular rectangular block R and T_0 is equal to $A_v\pi/\Delta\theta$.

For a particular angle θ and position r_0, let A_b be the average CPU time needed to complete the evaluation of $E(r_0)$ according to Eqs. (2) or (3). Then, the number of values of r_0 for which these computations have to take place is $(q_4 - q_1)/\Delta r$ and the corresponding CPU time is $(q_4 - q_1)A_b/\Delta r$.

Next, we wish to add up the CPU times for all values of θ ($0 < \theta \leqslant \pi$). It is straightforward to see that contributions for angles θ greater than $\pi/2$ are equal to the contributions corresponding to angles $\pi - \theta$.

Therefore, the total time T_r for all angles θ is given by

$$\begin{aligned} T_r &= \frac{2A_b}{\Delta\theta\Delta r}\int_0^{\pi/2} (q_4 - q_1)\,\mathrm{d}\theta \\ &= \frac{2A_b}{\Delta\theta\Delta r}\int_0^{\pi/2} (X\cos\theta + Y\sin\theta)\,\mathrm{d}\theta \\ &= \frac{2A_b(X + Y)}{\Delta\theta\Delta r}. \end{aligned} \tag{7}$$

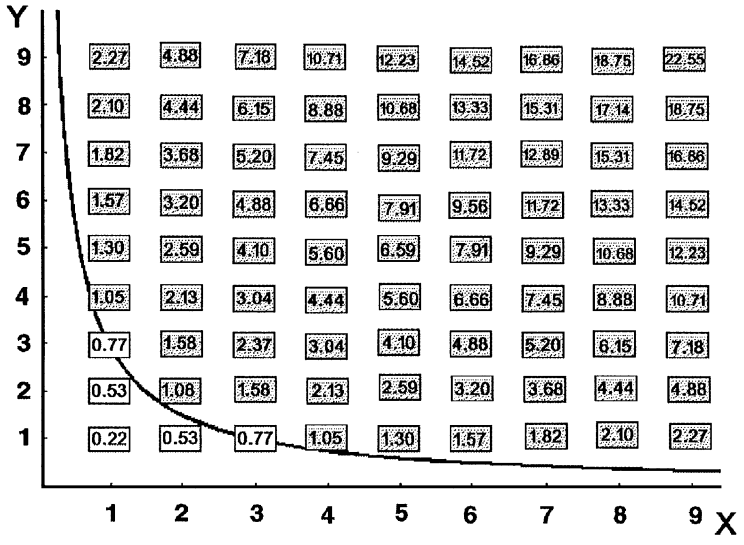


Fig. 2. The ratio of the CPU time required by PHT over the CPU time required by BHT for rectangular blocks of different sizes. The curve represents the theoretical result for ratio equal to one.

We can now obtain the final formula for the ratio of CPU times as follows:

$$\frac{T_{BHT}}{T_{PHT}} = \frac{2A_b(X + Y) + A_v\pi\Delta r}{XY\pi A_p\Delta r}. \tag{8}$$

The values of the coefficients A_p, A_b and A_v depend on the specific platform used for the computations. It can easily be seen, however, that A_p and A_b are of the same order (Eq. (2) shows that usually one to two additions and one to two multiplications contribute to A_b and this has to be compared with the two multiplications and one addition needed to compute A_p). On a Pentium platform at 150 MHz we have obtained $A_b/A_p = 0.71$ and $A_v/A_p = 3.29$.

BHT is faster than PHT for $T_{BHT}/T_{PHT} < 1$, i.e. for

$$XY\pi A_p\Delta r - 2A_b(X + Y) - A_v\pi\Delta r > 0. \tag{9}$$

In a X–Y plot, this corresponds to points above the curve defined by $XY\pi A_p\Delta r - 2A_b(X + Y) - A_v\pi\Delta r = 0$. In Fig. 2, experimental values of T_{PHT}/T_{BHT} are shown for different values of X and Y and $\Delta r = 5$. These are in good agreement with the theoretical curve. In our experiments, described in Section 5, we shall use relation (9) as our criterion for incrementing the accumulator array cells by applying the BHT formulas. Blocks that do not pass this criterion will be decomposed into individual points and the usual PHT formula will be applied for each of these points. In this way the maximum processing time economy will be achieved.

4. Image decomposition

For the BHT algorithm to be applied, the image $I(x, y)$ must first be decomposed into rectangular blocks of foreground pixels. To this end, image decomposition algorithms must be applied. These algorithms must be fast enough, so that the processing time advantage of the BHT over the PHT is not forfeited in the image decomposition stage. The complexity analysis of Section 3 can be taken into account, in order to indicate which types of decomposition algorithms fulfill this criterion.

We propose two different algorithms for image decomposition into rectangular blocks. The first just extracts the coordinates of all primitive rectangular blocks that form an image, while the second can also segment the image by identifying clusters of connected rectangular block components.

4.1. Rectangular blocks of the image

Our objective is to decompose a binary image into rectangular blocks, in such a way as to accelerate the evaluation of the HT accumulator array. From the discussion of Section 3, it follows that if the image has been decomposed into N rectangles of dimensions X_i, Y_i, $i = 1, \ldots, N$, the total time needed to complete the BHT is given by

$$T_{BHT} = \frac{2A_b}{\Delta\theta\Delta r}\sum_{i=1}^{N}(X_i + Y_i) + \frac{A_v\pi}{\Delta\theta}N. \tag{10}$$

Clearly, fastest BHT processing can be obtained when the expression in Eq. (10) takes its minimum value. The corresponding optimization problem is an NP complete problem. To find the optimal or suboptimal solutions to this problem would be time-consuming and would annihilate the advantage gained by the acceleration in the HT analysis stage. Indeed, note that the total time taken by the PHT to compute the accumulator is proportional to $N_p\pi/\Delta\theta$, where N_p is the number of foreground pixels in the image and $\pi/\Delta\theta$ is the number of angular bins in the accumulator array space. Clearly, if BHT processing is to be faster, this order of magnitude must not be exceeded by the time taken to form the rectangles and, therefore, any algorithm that would scan the image more than $\pi/\Delta\theta$ times must be excluded.

The following algorithm involves two top-down scans of the image and was found to work very well in practice.

A top-down scan of the image is performed until a foreground pixel (x_0, y_0) is found. Then a search is conducted for the "best fitting block" at (x_0, y_0), that is formed as follows: First, we identify the rectangle $T1$ which is formed if we move point (x_0, y_0) to the right until a background pixel is found and then we move the resulting horizontal segment parallel to itself and downwards, until it contains at least one background pixel. Similarly, we identify the rectangle $T2$ which is formed if we move point (x_0, y_0) downwards until a background pixel is found and then we move the resulting vertical segment parallel to itself and to the right, until it contains at least one background pixel. Between $T1$ and $T2$, the block with the larger area is the best fitting block at (x_0, y_0). If the height/width ratio of the best fitting block is smaller than a predetermined value $\rho > 1$, the block is accepted as part of the image decomposition and the coordinates of its upper left and lower right vertices are stored in arrays XF, YF, XL, YL for further processing by the BHT. All pixels of the accepted block are transformed to background pixels and the procedure is repeated until all the image pixels are scanned.

The condition involving the height/width ratio is included in the algorithm for the following reason: Since the top-down scan locates points where the tangent to the image perimeter is parallel to the x-axis, best fitting blocks tend to be line segments perpendicular to the x-axis. It is better to reject these segments in favor of less elongated blocks of larger area, in order to reduce the number of blocks in the final image decomposition. A typical situation is illustrated in Fig. 3. Fig 3b shows the decomposition of the image in Fig. 3a that is obtained if the height/width ratio condition is relaxed (effectively $\rho \to \infty$). The more efficient decomposition of Fig. 3c is obtained after two passes, one with $\rho = 5$ and one with $\rho \to \infty$.

Since the top-down scan procedure may not extract all rectangular blocks, and there may still exist remaining foreground pixels, we apply a second pass of the algorithm with $\rho \to \infty$, so that all remaining image pixels are eventually grouped in rectangular blocks.

To give a more formal description of the algorithm, let us define a function $B(x_1, y_1, x_2, y_2)$ that informs us if the pixels with coordinates (x_1, y_1) and (x_2, y_2) are opposite vertices of a rectangular block consisting of foreground pixels:

$$B(x_1, y_1, x_2, y_2) = \begin{cases} 1 & \text{if } I(x, y) = 1 \ \forall\ x, y : x \in [x1, x2] \wedge y \in [y1, y2] \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $x_1, x_2 \in [1, 2, \ldots, x_{max}]$ and $y_1, y_2 \in [1, 2, \ldots, y_{max}]$.

Each pass of the algorithm corresponding to a value of ρ is as follows:

Step 1: Set $iter = 1$.

Step 2: Perform a top-down scan of the image to find a pixel of coordinates (x_0, y_0): $I(x_0, y_0) = 1$.

Step 3: Find the opposite vertex (x_{op}, y_{op}) of the "best fitting block" at (x_0, y_0), as follows:

(a) Find $x_1 \geqslant x_0$: $B(x_0, y_0, x_1, y_0) = 1 \wedge\ x_1 - x_0 = \max$.

(b) Find $y_1 \geqslant y_0$: $B(x_0, y_0, x_1, y_1) = 1 \wedge\ y_1 - y_0 = \max$.

(c) Find $y_2 \geqslant y_0$: $B(x_0, y_0, x_0, y_2) = 1 \wedge\ y_2 - y_0 = \max$.

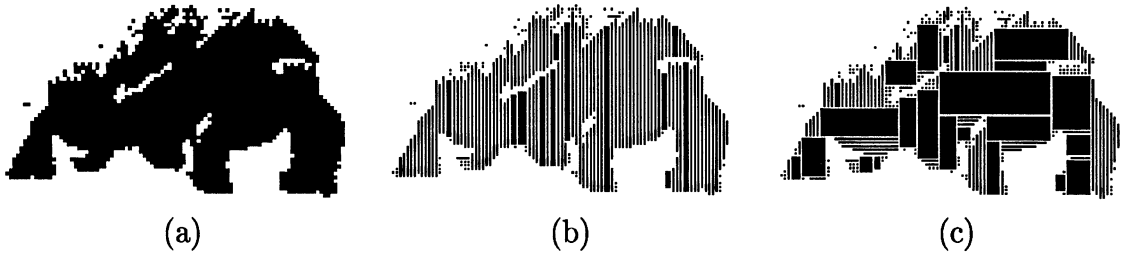


Fig. 3. Role of the parameter ρ (see text) in image decomposition into rectangular blocks: (a) original image; (b) decomposition obtained with $\rho = 1$; (c) decomposition obtained after two passes, one with $\rho = 5$ and one with $\rho = \infty$.

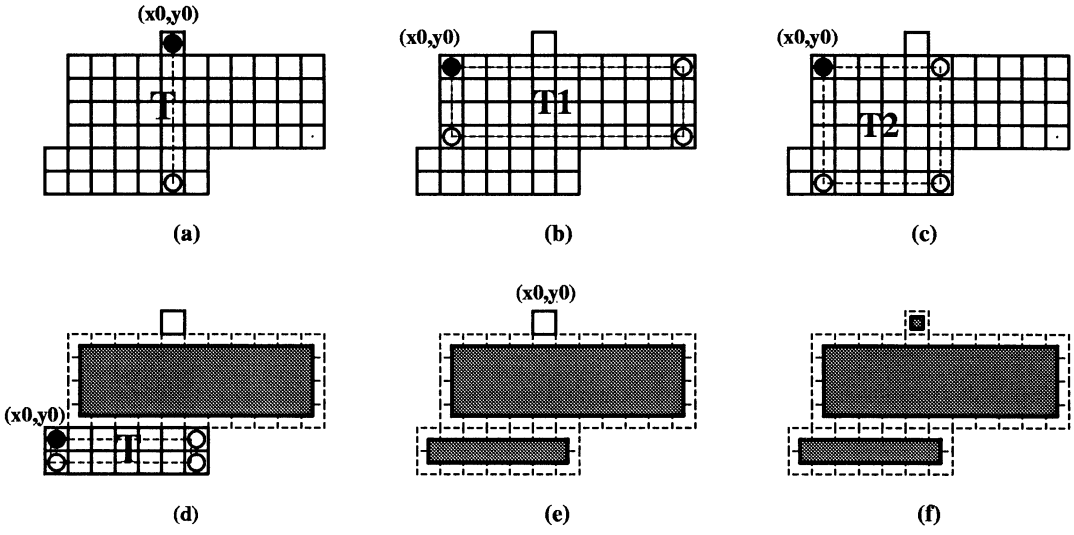


Fig. 4. Steps followed for block decomposition of a simple binary image.

(d) Find $x_2 \geqslant x_0$: $B(x_0, y_0, x_2, y_2) = 1 \wedge x_2 - x_0 = \max$.
(e) Find $q \in \{1, 2\}$: $(x_q - x_0)(y_q - y_0) = \max$. Set $x_{op} = x_q$ and $y_{op} = y_q$.

Step 4: If $(y_{op} - y_0)/(x_{op} - x_0) < \rho$, then:
(a) Set $XF[iter] = x_0$, $XL[iter] = x_{op}$, $YF[iter] = y_0$, $YL[iter] = y_{op}$.
(b) Set $I(x, y) = 0 \ \forall \ x \in [XF[iter] \ldots XL[iter]] \wedge y \in [YF[iter] \ldots YL[iter]]$.
(c) Set $iter = iter + 1$.
(d) Until there remains no unscanned pixel, continue with the top-down scan of step 2.

The decomposition of an image into rectangular blocks is demonstrated in a simple case in Fig. 4. Assume that during the first pass of the algorithm we set $\rho = 5$. Starting from the original image we find pixel (x_0, y_0) with a top-down scan (Fig. 4a). The best fitting block is a vertical line, denoted by T (dashed line in Fig. 4a). In this case, the height/width ratio is greater than ρ and T is not accepted as part of the image decomposition. Moving on to the pixel (x_0, y_0) of Fig. 4b, we form the two candidate blocks $T1$ (Fig. 4b) and $T2$ (Fig. 4c), of which $T1$ has the larger area. Since its height/width ratio is less than ρ, $T1$ is accepted as part of the image decomposition and all pixels of $T1$ are removed from the image. Moving on with the top-down scan to pixel (x_0, y_0) of Fig. 4d, we similarly accept block T as part of the image decomposition (Fig. 4d and e) and remove its pixels from the image. The remaining pixel is incorporated in the image decomposition during the second pass of the algorithm, with $\rho \to \infty$. In this way we have decomposed the original image into three rectangular blocks (Fig. 4f).

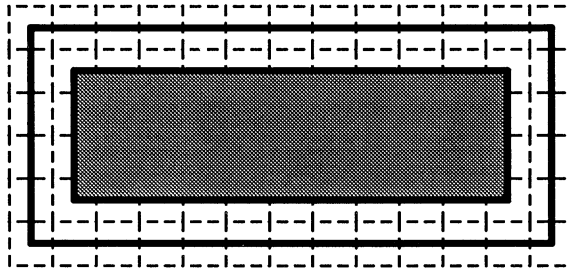


Fig. 5. The surrounding line of a rectangular block.

4.2. Rectangular blocks of connected image components

The previous algorithm is used to detect rectangular blocks in a digital image following a top-down image analysis. Using an extension of this algorithm, it is possible to group together all blocks that belong to the same connected component of the image. To start with, the "best fitting block" at the first foreground pixel is found using a top-down scan. Then, a search is conducted for all "best fitting blocks" at all foreground pixels which belong to the "surrounding lines" of already detected blocks, i.e. are immediately adjacent to the perimeter of such blocks and belong to their exterior (see Fig. 5). This process is repeated iteratively, transforming the pixels of all detected blocks to background pixels, until no foreground pixel remains on the surrounding lines of all blocks of the current connected image component. There follows a complete description of the algorithm:

Step 1: Perform a top-down scan of the image to find a pixel at coordinates (x_0, y_0): $I(x_0, y_0) = 1$.

Step 2:

(a) Find the opposite vertex (x_{op}, y_{op}) of the "best fitting block" at (x_0, y_0), following steps 3(a)–(e) of the algorithm in the previous section.

(b) Set $BlockNum = 1$.

(c) Set $XF[1] = x_0$, $XL[1] = x_{op}$, $YF[1] = y_0$, $YL[1] = y_{op}$.

(d) Set $I(x, y) = 0 \ \forall \ x \in [XF[1] \dots XL[1]] \wedge y \in [YF[1] \dots YL[1]]$.

Step 3: Set $iter = 1$.

Step 4: For every (x, y) belonging to the surrounding line of the block with opposite vertices at $(XF(iter), YF(iter))$ and $(XL(iter), YL(iter))$ and obeying $I(x, y) = 1$:

(a) Find the opposite vertex (x_{op}, y_{op}) of the "best fitting block" at (x, y), following steps 3(a)–(e) of the algorithm in the previous section.

(b) Set $BlockNum = BlockNum + 1$.

(c) Set $XF[BlockNum] = x$, $XL[BlockNum] = x_{op}$, $YF[BlockNum] = y$, $YL[BlockNum] = y_{op}$.

(d) Set $I(x, y) = 0 \ \forall \ x \in [XF[BlockNum] \dots XL[BlockNum]] \wedge \ y \in [YF[BlockNum] \dots YL[BlockNum]]$.

Step 5: If $iter < BlockNum$ then $iter = iter + 1$ and go to Step 4.

Once the process has been completed, the coordinates of all blocks that form the first object of the image are stored in the matrices $XF[i]$, $XL[i]$, $YF[i]$, $YL[i]$, for $i = 1 \dots BlockNum$. The above process is repeated until all image objects have been segmented.

5. Experimental results

In order to demonstrate the efficiency of our methods, we apply them to various image processing problems. All experiments have been performed using a Pentium processor at 150 MHz. For both PHT and BHT, we use look-up tables for all function calculations involving $\cos\theta$ and $\sin\theta$ in order to speed up all processes. In all cases, the Hough field produced by using BHT is very similar to that produced by using PHT. This is clearly shown in Fig. 6, where the original image, the Hough transform field for PHT, the Hough transform field for BHT and the relative difference between the two fields (difference between BHT and PHT field values divided by PHT field value) in the vicinity of the maximum are presented. The presence of a non-zero relative difference is due to the discrete image grid and amounts to less that 1.5% per cell value in the vicinity of local maxima in the HT field.

It has become obvious in Section 3 that the BHT method offers more economy in terms of CPU time when large rectangular blocks are present in the processed image. It is therefore natural that more economy can be achieved for image processing applications whereby edge detection is not appropriate as a preprocessing tool, since it is more likely that large blocks can be found in such applications. Our first two examples fall into this general category of applications. The third example is a document skew detection application which can be dealt with using either the original or the edge image. In our final two examples it is beneficiary to use the edge image. In all examples we compare the CPU time taken by block decomposition (or segmentation) followed by BHT to the time taken by PHT.

Example 1 (*Axis determination*)**.** In the first application, HT is used for estimating the central axis of a digitized skeleton image, shown in Fig. 7a. This is a 395×719 pixels image scanned at 300 dpi. In this case, there is no point in preprocessing the image using edge detection, since the desired axis lies in the interior of the object. Instead, it is beneficial to employ RLSA preprocessing [18–20], which is a one-pass algorithm facilitating large block extraction without slowing down the preprocessing stage. Fig. 7b shows the same image after thresholding, while Fig. 7c shows the result of RLSA smoothing and the direction obtained using the PHT or BHT algorithm. For the application of BHT, rectangular blocks are extracted using the algorithm of Section 4.1. Table 1 shows results concerning processing times using PHT and BHT (the latter being applied for rectangular blocks that pass the criterion of relation (9). Preprocessing time (i.e. time spent for RLSA plus time spent for block identification), time consumed by each HT variant and total processing time are shown. The fastest time (0.6 s) is obtained using BHT in conjunction with RLSA preprocessing, and is less than 1/5 of the best time obtained using PHT (3.5 s).

Example 2 (*Segmentation and axis determination*)**.** In the second application, HT is used for identifying the axes of elongated objects in the field of view of a camera. We process a 512×512 pixel field of view of a CCD camera depicting a number of discrete objects (keys), shown in Fig. 8a. Again, only RLSA preprocessing is used, whose result is shown in Fig. 8b. There is no point in using edge detection, since the desired axes lie in the interior of the objects (keys). In this case, it is useful to segment the image into its individual connected components before applying the BHT to each object. To this end, we use the block segmentation algorithm of Section 4.2. The result of axis identification using BHT is shown in Fig. 8c. In order to apply the PHT on the image for reasons of comparison, we have first segmented it using connected component analysis (this counts as part of the preprocessing time). In Table 2, performances of BHT and PHT in terms of processing time are shown. Clearly, the best time obtained by BHT (0.9 s) is equal to 1/7 of the fastest time obtained by PHT (6.3 s) in this application.

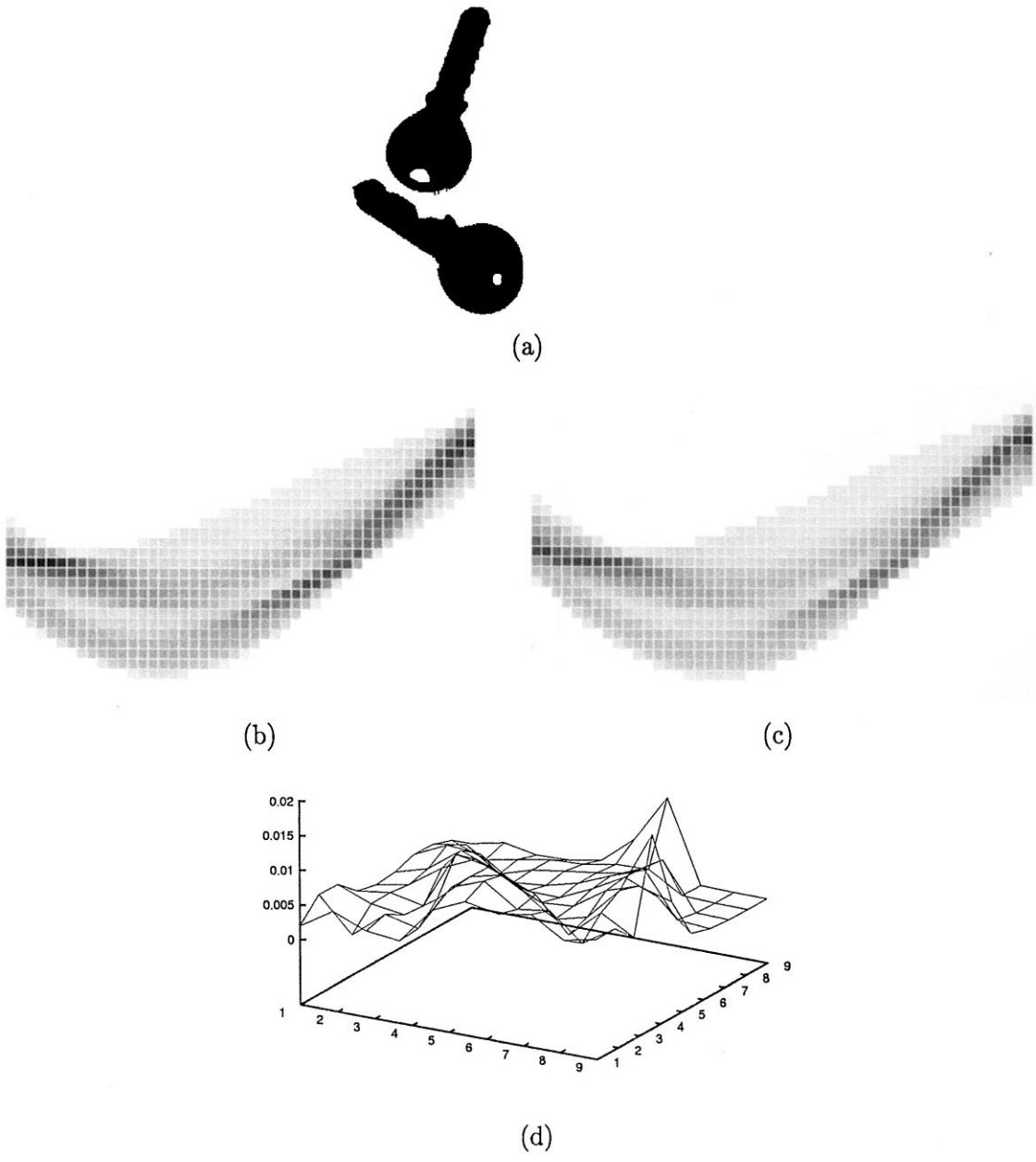


Fig. 6. Comparison of PHT and BHT accumulator arrays. (a) original image; (b) PHT accumulator array; (c) BHT accumulator array; (d) relative difference between BHT and PHT accumulator arrays in the vicinity of the maximum.

Example 3 (*Skew detection*)**.** There are a number of line finding problems that can be solved with some efficiency by application of HT either on the original binary image or an image resulting from the original image after an edge detection method has been applied. A characteristic example is estimation of the skew of a digital image document.

HT has been applied successfully to this kind of problem, achieving high detection accuracies, but it is rather slow because of its high computational complexity [21–25]. Fig. 9 shows a digitized document with a skew angle of 3° and its skew corrected form using HT. This is a 2200×1545 pixel image scanned at 200 dpi. It corresponds to a mixed document containing text with fonts of different sizes and an image. In order to apply the HT, we have used various preprocessing methods which can be useful for accelerating the skew detection process. More specifically, before applying the PHT to the image, we

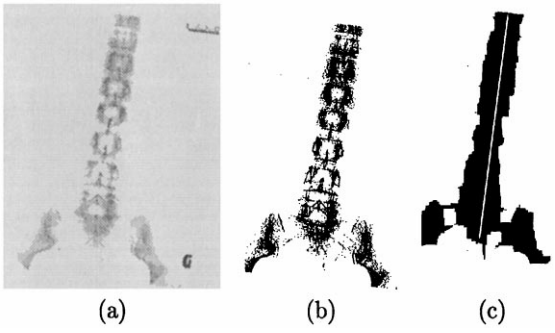


Fig. 7. Linear direction evaluation using rectangular block decomposition; (a) original image; (b) thresholded image; (c) final smoothed image and prevalent linear direction.

Table 1
Processing times for PHT and BHT applied on the image of Fig. 7a. Preprocessing time, actual HT evaluation time and total processing time are shown

	PHT time (s)			BHT time (s)		
RLSA	Pre-Proc.	HT	Total	Pre-Proc.	HT	Total
No	0	3.5	3.5	0.2	1.7	1.9
Yes	0.3	8.5	8.8	0.5	0.1	0.6

employ edge detection and RLSA preprocessing, as well as their combination (RLSA followed by edge detection). The resulting images are shown in Fig. 10. For BHT application, blocks are extracted using the block decomposition algorithm of Section 4.1.

To find the prevalent orientation using the HT variants, we first compute the HT accumulator array (400 values of r were used and θ varied from -5 to 5° in increments of 0.1°). Then, in order to eliminate cells with small contributions, we apply a lower threshold T_0 which is taken equal to the average value of the contribution of all cells in the accumulator array. Finally, we add the contributions of all remaining cells that correspond to a certain value of θ [21]. The result is a one-dimensional array (integrated accumulator array) of values, with each value corresponding to a certain θ. The skew angle can be estimated by finding the maximum value in this array.

In this case it is useful to assess the performance of BHT both in terms of accuracy in the determination of the desired linear direction and in terms of economy in CPU time. To this end, we show in Table 3 detailed results concerning processing time and determined skew angle for four different orientations of the initial document in increments of 1°. Preprocessing time (i.e. time spent for RLSA and/or edge detection plus time spent for block identification), time consumed by the HT variant and total processing time are shown.

It is evident from Table 3 that PHT and BHT application leads to very similar determinations of the skew angle. For three of the preprocessing tool combinations (no preprocessing, edge detection only, RLSA only) both HT variants (PHT and BHT) have always corrected the skew with maximum error of 0.1°. By contrast, the fourth preprocessing method (RLSA followed by edge detection) leaves too little information in the resulting image (see Fig. 10c), so that the obtained angles are often unacceptable (errors of up to 0.5° occur).

For all preprocessing method combinations, BHT is faster than PHT. The ratio of CPU time taken by BHT to CPU time taken by PHT is approximately 1:3 (for the original image), 1:16 (for RLSA preprocessing), 1:1.2 (for edge detection) and 1:2 (for combined RLSA and edge detection). Of the three preprocessing methods that

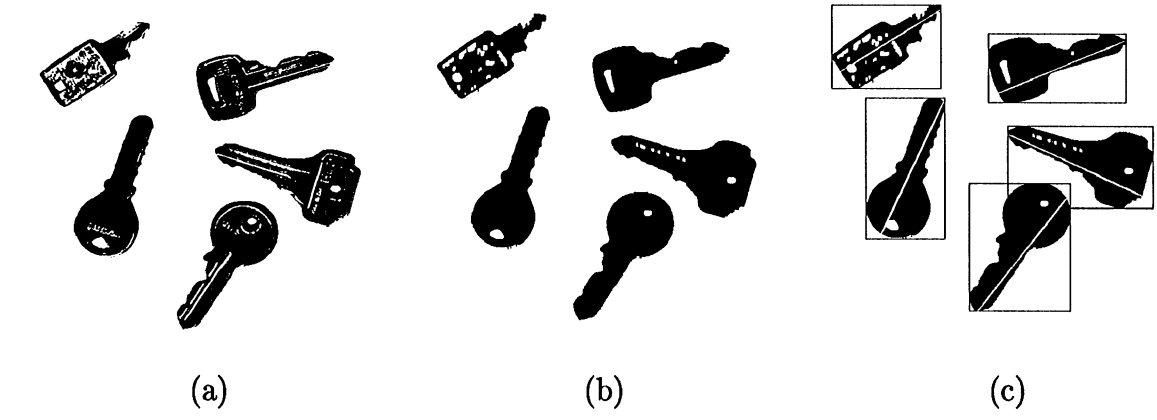


Fig. 8. Image segmentation and linear direction evaluation using rectangular block decomposition: (a) original image; (b) smoothed image using RLSA; (c) final segmented image and prevalent linear directions.

Table 2
Processing times for PHT and BHT applied on the image of Fig. 8a

	PHT time (s)			BHT time (s)		
RLSA	Pre-Proc.	HT	Total	Pre-Proc.	HT	Total
No	0.2	6.1	6.3	0.2	1.3	1.5
Yes	0.5	8.4	8.9	0.6	0.3	0.9

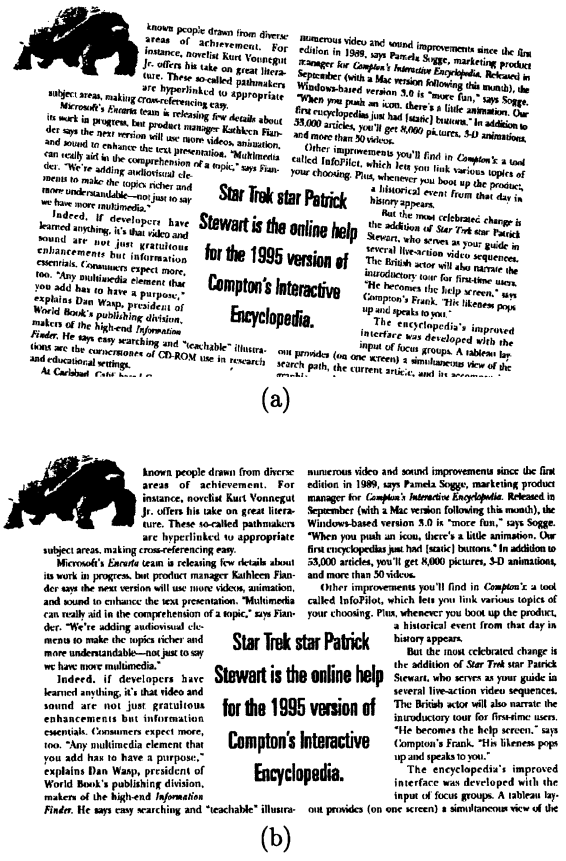


Fig. 9. Skew detection using HT; (a) original document. (b) skew detected and corrected version.

give acceptable results in terms of skew detection accuracy, the best method for BHT (RLSA preprocessing) takes 6–7 s, while the best method for PHT (edge detection) takes 21 s, and therefore the BHT method offers a threefold CPU time economy when compared with PHT.

Example 4 (*Application of HT variants on edge image*)**.** Next, we consider processing of images whereby strong edge gradients are present and therefore edge detection is essential [7]. In this case BHT is still applicable, and, as explained in the beginning of this section, gives the same

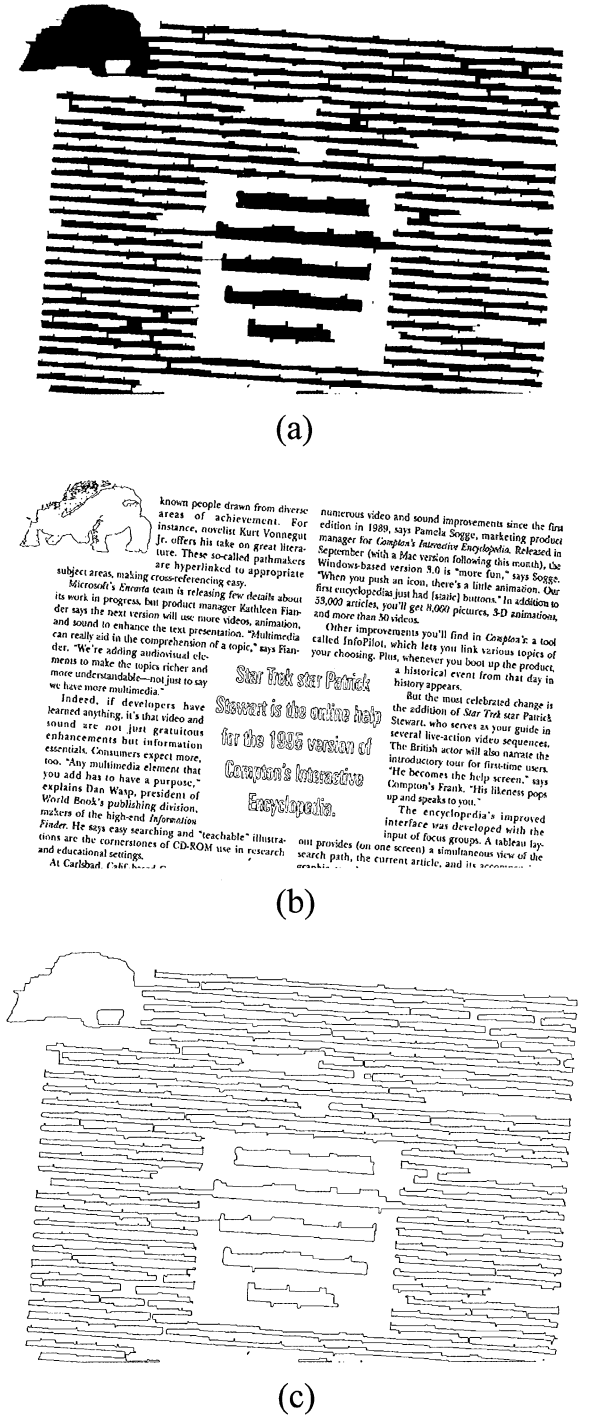


Fig. 10. Results of preprossesing for the document image of Fig. 9a: (a) image after application of RLSA; (b) image after edge detection; (c) image after application of RLSA and edge detection.

Table 3
Skew detection results obtained after processing the document of Fig. 9a positioned at four different orientations (true skew angle of 0–3° at increments of 1°). Different preprocessing tools (RLSA and edge detection — ED) and HT versions (PHT and BHT) are applied on the images. Results concerning point-block statistics, processing time and estimation of skew angle are shown

			PHT					BHT					
				Time (s)						Time (s)			
Skew (True)	RLSA	ED	Points	Pre-Proc.	HT	Total	Skew (estimate)	Blocks	Points	Pre-Proc	HT	Total	Skew (estimate)
0°	No	No	324 894	0	37	37	0°	18 191	12 948	2	8	10	0°
0°	Yes	No	997 657	3	110	113	− 0.1°	1653	184	5	1	6	− 0.1°
0°	No	Yes	178 564	2	19	21	− 0.1°	15 004	65 274	4	12	16	0°
0°	Yes	Yes	94 147	5	11	16	− 0.5°	1952	1982	7	1	8	− 0.4°
1°	No	No	323 502	0	37	37	1°	19 086	14 642	2	8	10	1.1°
1°	Yes	No	999 768	3	110	113	0.9°	2367	285	5	2	7	1°
1°	No	Yes	177 839	2	19	21	1°	15 217	67 406	4	13	17	1.1°
1°	Yes	Yes	93 175	5	11	16	0.6°	2599	2065	7	1	8	0.5°
2°	No	No	317 046	0	36	36	2°	19 758	16 439	2	9	11	2.1°
2°	Yes	No	979 851	3	109	112	1.9°	2944	417	5	2	7	2°
2°	No	Yes	175 021	2	19	21	2°	15 133	69 239	4	13	17	2.1°
2°	Yes	Yes	91 411	5	11	16	1.6°	3168	2339	7	1	8	1.7°
3°	No	No	310 860	0	35	35	3°	20 510	18 131	2	10	12	3.1°
3°	Yes	No	963 757	3	107	110	2.9°	3496	559	5	2	7	3°
3°	No	Yes	172 608	2	19	21	3°	15 118	70 990	4	14	18	3.1°
3°	Yes	Yes	89 571	5	11	16	2.6°	3698	2684	7	2	9	2.6°

prevalent lines as PHT (since their accumulator fields are almost identical). An interesting question is whether BHT is competitive to PHT in terms of processing time. Given a rectangular block, the BHT algorithm checks whether condition (9) is satisfied. If it is satisfied, the contribution of the whole block to the accumulator array is calculated, with CPU time advantage over the corresponding application of PHT. If, on the other hand, the condition is not satisfied, the contribution of each block point is calculated separately and the consumed CPU time is equal to the CPU time taken by PHT. In the most adverse case that no acceptable blocks are found, BHT can in principle be somewhat slower than PHT because of the overhead CPU time needed to find the blocks and check if condition (9) is satisfied for each block. However, even though no blocks of very large area can be found in edge detected images, enough line segments of adequate length usually pass the criterion of relation (9) and therefore some acceleration is observed. The skew detection example confirms this statement. Indeed, as can be seen in Table 3, when only edge detection is applied, a large number (about 15 000) of blocks that pass the criterion of relation (9) is still formed and BHT is still a little faster than PHT (CPU times for BHT vary between 16 and 18 s, while CPU time for PHT is 21 s).

A more typical example of this case is the edge image of Fig. 11. Results of the application of HT variants on this edge image are shown in the first row of Table 4. We found that about 2/3 of the total number of foreground pixels in the image (20 360 out of 29 833) are grouped into elongated blocks that pass the criterion of relation (9) and BHT is still faster than PHT (3.5 s are needed for BHT, while 4.3 s are needed for PHT). Our experiments support the conclusion that BHT is usually faster than PHT even when edge detection is used and can therefore

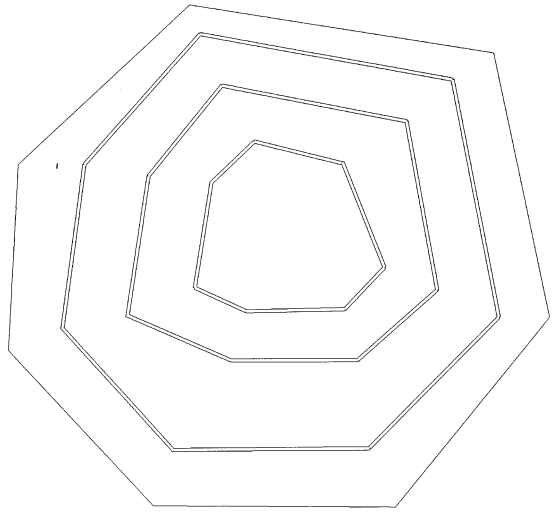


Fig. 11. An edge image with strong edge gradient information.

Table 4
Results of the application of PHT and BHT on the edge image of Fig. 11. PHT and BHT are applied before and after the angle θ is restricted using edge gradient information and processing times are displayed

	PHT time (s)			BHT time (s)		
	Pre-Proc.	HT	Total	Pre-Proc.	HT	Total
Angle not restricted	0	4.3	4.3	1.7	1.8	3.5
Angle restricted	0.8	1.2	2.0	2.0	0.4	2.4

be used safely in most applications where identification of linear formations is required.

Finally, we note that in the case of edge detected images, gradient information can be used to restrict the span of the angle θ that must be examined for filling in the PHT accumulator array cells [5,6]. With the application of 3×3 edge detector operators (e.g. Sobel, Prewitt) to the edge image, this span is reduced to a total of $\pi/4$ [1]. As a result, the processing time taken by PHT is also reduced. The decomposition of the edge image into rectangular blocks can also be used to impose constraints on the angular directions of the accumulator array cells that must be actually incremented. Clearly, for a rectangular block with $X > Y$, the larger the ratios of X/Y, the more horizontal the corresponding edge. It is reasonable to increment only accumulator array cells corresponding to θ with

$$\frac{\pi}{2} - \tan^{-1}\frac{Y}{X} \leqslant \theta \leqslant \frac{\pi}{2} + \tan^{-1}\frac{Y}{X}. \tag{12}$$

Similarly, $X < Y$ corresponds to a more or less vertical edge and therefore only accumulator array cells corresponding to θ with

$$0 < \theta \leqslant \tan^{-1}\frac{Y}{X} \text{ or } \pi - \tan^{-1}\frac{Y}{X} \leqslant \theta \leqslant \pi \tag{13}$$

need be incremented. The corresponding span of θ can be less than the value of $\pi/4$ imposed on the angle θ by gradient information obtained using the common 3×3 edge filters (Sobel, Prewitt). This restriction of the angle θ can lead to a further significant reduction of the processing time taken by the BHT.

Consider, for example, the edge image of Fig. 11. Apart from the application of PHT and BHT as before, the image is processed using two further methods. The first method (angle restricted PHT) involves application of the Prewitt edge detector operators to the neighborhood of each foreground pixel in order to restrict the angle θ, and subsequent application of PHT. The second method (angle restricted PHT) involves application of the block decomposition algorithm of Section 4.1 to the edge image. For blocks passing the criterion of relation (9), the angle θ is constrained using relations (12) or (13). As usual, blocks not passing the criterion are decomposed into points and PHT is applied to them with the angle θ restricted using the Prewitt operators. Results of the application of different HT variants on the image of Fig. 11 are also shown in Table 4, in the row labelled "angle restricted". Note that further restriction of the angle using the block decomposition has led to a very significant reduction in the CPU time taken by the HT application itself (from 1.2 s for PHT to 0.4 s for BHT). When the overhead preprocessing time needed to decompose the image into blocks and restrict the angle is also taken into account, the total CPU time is still comparable to the total time for the PHT application (2.4 s for BHT versus 2.0 s for PHT).

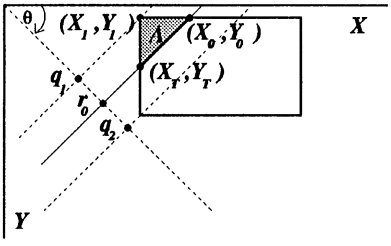
6. Conclusion and further research

In this paper, algorithms for binary image decomposition into rectangular blocks of foreground pixels and for image segmentation based on this decomposition were introduced. A novel fast method was presented for applying the HT to binary images, based on rectangular block decomposition. The efficiency of the combination of these approaches in determining prevalent linear features in binary images was demonstrated in various image processing applications, where significant acceleration was achieved compared to the classical PHT implementation.

As it becomes evident from the results in the experimental section, the algorithms described in this paper offer most economy in terms of CPU time in cases where edge detection is unacceptable as a preprocessing tool. Such a case, which we are currently investigating, is the application of the BHT algorithm to the problem of determining linear formations of geological interest using thresholded images of geophysical maps and satellite images in the framework of a GIS system. In this application, the desired linear formations coincide with the axes of elongated foreground objects or clusters of objects. Also of interest is the extention of the method to the location of other geometrical objects besides straight lines. As the parametric description of the desired geometrical shape becomes more complicated, the computational benefit from the block decomposition becomes less significant. However, preliminary calculations show that there is considerable benefit in applying the rectangular block decomposition method for locating conical sections (second degree curves), a problem with many practical applications. Finally, it would be interesting to extend the rectangular block decomposition method for the calculation of shape descriptors (various types of moments, Fourier descriptors, etc.) and to systematically study the benefits in computational complexity gained by this application.

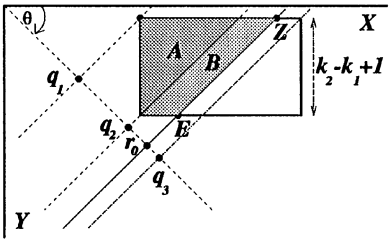
Appendix

A sketch of the derivation of the BHT algorithm is given forthwith.



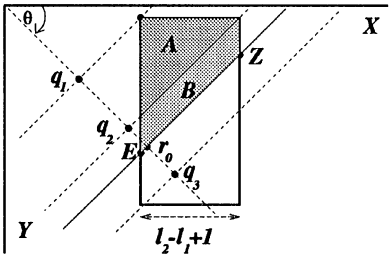
(a) $q_1 \leqslant r_0 < q_2$

$$\left.\begin{aligned} &E(r_0) = A = \frac{X_0 - X_1}{Y_T - Y_1} \\ &Y_1 = Y_0 \Rightarrow r_0 - q_1 = \cos\theta(X_0 - X_1) \\ &\tan\theta = \frac{X_0 - X_1}{Y_T - Y_1} \end{aligned}\right\} \Rightarrow E(r_0) = \frac{(r_0 - q_1)^2}{2\cos\theta\sin\theta}$$



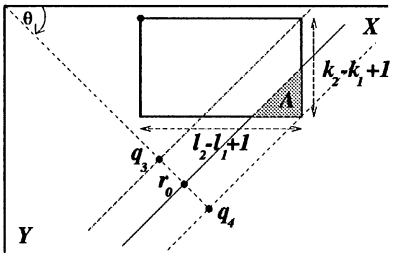
(b) $q_2 \leqslant r_0 < q_3 \wedge Y_1 = Y_2$

$$\left.\begin{aligned} &E(r_0) = A + B \\ &A = \frac{(q_2 - q_1)^2}{2\cos\theta\sin\theta} \\ &B = (EZ)\ (r_0 - q_2) \\ &\cos\theta = \frac{k_2 - k_1 + 1}{(EZ)} \end{aligned}\right\} \Rightarrow E(r_0) = \frac{(q_2 - q_1)^2}{2\cos\theta\sin\theta} + \frac{(r_0 - q_2)\ (k_2 - k_1 + 1)}{\sin\theta}$$



(c) $q_2 \leqslant r_0 < q_3 \wedge X_1 = X_2$

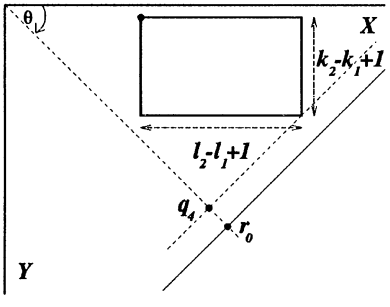
$$\left.\begin{aligned} &E(r_0) = A + B \\ &A = \frac{(q_2 - q_1)^2}{2\cos\theta\sin\theta} \\ &B = (EZ)\ (r_0 - q_2) \\ &\sin\theta = \frac{l_2 - l_1 + 1}{(EZ)} \end{aligned}\right\} \Rightarrow E(r_0) = \frac{(q_2 - q_1)^2}{2\cos\theta\sin\theta} + \frac{(r_0 - q_2)\ (l_2 - l_1 + 1)}{\cos\theta}$$



(d) $q_3 \leqslant r_0 < q_4$

$$\left.\begin{aligned} &E(r_0) = (k_2 - k_1 + 1)\ (l_2 - l_1 + 1) - A \\ &A = \frac{(q_4 - r_0)^2}{2\cos\theta\sin\theta} \end{aligned}\right\} \Rightarrow$$

$$\Rightarrow E(r_0) = (k_2 - k_1 + 1)\ (l_2 - l_1 + 1) - \frac{(q_4 - r_0)^2}{2\cos\theta\sin\theta}$$



(e) $r_0 \geqslant q_4$

$$E(r_0) = (k_2 - k_1 + 1)\ (l_2 - l_1 + 1)$$

References

[1] M. Sonka, V. Hlavac, R. Boyle, Image Processing, Analysis and Machine Vision, Chapman & Hall, London, 1993.

[2] M.H. Singer, A general approach to moment calculation for polygons and line segments, Pattern Recognition 26 (1993) 1019–1028.

[3] D. Wang, V. Haese-Coat, J. Ronsin, Shape decomposition and representation using a recursive morphological operation, Pattern Recognition 28 (1995) 1783–1792.

[4] R.D. Duda, P.E. Hart, Use of the Hough transform to detect lines and curves in pictures, Commun. ACM 15 (1972) 11–15.

[5] D.H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, Pattern Recognition 13 (1981) 111–122.

[6] D.H. Ballard, D. Sabbah, Viewer independent shape recognition, IEEE Trans. Pattern Anal. Machine Intelligence 5 (1983) 653–659.

[7] X. Lin, W.G. Wee, SDFS: a new strategy for the recognition of object using range data, Proc. Int. Conf. Robotics and Automation, 1986, pp. 770–775.

[8] C.S. Kannan, H.Y.H. Chuang, Fast Hough transform on a mesh connected processor array, Inform. Process. Lett. 33 (1990) 243–248.

[9] H.Y.H. Chuang, C. Ling, An efficient Hough transform algorithm on SIMD hypercube, Proc. Intern. Conf. on Parallel and Distributed Systems, Hsinchu, Taiwan, 1994, pp. 236–241.

[10] S. Olariu, J. L. Schwing, J. Zhang, Computing the Hough transform on reconfigurable meshes, Image Vision Comput. 11 (1993) 623–628.

[11] W.A. Götz, H.J. Druckmüller, A fast digital radon transform — an efficient means for evaluating the Hough transform, Pattern Recognition 28 (1995) 1985–1992.

[12] L. Yang, Z. He, Detection of line segments using a fast dynamic Hough transform, Proc. IEEE Int. Symp on Circuits and Systems, Chicago, IL, USA, Vol. 1, 1993, pp. 543–546.

[13] M.A. Li Lavin, R.J. LeMaster, Fast Hough transform: a hierarchical approach, Comp. Vision Graph. Image Process. 36 (1986) 139–161.

[14] L.T.S. Lam, W.C.Y. Lam, D.N.K. Leung, A modified Hough transform: knowledge-based boundary convergence algorithm, Proc. 2nd Int. Conf. Image Processing, Singapore, 1992, pp. 226–230.

[15] N. Guil, J. Villalba, E.L. Zapata, A fast Hough transform for segment detection, IEEE Trans. Image Process. 4 (1995) 1541–1548.

[16] S.C. Jeng, W.H. Tsai, Fast generalized Hough transform, Pattern Recognition Lett. 11 (1990) 725–733.

[17] L. Ke-qing, T. Qi, A fast Hough transform for inspecting accurate needle-type meter gauges, Proc. IAPR Workshop on Computer Vision, 1988, pp. 195–198.

[18] K.Y. Wong, R.G. Casey, F.M. Wahl, Document analysis system, IBM J. Res. Devel. 26, (1982) 647–656.

[19] F.M. Wahl, K.Y. Wong, R.G. Casey, Block segmentation and text extraction in mixed text/image documents, Comput. Graphics Image Process. 20 (1982) 375–390.

[20] D. Wang, S.N. Shihari, Classification of newspaper image blocks using texture analysis, Computer Vision Graphics Image Process. 47 (1989) 327–352.

[21] B. Yu, A.K. Jain, A robust and fast skew detection algorithm for generic documents, Pattern Recognition 29 (1996) 1599–1629.

[22] D.S. Le, G.R. Thoma, H. Wechsler, Automated page orientation and skew angle detection for binary document images, Pattern Recognition 27 (1994) 1325–1344.

[23] H. Yan, Skew correction of document images using interline cross-correlation, CVGIP: Graphical Models Image Process. 55 (1993) 538–543.

[24] S.C. Hinds, J.L. Fisher, D.P. d'Amato, A document skew detection method using run-length encoding and the Hough transform, Proc. 10th Int. Conf. on Pattern Recognition 1990, pp. 464–468.

[25] S.N. Shihari, V. Govindaraju, Analysis of textual images using the Hough transform, Mach. Vis. Appl. 2 (1989) 141–153.

About the Author—STAVROS J. PERANTONIS was born in Athens, Greece, in 1960. He received his Diploma Degree in Physics from the University of Athens in 1984. He also holds the M.Sc. Degree in Computer Science from the University of Liverpool and the D. Phil. Degree in Physics from the University of Oxford. He is currently a member of the Research staff at the Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", Athens, Greece. His research interests involve image processing as well as the theory and applications of neural networks. Dr. Perantonis has authored or coauthored more than 60 publications in journals, book chapters and international conference proceedings. He has also acted as principal researcher or participated in numerous national or EU funded projects.

About the Author—BASILIOS GATOS was born in Athens, Greece, in 1967. He received his Electrical Engineering Diploma in 1992, and his Ph.D. degree in 1998, both from the Electrical and Computer Engineering Department of Democritus University of Thrace, Xanthi, Greece. His Ph.D. thesis is on Optical Character Recognition Techniques. In 1993 he was awarded a scholarship from the Institute of Informatics and Telecommunications, NCSR "Demokritos", where he worked till 1996. Dr. Gatos is currently working at Lambrakis Press S.A. in the field of digital preservation of old newspapers. His main research interests are in image processing and document image analysis, OCR and pattern recognition. He has 15 publications in journals and international conference proceedings and has participated in three research programs funded by the European community. Dr. Gatos is a member of the Technical Chamber of Greece.

About the Author—NIKOS PAPAMARKOS was born in Alexandroupoli, Greece, in 1956. He received his Diploma Degree in Electrical and Mechanical Engineering from the University of Thessaloniki, Thessaloniki, Greece, in 1979 and the Ph.D. Degree in Electrical Engineering in 1986, from the Democritus University of Thrace, Greece. From 1987 to 1990 Dr. Papamarkos was a Lecturer, from 1990 to 1996 Assistant Professor in the Democritus University of Thrace where he is currently Associate Professor since 1996. During 1987 and 1992 he has also served as a Visiting Research Associate at the Georgia Institute of Technology, USA. His current research interests are in digital signal processing, filter design, optimization algorithms, image processing, pattern recognition and computer vision. Dr Nikos Papamarkos is a member of IEEE and a member of the Greek Technical Chamber.