PERGAMON

Pattern Recognition 34 (2001) 1751–1764

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Linear band detection based on the Euclidean distance transform and a new line segment extraction method

Jeong-Hun Jang, Ki-Sang Hong*

Image Information Processing Laboratory, Electrical and Computer Engineering Division, POSTECH, San 31, Hyoja-dong, Nam-Ku, Pohang, Kyungbuk 790-784, South Korea

Received 25 June 1999; received in revised form 22 June 2000; accepted 22 June 2000

Abstract

A linear band, which is a straight line segment with some width (i.e., thickness), is a more structured, higher-level feature compared to edge or line features. In spite of the usefulness of linear bands as features, papers dealing with their detection problem are rare. In this paper, we propose a new method for detecting linear bands in gray-scale images. We first talk about our opinion on what types of linear bands a desirable detector should be able to detect, and then give a description on how we designed our detector to achieve the goal. Our method consists largely of two parts: (1) extracting the candidate center line pixels of the linear bands contained in an input gray-scale image (sub-parts: edge detection, Euclidean distance transform, ridge detection in a distance map, and noisy ridge pixel removal), (2) extracting line segments from the result of (1) using our new line segment detection method (sub-parts: modified Hough transform, base line segment grouping, redundant line segment removal, and postprocessing). Experimental results show that our method is practical and robust. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Linear band detection; Line segment detection; Euclidean distance transform; Ridge detection; Modified Hough transform; Base line segment grouping

Nomenclature

b_i	the ith base line segment.
$L(b_i)$	the index number of b_i (i.e., $L(b_i) = i$).
$N(b_i)$	the number of member pixels of b_i.
l_i	the ith line segment.
$L(l_i)$	the index number of l_i (i.e., $L(l_i) = i$).
$N(l_i)$	the number of member pixels of l_i.
$L(\mathbf{p})$	the label assigned to the pixel $\mathbf{p}$.
P	a 1D array containing pixel coordinates.
r_j	the jth run in p.
$L(r_j)$	the label assigned to r_j.
$N(r_j)$	the number of member pixels of r_j.
S	a list of base line segments chosen along the guide line g.
$S(\mathbf{p})$	a set of line segments sharing $\mathbf{p}$ as a common member pixel.

1. Introduction

Feature extraction in gray-scale images is one of the important research topics in the computer vision and pattern recognition communities. Properly extracted features are the cornerstones of successful higher-level processing such as matching and recognition. Edges, lines, and corners are frequently used as features because of their generality, and many papers have been published concerning them.

In this paper, we consider the problem of detecting another useful feature, called a *linear band*, in a gray-scale

* Corresponding author. Tel.: + 82-54-279-2216; fax: + 82-54-279-5586.

E-mail addresses: jeonghun@postech.ac.kr (J.-H. Jang), hongks@postech.ac.kr (K.-S. Hong).

image. A linear band is a straight line segment with some width (i.e., thickness). It is usually found in images taken from artificial objects or structures such as roads, buildings, etc. It can be said that a linear band is a more structured, higher-level feature compared to edge or line features.

In spite of the usefulness of linear bands as features, papers dealing with their detection problem are rate. Lo and Tsai [1] proposed the gray-scale Hough transform, which is an extension of the standard Hough transform, to detect linear bands in gray-scale images. They first show how their method works when applied to a binary image containing linear bands, and then modified the method to apply it to a gray-scale image containing different gray levels of liner bands. Their basic idea is to use all the pixels when voting in parameter space to avoid image thresholding or thinning. Their proposed parameter space is three-dimensional, i.e., a third axis for gray-scale values is added to the conventional parameter space. Though their idea is interesting, we think that many problems may occur when using it in practical applications. Their method is expected to be slow due to a large number of votes, and requires a large amount of memory due to the 3D structure of the parameter space. Also, it does not seem to be able to handle linear bands of non-uniform brightness and width.

Our method is practical and applicable to real, natural images. Moreover, our method is flexible and robust. To clarify the problem to be solved, four basic types of linear bands we need to find are shown in Fig. 1. In Fig. 1(a), the linear band is brighter than the area on either side. On the contrary, the linear band in Fig. 1(b) is darker than the area on either side. The linear band in Fig. 1(c) is brighter than the area on one side, but darker than the area on the other side. Whereas these three types are distinguished based on their cross-sectional shapes, the linear band in Fig. 1(d) is different from the others in that its width gradually varies along its center line. In addition, the brightness of the linear bands mentioned above may not be uniform. Our goal is to design a method that is able to detect such linear bands. The algorithm should also allow a user to choose a linear band of desired width and length.

The overall flow of our method is shown in Fig. 2. We first obtain an edge map from an input gray-scale image and then perform the Euclidean distance transform on the edge map to get a distance map. The distance map provides rich information that is helpful in finding the desired linear bands, for example, center line positions or widths of the linear bands. In fact, the basic idea of introducing the Euclidean distance transform originates from the curvilinear structure detection algorithm proposed by Jang and Hong [2]. In the next step, ridge pixels (i.e., local maxima) are detected from the distance map. The ridge pixels serve as candidate center line pixels of the linear bands. Noisy ridge pixels are filtered out by considering the characteristics of each ridge pixel. Then proper line segments are extracted from the ridge map using our new line segment detection method proposed in this paper. Our method overcomes several weaknesses of the conventional Hough transform technique. Each line segment found serves as a kind of backbone of the corresponding linear band. Finally, the line segments (i.e., linear bands) are split by some rule and unnecessary ones are removed. Optionally, the least-squares line fitting technique may be used to obtain correct end points of the line segments. Then a user can select the linear bands of desired width and length using the information obtained in the above steps.

This paper is organized as follows. In Section 2, the proposed method is described step by step. At the end of each step, an example of the processing result is shown. Experimental results for some real images are shown in Section 3. Conclusions are given in Section 4.

2. Proposed method

2.1. Detecting edgels

The proposed algorithm begins with the detection of edgels in an input gray-scale image. Detected edgels serve as boundary pixels of meaningful structures. In our work, the *Canny edge detector* [3] is used because of its scale-space representation capability. Canny's method consists

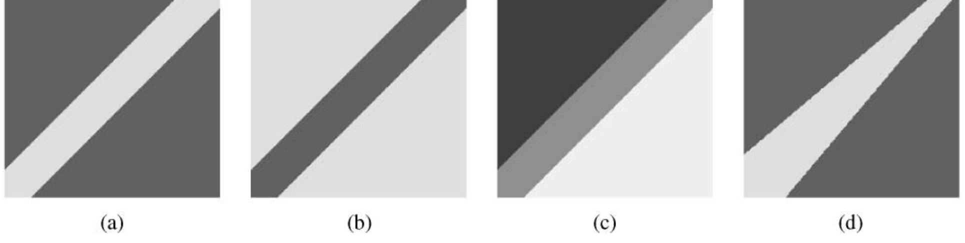


Fig. 1. Four types of linear bands to be detected.

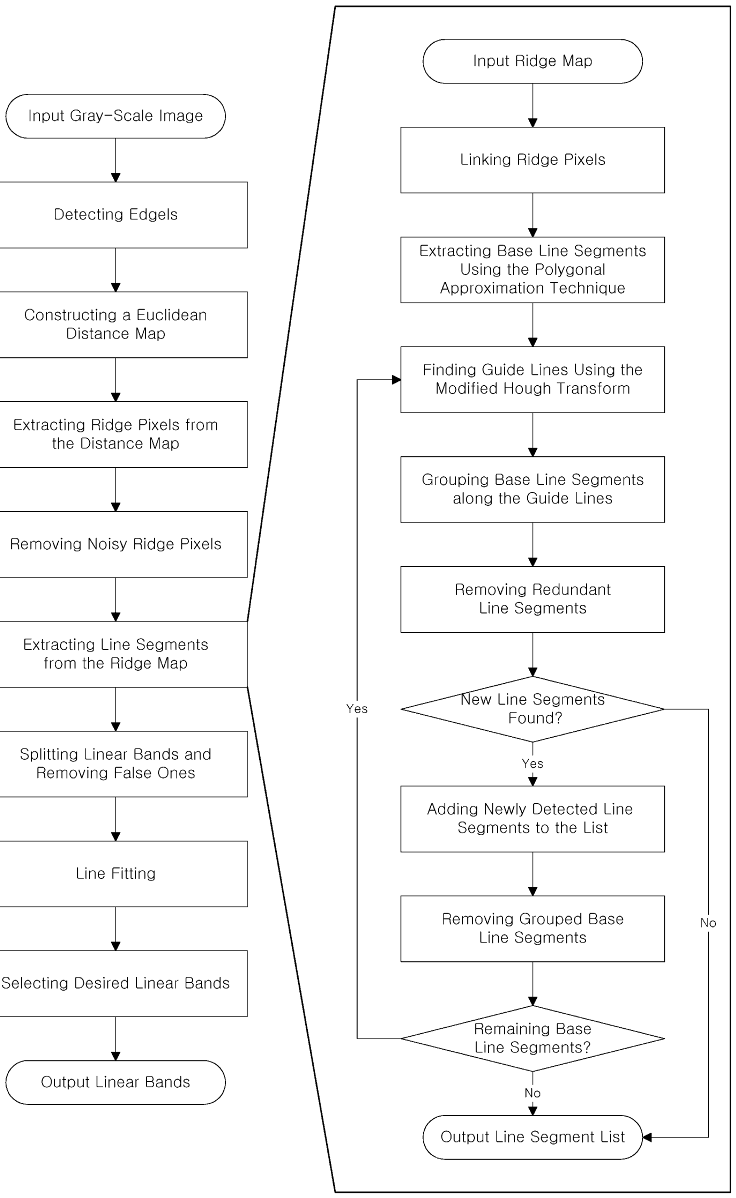

Input Gray-Scale Image
Detecting Edgels
Constructing a Euclidean Distance Map
Extracting Ridge Pixels from the Distance Map
Removing Noisy Ridge Pixels
Extracting Line Segments from the Ridge Map
Splitting Linear Bands and Removing False Ones
Line Fitting
Selecting Desired Linear Bands
Output Linear Bands
Input Ridge Map
Linking Ridge Pixels
Extracting Base Line Segments Using the Polygonal Approximation Technique
Finding Guide Lines Using the Modified Hough Transform
Grouping Base Line Segments along the Guide Lines
Removing Redundant Line Segments
New Line Segments Found?
Yes
Adding Newly Detected Line Segments to the List
Removing Grouped Base Line Segments
Remaining Base Line Segments?
Yes
No
No
Output Line Segment List

Fig. 2. Overall flow of the proposed method.

mainly of four parts: Gaussian smoothing, gradient computation, non-maxima suppression, and hysteresis thresholding. Three parameters are involved in the algorithm: the size of a Gaussian kernel σ in Gaussian smoothing, and two thresholds T_{h1} and T_{h2} ($T_{h1} < T_{h2}$) in hysteresis thresholding.

An input gray-scale image is given in Fig. 3(a), which will be used as an example throughout this paper. Its edge image is shown in Fig. 3(b), where the parameter values used are $\sigma = 1.5$, $T_{h1} = 1.0$, and $T_{h2} = 4.0$.

2.2. *Constructing a Euclidean distance map*

After obtaining an edge map from the input image, the Euclidean distance transform is carried out on the edge map. We slightly modified the *region growing Euclidean distance transform* algorithm proposed by Cuisenaire [4] to get a distance map. Each pixel site of the resulting distance map has a value of a distance to the nearest edgel and the position of that edgel. The algorithm starts computing a distance at each edgel, and the computation

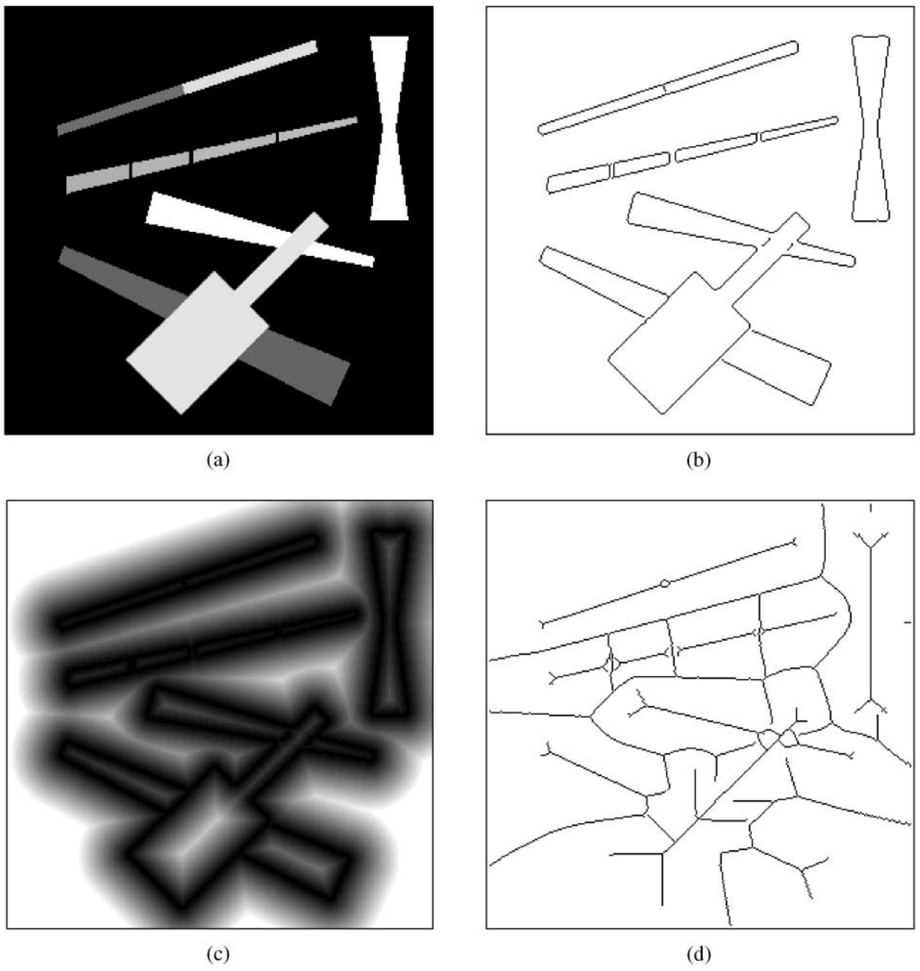


Fig. 3. Processing results: (a) input gray-scale image, (b) edge image, (c) distance map, (d) ridge map (cont. in Fig. 4).

region grows outward. Since former results are used to calculate a distance at a new pixel site, the algorithm is fast and efficient. The distance map obtained by applying the algorithm to the edge image in Fig. 3(b) is shown in Fig. 3(c), where distance values greater than 35.0 are thresholded to 35.0 for the purpose of display.

2.3. Extracting ridge pixels from the distance map

Ridge pixels (i.e., local maxima) of the distance map serve as candidate center line pixels of the linear bands included in the image. We implemented the algorithm proposed by Arcelli and Baja [5] which is specially designed to detect ridges in a distance map. One of the advantages of this method is the guaranteed connectivity of ridges. In this algorithm, strong ridge pixels are searched for in the raster scan order, and for each strong ridge pixel found, weak ridge pixels are tracked starting from it. Strong ridge pixels can be found using eight predefined operators. The tracking of weak ridge pixels is based on gradient computation between two neighboring pixels. Extracted ridges are thinned to be one pixel wide. The resulting ridge map obtained from the distance map in Fig. 3(c) is shown in Fig 3(d).

2.4. Removing noisy ridge pixels

As can be seen in Fig. 3(d), the step of Section 2.3 produces a number of superfluous ridge pixels that seem to make little contribution to the detection of linear bands. Therefore, we need to remove as many such noisy ridge pixels as possible to help extract correct linear bands in the next steps. As a measure of how eligible a ridge pixel is as a member of center line pixels of a linear band, we introduce the parameter ϕ [6]. The meaning of

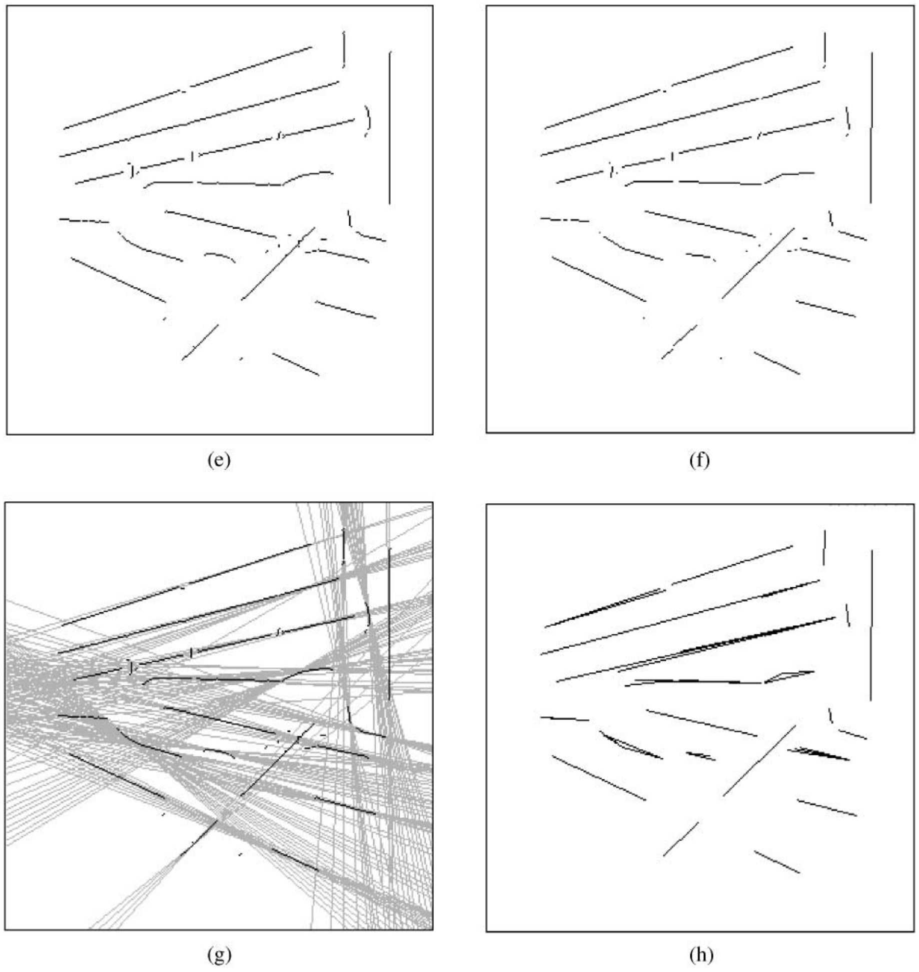


Fig. 4. (e) Filtered ridge map, (f) polygonal approximation result, (g) guide lines found, (h) base line segment grouping result (cont. in Fig. 5).

ϕ can be understood easily from Fig. 6, where $\mathbf{p}$ and $\mathbf{n}$ represent a ridge pixel and one of its eight neighbors, respectively, and $\mathbf{e_p}$ and $\mathbf{e_n}$ represent the edgels closest to $\mathbf{p}$ and $\mathbf{n}$, respectively. The parameter ϕ of a ridge pixel $\mathbf{p}$ is given by

$$\phi(\mathbf{p}) = \max_{\mathbf{n}\in N_\mathbf{p}} \angle \mathbf{e_p p e_n} = \max_{\mathbf{n}\in N_\mathbf{p}} \frac{180}{\pi} \arccos \frac{\overrightarrow{\mathbf{pe_p}} \cdot \overrightarrow{\mathbf{pe_n}}}{\|\overrightarrow{\mathbf{pe_p}}\| \|\overrightarrow{\mathbf{pe_n}}\|}, \quad (1)$$

where $N_\mathbf{p}$ denotes a set of eight neighboring pixels of $\mathbf{p}$. The pixels $\mathbf{e_p}$ and $\mathbf{e_n}$ in Fig. 6 are, in fact, only approximations of two touching points of the largest inner circle (called a *maximal disk*) centered on $\mathbf{p}$. The property of ϕ is well illustrated in Fig. 7. Noisy ridge pixels occurring due to edge gaps or small protrusions of edge contours (e.g., $\mathbf{p}_1$ and $\mathbf{p}_2$ in Fig. 7) usually have very small values of ϕ. Therefore, such pixels can be removed easily by thresholding the values of ϕ. The parameter ϕ of $\mathbf{p}$ can also be interpreted as a measure of parallelism of the two local edge lines tangent to the maximal disk of $\mathbf{p}$. Thus, computing ϕ is also useful for excluding highly tapered linear bands. The result in Fig. 4(e) was obtained by removing the ridge pixels whose value of ϕ was less than $T_\phi = 140.0^\circ$ from the ridge map in Fig. 3(d).

In addition to the parameter ϕ, the distance value of $\mathbf{p}$ on the Euclidean distance map, $\mathrm{D}(\mathbf{p})$ can also be used to discard unnecessary ridge pixels if the range of width of the linear bands a user wants to find has already been determined, since $2\mathrm{D}(\mathbf{p})$ represents the approximate width of the local linear band at $\mathbf{p}$.

2.5. Extracting line segments from the ridge map

In this section, we try to extract line segments from the ridge map obtained in Section 2.4. The extracted

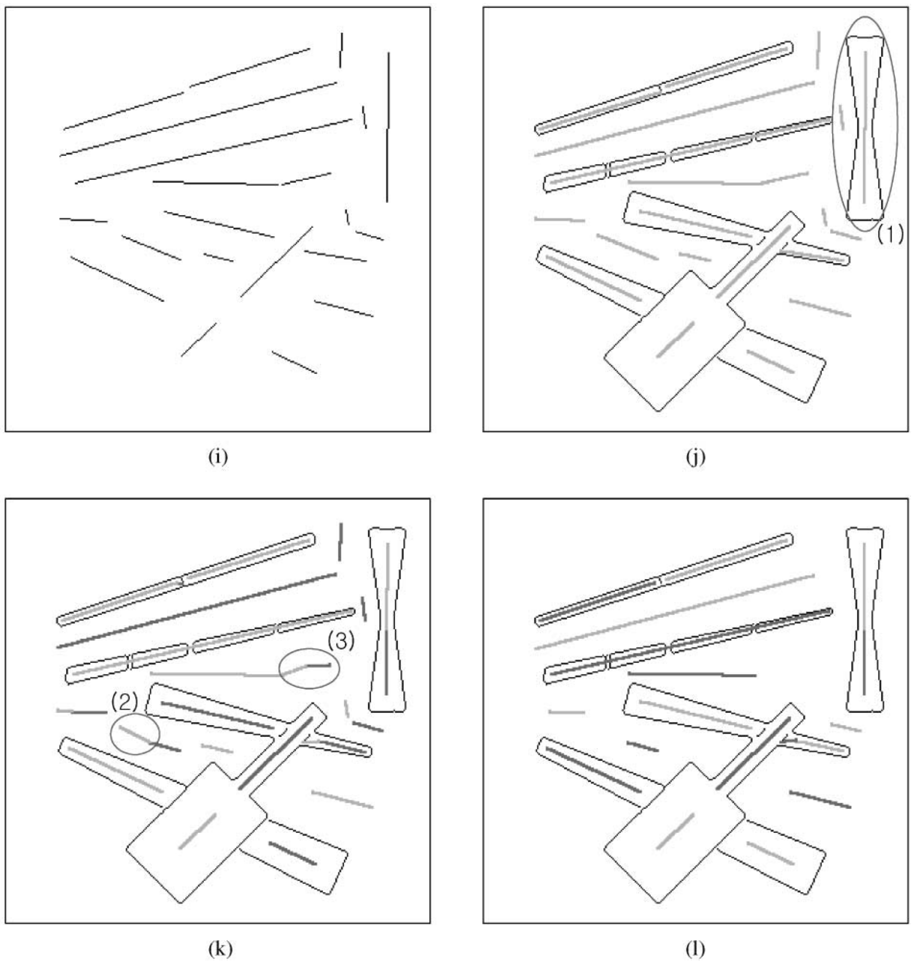


Fig. 5. (i) Result obtained after removing redundant line segments, (j) line segments found, (k) linear band split result, (l) final result.

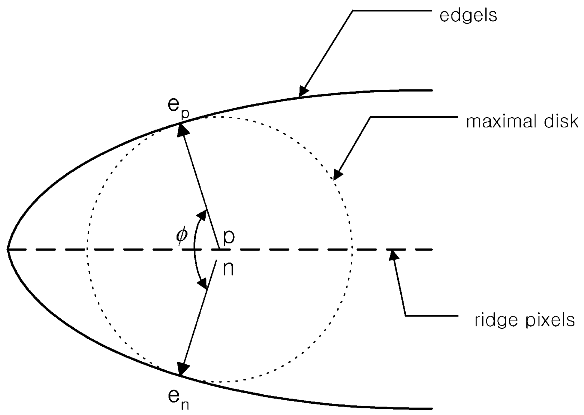


Fig. 6. Definition of the parameter ϕ.

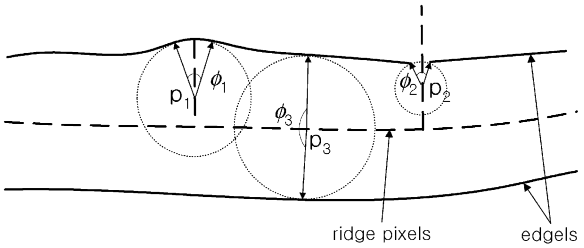


Fig. 7. Property of the parameter ϕ.

line segments represent the center line segments of the corresponding linear bands. Up to date, many methods have been proposed for line segment detection, and they can be divided mainly into two approaches:

Approach 1. Line equations are obtained using the Hough transform, and then line segments are extracted along each line [7–9].

Approach 2. Line segments consisting of connected pixels are found, and then they are iteratively grouped according to some grouping criteria [10–12].

As every method does, the above approaches also have their own disadvantages. Among them, several disadvantages that attract our attention are as follows:

Approach 1

- It is difficult to determine exact peaks of an accumulator array which indicate true lines contained in an image. For example, slightly curved segments as well as adjacent parallel line segments give rise to closely located peaks.
- Illusionary lines are sometimes detected with the Hough transform. This problem, the *connectivity problem*, arises because the connectivity among pixels are not taken into account in the Hough transform method.
- It usually takes much computation time to find lines using the Hough transform method because for each pixel, all the lines passing through that pixel are voted for.

Approach 2

- Line segment grouping is performed based on measuring the *groupability* such as proximity, collinearity, parallelism, etc., between two line segments. However, such measures become unreliable as the line segments under consideration get shorter. Think about an extreme case where most pixels are isolated. Obviously, this method will not work in such a case.
- Its process is purely local. It cannot find more favorable line segments from a global point of view.
- It is not easy to implement the algorithm. This approach requires sophisticated data structure handling.

There might be other disadvantages not mentioned above. In any case, many researchers have proposed methods that can overcome some of the drawbacks of the original methods [13–19]. Though line segment extraction is a very important step throughout our overall algorithm, we would like to avoid here a performance comparison among the proposed line segment detection methods, because line segment detection is not a main issue in this paper.

In this paper, we propose another new, efficient line segment detection algorithm. It can be said that our method combines the advantages of both approaches to alleviate their drawbacks. The overall flow of our line segment extraction algorithm is shown in the right column of Fig. 2. We first make a list of curve segments, each of which consists of connected ridge pixels, using the simple *eight-directional following* algorithm. Then, each curve segment is approximated by piecewise straight line segments using a simple algorithm called the *iterative endpoint fit-and-split algorithm* [20]. This algorithm requires only one parameter, T_{d1} (distance threshold), which determines whether to split the cure segment at the given position or not. The extracted straight line segments are called *base line segments* in this paper, since real line segments are formed by grouping these elementary line segments. Note that one isolated ridge pixel is treated as a base line segment of length one. Fig. 4(f) shows the base line segments extracted from the ridge map in Fig. 4(e), where $T_{d1} = 2.0$.

Now we need to group the base line segments to obtain real line segments. Instead of following the conventional grouping scheme of Approach 2, where the groupability is checked for all possible pairs of base line segments, we use a new *modified Hough transform* technique to find candidate base line segments that may belong to the same real line segment. A line found by the modified Hough transform serves as a *guide line* near which candidate base line segments are checked for grouping into one line segment. The guide line enables us to group the base line segments from a global point of view. By using such a guide line, we can avoid the problem raised by the conventional grouping algorithm of Approach 2.

In this paper, we use the polar form of a line equation, which is given by

$$r = x \cos\theta + y \sin\theta, \tag{2}$$

where (r, θ) are the two polar parameters of the line. The range of (r, θ) is given by

$$-\sqrt{W^2 + H^2} \leqslant r \leqslant \sqrt{W^2 + H^2}, \quad -\frac{\pi}{2} \leqslant \theta < \frac{\pi}{2} \tag{3}$$

for a ridge map of size $W \times H$. In the standard Hough transform (SHT) method, given a ridge pixel (x, y), all the cells (r, θ) satisfying Eqs. (2) and (3) are accumulated. This causes a heavy computational burden when many ridge pixels are contained in a ridge map.

We modified the SHT to reduce the computational cost. Let (x_i, y_i) be the ith member (ridge) pixel of some base line segment B, and let (r_0, θ_0) represent the polar parameters of a line passing through the two end pixels of B. In our method, all the cells (r, θ) satisfying the equation $r = x_i \cos\theta + y_i \sin\theta$ are accumulated in the range of

$$\theta_0 - \Delta\theta \leqslant \theta \leqslant \theta_0 + \Delta\theta, \quad \Delta\theta = \arctan\frac{d}{l}, \tag{4}$$

where d is a user-specified parameter ($d > 0$) and l is the length of B (see Fig. 8). Note that $\Delta\theta$ becomes larger as B gets shorter. That is reasonable because the parameter uncertainty of a line segment increases as the line segment becomes shorter. If $\theta_0 - \Delta\theta < -\pi/2$ or $\theta_0 + \Delta\theta \geqslant \pi/2$, the range of θ needs to be divided into two parts. For example, If $\theta_0 = -0.4\pi$ and $\Delta\theta = 0.2\pi$, then the range of θ is given by $[-0.5\pi, -0.2\pi] \cup [0.4\pi, 0.5\pi)$. Note that if the base line segment under consideration is composed of only one pixel, we follow the voting scheme of the SHT.

This method has some advantages over the SHT method. Firstly, owing to the considerable reduction of

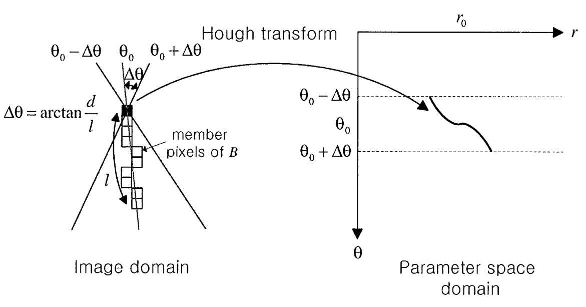


Fig. 8. Voting scheme of our method.

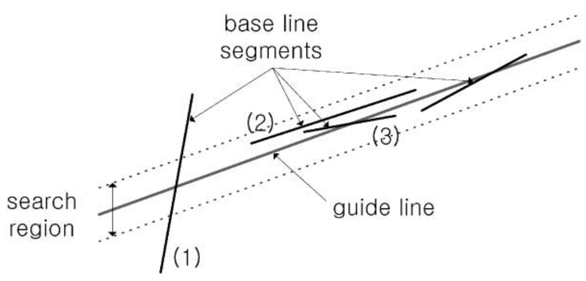


Fig. 9. Examples of the base line segments around the guide line.

the number of votes, it is usually faster than the SHT even when the time spent extracting base line segments is included. Secondly, it produces sharper peaks on an accumulator array, which makes it easier to find correct local maxima of cell values.

After voting has been carried out for every ridge pixel, peaks of an accumulator array are searched for. In our experiments, we simply selected as peaks the cells whose values are greater than those of their eight neighboring cells after some smoothing of the accumulator array. We acknowledge that it is not an elegant method because it usually yields much more peaks than necessary. However, as mentioned above, it is very difficult to extract exact peaks that indicate true lines only by examining the cell values of the accumulator array. In our method, falsely detected line segments caused by this simple peak selection scheme are removed using the algorithm *RemoveRedundantLineSegments* which will be explained later. In Fig. 4(g), guide lines computed from the base line segments in Fig. 4(f) are shown, where $d = 2.5$.

Once guide lines are obtained, we have to group base line segments along each guide line. In general, the base line segments to be grouped do not exactly overlap with the guide line. Therefore, we need to search for them near the guide line. It should be noted that *not* all the base line segments near the guide line are appropriate for grouping. For example, the base line segment (1) in Fig. 9 should be rejected because it is far from being parallel to the guide line. It is clear that the member pixels of (1) make little contribution to the construction of the guide line. It should also be noted that by rejecting such base line segments, we can considerably reduce the detection rate of illusionary line segments caused by the *connectivity problem* of the conventional Hough transform. The base line segment (2) in Fig. 9 should also be rejected though it is parallel to the guide line and completely included in the search region, because another base line segment (3) lies closer to the guide line. Such a case occurs when there exist two parallel line segments very close to each other in an image.

We propose a simple, efficient algorithm that can handle both the cases of (1) and (2) in Fig. 9. Given a guide line g, an algorithm for selecting appropriate base line segments along g is given as follows:

- Procedure: *SelectBaseLineSegments*

```
1. Do $L(\mathbf{p}) \leftarrow L(b_i)$ for each ridge pixel $\mathbf{p} \in b_I$
2. Find the nearest ridge pixels along the guide line $g$ and store them sequentially in the array $P$.
3. Count the member pixels of each run $r_j$ in $P$. A run is a set of consecutive pixels with the same label in $P$ (see Fig. 10). The label of $r_j$ is set equal to the common label of the member pixels of $r_j$.
4. Register $b_{L(r_j)}$ in the list $S$ if $N(r_j)/N(b_{L(r_j)}) \geqslant T_{r1}$, where $T_{r1}$ is a user-specified threshold ($0 \leqslant T_{r1} \leqslant 1$).
```

For example, in the case of Fig. 10, only the base line segment (4) will be rejected if T_{r1} is set to 0.6, because the followings are approximately calculated from the figure:

$$\frac{N(r_1)}{N(b_1)} = 1.0, \quad \frac{N(r_2)}{N(b_2)} = 1.0, \quad \frac{N(r_3)}{N(b_3)} = 0.75,$$
$$\frac{N(r_4)}{N(b_4)} = 0.5, \text{ and } \frac{N(r_5)}{N(b_5)} = 1.0.$$

We would like to give some comments on Step 2 of Procedure *SelectBaseLineSegments*. We use the simple DDA (digital differential analyzer) algorithm [21] to trace a guide line of a known equation. Let $\mathbf{p}$ denote a pixel on a guide line g that is being traced with the DDA algorithm. When searching for ridge pixels in the neighborhood of $\mathbf{p}$, in order to avoid including the same pixel more than once, the search is done only in the vertical direction if the slope $|\Delta y/\Delta x|$ of g is less than 1.0, and in the horizontal direction otherwise (see Fig. 11). Let $\mathbf{q}$ denote the ridge pixel closest to $\mathbf{p}$ in the direction determined by the value of the guide line's $|\Delta y/\Delta x|$. If $|\mathbf{q} - \mathbf{p}| \leqslant \delta$, $\mathbf{q}$ is stored in the array P, where δ is a user-specified parameter.

After obtaining the list S, grouping of the base line segments in S is performed. Note that the base line segments have been registered sequentially in S along

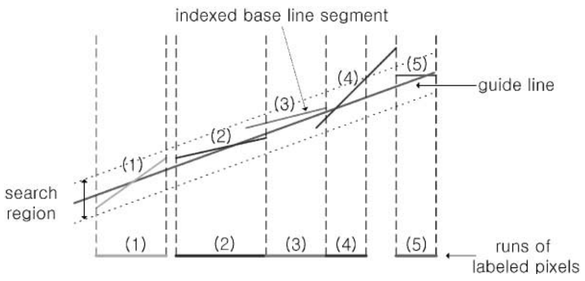


Fig. 10. Constructing runs of labeled pixels along the guide line.

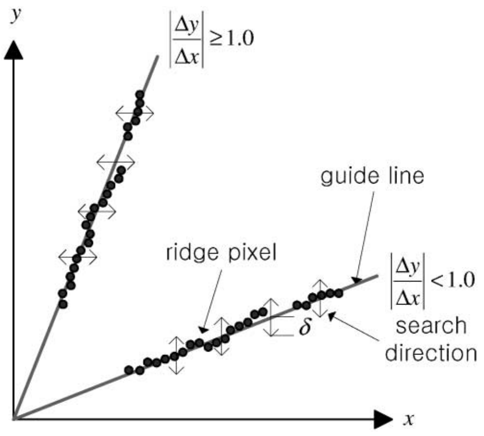

y
$\left|\frac{\Delta y}{\Delta x}\right| \geq 1.0$
guide line
ridge pixel
$\left|\frac{\Delta y}{\Delta x}\right| < 1.0$
δ
search direction
x

Fig. 11. Searching for ridge pixels around the guide line.

the guide line. Therefore, their grouping is also carried out sequentially. Let s_i represent the ith base line segment in S and let $E(s_i)$ denote a set of two end pixels of s_i. Two base line segments s_i and s_{i+1} are considered as belonging to the same real line segment if the conditions

$$\|\mathbf{p}-\mathbf{q}\| \leqslant T_g \quad \text{and} \quad |I(\mathbf{p})-I(\mathbf{q})| \leqslant T_b \tag{5}$$

are satisfied, where

$$\mathbf{p},\ \mathbf{q} = \operatorname{argmin}_{\mathbf{r}\in E(s_i),\ \mathbf{s}\in E(s_{i+1})} |\mathbf{r}-\mathbf{s}| \tag{6}$$

and $I(\mathbf{p})$ represents an intensity value of an input gray-scale image at $\mathbf{p}$. T_g and T_b are two grouping parameters. Fig. 4(h) shows the line segments obtained by grouping the base line segments in Fig. 4(f) along the guide lines in Fig. 4(g). The parameter values used are $T_{r1} = 0.8$, $\delta = 2$, $T_g = 15.0$, and $T_b = 20$. We eliminated the line segments shorter than 10 pixels after the grouping.

As can be seen in Fig. 4(h), near the true line segments, there are many redundant line segments that should be removed. As mentioned earlier, they are due to the inexact guide lines found in the accumulator peak selection stage. Here is our algorithm for removing the redundant line segments:

- Procedure: *RemoveRedundantLineSegments*

    1. Do $L(\mathbf{p}) \leftarrow L$ $(\operatorname{argmax}_{l\in S(\mathbf{p})} N(l)$ for each ridge pixel $\mathbf{p}$.
    2. Create a 1D accumulator array M with each cell of M set to 0.
    3. Do $M(L(\mathbf{p})) \leftarrow M(L(\mathbf{p})) + 1$ for each ridge pixel $\mathbf{p}$.
    4. Discard l_i if$M(i)/N(l_i) \leqslant T_{r2}$, where T_{r2} is a user-specified threshold $(0 \leqslant T_{r2} \leqslant 1)$.

In the above algorithm, $M(i)$, the score of a line segment l_i, becomes nearly equal to $N(l_i)$ if l_i is a locally dominant line segment, while other line segments sharing some member pixels with the dominant line segment come to have small values of M. Fig. 5(i) shows the result obtained after removing redundant line segments from the result in Fig. 4(h), where $T_{r2} = 0.8$. Note that the displayed line segments were drawn simply by connecting their two end pixels for convenience. One can see that most of the redundant line segments have been removed successfully.

Occasionally, some real line segments may fail to be detected with the method discussed up to now. For instance, if a short line segment lies almost parallel and very close to a long line segment, the peak indicating the short line segment may not be detected in the accumulator array because of the smoothing operation performed on it. The algorithm *RemoveRedundantLineSegments* may also eliminate some true line segments accidentally. To prevent real line segments from being undetected, the above method is *iteratively* applied until no new line segments are detected. Refer to the flow chart in the right column of Fig. 2. Fig. 5(j) shows the final line segment detection result. In this example, the final result seems almost equal to the result in Fig. 5(i) that was obtained at the first iteration. Generally, if an input image is not very complex, one iteration is sufficient to get a desired line segment detection result. When comparing Fig. 5(j) with Fig. 4(e), one can see that all the line segments have been extracted correctly except for a few very short segments. The short segments were removed in the base line segment grouping stage.

2.6. Splitting linear bands and removing false ones

Let us take a look at (1) in Fig. 5(j). Although the center line segment of (1) is straight, (1) does not seem to be a true linear band because its outlines are far from being straight. The method explained so far is unable to handle such a case because it does not take an outline shape into account at all. We need some way to split such an unnatural linear band at appropriate positions. Fortunately, in Section 2.2 we already obtained the information that is useful for the split operation, *the distance values* on the distance map. Let $D(\mathbf{p})$ denote a distance value at a pixel $\mathbf{p}$ and let $\mathbf{p}_i$ denote the ith member pixel of the center line segment of a linear band $l (i = 1, 2, \ldots, n)$. Assuming that the member pixels are ordered along the center line segment, we can now generate a set of 2D points: $B = \{(1, D(\mathbf{p}_1)), (2, D(\mathbf{p}_2)), \ldots, (n, D(\mathbf{p}_n))\}$. If both of the outlines of l are straight, the points of B will constitute a straight line segment. Otherwise the points will take the shape of some curve segment. In order to split l at suitable positions, we make use of the iterative end-point fit-and-split algorithm already used in Section 2.5, where B is used as an input to the algorithm. Fig. 5(k) shows the linear band split result, where T_{d2}, the parameter of the split algorithm, is set to 2.5. In this figure, the center line segments of the linear bands are marked with two different colors to make adjoining center line segments more distinguishable.

The final step of our method is to remove some false linear bands from the result obtained above. These false linear bands (e.g., (2) or (3) in Fig. 5(k)) often appear in the neighborhood of object corners. They can be characterized by their asymmetrical side line segments. In other words, the length ratio of their two side line segments is far greater or less than 1.0. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ represent the two end pixels of the center line segment of a linear band l and let $E_1 = \{\mathbf{e}_{11}, \mathbf{e}_{12}\}$ and $E_2 = \{\mathbf{e}_{21}, \mathbf{e}_{22}\}$ represent the sets of edgels that are closest to $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively. E_1 and E_2 were already obtained in Sections 2.2 and 2.4. To compute the length of each side line segment, we need to know which pair of edgels belong to the same side line segment. This can be done by checking the sign of $f(\mathbf{e}_{ij})$ if the center line equation of l is given by $f(\mathbf{x}) = 0$. Edgels of the same sign are paired. Assume that $\{\mathbf{e}_{11}, \mathbf{e}_{21}\}$ and $\{\mathbf{e}_{12}, \mathbf{e}_{22}\}$ are the resulting pairs. If $\min\,(d_1/d_2, d_2/d_1) \leqslant T_{r3}, l$ is removed, where $d_1 = \|\mathbf{e}_{11} - \mathbf{e}_{21}\|$, $d_2 = \|\mathbf{e}_{12} - \mathbf{e}_{22}\|$, and T_{r3} is a user-specified parameter $(0 \leqslant T_{r3} \leqslant 1)$. The result in Fig. 5(l) was obtained by setting T_{r3} to 0.6. It is certain that the result in Fig. 5(l) is more acceptable to a human than the one in Fig. 5(j).

Now, picking linear bands of desired widths and lengths is very easy because all the information necessary for the selection has been obtained in the previous steps.

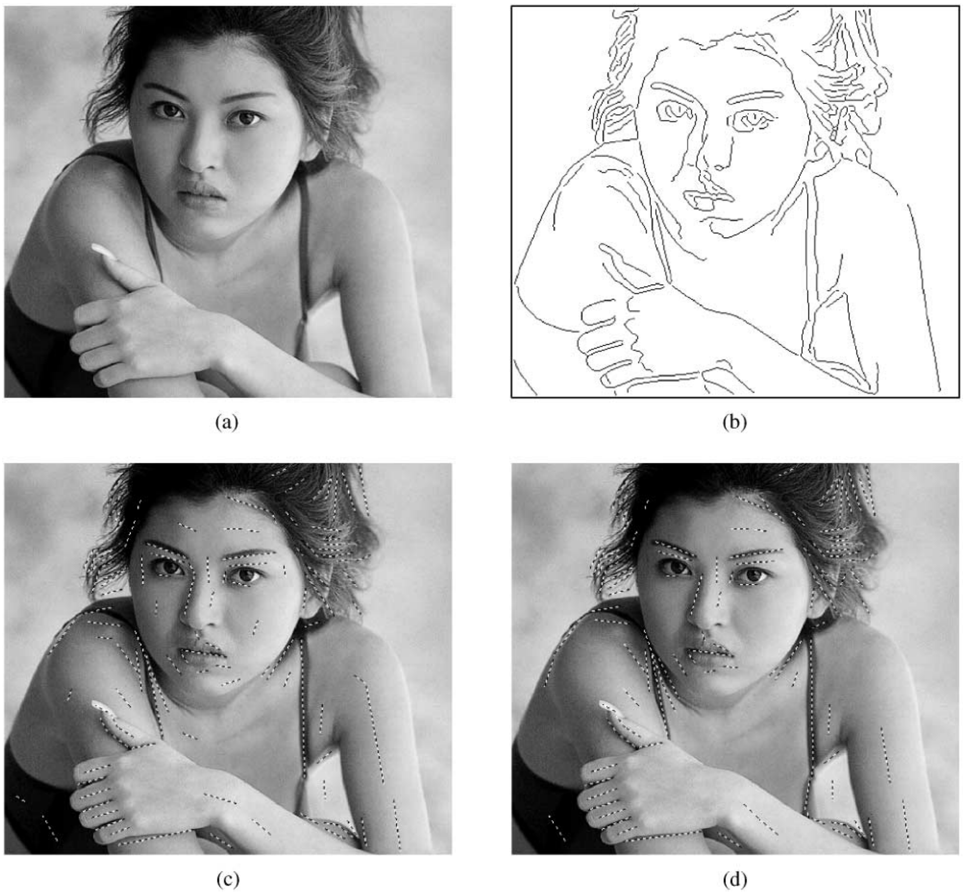


Fig. 12. Example 1: (a) input image, (b) edge image, (c) linear bands found during the first iteration of the method introduced in Section 2.5, (d) final linear band detection result.

3. Experimental results

In Section 2, we showed the feasibility of our method by using a simple synthetic image as an input to the method. In this section, we show several examples of the results obtained by applying our method to some real images.

Example 1 is shown in Fig. 12, where the parameter values used are $\sigma = 2.0$, $T_{h1} = 2.0$, $T_{h2} = 6.0$, $T_{\phi} = 150.0$, $T_g = 5.0$, and $T_{r3} = 0.5$. The other parameter values are the same as those used in Section 2. Though many parameters are associated with our method, the parameters of actual importance, which need to be adjusted from image to image, are usually the above mentioned parameters, In Fig. 12(b), the edge image obtained with the Canny edge detector is shown. Since linear bands are extracted on the basis of an edge detection result, it is important to set parameters of an edge detector properly so that the desired linear band structures are visible in an edge image. Once we obtain the edge detection result, we can guess a linear band detection result approximately by examining the edge image. Fig. 12(c) shows the linear bands found during the first iteration of the method introduced in Section 2.5, and Fig. 12(d) shows the final result. Linear bands shorter than 12 pixels were eliminated from the results. As can be seen in Figs. 12(c) and (d), in general, most of the desired linear bands are detected during the first iteration of the method in Section 2.5. It should be noted that since a few base line segments usually remain ungrouped after the first iteration, computation at following iterations becomes much faster than that of the first iteration. It should also be noted that as mentioned in Section 2.5, the computation time spent completing the first iteration

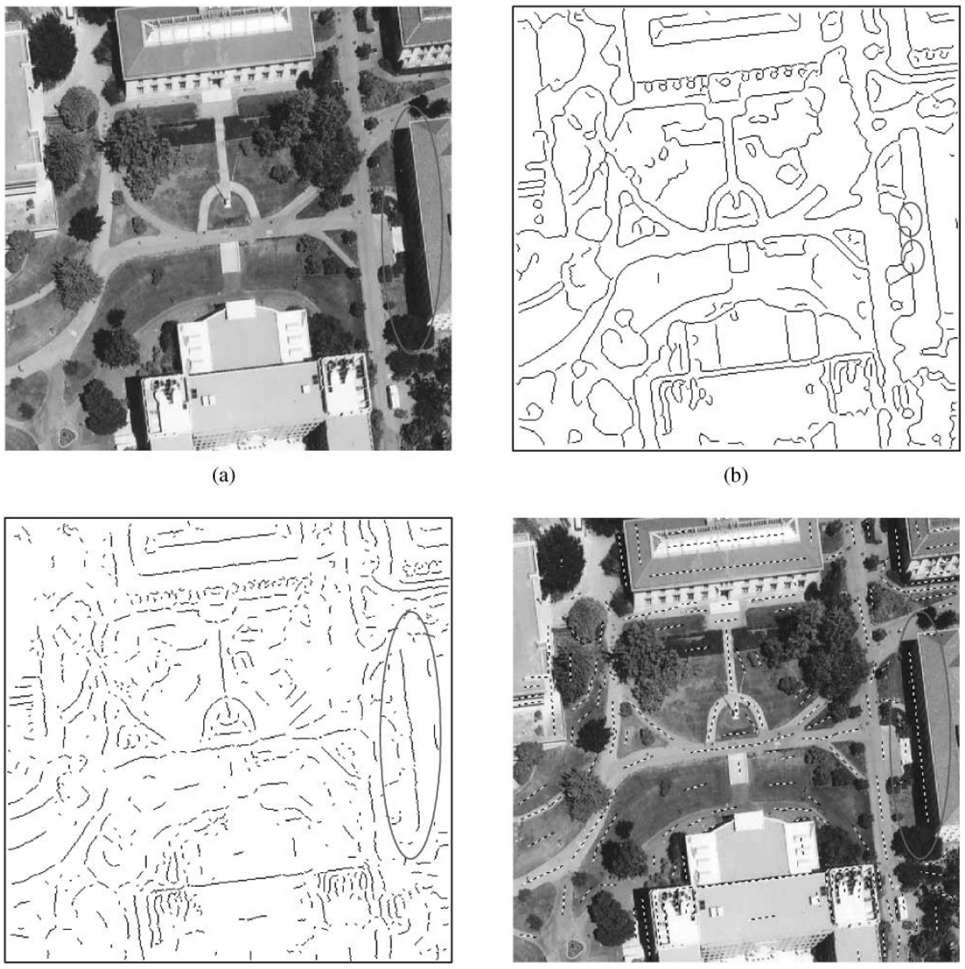

(a) (b) (c) (d)

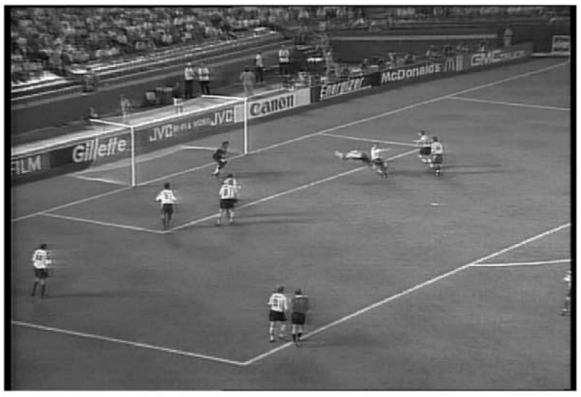
Fig. 13. Example 2: (a) input image, (b) edge image, (c) filtered ridge map, (d) final linear band detection result.

(including base line segment extraction) is usually less than that of the SHT method. For instance, in the case of Example 1 (image size: 457×401), it takes 1.59 s (with an Intel Pentium Pro 200 MHz) to complete the first iteration, which is far less than the 3.82 s needed for the SHT method. Note that in the SHT method, we used the adaptive cluster detection method proposed by Risse [13] to find local peaks of the accumulator array. The method is composed of three steps: finding a global maximum in the accumulator array, identifying the corresponding line, and decrementing all accumulator cells which are to be incremented when all pixels on this line are processed. The three steps are repeated until maximum becomes less than a given threshold. According to the experimental results of Ref. [9], Risse's method yielded the highest detection accuracy among the various Hough transform methods tested.

Example 2 in Fig. 13 shows another linear band detection result. The parameter values used are $\sigma = 2.5$, $T_{h1} = 2.5$, $T_{h2} = 6.0$, $T_{\phi} = 150.0$, $T_g = 8.0$, and $T_{r3} = 0.4$. In this example, linear bands shorter than eight pixels were removed from the result. The dark shaded region on the right side of Fig. 13(a) clearly looks like a linear band structure. However, due to the imperfection of the edge detector used, there are two large gaps on the left side of the linear band structure in Fig. 13(b). But from Figs. 13(c) and (d), one can see that our detector is not sensitive to such gaps.

As can be seen in Figs. 12(d) and 13(d), the proposed detector is successful in extracting most of the perceivable linear structures from the real images, including significant structures such as fingers or arms in Fig. 12(d) and roads in Fig. 13(d).

The last example, Example 3, is shown in Figs. 14 and 15. In Fig. 15, two linear band detection results are shown for comparison, where (c) was obtained with the proposed method and (d) was obtained with the SHT method instead of the new line segment extraction method explained in Section 2.5. Parameter values used are $\sigma = 1.5$, $T_{h1} = 1.0$, $T_{h2} = 5.0$, $T_{\phi} = 150.0$, $T_g = 4.0$, and $T_{r3} = 0.4$. We removed linear bands whose average width is greater than five pixels to get only thin linear


(a)

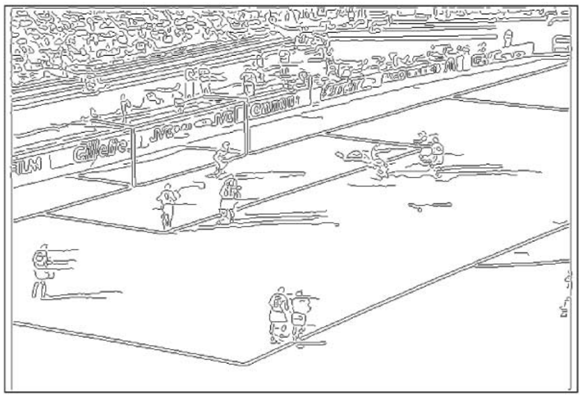
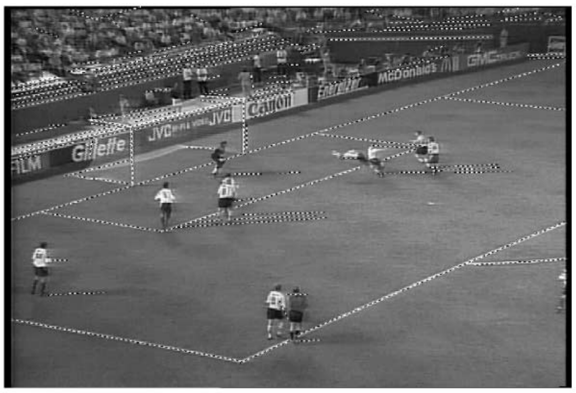

(b)

Fig. 14. Example 3: (a) input image, (b) edge image (cont. in Fig. 15).


(c)

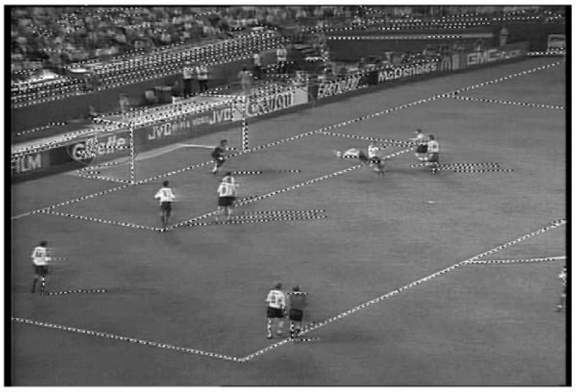

(d)

Fig. 15. (c) Linear bands detected with the proposed method, (d) linear bands detected with the SHT method.

bands. Linear bands shorter than 20 pixels were also removed. As was expected, some illusionary linear bands were detected with the SHT method. For example, several false line segments are marked across the characters on the advertisement boards in Fig. 15(d). However, such line segments were successfully rejected by our method as can be seen in Fig. 15(c).

4. Conclusions

In this paper, we have proposed a new method for detecting a useful feature, called a linear band, in a gray-scale image. We first gave our opinion on what types of linear bands a desirable detector should be able to detect, and then explained our method step by step. There are two important contributions of this paper: (1) a Euclidean distance map constructed based on the edge map is used to get information about linear bands contained in an input image, and (2) a new, efficient line segment extraction algorithm is proposed and incorporated into the linear band detection algorithm. The experimental results showed that our method works well on real images.

Acknowledgements

The authors would like to thank the Ministry of Education of Korea for its financial support towards the Electrical and Computer Engineering Division at POSTECH through its BK21 program.

References

[1] R.C. Lo, W.H. Tsai, Gray-scale Hough transform for thick line detection in gray-scale images, Pattern Recognition 28 (5) (1995) 647–661.

[2] J.H. Jang, K.S. Hong, Detection of curvilinear structures using the Euclidean distance transform, Proceedings of the IAPR Workshop on Machine Vision Applications, Chiba, Japan, 1998, pp. 102–105.

[3] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (6) (1986) 679–698.

[4] O. Cuisenaire, Region growing Euclidean distance transforms, Proceedings of the International Conference on Image Analysis and Processing, Vol. 1, Florence, Italy, 1997, pp. 263–270.

[5] C. Arcelli, G.S. Baja, Ridge points in Euclidean distance maps, Pattern Recognition Lett. 13 (1992) 237–243.

[6] G. Malandain, S.F. Vidal, Euclidean skeletons, Image Vision Comput. 16 (1998) 317–327.

[7] J. Illingworth, J. Kittler, A survey of the Hough transform, Comput. Vision Graphics Image Process 44 (1988) 87–116.

[8] V.F. Leavers, Which Hough transform?, Comput. Vision Graphics Image Process. Image Understanding 58 (2) (1993) 250–264.

[9] H. Kälviäinen, P. Hirvonen, L. Xu, E. Oja, Probabilistic and non-probabilistic Hough transforms: overview and comparisons, Image Vision Comput. 13 (4) (1995) 239–252.

[10] E. Trucco, A. Verri, in: Introductory Techniques for 3D Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1998, pp. 114–117.

[11] M. Boldt, R. Weiss, E. Riseman, Token-based extraction of straight lines, IEEE Trans. Systems Man Cybernet. 19 (6) (1989) 1581–1594.

[12] P. Nacken, A metric for line segments, IEEE Trans. Pattern Anal. Mach. Intell. 15 (12) (1993) 1312–1318.

[13] T. Risse, Hough transform for line recognition: complexity of evidence accumulation and cluster detection, Comput. Vision Graphics Image Process. 46 (1989) 327–345.

[14] J. Princen, J. Illingworth, J. Kittler, A hierarchical approach to line extraction based on the Hough transform, Comput. Vision Graphics Image Process. 52 (1990) 57–77.

[15] L. Xu, E. Oja, P. Kultanen, A new curve detection method: randomized Hough transform (RHT), Pattern Recognition Lett. 11 (1990) 331–338.

[16] N. Kiryati, Y. Eldar, A.M. Bruckstein, A probabilistic Hough transform, Pattern Recognition 24 (4) (1991) 303–316.

[17] S.Y.K. Yuen, T.S.L. Lam, N.K.D. Leung, Connective Hough transform, Image Vision Comput. 11 (5) (1993) 295–301.

[18] N. Guil, J. Villalba, E.L. Zapata, A fast Hough transform for segment detection, IEEE Trans. Image Process. 4 (11) (1995) 1541–1548.

[19] M.C.K. Yang, J.S. Lee, C.C. Lien, C.L. Huang, Hough transform modified by line connectivity and line thickness, IEEE Trans. Pattern Anal. Mach. Intell. 19 (8) (1997) 905–910.

[20] R. Haralick, L. Shapiro, in: Computer and Robot Vision, Vol. 1, Addison-Wesley, Reading, MA, 1992, pp. 563–565.

[21] J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, R.L. Phillips, in: Introduction to Computer Graphics, Addison-Wesley, Reading, MA, 1996, pp. 70–80.

About the Author—JEONG-HUN JANG was born in Seoul, South Korea, in 1970. He received the B.S. degree in Electrical Engineering from Hanyang University, Korea, in 1994 and the M.S. degree in Electrical & Electronic Engineering from POSTECH, Korea, in 1996. He is now in the Ph.D. program at POSTECH. His research interests include feature extraction, character detection & recognition, and face detection & recognition.

About the Author—KI-SANG HONG received the B.S. degree in Electronic Engineering from Seoul National University, Korea, in 1977, and the M.S. degree in Electrical & Electronic Engineering from KAIST, Korea, in 1979. He also received the Ph.D. degree from KAIST in 1984. During 1984–1986, he was a researcher in the Korea Atomic Energy Research Institute and in 1986, he joined POSTECH, Korea, where he is currently an Associate Professor of Electrical & Electronic Engineering. During 1988–1989, he worked in the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA, as a visiting professor. His current research interests include camera self-calibration, mixed reality, image mosaicking, and SAR image analysis.