Pergamon

Pattern Recognition, Vol. 30, No. 10, pp. 1651–1660, 1997

0031-3203/97 $17.00+.00

PII: S0031-3203(96)00185-9

EXTRACTION OF LINE FEATURES IN A NOISY IMAGE

JOON-WOONG LEE* and IN-SO KWEON

Department of Automation and Design Engineering, Korea Advanced Institute of Science and Technology, 207-43, Cheongryangridong, Dongdaemoongu, Seoul, South Korea

(*Received* 14 *June* 1996; *in revised form* 31 *October* 1996)

Abstract—Finding line segments in an intensity image has been one of the most fundamental issues in the area of computer vision. In complex scenes, it is hard to detect the locations of point features. Line features are more robust in providing greater positional accuracy. In this paper we present a robust "line feature extraction" algorithm which extracts line features in a single pass without using any assumptions and constraints. Our algorithm consists of six steps: (1) edge extraction, (2) edge scanning, (3) edge normalization, (4) line-blob extraction, (5) line-feature computation and (6) line linking. By using an edge scanning, the computational complexity due to too many edge pixels is drastically reduced. Edge normalization improves the local quantization error induced from the gradient space partitioning and minimizes perturbations on edge orientation. We also analyze the effects of edge processing, and the least squares-based method and the principal axis-based method on the computation of line orientation. We show its efficiency with some real images. © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Line features extraction Edge processing Edge normalization Line linking

1. INTRODUCTION

Finding line segments in an intensity image has been one of the most fundamental issues in the area of computer vision. Although much work has been done since the 1960s, the robust line segment extraction has remained as a difficult and open problem in many areas of computer vision. There are several methods to extract line segments such as the Hough transform,[1–6] polygonal approximation,[7–11] arm method,[7] chain coding,[8,12] and ϕ–s curve.[7] The well-known Hough technique has some limitations such as low peaks for short lines in Hough space and limited accuracy in line parameter estimation caused by the quantization of Hough space. Other methods also have several problems such as the dependence on iteration, the inability to an unconstrained complex scene, heuristic parameterization, poor localization accuracy and long processing time. Bimbo *et al.*[1] divided the image into multiple tiles and extracted line segments for each tile using the Hough transform. They used a neural network to obtain road boundary line using the extracted line segments. This method is task-oriented and depends on prior knowledge. Burns *et al.*[13] proposed a line extraction algorithm based on a gradient-based and region-based approach. They effectively quantized the gradient orientation of edge pixel. Kahn *et al.*[14] used a similar approach to Burns *et al.*[13] and improved the speed. Yuan and Suen[12] proposed an algorithm to determine the straightness of a digital arc by chain coding in O(n) time. Pikaz and Dinstein[10] improved the general polygonal approximation and Chung *et al.*[9] discussed a polygonal approximation using a competitive Hopfield neural network. Breuel[6] introduced a variety of statistical error models to extract the maxima of the probabilistic Hough transform and the generalized Hough transform. Juli *et al.*[3] optimized the general Hough transform algorithm by reducing the complexity and memory requirements.

In this paper, we present a line extraction algorithm to solve the problems of previous approaches. The first step in the line segment extraction is extracting edge pixels. We compute the magnitude and the direction of the gradient for each pixel using the Deriche operator,[15] and remove edge pixels with a low gradient norm by non-local maximum suppression[16] and hysteresis thresholding.[16] Deriche's edge operator gives a comparatively robust result in getting gradient orientation of an edge pixel even under salient intensity variation or noise effect compared to other gradient edge operators such as Sobel, Prewitt, DOG filter, etc. According to the gradient space partitioning scheme proposed by Burns *et al.*,[13] we coarsely quantize the gradient direction and assign a gradient direction code to each edge pixel. The gradient space partitioning, however, has the intrinsic ambiguity problem in its codes due to a quantization error in the boundary of the two partitioned sections. In addition to this ambiguity, a salient intensity variation or noise effect along the edge profile provides perturbations on the edge orientation. An edge normalization method overcomes the ambiguity and the random orientation along the edge profile by using the maximum-likelihood decision criterion.[17] Edge operators generally produce incomplete boundaries and false edges. Such segmentation defects often require post-processing to link broken edges and to remove short edges. Edge scanning, which provides the connected edge chains, is used to increase the computa-

* Author to whom all correspondence should be addressed. E-mail: kweon@design.kaist.ac.kr.

tional efficiency, linking broken edges while removing short edges. The connected edge pixels with an identical gradient direction code are then grouped into a *line-blob* using the blob-coloring technique.[7] Finally, we compute the line features for each line-blob.

Our proposed algorithm has several benefits: For any unconstrained outdoor complex scene, the algorithm extracts line features in a single pass, without any assumptions and constraints, with the minimum use of heuristic parameters.

We show the efficiency of the algorithm with some real images.

2. EXTRACTING LINE SEGMENTS

The proposed line extraction algorithm is organized as shown in Fig. 1.

2.1. Edge extraction

In edge processing, we obtain the magnitude and the direction of the gradient for each pixel using the Deriche operator[15] and remove the edge pixels with low gradient norms by the non-local maximum suppression[16] and the hysteresis thresholding.[16] In order to check whether an edge pixel is a local maximum of the gradient norm in the direction of the gradient, we have to interpolate gradient norms on both directions using the gradient values of the neighboring pixels and compare the gradient norm of the pixel to the interpolated gradient norms. The edge pixel is not suppressed when the gradient norm of the pixel is strictly greater than the interpolated values. In the hysteresis thresholding, we use two thresholds T_1 and T_2 with $T_2 < T_1$. If the gradient norm of an edge pixel is greater than T_1, we do not remove the neighboring edge pixels which have gradient norm greater than T_2. These two processes deliver the effect of edge thinning and reduce the computational complexity. They provide better localization accuracy of an extracted line segment by eliminating non-uniform thick parts of edges than an edge thinning followed by a simple thresholding. Using the gradient space partitioning, we assign a gradient direction code to each edge pixel. The range of gradient direction is quantized into eight sections depending on the angle as shown in Fig. 2. We assign one of four direction codes to each section.

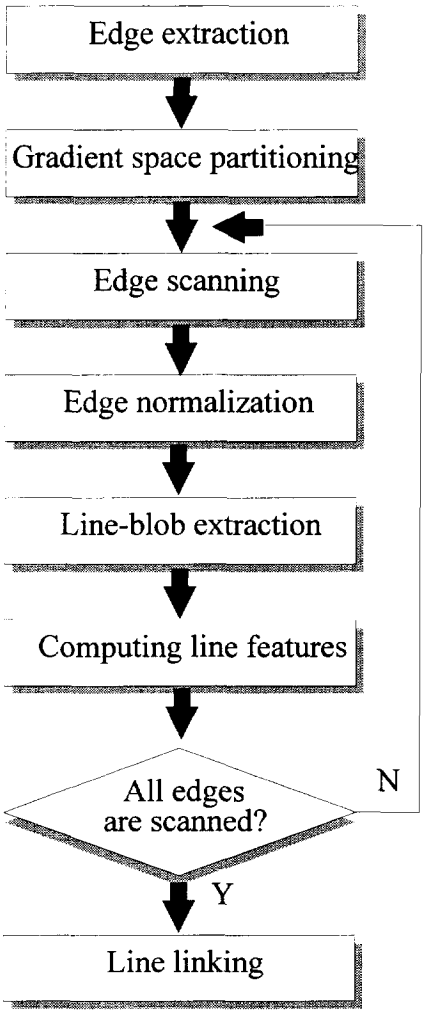


Fig. 1. Overall procedure of a line segment extraction.

2.2. Line-blob extraction and computation of line features

2.2.1. Edge scanning. Prior to the computation of line features, we have to solve several problems such as the processing of a large amount of data due to excessive edges, linking broken edges, removing short edges and reducing the quantization error of the gradient space partitioning. To solve these problems, we use the edge scanning which produces connected edge chains. For example, knowing the length of scanned edges, we can remove short edges and control the maximum data size in the line-blob extraction process. We also use the

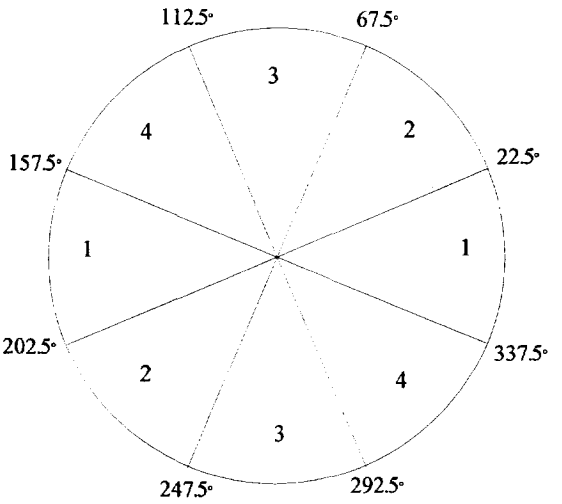


Fig. 2. Four-directional gradient space partitioning.

contextual relationship of scanned edge pixels in the edge normalization process. In the edge scanning process, we traverse the connected edge chain through the eight-directional chain coding[7,8] which generates the direction code of each pixel on the connected edge chain according to the predetermined numbering scheme.

2.2.2. Edge normalization. Edge normalization enhances edge orientation by reducing the random phenomena of edge locality that may be introduced in the edge gradient partitioning process. Here, the examples of *locality* are shown in Fig. 3, marked as A and B. This random gradient direction code occurs due to a quantization error around the boundary of a partitioned gradient space or due to a salient intensity variation and noise effect. To solve this problem, edge normalization is carried out. For successive two-edge pixels, p_k and p_{k+1} on the scanned edge, we check whether d_k is different from d_{k+1} which are the gradient direction codes of p_k and p_{k+1}, respectively. If d_k is not equal to d_{k+1}, n pixels before p_k and after p_{k+1} are used to compute a probability density function of $P(z|m_1)$ and $P(z|m_2)$ defined by

$$P(z|m_1) = \frac{\sum_{i=k-n}^{k+n+1} f_i}{(2n+2)},$$
$$f_i = \begin{cases} 1 & \text{if } d_i = d_k,\ i = k-n, \ldots, k+n+1, \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

$$P(z|m_2) = \frac{\sum_{i=k+1}^{k+n+1} f_i}{(n+1)},$$
$$f_j = \begin{cases} 1 & \text{if } d_j = d_k + 1,\ i = k+1, \ldots, k+n+1, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where m_1={gradient code of $p_{k+1} = d_k$} and m_2={gradient code of $p_{k+1} = d_{k+1}$} represent message spaces for the decision rule.

If the *likelihood ratio* $\Lambda(z)$ is defined as

$$\Lambda(z) = \frac{P(z|m_1)}{P(z|m_2)}, \quad (4)$$

then we decide the decision rule associated with message spaces

$$\Lambda(z) \underset{m_2}{\overset{m_1}{\gtrless}} 1. \quad (5)$$

If message m_1 is decided, the gradient direction code d_{k+1} changes to d_k. Figure 3 shows an example of edge normalization.

Random localities occur at the positions marked A and B in Fig. 3(a). While the dominant direction code of scanned edges is 2 as shown in Fig. 3(a), the other direction codes 1 and 3 are produced. By using the likelihood ratio test of equation (5), we obtain the enhanced edge image as shown in Fig. 3(b). The gradient direction codes of pixels indicated by A and B in

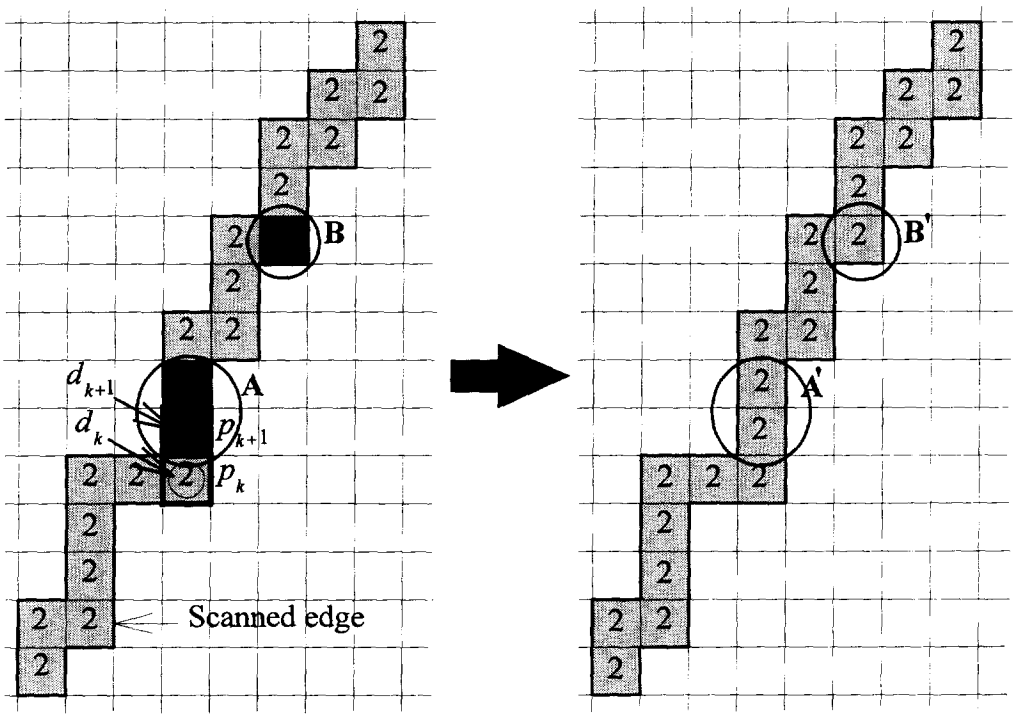


Fig. 3. Edge normalization.

Fig. 3(a) are converted to the ones indicated by A′ and B′ in Fig. 3(b).

2.2.3. Line-blob extraction and computation of line features. The connected edge pixels with an identical gradient direction code are grouped into a *line-blob* using the blob-coloring technique.[7] While the previous blob-coloring algorithm deals with only binary images to extract a blob, the algorithm used in this paper handles quintuple images which are four quantized gradient direction codes and a background. For each extracted *line-blob*, we compute line features such as the mid-point, the intercept of y-axis, the orientation, end points and the length.

Unconstrained complex scenes produce too many edge elements, which result in too many line-blobs to be managed all together. Therefore, we control the data size through edge scanning. In addition, we do not know how many edge pixels belong to a line-blob. Accordingly, a proper data structuring for a line-blob extraction is required to save memory and to reduce processing time. An efficient linked-list-type data structure[18] well accommodates this purpose. In Appendix A, we present the computation of line features based on the extracted line-blobs.

2.3. Line linking

For a cluttered scene, the current approaches to line extraction generally produce broken segments even for a single scene line. In order to link broken line segments, we give three definitions.

Definition 1 (*co-linearity*). For any two lines L_1 and L_2 as shown in Fig. 4, the two lines are *collinear* if

$$\max(l_1, l_2) < \tau_1, \tag{6}$$

where l_1 and l_2 are lengths from the mid-point P_1 of L_1 to L_2 and from the mid-point P_2 of L_2 to L_1, respectively.

Definition 2 (*overlappedness*). In Fig. 4, line L_3, which is orthogonal to line L_1, passes through P_1, and intersects the line L_2 at P_3. If P_3 belongs to the MBR which stands for a minimum building rectangle encompassing L_2, two lines L_1 and L_2 are assumed to be *overlapped.*

Definition 3 (*adjacency*). If the minimum distance between end points of two lines is smaller than a preset threshold τ_2, these two lines are adjacent. In Fig. 4, if $l_3 < \tau_2$, two lines L_1 and L_2 are adjacent.

If any two lines are *collinear*, not *overlapped*, and *adjacent*, we merge them to make a single line.

3. ANALYSIS OF THE EFFECTS OF EDGE PROCESSING AND LINE ORIENTATION COMPUTATION METHODS

In this section, we analyze the effects of edge processing and line orientation computation methods such as the least-squares based method and the principal-axis based method to the line orientation. To do this, we use a real image as shown in Fig. 5. In most cases, simple edge thresholding leads to large localization errors and many multiple detections of a single edge as shown in Fig. 5(b). Therefore, the thresholding is often combined with the detection of local maxima of the edge elements in some suitably chosen direction in the image plane.[16,19] Non-local maximum suppression and hysteresis thresholding can efficiently implement an edge thinning as shown in Fig. 5(e). However, edge thinning followed by the simple thresholding does not take into account the gradient norm of each edge pixel and thus often produces segmentation defects such as broken smaller chains and inaccurate edge localization as shown in Fig. 5(d). On the other hand, edges obtained by non-local maximum suppression and hysteresis thresholding do not show such effects as shown in Fig. 5(e). We applied the same values for hysteresis thresholding and obtained edges shown in Figs 5(b) and (e).

In Fig. 5(c), which shows the extended edge image encompassed by the rectangle indicated by B in Fig. 5(b), we note that pixels in the region marked by A and A′ have great intensity changes. Therefore, more edge pixels are left and dominate the line orientation. In Fig. 5(c), for example, lines ① and ③ are obtained by the principal axis of all pixels in the line-blob. Lines ② and ④ are the least-squares estimate obtained by using the pixels within the line-blob. Lines ① and ③ show better localization accuracy than lines ② and ④. We note that the least-squares based method is more sensitive to the region indicated by A and A′ than the principal axis-based method. Lines ① and ③, however, do not

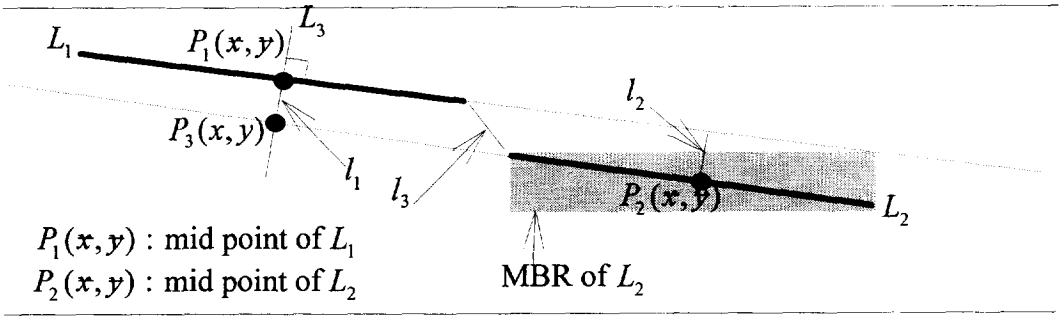


Fig. 4. Line linking.

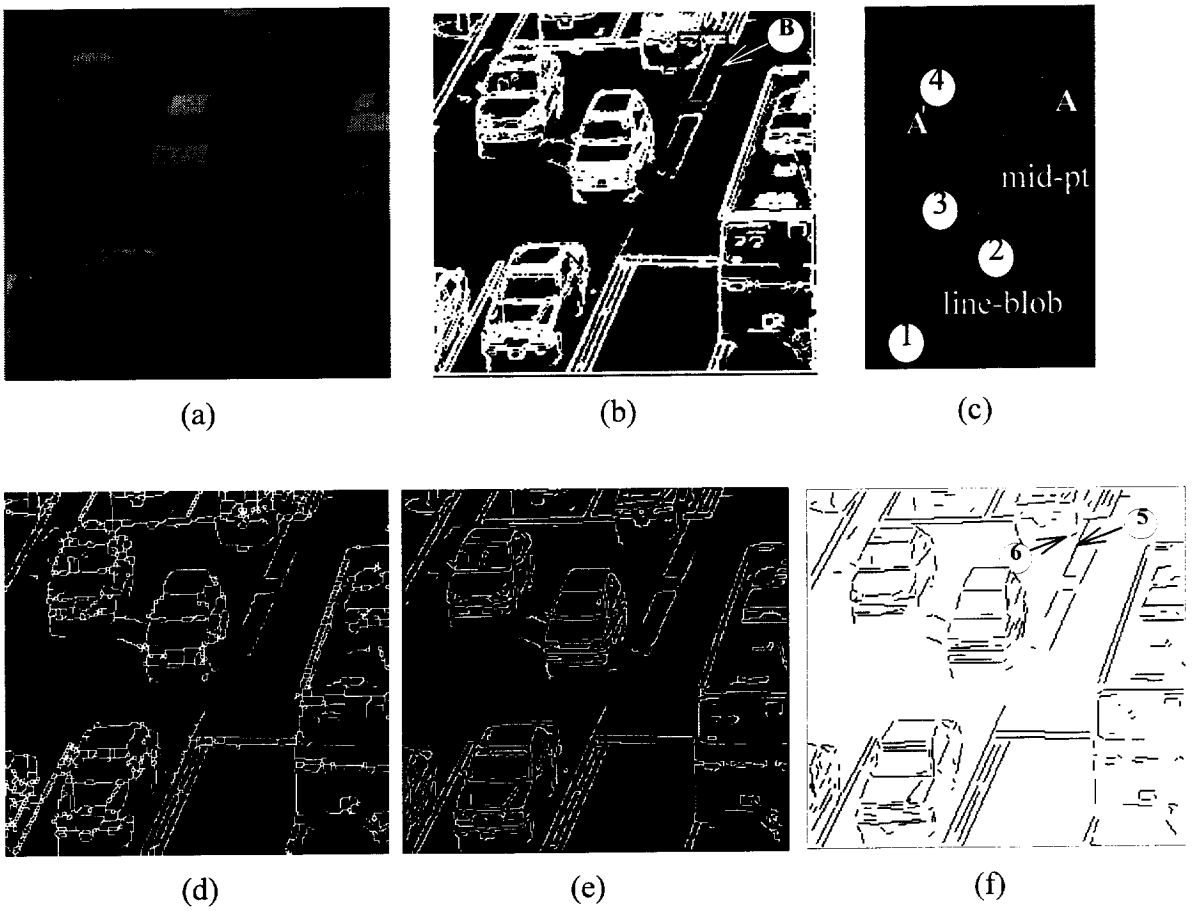


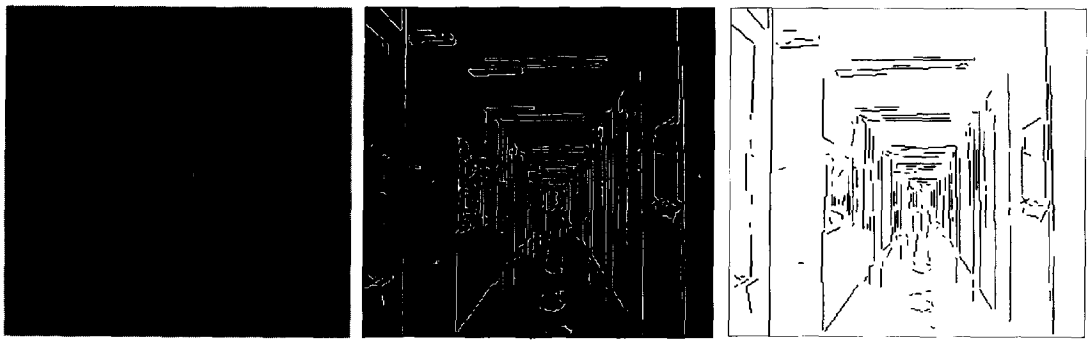
Fig. 5. Analysis of effects of edge processing and line orientation generation methods using a real image: (a) original image, (b) edges from hysteresis thresholding, (c) extended edges of rectangle indicated by B in (b) and line fitting results, (d) thinned edges from edges of (b), (e) edges from non-local maximum suppression and hysteresis thresholding and (f) extracted line segments using edges of (e).

satisfy the localization well. In Fig. 5(f), lines ⑤ and ⑥ are also obtained by the principal axis-based method. These lines show better localization than lines ① and ③. In conclusion, the principal axis to an edge from the non-local maximum suppression and hysteresis thresholding gives good results of localization for a line extraction.

4. EXPERIMENTAL RESULTS

The proposed algorithm for extracting line segments has been examined on a large number of real 512×512 (or 480×640) images which are composed of a laboratory image, a corridor image, two types of road images and an industrial welding panel image. In these experiments, we used Deriche's operator[15] to compute

Fig. 6. Laboratory image, its extracted edges and line segments. (a) Laboratory image. (b) Extracted edges. (c) Extracted line segments.

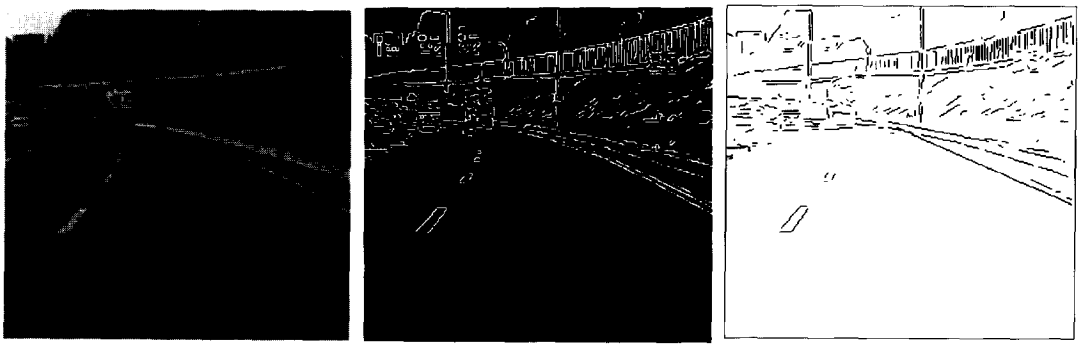


(a) Corridor image (b) Extracted edges (c) Extracted line segments

Fig. 7. Corridor image, its extracted edges and line segments for mobile robot navigation. (a) Corridor image. (b) Extracted edges. (c) Extracted line segments.

(a) Road image (b) Extracted edges (c) Extracted line segments

Fig. 8. Road image, its extracted edges and line segments for a car following system. (a) Road image. (b) Extracted edges. (c) Extracted line segments.



(a) Road image (b) Extracted edges (c) Extracted line segments

Fig. 9. Road image, its extracted edges and line segments for a lane tracking system. (a) Road image. (b) Extracted edges. (c) Extracted line segments.

the gradient norm and the gradient direction of an edge pixel and then applied the non-local maximum suppression and hysteresis thresholding to the edge pixels.

At first, we took an example of our laboratory image as shown in Fig. 6(a). Fig. 6(b) shows the extracted edges. Fig. 6(c) shows the extracted line segments composed of 471 segments. Before line linking, 520 line segments were extracted.

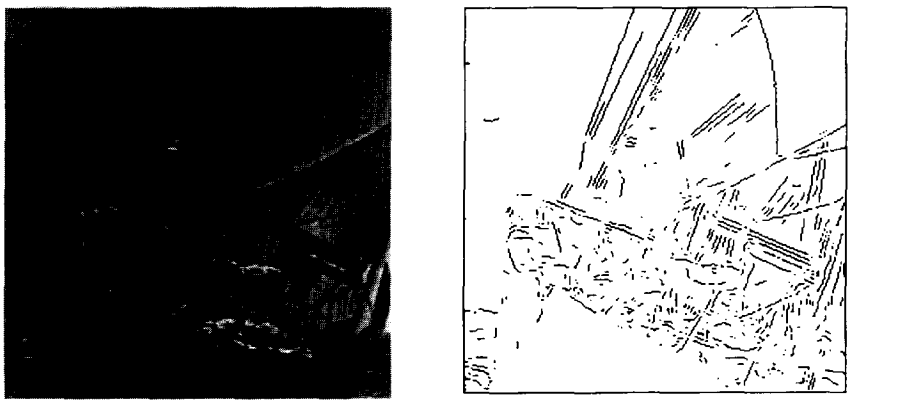
The following example is a corridor scene in which line segments are one of the most prominent geometric features. Fig. 7(a) shows an image of a corridor scene. Figures 7(b) and (c) show the extracted edges and the extracted line segments, respectively.

Figures 8 and 9 show the line segment extraction for outdoor road scenes.

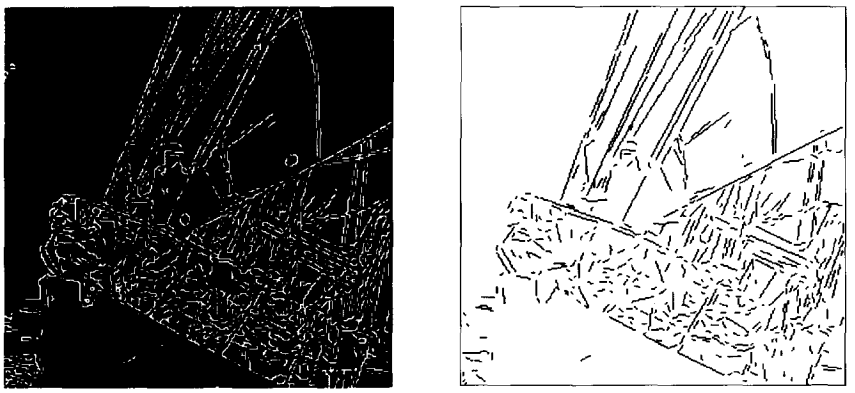
Figure 10(a) shows an image of a welding panel in the subassembly process of a shipbuilding factory. Figures

(a) Welding panel image (b) Extracted edges (c) Extracted line segments

Fig. 10. Welding panel image, its extracted edges and line segments for a panel recognition system in the subassembly process of a shipbuilding factory. (a) Welding panel image. (b) Extracted edges. (c) Extracted line segments.



(a) Yacht image (b) Line segments by F. Heijden [20]

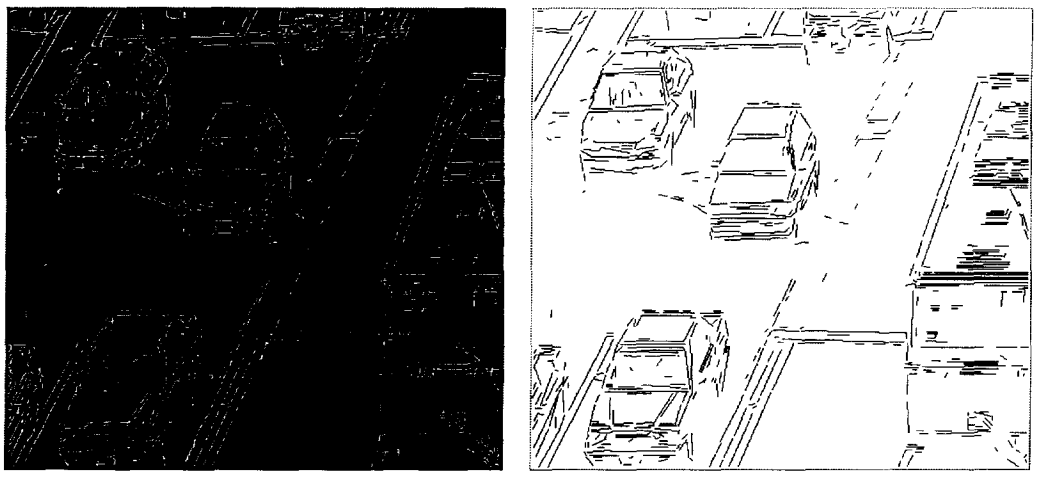


(c) Extracted edges (d) Extracted line segments by the proposed method

Fig. 11. Original image, its extracted edges and line segments. (a) Yacht image. (b) Line segments by Heijden.[19] (c) Extracted edges. (d) Extracted line segments by the proposed method.

10(b) and (c) show the extracted edges the extracted line segments, respectively.

Figure 11(a) shows an image of a yacht. Figure 11(b) shows a line map derived from a rotational invariant edge feature extraction and the resulting log-likelihood ratio, provided by the courtesy of Heijden.[19] His method emphasized local multiple step edges. Figures 11(c) and (d) show the extracted edges and line segments according to the proposed algorithm in this paper. Since these two methods use different approach to extract edge elements, it is very difficult to compare directly. The careful inspection reveals that the proposed algorithm produces line segments well in the regions where intensity discontinuities occurs strongly.



(a) Extracted edges (b) Extracted line segments

Fig. 12. Extracted edges and line segments of a road image. (a) Extracted edges. (b) Extracted line segments.

Deriche's edge operator[15] basically has two aspects: while it provides a comparatively robust result in getting edge gradient direction to salient intensity variation or noise effect it needs a lot of memory. As mentioned in the paper, we overcome the quantization error from gradient space partitioning and randomness due to a salient intensity variation and noise effect by using edge normalization. Therefore, we can freely select the edge operator according to the user's desire. For example, when the problem to acquire enough memory is severe it will be better to use the conventional derivative edge operators. Figure 12 shows an example of line extraction using Sobel edge operator for the same image shown in Fig. 5.

5. CONCLUSION

In this paper, we proposed a robust line segment extraction algorithm. Our algorithm extracts line features in a single pass, without any assumptions and constraints, with the minimum use of heuristic parameters. The algorithm has been implemented on an IBM PC or its compatible with a C30-based image processing board. By using edge scanning, blob-coloring and proper data structuring, the computational efficiency was highly improved. We minimized the local quantization errors induced from the gradient space partitioning and the occurrence of a random orientation involved in the detected edge profile due to a salient intensity variation or noise by edge normalization. We also analyzed the effects of edge processing and line orientation computation methods on line orientation. Even though it is scarcely occurred, if the gradient directions of connected edge pixels change too frequently we cannot guarantee the good quality of extracted line features. Experimental results with some real scenes showed that the proposed algorithm works well in any complex real environment.

APPENDIX A. COMPUTING LINE FEATURES

In presenting line features computation, we first introduce the data structure for representing a line-blob as shown in Fig. 13. "BLOB" is a structure-type array to represent attributes of a line-blob and "NODE" is also a structure-type variable to record the coordinates of an edge pixel of the line-blob. In "BLOB", *dir* and *size* represent the edge pixel's gradient direction code and the number of edge pixels in a line-blob, respectively, *min_row*, *max_row*, *min_col*, and *max_col* are used to

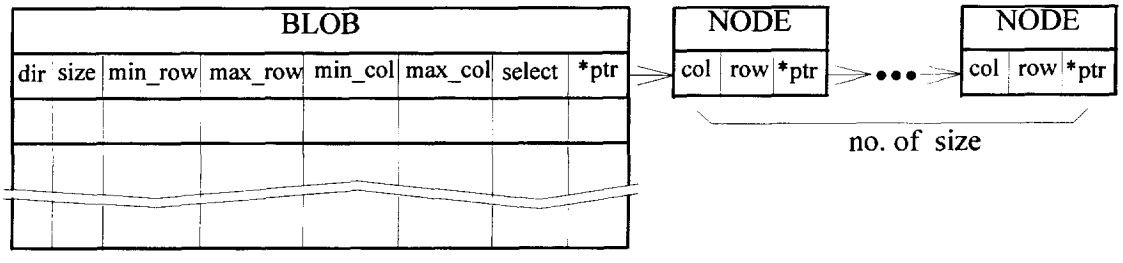


Fig. 13. Data structure for line-blobs.

represent the coordinates of the minimum rectangle encompassing a line-blob, *select* represents whether the blob label is used or not, and **ptr* points to the first "NODE" of the edge pixel belonging to the line-blob. In "NODE", *col* and *row* represent the pixel's coordinates, and **ptr* points to the "NODE" following the current node.

Second, we introduce the computation of line features such as the end points, the mid-point, the orientation, the intercept of the y-axis, and its length. The computation is based on the information recorded in "BLOB".

1. *Mid-point.* It corresponds to the center of the line-blob as defined by

$$(\bar{x},\bar{y}) = \left(\frac{\sum_{i=1}^{BLOB.size} x_i}{BLOB.size}, \frac{\sum_{i=1}^{BLOB.size} y_i}{BLOB.size}\right), \quad \text{(A.1)}$$

where x_i and y_i are column and row on the image plane, recorded in "NODE" expressed as a linked-list.

2. *Orientation.* It corresponds to the principal axis as defined by

$$\theta_s = \frac{1}{2}\tan^{-1}\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}, \quad \text{(A.2)}$$

where μ_{11}, μ_{20}, and μ_{02} are second order central moments[8] which are obtained using

$$\mu_{pq} = \sum_{x_i}\sum_{y_i}(x_i-\bar{x})^p(y_i-\bar{y})^q,$$
$$i = 1,\ldots,BLOB.size.p,q = 0,1,2,\ldots, \quad \text{(A.3)}$$

where x_i, y_i are column and row on the image plane, recorded in the "NODE". Special cases of θ_s are

- 0° when $\mu_{11}=\mu_{02}=0$, $\mu_{20}\neq 0$,
- 90° when $\mu_{11}=\mu_{20}=0$, $\mu_{02}\neq 0$, and
- 45° when $|\mu_{11}|=|\mu_{20}| = |\mu_{02}|$.

3. *Intercept of y-axis.*

$$\alpha = \bar{y} - \tan\theta_s\bar{x}. \quad \text{(A.4)}$$

4. *End points.* Let coordinates of the minimum rectangle of a *line-blob* as (x_0,y_0), (x_0,y_1), (x_1,y_0), (x_1,y_1), where x_0 and y_0 represent minimum column and row and x_1 and y_1 represent maximum column and row. Then, end points (x_s,y_s) and (x_e,y_e) are computed by:

① when line $|\tan\theta_s| > \tan^{-1}[(y_1-y_0)/(x_1-x_0)]$,

$$x_s = \frac{(y_0-\alpha)}{\tan\theta_s},\ x_e = \frac{(y_1-\alpha)}{\tan\theta_s},\ y_s = y_0,\ y_e = y_1, \quad \text{(A.5)}$$

② when line $|\tan\theta_s| < \tan^{-1}[(y_1-y_0)/(x_1-x_0)]$,

$$x_s = x_0,\ x_e = x_1,\ y_s = \tan\theta_s x_0 + \alpha,\ y_e = \tan\theta_s x_1 + \alpha. \quad \text{(A.6)}$$

REFERENCES

1. A. D. Bimbo, L. Landi and S. Santini, Determination of road directions using feedback neural nets, *Signal Process.* **32**, 147–160 (1993).
2. R. O. Duda and P. E. Hart, *Pattern Recognition and Scene Analysis*. Wiley, New York (1973).
3. N. Guil, J. Villaba and E. L. Zapata, A fast Hough transform for segment detection, *IEEE Trans. Pattern Analysis Mach. Intell.* **4**(11), 1541–1548 (1995).
4. P. L. Palmer, K. Kittler and M. Petrou, Using of attention with the Hough transform for accurate line parameter estimation, *Pattern Recognition* **27**(9), 1127–1134 (1994).
5. Y. Zhang and R. Webber, A windowing approach to detecting line segment using Hough transform, *Pattern Recognition* **29**(2), 255–265 (1996).
6. T. M. Breuel, Finding lines under bounded error, *Pattern Recognition* **29**(1), 167–178 (1996).
7. D. H. Ballard and C. M. Brown, *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey (1982).
8. R. G. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts (1992).
9. P. C. Chung, C. T. Tasi, E. L. Chen and Y. N. Sun, Polygonal approximation using competitive Hopfield neural network, *Pattern Recognition* **27**(11), 1505–1512 (1994).
10. A. Pikaz and I. Dinstein, Optimal polygonal approximation of digital curves, *Pattern Recognition* **28**(3), 373–379 (1995).
11. K. Wall and P. E. Danielsson, A fast sequential method for polygonal approximation of digitized curves, *CVGIP* **28**, 220–227 (1984).
12. J. Yuan and C. Y. Suen, An optimal O(n) algorithm for identifying line segments from a sequence of chain codes, *Pattern Recognition* **28**(5), 635–646 (1995).
13. J. B. Burns, A. R. Hanson and E. M. Riseman, Extracting straight lines, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-8**(4), 425–455 (1986).
14. P. Kahn, L. Kitchen and E. M. Riseman, A fast line finder for vision-guided robot navigation, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**(11), 1098–1102 (1990).
15. R. Deriche, Fast algorithm for low-level vision, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**(1), 78–87 (1990).
16. O. Faugeras, *Three-Dimensional Computer Vision—A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts (1993).
17. J. L. Melsa and D. L. Cohn, *Decision and Estimation Theory*. McGraw-Hill, New York (1978).
18. E. Horowits and S. Sahni, *Fundamentals of Data Structure in Pascal*. Computer Science Press, Rockville, Maryland (1984).
19. F. van deer Heijden, Edge and line feature extraction based on covariance models, *IEEE Trans. Pattern Analysis Mach. Intell.* **17**(1), 16–32 (1995).

About the Author—JOON-WOONG LEE received BS degree in Industrial Engineering in 1984 from Cheonnam National University and MS degree in CAD/CAM in 1986 from KAIST (Korea Advanced Institute of Science and Technology), Seoul, S. Korea. He has worked in the Production Engineering Division and Electronic Research Laboratory in KIA Motors Corp. since 1986. Now he is working toward a Ph.D. degree in Computer Vision at KAIST. His current research interest includes object recognition, image processing, moving object tracking and intelligent vehicle.

About the Author — IN-SO KWEON received the BS and ME degrees in Mechanical Design and Production Engineering from Seoul National University, Seoul, S. Korea, in 1983 and the Ph.D. degree in Robotics from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1990. He is an Associate Professor of Electrical Engineering at KAIST. Before joining KAIST, he was a Visiting Research Scientist in the Information Systems Laboratory at Toshiba Research and Development Center and he worked on behavior-based mobile robots and motion vision research. His current research interest includes industrial application of vision, motion analysis, invariant vision and mobile robots.