Proceedings of the 11th
KACC, October 1996

Line Feature Extraction in a Noisy Image

° Joon-Woong Lee *, Hak-Seo Oh **, In-So Kweon ***

* KIA Technical Center, Kia Motors
781-1, Soha-Dong, Kwang Myung-Shi, Kyunki-Do, Korea.
Tel:+82-2-801-2049; Fax:+82-2-801-2400
** Production Engineering Center, Samsung Electronics
416, Maetan-3Dong, Suwon City, Kyungki-Do, Korea.
Tel:+82-331-200-2432; Fax:+82-331-200-2454
*** Department of Automation and Design Eng., Korea Advanced Institute of Science and Technology
207-43, Cheongryangridong, Dongdaemoongu, Seoul, Korea.
Tel:+82-2-958-3455; Fax:+82-2-968-1638

Abstract: Finding line segments in an intensity image has been one of the most fundamental issues in 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 features extraction" algorithm which extracts line feature in a single pass without using any assumptions and constraints. Our algorithm consists of five steps: (1) edge scanning, (2) edge normalization, (3) line-blob extraction, (4) line-feature computation, and (5) line linking. By using 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.

Keywords: line extraction, edge extraction, edge scanning, edge normalization, line-blob extraction, 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[2, 7,8], polygonal approximation[3,9,10,14], arm method[3], chain coding[11] and φ-s curve[3]. The well-known Hough technique is probably the most popular method. This method, however, 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.*[2] 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.*[4] 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.*[1] used a similar approach to Burns *et al.*[4] and improved the speed. Yuan and Suen[11] 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[8] introduced a variety of statistical error models to extract the maxima of the probabilistic Hough transform and the generalized Hough transform. Guil *et al.*[7] 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[6], and remove the edge pixels with low gradient norm by non-maximum suppression[5] and hysteresis thresholding[5]. According to the gradient space partitioning scheme proposed by Burns *et al.*[4], 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[12]. Edge scanning, which provides the connected edge chains, is used to increase the computational 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[3]. Finally, we compute the line features for each line-blob.

Our proposed algorithm has several benefits: For any unconstrained outdoor complex scene our 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 segments extraction algorithm is organized as shown in Fig. 1.

2.1 Edge processing

In edge processing, we obtain the magnitude and the direction of the gradient for each pixel using the Deriche operator[6] and remove the edge pixels with low gradient norms by the non-local maximum suppression[5] and the hysteresis thresholding[5]. These two processes deliver the effect of edge thinning and reduce the computational complexity. 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.

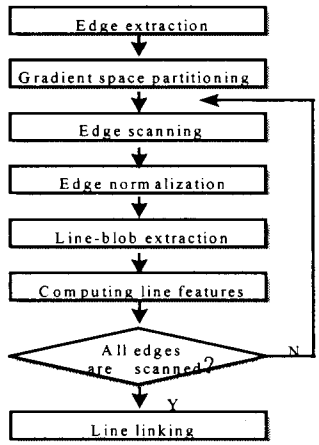


Fig. 1. Overall procedure of a line segment extraction

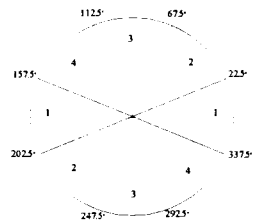


Fig. 2. Four-directional gradient space partitioning

2.2 Line-blob extraction and computation of line features

A. 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. We also use the contextual relationship of scanned edge pixels in the edge normalization process. The eight-directional chain coding[3] is used to scan the edges.

B. 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:

$$P(z|m_1) = \frac{\sum_{j=k-n}^{k+n+1} f_j}{2n+2}, \quad f_j = \begin{cases} 1 & \text{if } d_j = d_k, \ j = k-n,\cdots,k+n+1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$P(z|m_2) = \frac{\sum_{j=k+1}^{k+n+1} f_j}{n+1}, \quad f_j = \begin{cases} 1 & \text{if } d_j = d_{k+1}, \ j = k+1,\cdots,k+n+1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $m_1 = \{\text{gradient code of } p_{k+1} = d_k\}$ and

$m_2 = \{\text{gradient code of } p_{k+1} = d_{k+1}\}$

represent message spaces for the decision rule.

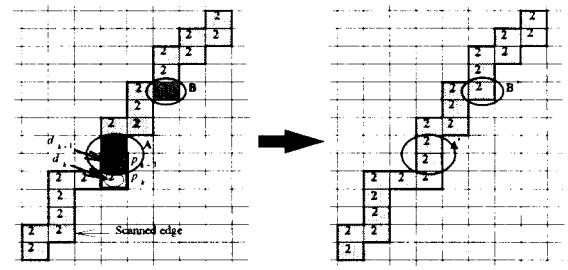
If the *likelihood ratio* $\Lambda(z)$ is defined as

$$\Lambda(z) = \frac{P(z|m_1)}{P(z|m_2)} \tag{3}$$

then we decide the decision rule associated with message spaces

$$\Lambda(z) \underset{m_2}{\overset{m_1}{\gtrless}} 1 \tag{4}$$

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



(a) Before normalization (b) After normalization

Fig. 3. Edge normalization

The gradient direction codes of pixels indicated by A and B in Fig. 3(a) are converted to the ones indicated by A′ and B′ in Fig. 3(b)

C. 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[3]. 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 mid-point, intercept of y-axis, orientation, end points and length.

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 construct 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

$$\max(l_1, l_2) < \tau_1 \tag{5}$$

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*.

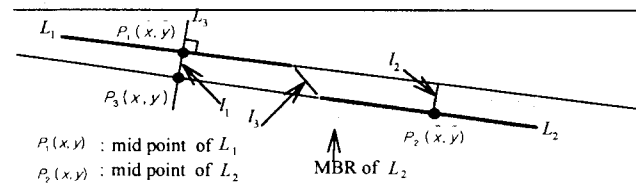


Fig. 4. Line linking

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

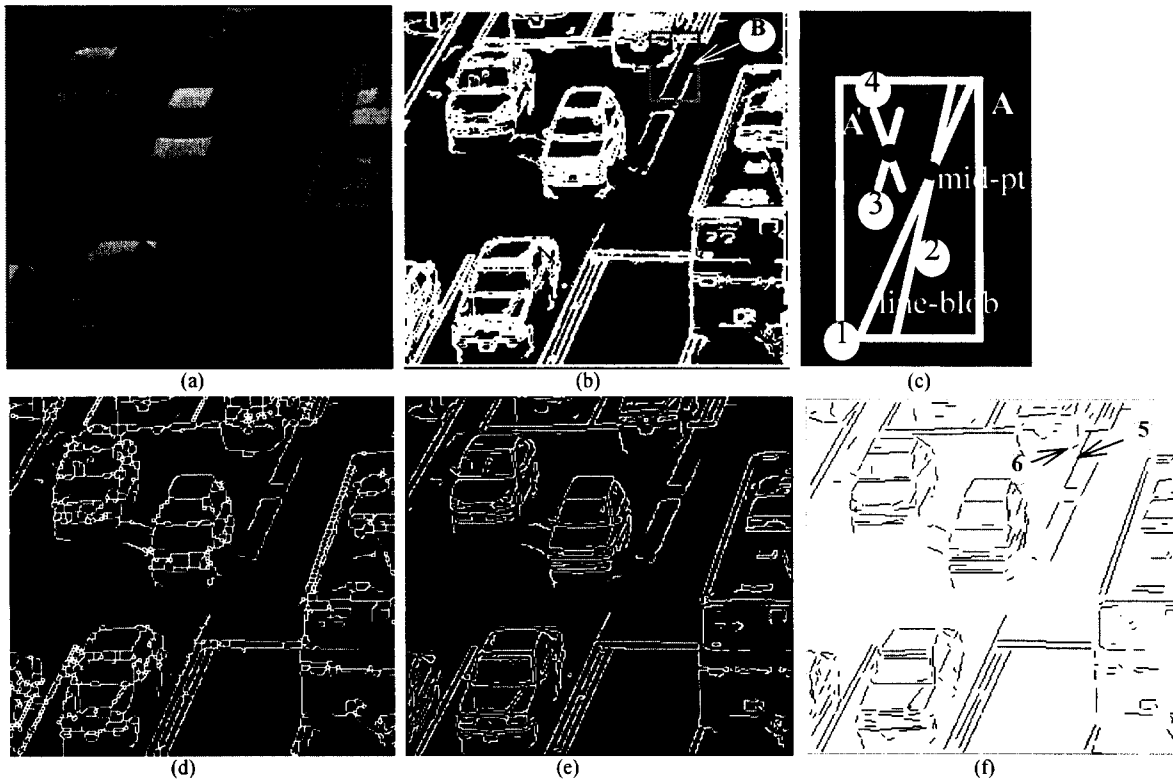


Fig. 5. Analysis of effects of edge processing and line orientation generation methods using a real image: (a)Original image, (b)Edges from hysteresis thresholding, (c)Extended edges of rectangle indicated by B in (b) and line fitting results, (d)Thinned edges from edges of (b), (e)Edges from non-local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

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 principal axis-based method to the line orientation. To do this, we use a real image as shown in Fig. 5. In most cases, edge thresholding leads to large localization errors and many multiple detections of a single edge as shown in Fig. 5(b). 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 often produces segmentation defects such as broken smaller chains and inaccurate edge localization as shown in Fig. 5(d). Edges obtained by non-local maximum suppression and hysteresis thresholding do not show such effects as shown in Fig. 5(e).

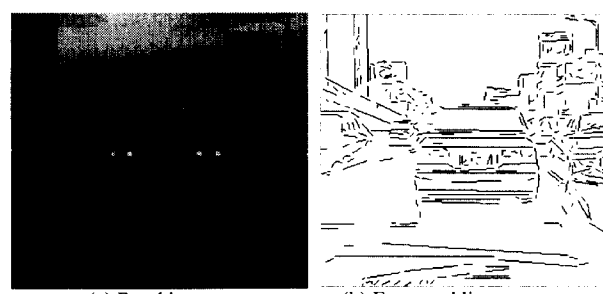
In Fig. 5(c), which shows the extended edge image encompassed by the rectangle indicated by B in Fig. 5(b), we note that pixels in the region marked by A and A′ have great intensity changes. Therefore, more edge pixels are left and dominate the line orientation. In Fig. 5(c), for example, lines ① and 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 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 images which are composed of a laboratory image and two types of road images. At first, we took an example of our laboratory image as shown in Fig. 6(a).

(a) Laboratory image (b) Extracted line segments

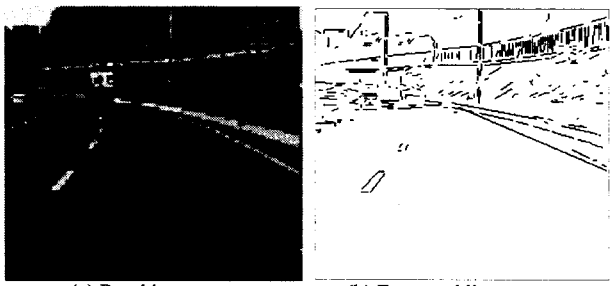
Fig. 6. Laboratory image, its line segments



(a) Road image (b) Extracted line segments

Fig. 7. Road image, its and line segments for a car following system

Fig. 6(b) shows the extracted line segments composed of four-hundred-seventy-one segments. Before line linking, five-hundred twenty line segments were extracted. Fig. 7 and Fig. 8 show the line segments for outdoor road scenes.

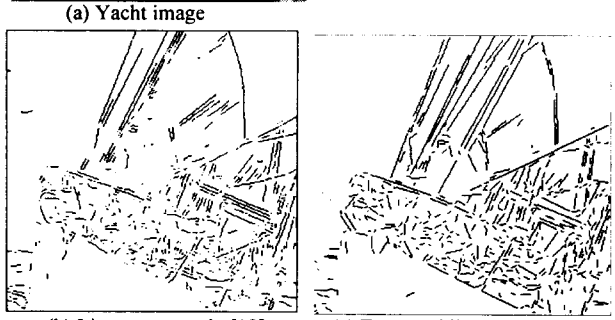


(a) Road image (b) Extracted line segments

Fig. 8. Road image, its line segments for a lane tracking system

The following example is an image of a yacht as shown in Fig. 9(a). Fig. 9(b) shows a line map derived from a rotational invariant edge feature extraction and the resulting log-likelihood ratio provided by courtesy of F. Heijden[13]. His method emphasized local multiple step edges. Fig. 9(c) shows the extracted 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) Yacht image



(b) Line segments by[13] (c) Extracted line segments

Fig . 9. Original image, its extracted edges and line segments

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. Experimental results with some real scenes showed that the proposed algorithm works well in any environment. We expect our work will contribute to many areas of computer vision.

REFERENCES

[1] P. Kahn, L. Kitchen and E. M. Riseman, A Fast Line Finder for Vision-Guided Robot Navigation, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 12, No. 11, pp.1098-1102, 1990.

[2] A.D. Bimbo, L. Landi and S. Santini, Determination of Road Directions Using Feedback Neural nets, *Signal Processing*, Vol. 32, pp. 147-160, 1993.

[3] D.H. Ballard and C.M. Brown, Computer Vision, Prentice-Hall, Englewood Cliffs NJ(1982).John Wiley & Songs, Inc. 1976.

[4] J.B. Burns, A.R. Hanson, and E.M. Riseman, Extracting Straight Lines, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol.PAMI-8, No. 4 pp. 425-455,1986.

[5] O. Faugeras, Three Dimensional Computer Vision - A Geometric Viewpoint, The MIT Press, England. 1993.

[6] R. Deriche, Fast Algorithm for Low-Level Vision, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 12, No. 1, pp. 78-87,1990.

[7] N. Guil, J. Villaba and E. L. Zapata, A Fast Hough Transform for Segment Detection, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 4, No. 11, pp. 1541-1548,1995.

[8] T. M. Breuel, Finding Lines Under Bounded Error, *Pattern Recognition*, Vol. 29, No. 1, pp. 167-178, 1996.

[9] P.C. Chung, C.T. Tasi, E.L. Chen and Y.N. Sun, Polygonal Approximation Using Competitive Hopfield Neural Network, *Pattern Recognition*, Vol. 27, No. 11, pp. 1505-1512, 1994.

[10] A. Pikaz and I. Dinstein, Optimal Polygonal Approximation of Digital Curves, *Pattern Recognition*, Vol. 28, No. 3, pp. 373-379, 1995.

[11] J. Yuan and C.Y. Suen, An Optimal O(n) Algorithm for Identifying Line Segments from a Sequence of Chain Codes, *Pattern Recognition*, Vol. 28, No. 5, pp. 635-646, 1995.

[12] J.L. Melsa and D.L. Cohn, Decision and Estimation Theory, McGraw-Hill, New York, 1978.

[13] F. van deer Heijden, Edge and Line Feature Extraction Based on Covariance Models, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 17, No. 1, pp. 16-32, 1995.

[14] K. Wall and P.E. Danielsson, A Fast Sequential Method for Polygonal Approximation of Digitized Curves, *CVGIP*, Vol. 28, pp. 220-227, 1984