

A straight line detection using principal component analysis

Yun-Seok Lee, Han-Suh Koo, Chang-Sung Jeong *

Department of Electronics Engineering, Korea University, Anam-Dong, Seongbuk-Gu, Seoul, Republic of Korea

Received 16 August 2005; received in revised form 30 March 2006

Available online 22 June 2006

Communicated by G. Sanniti di Baja

Abstract

A straight line detection algorithm is presented. The algorithm separates row and column edges from edge image using their primitive shapes. The edges are labeled, and the principal component analysis (PCA) is performed for each labeled edges. With the principal components, the algorithm detects straight lines and their orientations, which is useful for various intensive applications. Our algorithm overcomes the disadvantages of Hough transform (HT) and other algorithms, i.e. unknown grouping of collinear lines, complexity and local ambiguities. The experimental results show the efficiency of our algorithm.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Straight line detection; Principal component analysis (PCA); Line descriptor; Edge image

1. Introduction

Straight line detection is a fundamental field in computer vision, e.g. vanishing point detection, parallel line detection and line matching across views (Lutton et al., 1994; Schmid and Zisserman, 1997). A few models have been proposed in literature; among them, Hough transform (HT) is the most well-known algorithm (Hough, 1962; Duda and Hart, 1972). In HT, each edge pixel is voted upon a quantized parameter space. Each cell in the accumulator array for the quantized parameter space corresponds to a straight line. Modified versions of HT have been proposed such as fast HT (Li et al., 1986), adaptive HT (Illingworth and Kittler, 1987), combinatorial HT (Ben-Tzvi and Sandler, 1990) and hierarchical HT (Princen et al., 1990). However, regardless of its robustness, the actual distribution of edges is not known, because HT is a mandatory grouping method for collinear lines.

In the method presented by Nevatia and Babu (1980), edge pixels are extracted using six 5×5 gradient masks for threshold detection and thinning. Edge pixels are linked to other pixels which are identified as the same linear segment. The contours are approximated by relative straight line pieces using an iterative end-point fitting method.

In (Burns and Hanson, 1986), pixels are grouped into line-support regions of similar gradient orientation. The intensity surface associated with each line-support region is approximated by a planar surface. Straight lines are extracted by intersecting this fitted plane with a horizontal plane representing the average intensity of the region weighted by a local gradient magnitude. Although long lines can be extracted, the method is rather complex.

In (Venkateswar and Chellapa, 1992), a straight line extractor for aerial images is proposed. The extractor first scans the edge image to generate a label image using various templates for edge direction. Next, the extractor assigns the same label to the edge pixels in the same line. Then, the lines are merged, if they are identified as a collinear line. The disadvantage of this method, however, is that it has to consider a large number of templates.

* Corresponding author. Tel.: +82 2 929 0985; fax: +82 2 926 7620.
E-mail addresses: leey@snoopy.korea.ac.kr (Y.-S. Lee), hskoo@korea.ac.kr (H.-S. Koo), csjeong@charlie.korea.ac.kr (C.-S. Jeong).

Recently, Guru et al. (2004) proposed a line detection algorithm based on small eigenvalue. The small eigenvalue of the covariance matrix of the edge pixels covered by a mask of a proper size is estimated. The mask is moved pixel by pixel, so that each edge pixel has a number of eigenvalues because of the overlapping of masks. Consequently, each edge pixel is associated with the minimum eigenvalue of all the small eigenvalues that are assigned to it; and if the minimum eigenvalue is less than a pre-defined threshold value, the pixel is said to be linear edge pixel. However, the performance can be damaged by either unsuitable size of mask or excessive noise pixels in the edge image. More recently, they presented papers for the algorithm using principal component analysis (PCA) (Nagabhushan et al., 2005; Shekar et al., 2006). Especially, they proposed an algorithm for edge detection using eigenvalue (Nagabhushan et al., 2005), so they can have straightforward procedure for line detection using PCA because the line detection needs an edge image for the first time. Besides, in their paper for object recognition (Shekar et al., 2006), they used their line detection method as preprocessing, and again they used PCA to obtain principal component vectors of objects. In this way, they have proposed various applications for PCA from edge detection to object recognition.

In this paper, we describe an efficient algorithm for straight line detection. As discussed in the following section, the algorithm extracts straight lines from edge image using PCA. First, we divide edge segments into row- and column-directional edge pieces to label them separately. Second, we analyze each labeled line to obtain its principal direction and straightness. Straight lines are then detected, if the small eigenvalue of line is less than a pre-defined value, which may be changed relative to the length of the line. Section 3 will discuss the threshold for determining straightness of lines. In Section 4, experimental results will be shown. The discussion and conclusion will be addressed in Section 5.

1.1. Terminology

We discriminate “straight line” from line. We also use “primitive” as a basic segment of a line which is like a bar. Therefore, “row edge” consists of row primitives and “column edge” consists of column primitives. The edge pixel that has no neighbor in four directions is called “single” primitive, and the pixel that can be an element of both row and column primitives is called “cross” primitive (see Fig. 3(b)).

2. Proposed algorithm

Edge images have various line segments such as curve, circle and straight line. In digital coordinates, however, there exist only two types of straight lines, i.e. row-directional line and column-directional line. The combination of those lines makes a curve roughly. Thus, we can say that all edge segments are basically straight lines in a digital scheme, but the segments of a line gradually change the orientation of the line to make an arbitrary shape. The purpose of our algorithm is to find the lines that have small variation of the orientation. The overall diagram of our algorithm is shown in Fig. 1.

2.1. Edge separation and labeling

We first extract row and column edges from the edge image detected by Canny edge detector (Canny, 1986), and the detected edges are labeled for the next step. The following Algorithm 1 shows the procedure for the extraction of row and column edges.

```

Algorithm 1: Row and column edge extraction ( $p[i][j]$ :
pixel at  $(i, j)$ )
for all  $i$  and  $j$ 
  if both  $p[i][j]$  and  $p[i][j + 1]$  are edge pixels
    register  $p[i][j]$  and  $p[i][j + 1]$  as row mark
  end if
  if both  $p[i][j]$  and  $p[i - 1][j]$  are edge pixels
    add column mark to the register of  $p[i][j]$  and
    of  $p[i - 1][j]$ 
  else if  $p[i][j]$  is an edge but it has no mark in the
  register
    register  $p[i][j]$  as single mark
  end if
end for

```

Some edge pixels are marked as both row and column edges, if they are center pixels of cross lines. Therefore, after above processing, there are four types of marks, i.e. row, column, row and column (or cross), and single. Next, row and column edge segments are labeled separately using edge connectivity.

Since the segments of a straight line run along the same direction, we separately label row and column edges using

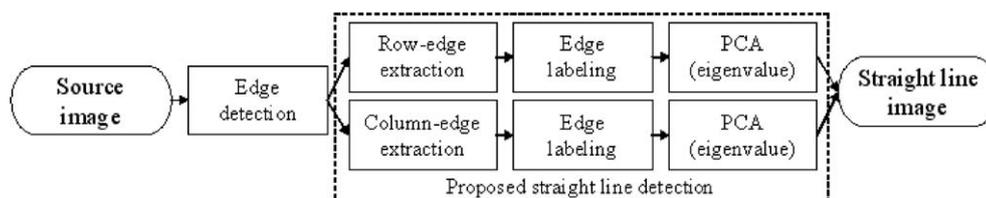


Fig. 1. The overall diagram of our algorithm.

8-neighbor connectivity to extract the primitive type of a straight line. First, the label register for the row has the labels of all the edges except for the column marked edges. In a similar way, the label register for the column has the labels except for the row marked edges. If a label consists of only single edges, it should remain only in one of the registers to avoid redundant processing. Thus, the column register is used for the label consisting of only single edges. The labeling algorithm is shown in Fig. 2 and the example of the algorithm is shown in Fig. 3.

2.2. Straight line detection using PCA

PCA is a well-known metric method that produces the base axes of a distribution of data (Duda et al., 2001). Given an ideal straight line in two dimension, it has some principal components deduced from the eigenvectors and

eigenvalues of the scatter matrix. The eigenvector means one main direction of the distribution of the pixels of a line and the eigenvalue means how long the distribution is. Generally, the first eigenvalue is larger than the second one, so that, in the case of ideal straight line, the second eigenvalue should be zero. However, a digital line is represented stepwise, so that the second eigenvalue of the line cannot be zero. The tolerance for that case will be addressed and determined in the next section. The scatter matrix is as follows:

$$S = \begin{pmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{pmatrix}. \quad (1)$$

If n is the number of pixels in a line and (x_i, y_i) is the coordinates of the i th pixel of the line,

$$s_{11} = \frac{1}{n} \sum_{i=1, \dots, n} (x_i - x_m)^2, \quad (2)$$

$$s_{12} = s_{21} = \frac{1}{n} \sum_{i=1, \dots, n} (x_i - x_m)(y_i - y_m), \quad (3)$$

$$s_{22} = \frac{1}{n} \sum_{i=1, \dots, n} (y_i - y_m)^2, \quad (4)$$

where

$$x_m = \frac{1}{n} \sum_{i=1, \dots, n} x_i, \quad y_m = \frac{1}{n} \sum_{i=1, \dots, n} y_i. \quad (5)$$

The large eigenvalue λ_1 and the small eigenvalue λ_2 of the scatter matrix are

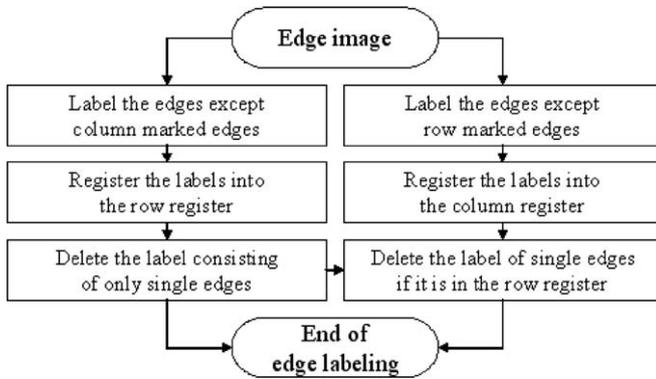


Fig. 2. The edge labeling algorithm.

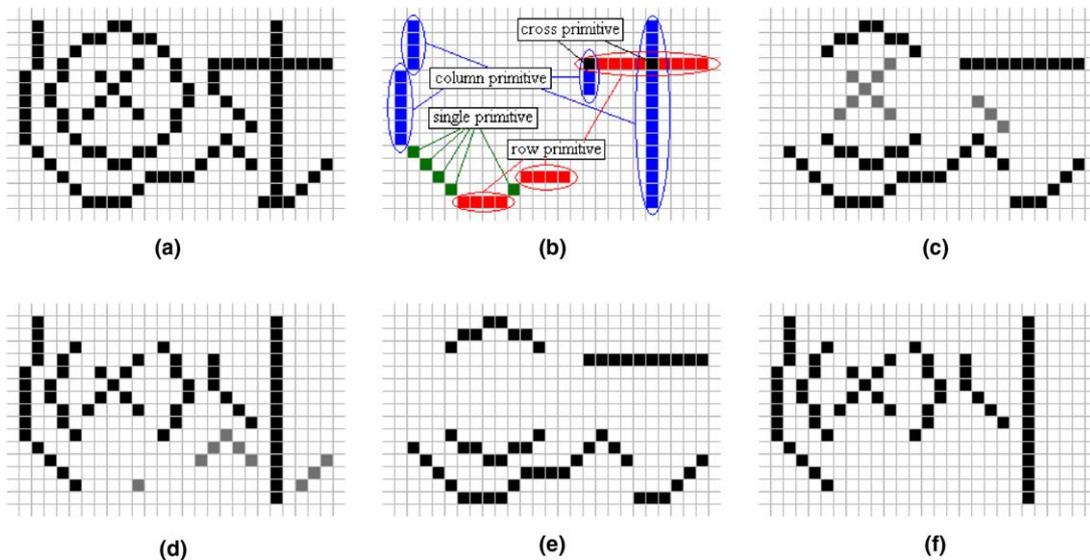


Fig. 3. Example of edge labeling: (a) edge image, (b) blue, red, green, and black is column, row, single, and cross primitive respectively, (c) is the edge image except column marked edges, (d) is the edge image except row marked edges. After eliminating gray points in (c) and (d), the edges consisting of only single primitives remain in column edge image, (f) while row edge image and (e) does not have such edges. (For interpretation of references in color in this figure legend, the reader is referred to the web version of this article.)

$$\lambda_1 = \frac{1}{2} \left\{ s_{11} + s_{22} + \sqrt{(s_{11} - s_{22})^2 + 4s_{12}^2} \right\}, \quad (6)$$

$$\lambda_2 = \frac{1}{2} \left\{ s_{11} + s_{22} - \sqrt{(s_{11} - s_{22})^2 + 4s_{12}^2} \right\}. \quad (7)$$

We can only know the proportion of the elements of the eigenvector, and so the angle of the line, θ is defined as

$$\theta = \tan^{-1} \frac{(\lambda_1 - s_{11})}{s_{12}} \quad \text{or} \quad \theta = \tan^{-1} \frac{s_{21}}{(\lambda_1 - s_{22})}. \quad (8)$$

Above two equations are identical because $\det(\lambda_1 I - S) = 0$. In this way, the straightness of a line and its angle are obtained. Thus, the position and orientation of a line is easily taken from the eigenvector of the scatter matrix as it is obtained from HT. However, HT gives the information for a group of the lines in the same orientation, while PCA gives such information for a separate line.

2.3. Time complexity

If an image has k edge pixels, row and column edge separation has $O(2k) = O(k)$ because every pixel is compared to the next and below pixel. After that, if the edge images have l lines and the average length (pixels) of lines is a , the labeling processing takes $O(la^2)$, because in the worst case, every pixel of a line has its own label so that it will be compared to one another. However, PCA is an arithmetic calculation and it has $O(1)$. Therefore, the sum of the order is $O(k) + O(la^2) \times O(1) = O(k) + O(la^2)$, but roughly $k = la$; so the time complexity is $O(ka)$.

3. Determining the threshold for the small eigenvalue λ_2

A line should be long enough to validate its straightness and angle. For that reason, we use lines at least 30 pixels long to see how large their eigenvalues are. First of all, to see the variation of the small eigenvalue λ_2 of straight line, 16 straight lines are used with the range from $\pi/4$ to $\pi/2$ as shown in Fig. 4(a). The characteristics of straight lines beyond that range would be the same as these 16 lines, because of the symmetrical property of angle.

Table 1 shows the angles and small eigenvalues of the lines in Fig. 4(a). The small eigenvalues of the lines were less than 0.075; but we could not take that value as such for the threshold of the straightness on account of noise effects in edge and image. Thus, we made an experiment with the lines in Fig. 4(b) that might be considered as straight lines to some extent.

Table 2 shows PCA data of the lines in Fig. 4(b). Some of them could be detected for straight line; for example, 4th–11th lines can be said that they have a little straight-

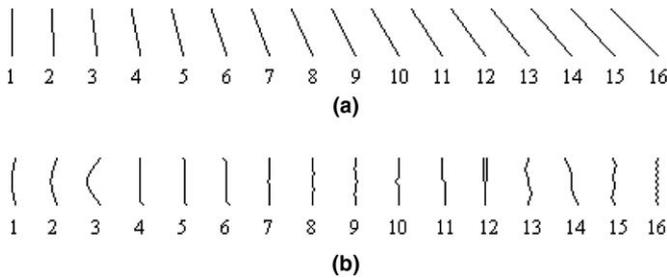


Fig. 4. (a) Short straight lines of the range from $\pi/4$ to $\pi/2$ in 16 parts and (b) noise lines which are likely to be straight lines.

Table 1
Angles and small eigenvalues of 16 short lines in Fig. 4(a)

Type	1	2	3	4	5	6	7	8
Angle	1.5708	1.5207	1.4496	1.3783	1.3249	1.2515	1.2091	1.1435
Eigenvalue	0.0	0.0621	0.0728	0.0750	0.0695	0.0662	0.0703	0.0639
	9	10	11	12	13	14	15	16
Angle	1.1055	1.0446	0.9918	0.9440	0.9035	0.8670	0.8317	0.7854
Eigenvalue	0.0498	0.0583	0.0610	0.0528	0.0481	0.0438	0.0401	0.0

Table 2
Angles and small eigenvalues of 16 noise lines in Fig. 4(b)

Type	1	2	3	4	5	6	7	8
Angle	1.5708	1.5708	1.5708	1.5519	1.5579	1.5329	1.5708	1.5708
Eigenvalue	0.5956	1.5822	7.1556	0.1298	0.0542	0.2259	0.0622	0.1156
	9	10	11	12	13	14	15	16
Angle	1.5708	1.5708	1.4710	1.5708	1.4901	1.3536	-1.5388	1.5708
Eigenvalue	0.1956	0.2933	0.2159	0.6667	0.7148	0.4406	0.4404	0.2489

Table 3
Small and large eigenvalues of five row lines in Fig. 5

Type	1	2	3	4	5
Size (pixels)	30	60	90	120	150
Angle	0.0782	0.0848	0.0898	0.0928	0.0938
Small eigenvalue	0.0735	0.0813	0.0817	0.0814	0.0820
Large eigenvalue	75.3765	302.0854	680.3905	1210.3186	1891.5013



Fig. 5. Five lines of different size.

ness properties. By this observation, we found that a value between 0.25 and 0.3 would be proper for the *absolute* threshold t_a .

As a line is getting longer, there might be short noise primitives destroying the straightness of the line. Therefore, long lines should be compensated by adjusting the threshold according to the length of the line. Although there is a little error, Table 3 shows that the large eigenvalue is proportional to the square of the length of line (refer to Fig. 5). Since we had an absolute threshold, t_a for at least 30-pixel-long line, the threshold for the small eigenvalue relative to the length, t_i can be as follows:

$$t_i = \frac{l_i^2}{30^2} t_a, \quad (9)$$

where l_i is the length of the i th line in row or column edge image, and the length is simply measured by the number of pixels of the line.

As was expected, the absolute threshold gives a small number of straight lines but very reliable result, while the *relative* threshold gives more lines with a little inaccuracy. The next section has more details.

4. Experimental results

4.1. Synthetic image

We compared our algorithm to the algorithm by Guru et al. with a synthetic image they had used (Fig. 6(a)). Fig. 6 shows the experimental result of the synthetic image with impulse noises (Fig. 6(b)). The algorithms differ in that our algorithm focuses on the detection of the entire line of the same label having straightness property, while their algorithm focuses on the local straightness within the size of mask. Thus, as seen in Fig. 6(d), four round corners of the rectangular on the left were missed, while our algorithm detected them (see Fig. 6(c)); and besides, their algorithm also detected straight parts of the circle, but ours did not detect any part of the circle. One of the advantages of our algorithm is that it eliminates most impulse noises, especially when they attached to straight lines, because such noisy parts may be labeled independent of the straight lines.

4.2. Real images

We tested our algorithm with two aerial images and three building images. Canny edge detector (Canny, 1986) was applied first, and then row and column edges were extracted from the edge image as Fig. 7(a) and (d) show. All the lines were labeled, and the PCA is performed for each label in both row and column edge images. As seen in Fig. 7(b), the Pentagon image Fig. 7 has a lot of lines with noise, so that some straight lines were missed as the result of the absolute threshold of 0.25 shows in Fig. 7(e). On the other hand, more straight lines are shown in Fig. 7, where the relative threshold proportional to the square of the length is used. The result of the absolute threshold gives short but fairly reliable lines, whereas the relative threshold gives many long lines. Lines that were likely to be the south of the Pentagon were not detected due to the noise; on the contrary, the detected lines are good in their positions and lengths.

In real image, we have to consider noise effect which appears in object boundaries and makes the straight boundaries curved. If we use only the absolute threshold,

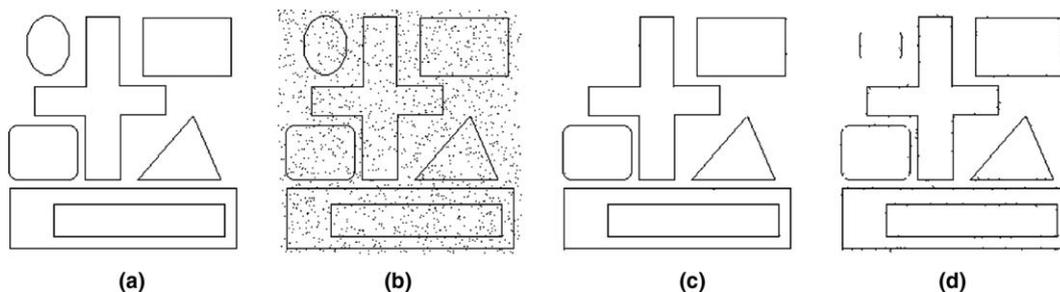


Fig. 6. Experimental results with a noise added synthetic image: (a) a synthetic image used by Guru et al., (b) noise added image, (c) the result of our algorithm and (d) the result of Guru et al. with the mask size of 7.

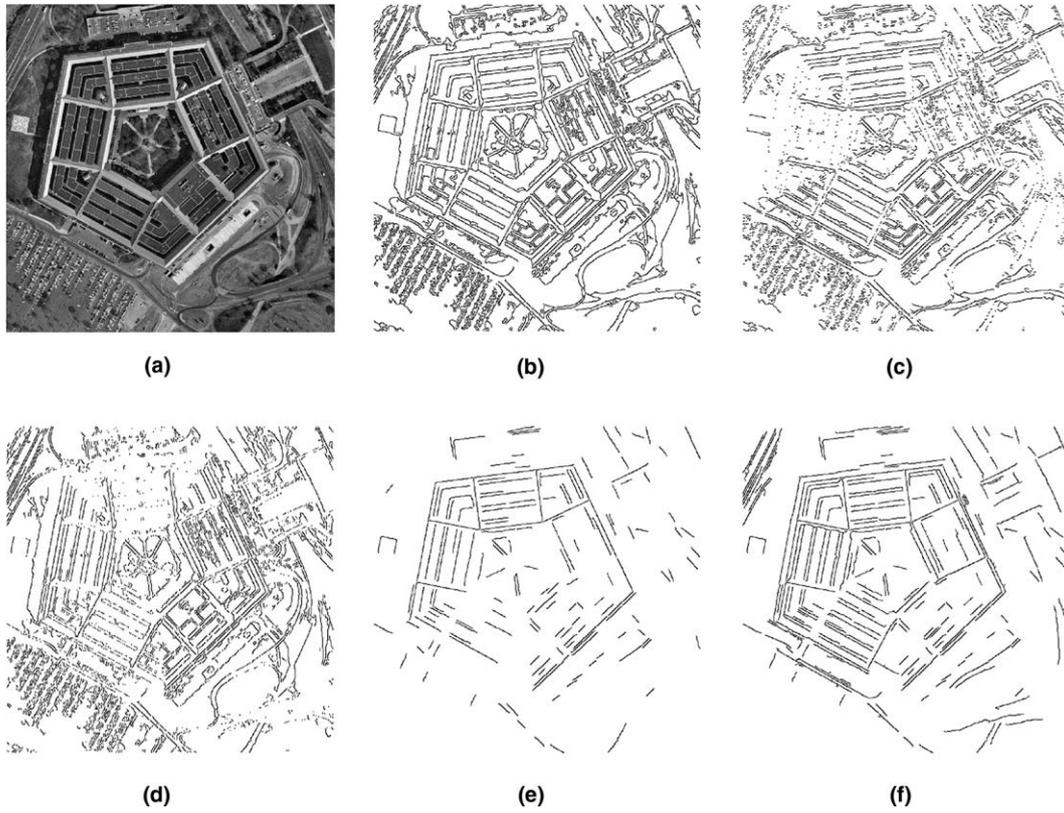


Fig. 7. (a) The pentagon image, (b) edge image, (c) row edges, (d) column edges, (e) straight lines using the absolute threshold and (f) straight lines using the relative threshold.

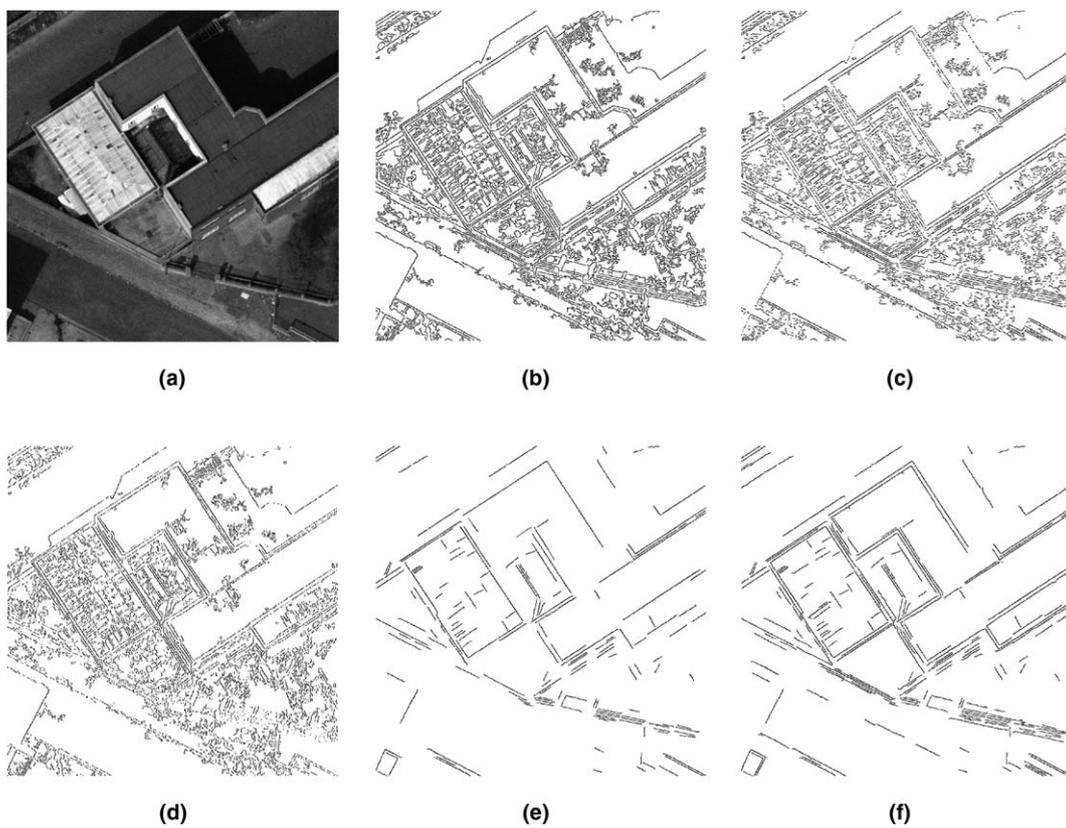


Fig. 8. (a) The aerial image, (b) edge image, (c) row edges, (d) column edges, (e) straight lines using the absolute threshold and (f) straight lines using the relative threshold.



Fig. 9. (a) House image, (b) church image and (c) college image.

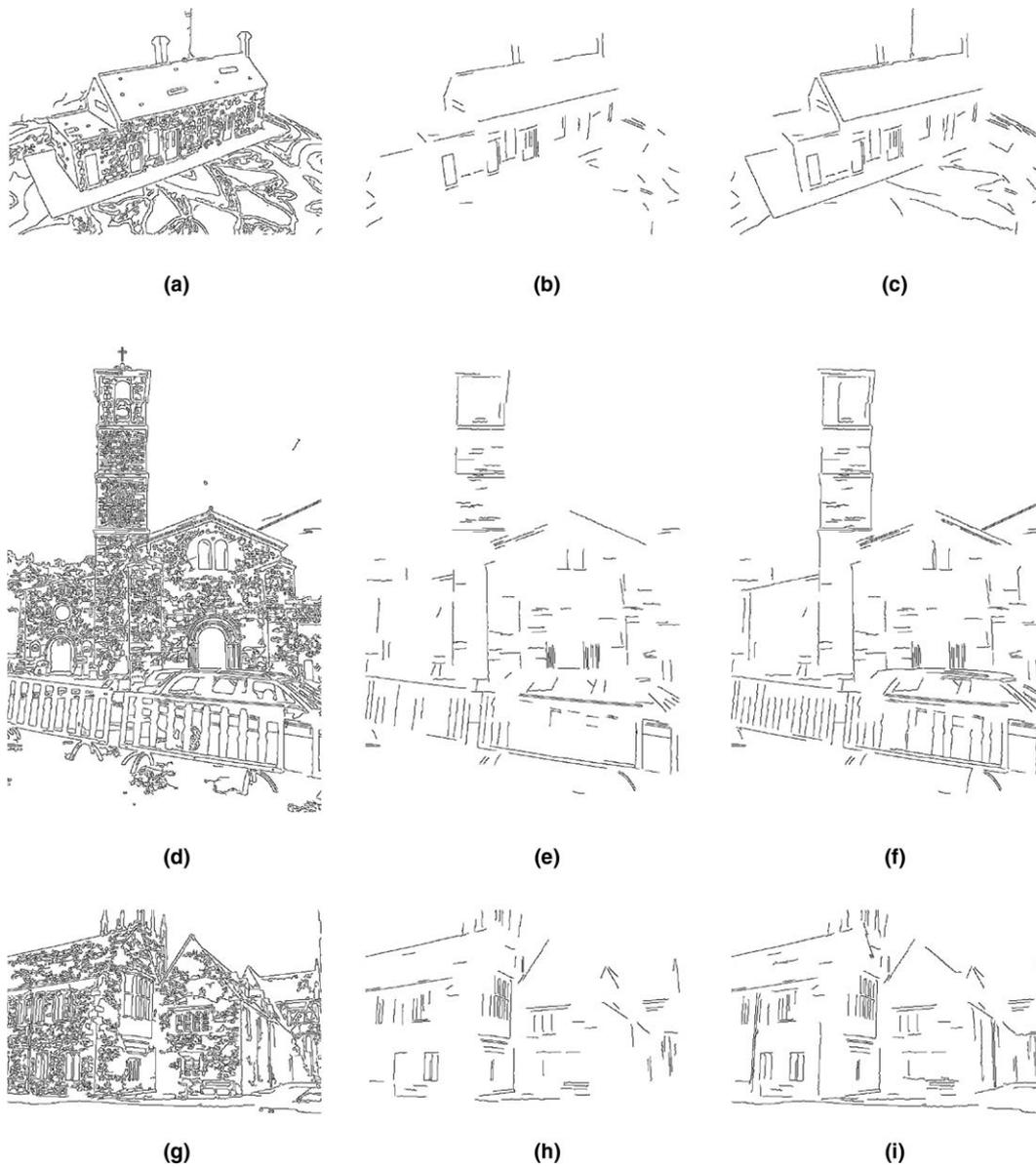


Fig. 10. The results of Fig. 9. Each row shows the result of house, church, and college respectively. Each column shows edge image, straight lines with the absolute threshold, and straight lines with the relative threshold.

there will be small number of detected lines because of noise. Besides, we found that the line is easily affected by noise as it is getting longer, so that we proposed another threshold that changes according to the length of a line to reduce noise effect and detect more lines. In summary, the application that needs exact straight line such as stereo line matching should use the absolute threshold only, but if the application that needs many straight lines such as object recognition has to use proposed relative threshold.

Aerial image Fig. 8(a) was also tested using the same procedure as above. As Fig. 8(c) and (d) shows, the separation of row and column lines from the edge image Fig. 8(b) is a simple and efficient preprocessing, because rectangular edges of roofs are easily divided into the base frame lines. We used the same threshold as used for the Pentagon image. Fig. 8(e) and (f) shows the results of the absolute and the relative threshold respectively. Fortunately, the result of the absolute threshold gives many straight lines

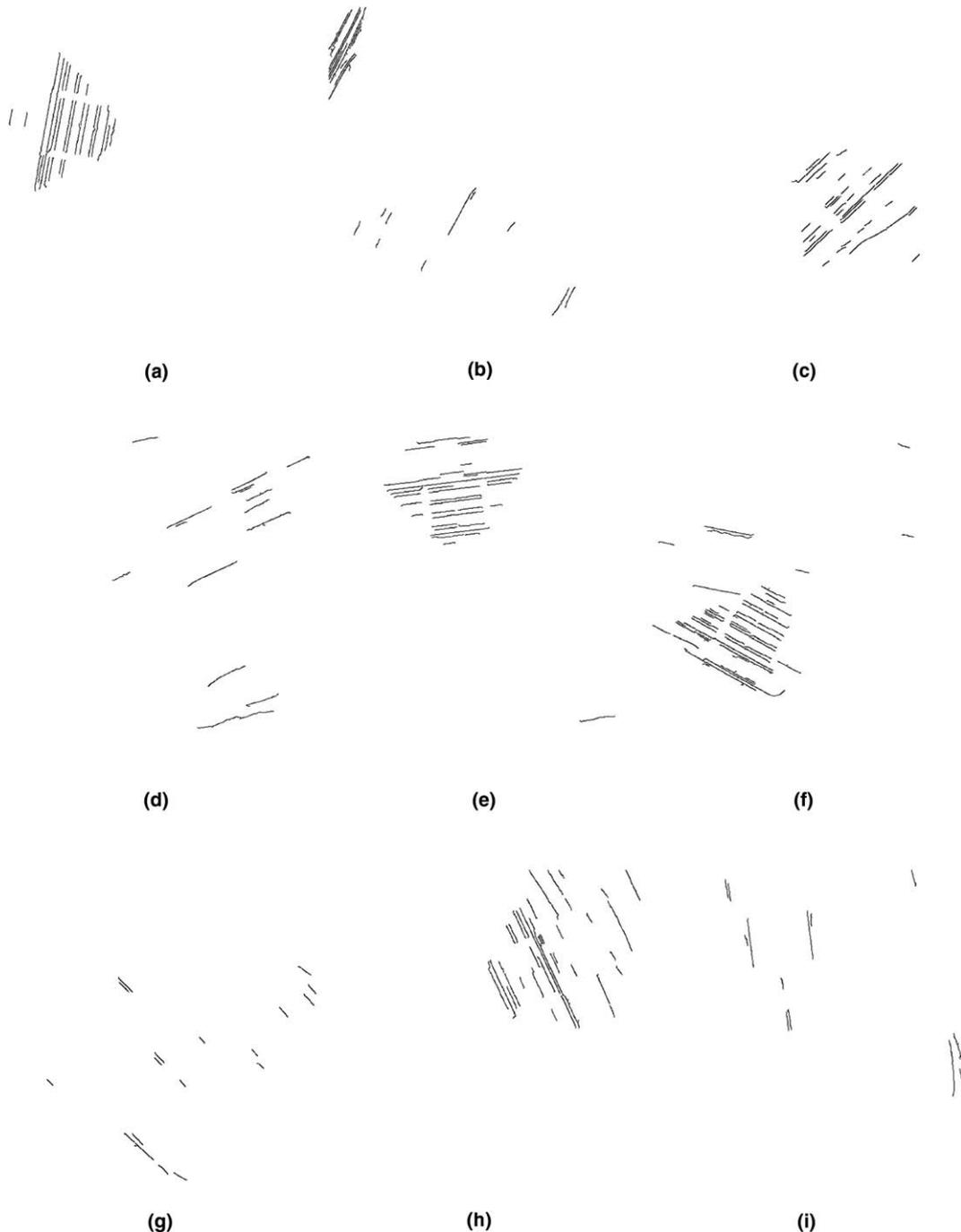


Fig. 11. Orientation estimation using the large eigenvalue of straight line for the pentagon image: (a) $-90^\circ \leq \theta \leq -70^\circ$, (b) $-70^\circ < \theta \leq -50^\circ$, (c) $-50^\circ < \theta \leq -30^\circ$, (d) $-30^\circ < \theta \leq -10^\circ$, (e) $-10^\circ < \theta \leq 10^\circ$, (f) $10^\circ < \theta \leq 30^\circ$, (g) $30^\circ < \theta \leq 50^\circ$; (h) $50^\circ < \theta \leq 70^\circ$ and (i) $70^\circ < \theta \leq 90^\circ$.

regardless of noise effect, whereas the result of the relative threshold has a few curve lines, but they are connected to long straight lines.

Other three images (see Fig. 9) were also tested with the same threshold, 0.25. Fig. 10(a)–(i) is the results of those images. The advantage of the relative threshold is that it defines the shapes of buildings more clearly as shown in the results, whereas the absolute threshold missed some

verges of buildings. The result of house image shows that important straight lines are well extracted. However, as shown in the edge image, local brightness ambiguities between object and background and between some parts of the image affected the edge localization, so that some straight lines were not detected. In church image, some bars of the front fence were missed because of the same reason of the case of house image. One of the roof lines of college

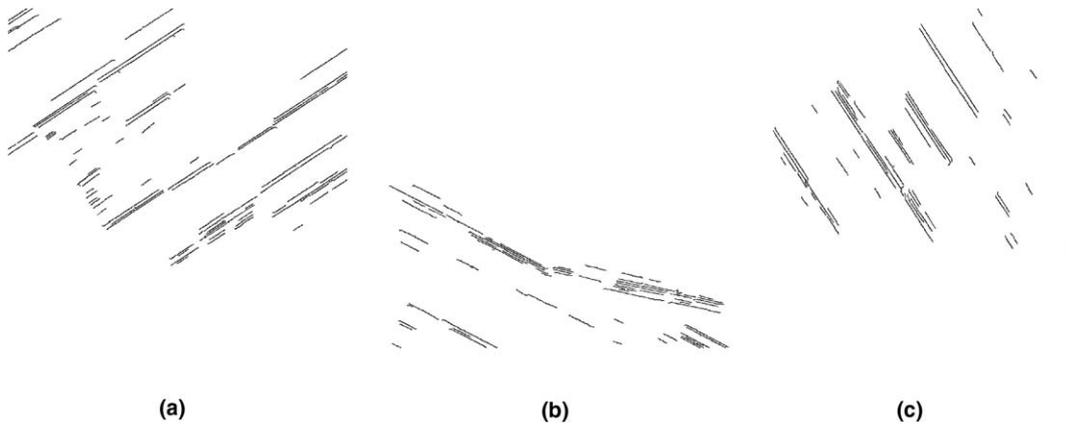


Fig. 12. (a)–(c) show the example of parallel line extraction for the Aerial image.

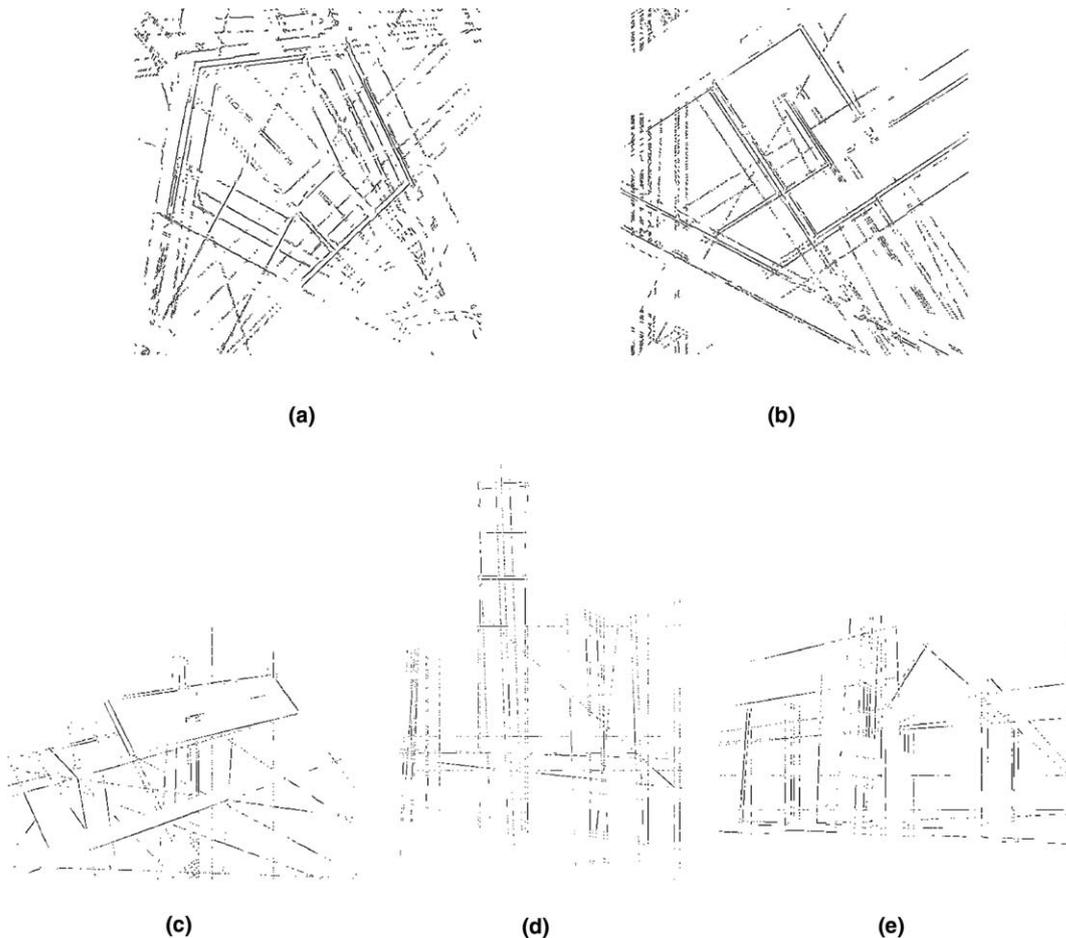


Fig. 13. The results of the standard HT: (a) the pentagon image, (b) the aerial image, (c) house image, (d) church image and (e) college image.

Table 4

Consuming time of our method and standard HT (s)

Image		Pentagon	Aerial	House	Church	College
Ours	Labeling	1.566	3.765	0.375	2.331	0.781
	The rest	0.172	0.266	0.062	0.203	0.109
	Total	1.738	4.031	0.437	2.534	0.890
HT	Total	8.584	12.716	6.472	10.741	6.555

image was also missed, and the windows of the college are so small that they are very sensitive to the noise, which means a line should be long enough to show its property. Nevertheless, all the results show a good efficiency of our algorithm, and show that the relative threshold is needed to get more information for further processing.

Figs. 11 and 12 are examples of the orientation estimation of straight lines. This work helps us do additional processing such as vanishing point detection, parallel line detection and line matching across views (Lutton et al., 1994; Schmid and Zisserman, 1997). Fig. 13 shows the results of HT with the same images we have used. However, too many edges deteriorate the performance, which is disadvantage of HT. Of course, estimated angle of a line of our result can be compared with that of HT, but our method estimates exact angle of a line, while HT obtains the approximate angle of a line from accumulator cells divided by pre-defined angle and distance. Moreover, the lines of HT are broken frequently. There are some reasons of this problem: First, HT is very sensitive to edge image. If an image has quite a lot of edges, the result of HT is very poor, and our test images have so many edges. Second, because HT gives line that is in the range of angle of an accumulator cell with a tolerance, the value from HT result is not exact. Third, HT chooses the peak value out of accumulator cells and deletes it before the next peak detection, so that some part of a line that is near the previous peak line may not be drawn. Fourth, since HT has a grouping characteristic, it often mistakes a group of edges for a line, i.e. when many edges spread over the same distance and angle, HT takes them as one line according to the accumulated votes.

We also present consuming time of our method and HT for each image in Table 4. The environment was Pentium 4 2 GHz and the platform was Visual C++ for MS-Windows. Obviously, large image takes more time like the Aerial image, and most of time of our method elapses in the edge labeling processing. We think that it is kind of trade-off between time and accuracy; that is, because we use labeling processing, we can get exact values of lines, but unfortunately it takes much of time. On the other hand, HT consumes long time because the quantization interval and peak detection take plenty of time.

5. Discussions and conclusion

Our algorithm basically uses a simple segmentation that separates row and column edges using primitive shape. In some cases, however, our algorithm fails. For instance, if

two straight lines meet with the difference between two angles small, they might have the same label, which makes the small eigenvalue for the label larger.

Chain code (Sonka et al., 1999) algorithm can be an alternative to the solution of the problem. If we use this algorithm, corners and points that have a great curvature should be detected. Besides, every end point and junction point should be extracted. There can be three types of composition of line: end-end line, end-junction line and junction-junction line. However, it may take so much time for processing. Moreover, the ambiguity about line segment and noise edge does hardly disappear.

The proposed algorithm is a simple and efficient. Especially, it is good for an edge image that has a lot of edges. In addition, our algorithm avoids mask processing that only uses local information. Solving the problem of two or more merged straight lines as mentioned above will be a future work for improving the performance of our algorithm.

Acknowledgements

This work was supported by the Brain Korea 21 and KIPA Information Technology Research Center. The Pentagon image was obtained from Carnegie Mellon Image Database and all the rest of the source images in experiment section were obtained from Visual Geometry Group at Oxford University.

References

- Ben-Tzvi, D., Sandler, M.B., 1990. A combinatorial Hough transform. *Pattern Recognition Lett.* 11 (3), 167–174.
- Burns, J.B., Hanson, A.R., 1986. Extraction straight lines. *IEEE Trans. Pattern Anal. Mach. Intell.* (4), 425–456.
- Canny, J.F., 1986. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 8 (6), 679–698.
- Duda, R.O., Hart, P.E., 1972. Use of Hough transformation to detect lines and curves in pictures. *Commun. ACM* 15 (1), 11–15.
- Duda, R.O., Hart, P.E., Stork, D.G., 2001. *Pattern Classification*, Second ed. Wiley Interscience (Chapter 3).
- Guru, D.S., Shekar, B.H., Nagabhushan, P., 2004. A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognition Lett.* 25, 1–13.
- Hough, P.V.C., 1962. Method and means for recognizing complex patterns, US Patent No. 3069654.
- Illingworth, J., Kittler, J., 1987. The adaptive Hough transform. *IEEE Trans. Pattern Anal. Mach. Intell.* 9 (5), 690–698.
- Li, H., Lavin, M.A., Le Master, R.J., 1986. Fast Hough transform: A hierarchical approach. *Comput. Vision Graphics Image Process.* 36, 139–161.
- Lutton, E., Maitre, H., Lopez-Krahe, J., 1994. Contribution to the determination of vanishing points using Hough transform. *IEEE Trans. Pattern Anal. Mach. Intell.* 16 (4), 430–438.
- Nagabhushan, P., Guru, D.S., Shekar, B.H., 2005. Eigen transformation based edge detector for gray images, *PRMI 2005. LNCS 3776*, 434–440.
- Nevatia, R., Babu, K.R., 1980. Linear feature extraction and description. *Comput. Graphics Image Process.* 13, 257–269.
- Princen, J., Illingworth, J., Kittler, J., 1990. A hierarchical approach to line extraction based on the Hough transform. *Comput. Vision Graphics Image Process.* 52 (1), 57–77.

- Schmid, C., Zisserman, A., 1997. Automatic line matching across views. Proc. IEEE Conf. Comput. Vision Pattern Recognition, 666–671.
- Shekar, B.H., Guru, D.S., Nagabhushan, P., 2006. Object recognition through the principal component analysis of spatial relationship amongst lines ACCV 2006. LNCS 3851, 170–179.
- Sonka, M., Hlavac, V., Boyle, R., 1999. Image Processing Analysis and Machine Vision, Second ed. International Thomson Computer Press (Chapter 6).
- Venkateswar, V., Chellapa, R., 1992. Extraction of straight lines in aerial images. IEEE Pattern Anal. Mach. Intell. 14, 1111–1114.