Real-Time Imaging **7**, 473–481 (2001)
doi:10.1006/rtim.2001.0233, available online at http://www.idealibrary.com on IDEAL®

A New Randomized Algorithm for Detecting Lines

Line detection is very important in image processing. In this paper, a new randomized algorithm for detecting lines is presented. The proposed algorithm is quite different from the previous parameter–based methods which vote on the parameter space. Our proposed novel algorithm does not need extra storage to maintain an accumulator array for representing parameter space. The main concept used in the proposed algorithm is that we first randomly select three edge points in the image and use a distance criterion to determine whether there is a candidate line in the image; after finding that candidate line, we apply an evidence–collecting process to further determine whether the candidate line is the desired line. Some experiments have been carried out to demonstrate the computational and robust advantages of the proposed algorithm when compared with the previous algorithms.

Teh-Chuan Chen* and Kuo-Liang Chung†

Department of Information Management, Institute of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, R. O. C.

Introduction

Detecting lines is one of the most fundamental problems in the image processing community [1]. The commonly used method for solving this problem is the Hough transform (HT) and its variants [2,3]. Originally, the HT was proposed by Hough [4] whose method is based on the slope and intercept parameters. Later, Duda and Hart [5] introduced the normal form of a line to avoid the infinite range of parameter space and their method is denoted by DHT for convenience. In DHT, a straight line in an image is parameterized in normal form by

$$\rho = x\cos\theta + y\sin\theta, \tag{1}$$

where (x, y) denotes the coordinates of the given edge point on the straight line; ρ and θ denote the normal distance from the origin to the line and the angle of the normal line associated with the positive x-axis, respectively. Essentially, the DHT consists of two phases: (1) the evidence–collecting phase and (2) the searching phase for finding the desired line. In the evidence–collecting phase, the DHT employs the voting technique. From Eqn. (1), for different θ's, each edge point (x, y) is transformed to many points in the (θ, ρ) parameter space which is quantized and represented by an accumulator array. After finishing the evidence–collecting phase for all edge points, the searching phase searches the cells whose scores are larger than the threshold in the accumulator array, then their corresponding parameters are recognized as the parameters of the desired lines in the image. As a result, to implement DHT, it requires large storage to save the accumulator array and some computational effort for voting and searching in the array.

*Also at: *Department of Information Management, Chung-Yu Junior College of Business Administration, 40 I Chi Road, Keelung 201, Taiwan, R. O. C.*
†Corresponding author. E-mail: klchung@cs.ntust.edu.tw, supported by NSC89-2218-E011-017

Previously, based on the random sampling technique, Fischler and Bolles [6, 7] presented the Random Sample Consensus (RANSAC). The RANSAC randomly selects two edge pixels each time and then other edge pixels are used to test whether the line determined by the two edge pixels is a true line in the image. But in some cases, the probability of two randomly picked edge pixels coming from a true line in an image is low and this will degrade the efficiency of the RANSAC.

In order to reduce the large storage and heavy computation needed in the DHT, Xu *et al.* [8, 9] proposed a randomized Hough transform (RHT) which also randomly selects two edges pixels each time. According to Eqn (1), the reason behind the RHT is that two edge points can be mapped into one corresponding point in the (θ, ρ) parameter space which represents the line passing through the two edge points. For this reason, the RHT randomly picks two edge points in the image each time with equal probability. Then, their corresponding mapped point in the parameter space is accumulated by voting in an appropriate type of accumulator such as the usual accumulation array or the dynamic linear list [9]. The above procedure is continued until some cells in the accumulator have satisfactory scores (i.e. larger than a given threshold) and each of them represents a candidate line. For each candidate line, it follows another evidence–collecting step which counts the number of edge points lying in the candidate line. This step is used to determine whether the candidate line is the desired line. Moreover, when a line is detected, the points lying in the line are taken out of the set of edge points in order to speed up the next line detection. Different lines are detected iteratively until a given stopping criterion is reached.

Later, Kälviäinen and Hirvonen [10] proposed a connective randomized Hough transform (CRHT) which improves the RHT by exploiting the connectivity of local edge pixels. The effectiveness and efficiency of the CRHT heavily depend on the connectivity of neighboring edge pixels. So, problems may arise if the testing image is in the presence of distortions and noises. To alleviate these problems, Kyrki and Kälviäinen [11] proposed an extended connective randomized Hough transform (ECRHT). The ECRHT also uses local information to improve RHT but allows gaps in a line segment. Basically, the RHT [8], CRHT, and ECRHT still use the voting technique to collect the evidence of lines in the accumulator. Thus, these three methods still belong to the type of HT-based methods and still need some extra storage for saving the accumulator. Although the CRHT and the ECRHT can be implemented without using the accumulator by setting the threshold in the accumulator to one [10,11], by employing this accommodation a bend curve could be recognized as many straight lines.

The DHT is a non–probabilistic approach, whereas the RANSAC, the RHT, the CRHT, and the ECRHT are the probabilistic approaches. For a comprehensive survey on the two kinds of approach refer to [2,3,12].

In this paper, we present a new Randomized Line Detection algorithm, called the RLD. The proposed RLD first randomly picks three edge points in the image and uses a distance criterion to determine whether there is a candidate line in the image. After finding a candidate line, it follows an evidence-collecting process to further determine whether the candidate line is the desired line. The proposed RLD is not based on the voting technique, so it does not need the accumulator. The proposed method leads to some advantages such as real–time speed, adaptive accuracy, and raise robustness. Some experiments have been carried out to confirm these advantages when compared with the previous methods, such as the DHT, the RANSAC, the RHT, and the ECRHT. Since experimental results [11] have shown that the ECRHT is more effective than the CRHT, we only include the ECRHT for comparison.

The Proposed Algorithm: RLD

This section consists of four subsections. The first subsection describes the basic idea of the RLD. The second subsection presents two criteria which can be used to determine whether the selected three edges points lie on a candidate line or not. The third subsection describes how to check whether the candidate line is a true line, i.e. the desired line, in an image. Finally, the formal algorithm of the proposed RLD is listed in the fourth subsection.

Basic idea

Let V denote the set of all edge points in an image. From the RHT [8], we know that if we randomly pick two edge points from V, the two points are probably taken from a straight line in the image. Two points can exactly determine a straight line. Therefore, when it is shown that many selected edge points are from the same straight line, it is very probable that the straight line a true line in the image. This is why RHT requires an accumulator to collect the evidence of line obtained from any two edge points we picked and needs to do this job iteratively in order to discover a candidate line in the accumulator. We now extend this idea, used in [8], to the case of randomly picking three edge points. Surprisingly, this extension leads to memory—and computation — saving effects.

Generally, three edge points can determine three straight lines. But if the selected three edge points are from the same straight line, the straight line seems very likely to be a true line in the image. For example, assume there is an image with only one straight line. When we randomly pick three edge points from the image and it shows that the three points are almost on the same line, i.e. collinear, we can then further determine a candidate line from the three edge points and check whether the candidate line is a true line or not.

Determine candidate lines

Here we present two methods, each used as a criterion, to determine whether three edge points are collinear. The first method is motivated by the fact that the area of the triangle formed by three collinear points is 0.

Let $v_i = (x_i, y_i)$ denote a pixel with coordinates (x_i, y_i) in an image. Given three pixels v_1, v_2, and v_3, if they are not collinear, they can form a triangle with those three points as its vertices. The area of the triangle can be easily calculated using the following formula:

$$\left|\frac{1}{2}[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)]\right|, \tag{2}$$

where $|z|$ denotes the absolute value of z.

If v_1, v_2, and v_3 are totally collinear, the value of Eqn (2) is 0. But as the image we dealt with is a digital image, it rarely occurs that many edge points lie exactly on a line. So, the purpose of line detection in a digital image is to detect a set of edge points which lie not exactly but roughly on a straight line (see Figure 1). When v_1, v_2, and v_3 are almost collinear and lie on a digital line, then the value of Eqn (2) is very small. So, Eqn (2) can be used to determine whether v_1, v_2, and v_3 are collinear.

There is a normalized problem which arises from using the area of a triangle [see Eqn. (2)] to check whether these three points are collinear. For example, as shown in Figure 1, v_1, v_2, v_3, and v_4 lie on a digital line, but the area of the triangle formed by v_1, v_3, and v_4 is six

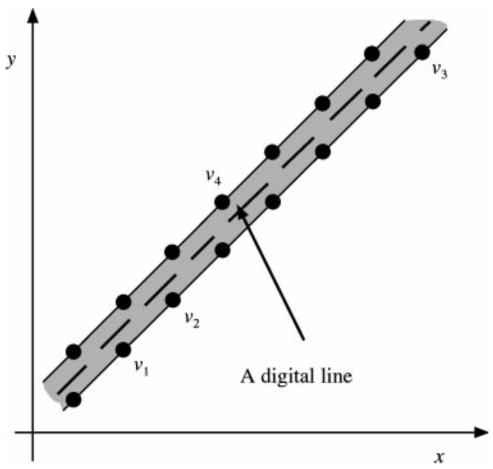

Figure 1. Pixels of a digital line are roughly on a straight line (the dash line).

times larger than the area of the triangle formed by v_1, v_2, and v_4. Therefore, we propose another criterion to avoid this case. Given three randomly picked edge points $v_i = (x_i, y_i)$, $v_j = (x_j, y_j)$, and $v_k = (x_k, y_k)$, if they are collinear, there exists one of the three points where the distance from it to the line passing through the other two points is small. For example, v_1, v_2, and v_3 are collinear in Figure 2. The distance from v_3 to the line passing through v_1 and v_2, denoted by $d_{3\to12}$ is small. Intuitively, we can regard v_3 as it lies on the line passing through v_1 and v_2 if the distance $d_{3\to12}$ is small enough.

Note that three points can determine three lines and each line has its own distance to the third point. As shown in Figure 2, the line passing through v_1 and v_2 satisfies two conditions: (1) It has the shortest distance to v_3 and (2) the distance is smaller than some tolerance value (e.g., 0.5). So, we can regard the line passing through v_1 and v_2 as the candidate line.

Therefore, we can use the distance between a point and a line, which determined by another two points, to check whether these three randomly chosen points are collinear. What we need to know is which line has the corresponding minimum distance among three such distances. To solve this problem, we can regard the three points as the vertices of a triangle. In the triangle, the distance from one vertex to the line passing through the other two vertices is the height corresponding to the side connected by the two vertices. There are three such heights and we need to find out the minimum one. In fact, the longest side has the corresponding minimum

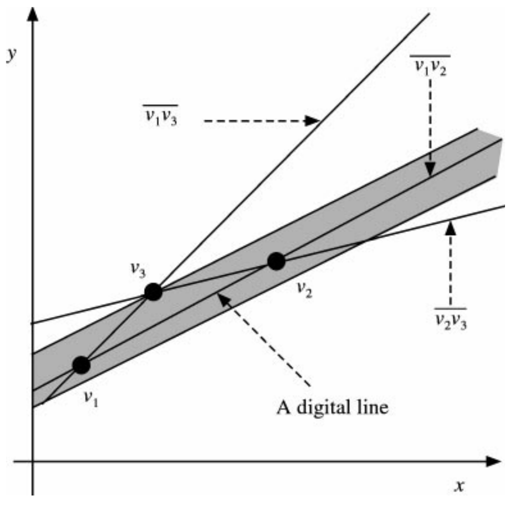

Figure 2. An example of three pixels in a digital line.

height in a triangle. This can be justified from the following theorem.

THEOREM 1. In a triangle formed by three vertices, there are three sides determined by connecting any two vertices and three corresponding heights being the distance from one vertex to the line extending from the opposite side. Then, the minimum height is the one whose corresponding side has the longest length among the three sides.
Proof. The area of a triangle is equal to the half of the length of each side multiplies its corresponding height. So, it is easy to see that the longest side has the shortest height.

Using Euclidean distance measure, the length of $\overline{v_iv_j}$ can be calculated by

$$\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} .$$

From Theorem 1, we first calculate the length of three sides, i.e. $\overline{v_iv_j}$, $\overline{v_jv_k}$, and $\overline{v_kv_i}$. Then, the minimum height is the one whose corresponding side is the longest one among the three sides. For example, when $\overline{v_iv_j}$ is the longest side, then the minimum height is the distance from v_k to the line passing through v_i and v_j.

When we know that the distance from v_k to the line passing through v_i and v_j, say $d_{k\to ij}$, is minimum, we need to calculate the value $d_{k\to ij}$ to check whether $d_{k\to ij}$ is small enough. The line passing through v_i and v_j can be represented as

$$(x_j - x_i)\,(y - y_i) = (y_j - y_i)\,(x - x_i)$$

or

$$(x_j - x_i)y + (y_i - y_j)x + x_iy_j - x_jy_i = 0.$$

Therefore, $d_{k\to ij}$ can be calculated by

$$d_{k\to ij} = \frac{|(x_j - x_i)y_k + (y_i - y_j)x_k + x_iy_j - x_jy_i|}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}.}. \quad (3)$$

If $d_{k\to ij}$ is not larger than a given tolerance T_d, for example $T_d = 0.5$, we conclude that v_i, v_j, and v_k are collinear and the line passing through v_i and v_j is our candidate line. Furthermore, v_i and v_j are called the agent points of the candidate line.

There is, however, an undesirable case that should be taken care of. When v_i, v_j, and v_k are so close, the minimum distance between point and line may be smaller than T_d, but the true line is not determined by any two points (see Figure 3). To avoid this case, we claim that the distance between two agent points of a candidate line must be greater than a given threshold T_{min}. If so, it means that the candidate line is stretched enough.

Determine true lines

After we detect a candidate line with agent points v_i and v_j, whether the candidate line is a true line in an image can be checked by the following evidence–collecting process. Initially, we set a counter $C = 0$ for this candidate line. For each edge point v_k in V, Eqn (3) is used to calculate the value $d_{k\to ij}$. If $d_{k\to ij}$ is not larger than the given threshold T_d, we increment the counter C

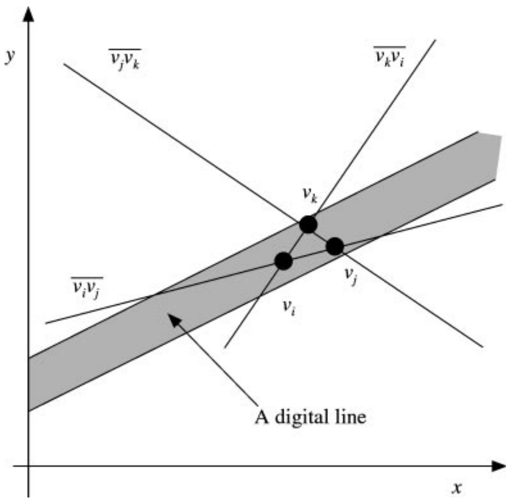

Figure 3. An example of three close pixels in a digital line.

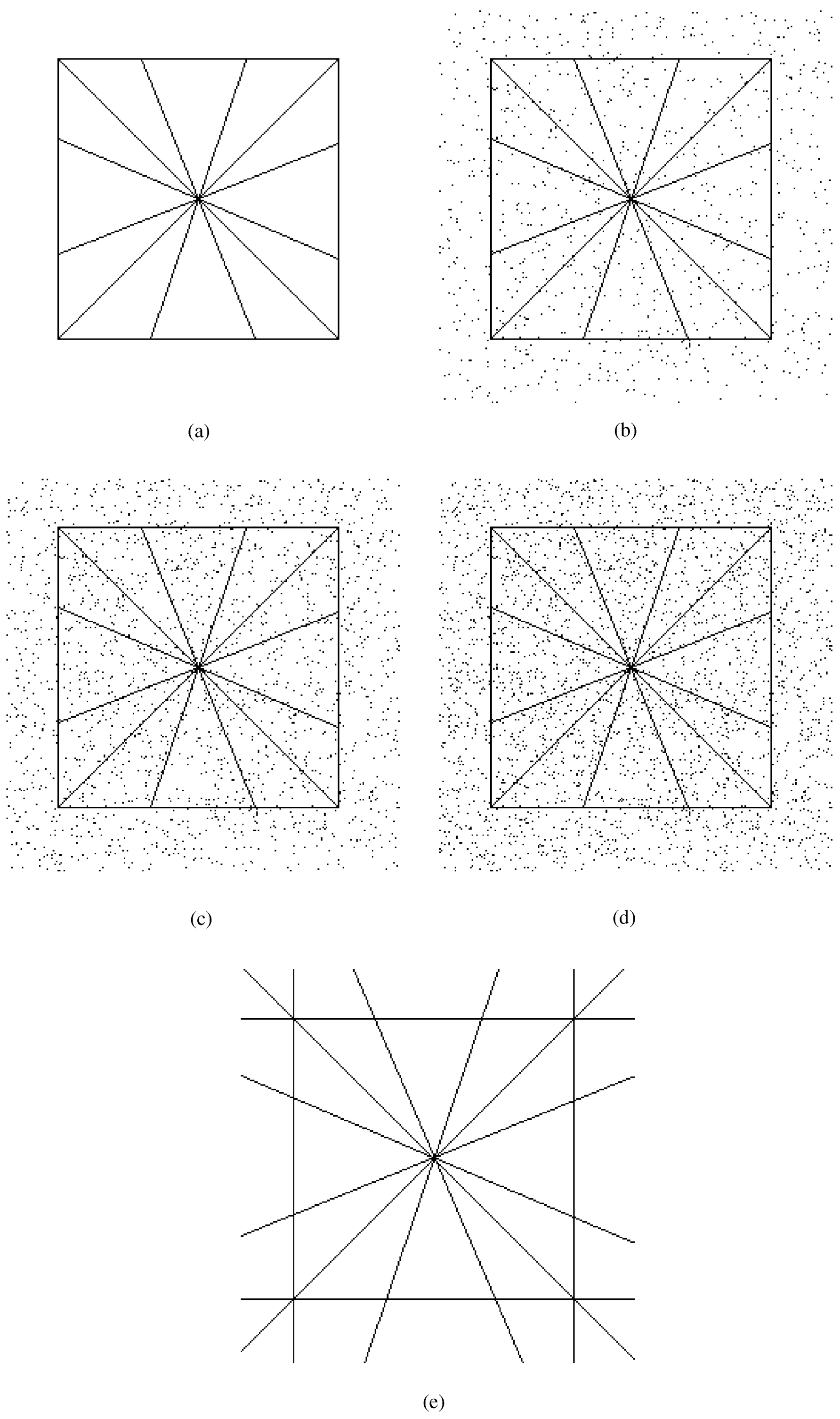

Figure 4. The testing synthetic images. (a) The original synthetic image; (b) the image, Noise1, with 903 randomly adding pixels; (c) the image, Noise2, with 1805 randomly adding pixels; (d) the image, Noise3, with 2708 randomly adding pixels; (e) the resulting image.

by one and take v_k out of V; otherwise we proceed next edge point. We continue the evidence–collecting process until all the edge points in V have been examined. Assume that n_p edge points have been taken out, i.e. $C = n_p$, and then we check whether the counter C is larger than the given threshold T_l or not. Once C is larger than T_l, we conclude that the candidate line is a true line. Otherwise, the candidate line is a false line and we return those n_p edge points into the set V.

The proposed RLD algorithm

From the above description, this subsection presents the formal RLD algorithm consisting of six steps and these steps are stated as follows:

Step 1. Putting the coordinates of all edge pixels $v_i = (x_i, y_i)$ into the set V and initialize the failure counter $f = 0$. Let T_f and T_l be two given thresholds, where T_f denotes the number of failures that we can tolerate; a line in an image should have at least T_l pixels. Moreover, let $|V|$ denote the number of edge points retained in V.

Step 2. If $f = T_f$ or $|V| < T_l$, then stop; otherwise, we randomly pick three points $v_i, i = 1, 2, 3$, out of V in such a way that all points of V have an equal probability to be taken as v_i. When v_i has been taken, set $V = V - \{v_i\}$.

Step 3. Using Euclidean distance measure, we calculate the lengths, $\overline{v_1v_2}$, $\overline{v_1v_3}$, and $\overline{v_2v_3}$. Finding out the longest one among these three lengths. Assume the longest one is $\overline{v_1v_2}$, then we calculating $d_{3\to12}$ by Eqn. (3). If $d_{3\to12} \leq T_d$ and the length of $\overline{v_1v_2}$ is larger than T_{min}, where T_{min} is a given threshold which denotes the length of the longest side being longer than T_{min}, we regard the line passing through v_1 and v_2 as a candidate line and goto Step 4; otherwise, put $v_i, i = 1, 2, 3$, back to V; perform $f = f + 1$; go to Step 2.

Step 4. Set the counter $C = 0$. For each v_k in V, check whether $d_{k\to12}$ is not larger than the given tolerance T_d. If yes, $C = C + 1$ and take v_k out of V.

Step 5. Assume there are n_p edge points, i.e. $C = n_p$, with $d_{k\to12} \leq T_d$. If $C \geq T_l$, go to Step 6; otherwise, regard the candidate line be a false line, return these n_p edge points into $V, f = f + 1$, and go to Step 2.

Step 6. The candidate line has been detected as a true line. Set $f = 0$ and go to Step 2.

Following the point distance criterion proposed by Kälviäinen and Hirvonen [8], Step 2 can be modified as follows:

Step 2. If $f = T_f$ or $|V| < T_l$, then stop; otherwise, we randomly pick three points $v_i, i = 1, 2, 3$, out of V in such a way that all points of V have an equal probability to be taken as v_i and the distance between any two of the three pixels should be not larger than a predefined value T_{max}. When v_i has been taken, set $V = V - \{v_i\}$.

For convenience, the RLD_ND (RLD_D) is used to denote the proposed RLD without (with) the point distance criterion. Note that the point distance criterion can also be applied to the RANSAC which randomly picks two pixels each time. Similarly, the RANSAC_ND (RANSAC_D) is used to denote the RANSAC without (with) the point distance criterion. RHT_ND (RHT_D) is used to denote the RHT without (with) the point distance criterion.

Experimental Results

The performance of the RLD is tested on one synthetic image and two real images. A 256×256 synthetic image with 1805 edge points is shown in Figure 4(a). The synthetic image consists of lines with different slopes. In order to test the robustness of the RLD, 903 (about half of 1805), 1805, and 2708 extra noisy pixels are added to the synthetic image and the resulting images are shown in Figure 4(b), Figure 4(c), and Figure 4(d), respectively. Besides the RLD_ND and the RLD_D, all the related line detection methods, such as the DHT, the RANSAC_ND, the RANSAC_D, the RHT_ND, the RHT_D, and the ECRHT, are implemented in order to compare their performance.

In our implementations, any one of these methods we choose for comparison can correctly detect the lines in

Table 1. Time performance comparison for synthetic images

Method	Original	Noise 1	Noise 2	Noise 3
DHT	3606	5490	7182	8962
RANSAC_ND	57	1575	3639	9608
RANSAC_D	41	432	1039	2761
RHT_ND	341	1105	2113	6048
RHT_D	283	577	1133	2339
ECRHT	425	374	393	697
RLD_ND	17	78	188	361
RLD_D	18	44	94	170

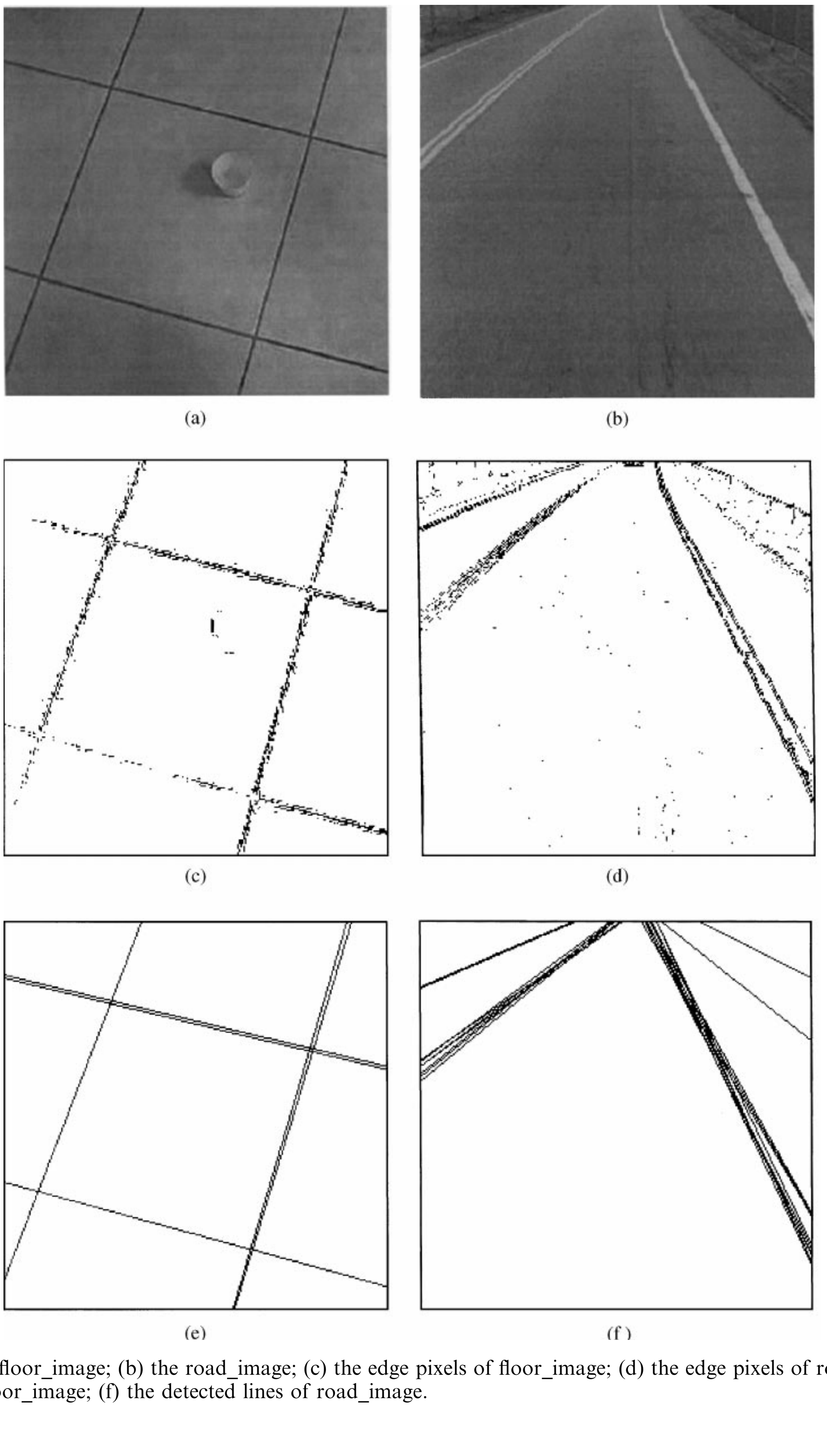

Figure 5. (a) The floor_image; (b) the road_image; (c) the edge pixels of floor_image; (d) the edge pixels of road_image; (e) the detected lines of floor_image; (f) the detected lines of road_image.

the testing images. For example, using our proposed RLD, the resulting image is shown in Figure 4(e), where each line is drawn by passing the two agent points, which represent the line that we have detected.

The execution time of all the implementations are examined by using the Pentium 233 personal computer and the Microsoft Windows 98 environment. The programming language used is the Borland C++ Builder version 4. Table 1 illustrates the experimental results. In Table 1, the first column denotes the name of the line detection method used and the next four columns begin with the names of the synthetic images. The execution time required in each method is measured in terms of milliseconds and the related simulations are tested based on 1000 simulations for each method.

From Table 1, it is observed that the two variants of the proposed RLD, the RLD_ND and the RLD_D, are the fastest ones among the concerning methods. Table 1 reveals that any method with the point distance criterion has better time performance when compared with the one without the point distance criterion.

Further, as shown in Figure 5(a) and Figure 5(b), two real images, floor_image and road_image, are used for time performance comparison. Each image is of size 256×256. The edge pixels of the two images are shown in Figure 5(c) and Figure 5(d), respectively. In order to test the robustness of the RLD, the Laplacian operator [1] is used as the edge detection operator. Similar to the test of synthetic images, all the methods concerned have a similar detecting result. Figure 5(e) and Figure 5(f) show the resulting images of the RLD. The time performance comparison among the concerning methods is shown in Table 2.

Table 2 reveals that the two variants of the proposed RLD, the RLD_ND and the RLD_D, are the fastest amongst the methods concerned. Basically, the point distance criterion proposed in [10] can prune away some false lines leading to the computational effect. This is why the method with the point distance criterion has a better time performance when compared with the one without the point distance criterion.

Table 2. Time performance comparison for two real images

Method	floor_image	road_image
DHT	3641	4110
RANSAC_ND	3023	3632
RANSAC_D	753	1082
RHT_ND	1180	3073
RHT_D	483	1233
ECRHT	1062	1516
RLD_ND	195	207
RLD_D	83	95

Two Remarks About the Proposed RLD

In this section, two remarks are given to illustrate some other advantages of the proposed RLD method.

(1) In the DHT method or other line detection methods implemented with an accumulator array representing the parameter space, due to the fact that the quantization of the parameter space and the exact parameters of a line are often not equal to the quantized parameters, these methods seldom find the exact parameters of a line in the image [13]. But, the proposed RLD method is not based on the quantization of the parameter space. In the RLD method, the detected lines are based on the edge points that we pick. Therefore, the proposed method may detect the line in a more accurate way. Also, due to quantization of the parameter space, a cell in the accumulator array corresponds to a region which is not exactly bar-shaped but is like a bow–tie shape in the image space [14]. However, the proposed RLD method uses two agent points to represent a candidate line, followed by the distance between the point to the candidate line to test which other points belong to the line. Therefore, when we find a candidate line, the region included in the image to support whether it is a true line is exactly bar-shaped.

(2) In the HT–based methods, high resolution in parameter space necessitates large storage and more computation time. Therefore, there is a trade-off between the resolution of the detected line and the burden in storage and computation time. But, in the proposed RLD method, we use two agent points to represent a line and dynamically adjust the threshold T_d to fit any resolution of the line without increasing the storage and time requirement.

Conclusions

In this paper, we have presented a new randomized algorithm for line detection. The proposed RLD method is based on randomly picking three edge points in the image. Then, based on some newly proposed criteria, we work out a candidate line and check whether the candidate line is a true line. Unlike the HT–based methods, the proposed RLD method is not based on the parameter approach. Hence, it indeed does not need any extra storage for saving the accumulator array, which is needed in the HT–based methods. Furthermore, the proposed RLD has some advantages such as high speed, adaptive accuracy, and robust to noises. The experimental results have confirmed these advantages.

Acknowledgment

We would like to thank three anonymous referees and the Associate Editor Professor L. da Fontoura Costa for their valuable suggestions that lead to the improved presentation of this paper. We also appreciate Miss L. F. Chieu for her valuable corrections.

References

1. Gonzalez, R.C. & Woods, R.E. (1992) *Digital Image Processing*, Addison Wesley, New York.
2. Illingworth, J. & Kittler, J. (1988) "Surver: Survey of the Hough Transforms," *Computer Vision, Graphics, and Image Processing*, **44**: 87–116.
3. Leavers, V.F. (1993) "Survey: Which Hough Transform," *CVGIP: Image Understanding*, **58**(2), 250–264.
4. Hough P.V.C. "Method and Means for Recognizing Complex Patterns," U.S. Patent 3,069,654, Dec. 18, 1962.
5. Duda R.O. and Hart P. E. (1972) "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Commun. ACM*, **15**(1): 11–15.
6. Fischler M.A. & Bolles, R.C. (1981) "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Commun. ACM*, **24**(6): 381–395.
7. Fischler, M.A. & Firschein, O. (1978) *Intelligence: The Eye, the Brain, and the Computer*, Addison Wesley, pp. 279–280.
8. Xu, L., Oja, E. & Kultanan, P. (1990) "A New Curve Detection Method: Randomized Hough Transforms (RHT)," *Pattern Recognition Letters*, **11**(5): 331–338.
9. Xu, L. & Oja, E. (1993) "Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities," *CVGIP: Image Understanding*, **57**(2): 131–154.
10. Kälviäinen, H. & Hirvonen, P. (1997) "An Extension to the Randomized Hough Transform Exploiting Connectivity," *Pattern Recognition Letters*, **18**(1): 77–85.
11. Kyrki, V. & Kälviäinen, H. (2000) "Combination of Local and Global Line Extraction," *Real-Time Imaging*, **6**, 79–91.
12. Kälviäinen, H., Hirvonen, P. Xu, L. & Oja, E. (1995) "Probabilistic and Nonprobabilistic Hough Transforms: Overview and Comparison," *Image and Vision Computing*, **13**(4): 239–252.
13. van Veen, T.M. & Groen, F.C.A. (1981) "Discretization Errors in the Hough Transform," *Pattern Recognition*, **14**(1): 137–145.