ELSEVIER

Pattern Recognition Letters 22 (2001) 421–429

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Detecting line segments in an image – a new implementation for Hough Transform

Yu-Tai Ching *

Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu 30010, Taiwan, ROC

Received 27 March 2000; received in revised form 20 September 2000

Abstract

The conventional Hough Transform is a technique for detecting line segments in an image. The conventional Hough Transform transforms image points into lines in the parameter space. If there are collinear image points, the lines transformed from the points intersect at a point in the parameter space. Determining the intersection is generally carried out through the "voting method", which partitions the parameter space into squared meshes. A problem with the voting method involves determining the resolution required for partitioning the parameter space. In this paper, we present a solution to this problem. We propose to transform an image point into a belt, whose width is a function of the width of a line in the image. We then determine the intersection of numerous belts to detect a line segment. An iterated algorithm based the transformation for detecting line segments is presented in this paper.

Keywords: Hough Transform; Computational geometry; Geometric duality

1. Introduction

The *Hough Transform* is a technique in pattern recognition which has been applied to identify shapes of varying kinds in an image (Hough, 1962; Duda and Hart, 1972; Illingworthe and Kittler, 1988). The conventional Hough Transform identifies a line segment in an image. It is generally carried out though the following procedures. An image is preprocessed using thresholding, edge detection, or thinning in order to produce a binary image which could possibly contain line patterns of 1-pixel width. Each non-zero image point (i, j) is then transformed to a line $y - ix - j = 0$ in the parameter space. If there is a set, S, of collinear image points, the set of lines transformed from S have a common intersection. Using this transformation, a line having a slope close to ∞ (a vertical line) is difficult to detect because the set of points on a vertical line are transformed into a set of parallel lines. To overcome this problem, another transformation is applied. A point in the xy-coordinate system is transformed into a sinusoidal curve in (θ, ρ)-parameter space by the mapping $\rho = x\cos(\theta) + y\sin(\theta)$. Still, the set of curves intersect at a point if they are transformed from a set of collinear points.

Regardless the kinds of transformation applied, implementation of the Hough Transform requires the calculation of the intersections of numerous

* Fax: +886-3-572-1490.
E-mail address: ytching@cis.nctu.edu.tw (Y.-T. Ching).

0167-8655/01/$ - see front matter
PII: S0167-8655(00)00130-6

geometric objects (lines or curves). A typical approach for detecting the intersection is a discrete approach called "voting". The parameter space is first partitioned into squared meshes with a "proper resolution". Each square holds a counter which counts the number of lines passing through it. The square having the largest vote is the intersection point. A problem with this approach is determining the "proper resolution". There were many literatures about the Hough Transform in the past (Murphy, 1986; Kalviainen et al., 1995; Lo and Tsai, 1995; Yang et al., 1997; Lam and Yuen, 1996; O'Rourke and Sloan, 1984; Illingworth and Kittler, 1987). Very few approaches attempted to solve this problem (O'Rourke and Sloan, 1984; Illingworth and Kittler, 1987). These approaches were designed based on the observation that "high resolution is required only at the place where there are many intersection points". However, neither of these works discussed how fine a resolution would be necessary.

In this paper, we discuss the resolution issue and present a line segment detection algorithm. Since a line segment in the image actually has width, we propose to transform an image point into a belt. We show that the width of the belt is a function of the width of the line segment in the image. To find out the common intersection of many lines becomes determining the intersection of many belts. An iterative algorithm based on the line segmentation detection algorithm is presented to identify numerous lines in an image.

In the next section, we describe the transformation that was used in this study. The details of the proposed algorithm will be presented in Section 3. Section 4 contains the experimental results. The conclusion is in Section 5.

2. Geometric duality

The Hough Transform is known as the "geometric duality" in the area of computational geometry (Chazelle et al., 1983; Lee and Ching, 1985). One of the dual relationships transforms a point (a, b) into a line $ax + by = 1$ and vise versa in an xy-plane. This dual relationship is a transformation between a point and a line through a unit circle centered at $O = (0, 0)$. Consider a point p and its dual $D(p)$ as shown in Fig. 1. $D(p)$ is a line perpendicular to $\overline{O,p}$ and $d(O, D(p)) = 1/(d(O, p))$ where $d(P_1, P_2)$ is the distance between two geometric objects P_1 and P_2.

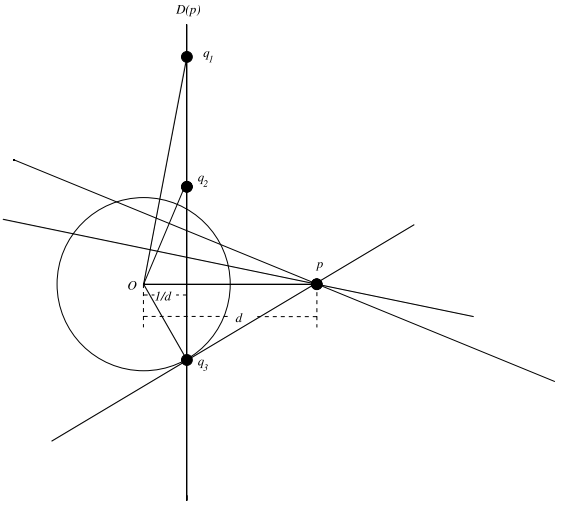

Fig. 1. p is transformed to $D(p)$ through a unit circle centered O. $d(O, p) = d$ and $d(O, D(p)) = 1/d$. q_1, q_2, and q_3 are points on $D(p)$, $D(q_1)$, $D(q_2)$, and $D(q_3)$ pass through p.

There are also degenerate cases in this transformation. The origin of the unit circle, O, does not have a line as its dual. Lines passing through the origin of the unit circle do not have duals either.

In this study, we modified the dual transformation so that a point, p, is transformed into a belt. In order to distinguish the duality of a point in the original transformation (a line) and the modified transformation (a belt), we use $D(p)$ to denote the line and $D'(p)$ to denoted the belt. $D'(p)$ is specified by a pair of parallel lines which are on both sides of $D(p)$.

3. The proposed algorithm

In this section, we show that the width of a transformed belt is a function of the width of the line segment in an image. The equations for the parallel lines representing the belt will be derived from the width of the belt. The algorithm for

detecting line segments in an image will also be presented in this section.

Suppose that an image point p is on the line L if $d(p,L) \leqslant \epsilon$, i.e., the width of a line segment is 2ϵ. Let the width of the belt, $D'(p)$, be 2δ. δ can be written in terms of ϵ as the following theorem.

Theorem. *Let d_1 and d_2 be $d(L,O)$ and $d(p,O)$ respectively*,

$$\delta = \frac{\epsilon}{d_2 d_1}. \tag{1}$$

Proof. We assume that p and O are on the different sides of L as shown in Fig. 2. Let q be the intersection between $\overline{O,p}$ and L. Since $\overline{p,r}$ is parallel to $\overline{O,s}$, we have

$$\frac{\epsilon}{d_1} = \frac{d(p,q)}{d_2 - d(p,q)}. \tag{2}$$

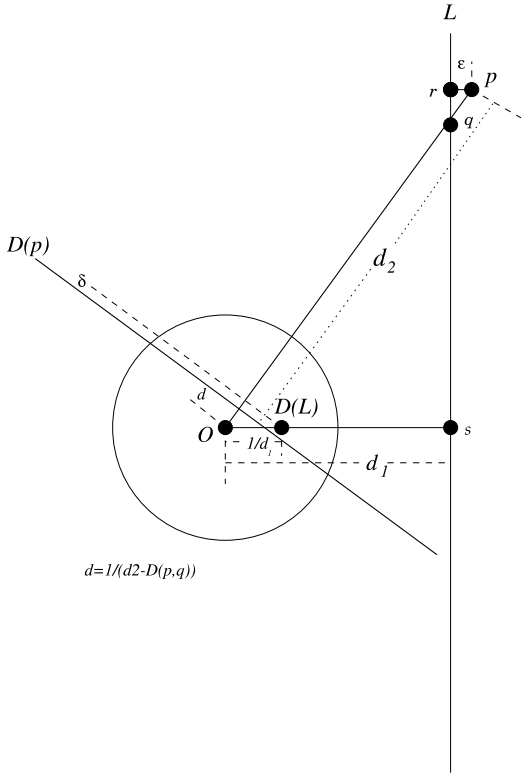

Fig. 2. Since p is not on L and $d(p,L) = \epsilon$, the dual of p does not pass through $D(L)$ but δ away from $D(L)$.

Since p is not on L, $D(p)$ does not pass through $D(L)$. Let $d(D(p), D(L)) = \delta$. We have

$$\frac{1}{d_2} + \delta = \frac{1}{d_2 - d(p,q)}. \tag{3}$$

From Eq. (2), we have

$$\epsilon d_2 - \epsilon d(p,q) = d_1 d(p,q)$$

and thus

$$d(p,q) = \frac{\epsilon d_2}{d_1 + \epsilon}.$$

From Eq. (3),

$$\begin{aligned}\delta &= \frac{1}{d_2 - d(p,q)} - \frac{1}{d_2} \\ &= \frac{1}{d_2 - (\epsilon d_2/(d_1+\epsilon))} - \frac{1}{d_2} \\ &= \frac{d_1+\epsilon}{d_2 d_1} - \frac{d_1}{d_2 d_1} \\ &= \frac{\epsilon}{d_2 d_1}.\end{aligned}$$

If p and O are on the same side of L, Eqs. (2) and (3) become

$$\frac{\epsilon}{d_1} = \frac{d(p,q)}{d_2 + d(p,q)}$$

and

$$\delta = \frac{1}{d_2} - \frac{1}{d_2 + (\epsilon d_2/(d_1 - \epsilon))}.$$

We still have

$$\delta = \frac{\epsilon}{d_2 d_1}. \qquad \square$$

Based on the above theorem, if $D(p)$ is $ax + by = 1$, $D'(p)$ are parallel lines on both sides of $D(p)$ and $\epsilon/(d_2 d_1)$ away from $D(p)$. Let $ax + by = C'$ and $ax + by = C''$ be these two lines. We show that $C' = 1 + \epsilon/d_1$ and $C'' = 1 - \epsilon/d_1$.

Given a point $p = (a,b)$ and its dual $D(p) = ax + by = 1$, $(0, 1/b)$ is a point on $D(p)$ and both $N' = (a,b)$ and $N'' = (-a,-b)$ are normal vectors for $D(p)$. Since $d_2 = \sqrt{a^2+b^2}$, the normalized N' and N'' are

$$N' = \left(\frac{a}{\sqrt{a^2+b^2}}, \frac{b}{\sqrt{a^2+b^2}}\right) = \left(\frac{a}{d_2}, \frac{b}{d_2}\right)$$

and

$$N'' = \left(-\frac{a}{\sqrt{a^2+b^2}}, -\frac{b}{\sqrt{a^2+b^2}} \right) = \left(-\frac{a}{d_2}, -\frac{b}{d_2} \right).$$

$D'(p)$ are parallel lines which must respectively pass through two points $N'\delta + (0, 1/b)$ and $N''\delta + (0, 1/b)$. By substituting the coordinates of these points to the line equation, we can calculate C' and C'' as in the following equations

$$a\left(\frac{\epsilon a}{d_1 d_2 d_2} \right) + b\left(\frac{\epsilon b}{d_1 d_2 d_2} + \frac{1}{b} \right) = C'$$

and

$$a\left(-\frac{\epsilon a}{d_1 d_2 d_2} \right) + b\left(-\frac{\epsilon b}{d_1 d_2 d_2} + \frac{1}{b} \right) = C''.$$

Since $d_2^2 = a^2 + b^2$, C' and C'' are respectively,

$$C' = 1 + \frac{\epsilon}{d_1} \tag{4}$$

and

$$C'' = 1 - \frac{\epsilon}{d_1}. \tag{5}$$

From Eqs. (4) and (5), C' and C'' depend on d_1 and ϵ. ϵ can be considered as a known factor. But d_1 is not known since we do not know where the line is. We now present a method to produce an estimation for d_1 so that we can bound δ with reasonable accuracy.

We restrict our discussion to detecting a line segment with a negative slope. Suppose that the dimensions of an image are W by H. If we centered the unit circle at $(0, 0)$, then d_1 ranges from 0 to $\sqrt{W^2 + H^2}$. That means $1/d_1$ ranged over $1/\sqrt{W^2 + H^2}$ to ∞. In this case, an estimation for d_1 could cause significant error. Let $T = \max(H, W)$. If we translate all of the image points (i, j) to $(i + cT, j + cT)$, $c > 1$, then we have

$$cT \leqslant d_1 \leqslant \sqrt{(cT + W)^2 + (cT + H)^2} \leqslant \sqrt{2}(c + 1)T.$$

Any estimation for d_1 in this range will cause the width of the belt to range roughly over $1/\sqrt{2}$ to $\sqrt{2}$ times the actual width for a sufficiently large c. When c is 9, the double precision variable in any programming language can provide sufficient numerical precision for calculating $1/(\sqrt{2}(c + 1)T)$. In order to detect a line segment having a positive slope, we center the unit circle at $(0, H)$ and translate an image point (i, j) into be $(i + cT, -j - cT)$.

We now present the line segment detection algorithm. Let S be a set of image points on a line segment L. There are two iterations. In the first iteration, we assume that the slope of L is negative. In this case we center the unit circle at $(0, 0)$ and translate all of the image points (i, j) to $(i + cT, j + cT)$. We randomly choose a set of m points, $\{p_i \mid i = 1, \ldots, m\}$, from S. We then look for a point q in the set which is the closest to the center of L. q is determined by finding the $D(q)$ which contains an interval intersected by the maximum number of the belts transformed from $S - q$. The dual of the center point of the interval on $D(q)$ is taken to approximate L.

We now describe the algorithm for finding the interval which is the intersection of the largest number of the belts. Each intersection of $D(q)$ and a belt is defined by a pair of boundary points. We order all of the boundary points along $D(q)$ from left to right and march along the line from left to right. When we enter an interval by passing through a left boundary point, we enter an interval with one additional belt passing through it. On the other hand, when we leave an interval, by passing through a right boundary point, we enter an interval with one less belt passing through it. If each interval has a counter associated with it, the interval that has the largest count is the interval that has the largest number of belts passing through it.

In the second iteration, we assume that L has positive slope. We center the unit circle at $(0, H)$ and translate each image point into $(i + cT, -j - cT)$. The following procedures are all the same as the above. If the largest interval count for the two iterations is greater than a given a threshold, then a line is detected. Otherwise we consider the detected line to be passing through some noise.

There are two implementation details, the first is to determine m, and the second is to set a threshold for the number of points on a detected line.

• m depends on the width of a line ϵ in the image. Suppose there is no noises in the image. A selection of a point p has probability $1/(2\epsilon)$ that p is at the center of a line. The probability that a consecutive points of m selections are all not at the center is $(1-(1/(2\epsilon)))^m$. Suppose $\epsilon = 1.5$, the probability that all of the selected points are not close to the center of a line is 0.08779 when $m = 7$. We have a very good chance that one of the seven selections is close to the center of a line. Note that, the probability is independent of the number of lines in the images. If there are noises in the images, the probability that a point at the center is less than $1/(2\epsilon)$. We can increase m to improve the accuracy. In the experiment, $m = 9$ for $\epsilon = 1.5$ can have very stable performance.

• We need to set a threshold that a detected line must contain as many points as the preset threshold. This is required because noise might cause the detection of false lines. If we know that there are many points on a line, we prefer to set the threshold higher so that the noise has less effect.

If there is more than one line in the image, we apply the above algorithm iteratively. In each iteration, we find a line and remove all of the image points that have a distance to the line less than ϵ. If the number of image points removed is greater than the threshold, a line is detected. The iterative algorithm stops when the number of points in the images is less than the given threshold.

4. Experimental results

This experiment was performed on many data sets. All of the images were preprocessed using a Sobel filter. The resulting images after Sobel filtering were normalized so that the gray scales ranged from 0 to 255. We then took a threshold, `TH`, of the gray scale for boundary point detection. To reduce the noise, `TH` was generally set to a high value, for example a range of 60–100, to detect boundary points in an actual image.

There are several parameters in the proposed algorithm. The parameters are

1. ϵ, the width of a line in the image,
2. d_1, the distance between a line to the origin,
3. m, the number of trials for detecting a line, and
4. N, the minimum number of points in a detected line.

The width of a line depends upon the input image. The proposed algorithm is not sensitive to the

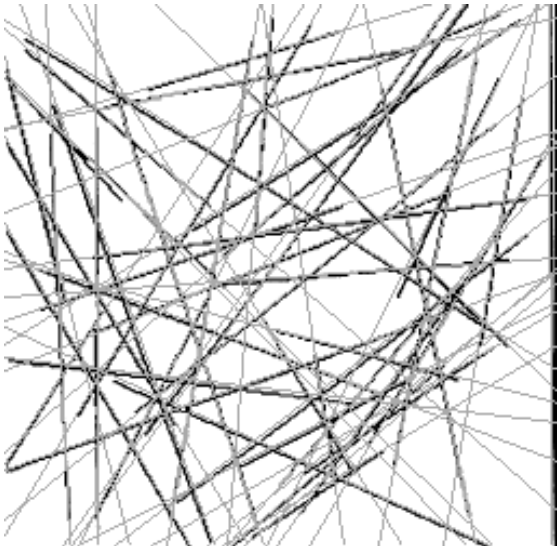

Fig. 3. An image consists of line segments and the detected lines.

Fig. 4. Image of a house and the detected lines.

width selected. Generally speaking, we set ϵ to 1.0–1.5. If there are bold lines in the images, the bold lines will be detected several times. d_1 ranged from $9T$ to $10\sqrt{2}T$. The proposed algorithm does not sensitive to d_1 within this range. We set d_1 to be $9.5\sqrt{2}T$. The number of trials, m, to detect a line could affect the accuracy of the results. $m = 9$ worked well for most of the cases. The parameters were respectively, $m = 9$, $\mathtt{TH} = 80$, and $N = 70$ in processing the real images shown in Figs. 3–5.

Figs. 3–6 show the results for different images containing line segments. Fig. 3 is a test data set in (Kalviainen et al., 1995). The images were obtained from a hard copy scan. Fig. 4 is an image of

Fig. 5. Image of a building with many windows.

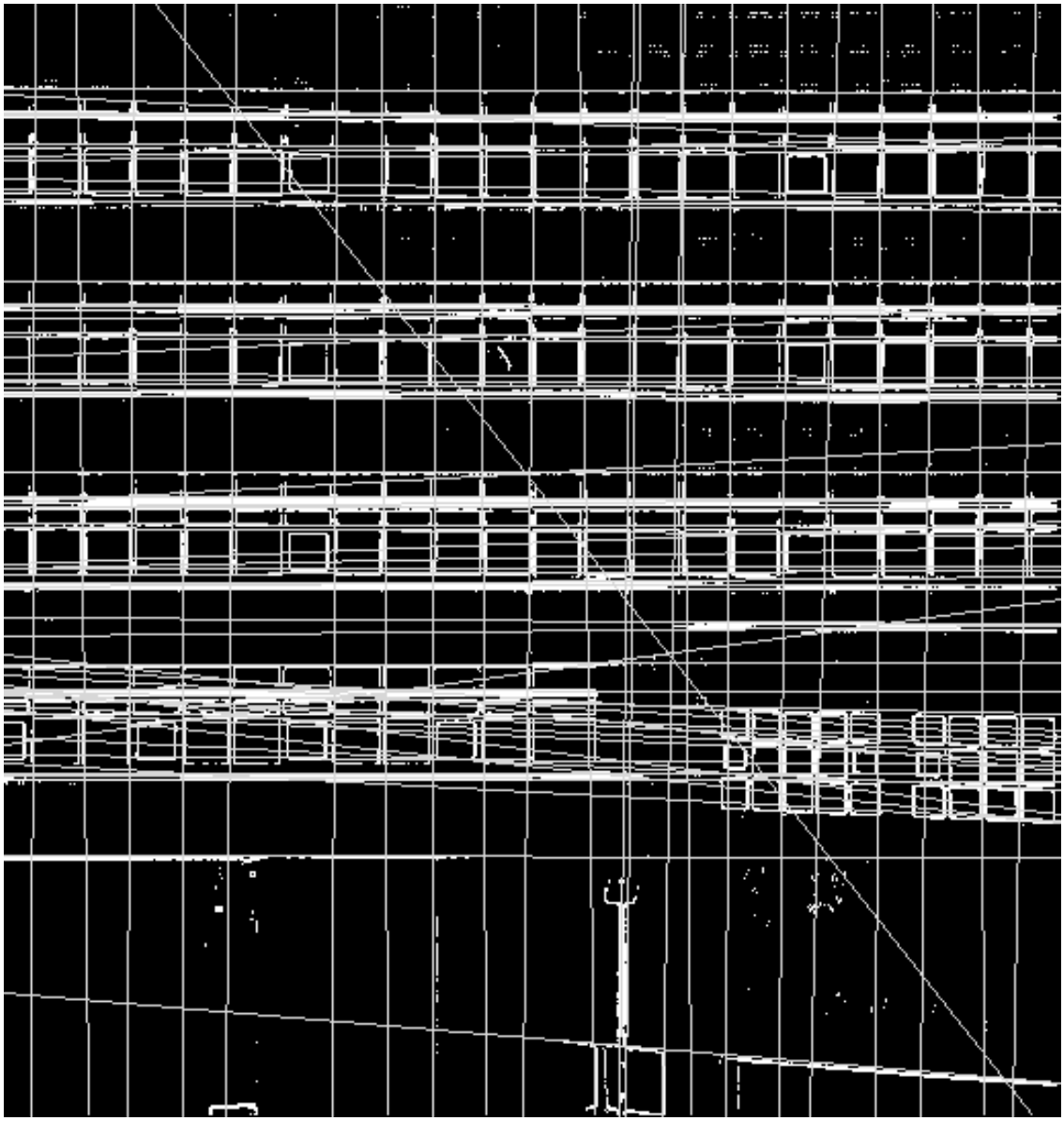

Fig. 6. The result after Sobel filtering and the detected lines.

a house. Fig. 5 is a picture of a building which contains many lines. The result after the Sobel filter and the lines were detected is shown in Fig. 6. The parameters and the computing time for these images are summarized in Table 1.

We also tested the robustness of the proposed approach by adding discrete noises to an image. Fig. 7(a) is a synthetic image containing 22 lines. Fig. 7(b) and (c) are the same images containing noises. We randomly generated 1600 and 3200 points uniformly distributed in these images. We first set $m = 7$ and $N = 40$. In Fig. 7(a) and (b), 22 lines in both images were accurately detected. When the number of noise points reached 3200,

Table 1
The parameters and the computing time for line segments detection in Figs. 5–7

Image	TH	Number of points	ϵ	N	m	Lines detected	Computing time (s)
Lines 251 × 256	100	10200	1.5	10	3	52	14.18
House 256 × 256	80	3182	1.5	70	9	49	13.72
Library 516 × 543	60	28718	1.3	95	5	83	110

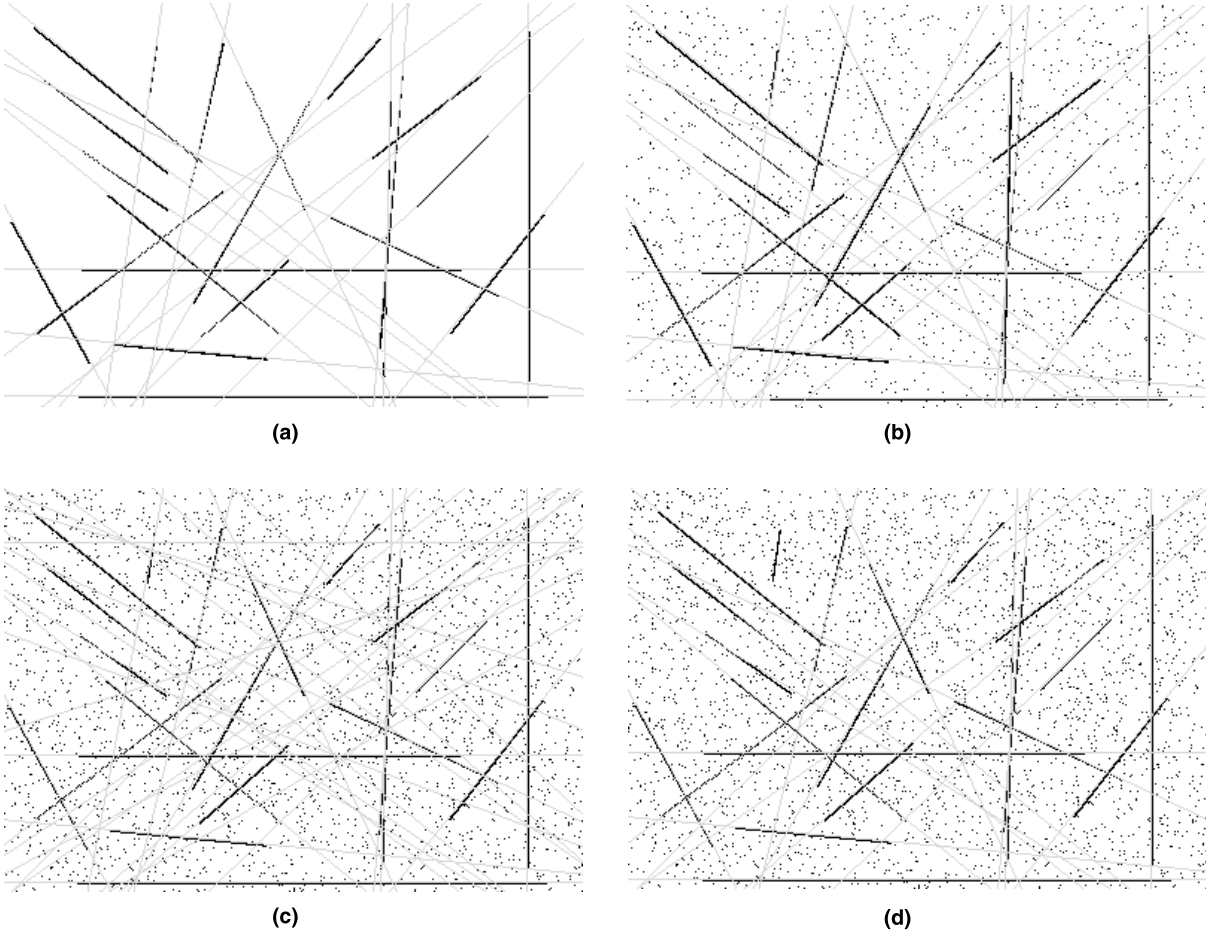

Fig. 7. This figure demonstrates the robustness of the proposed algorithm by adding discrete noises in the image. (a) The original synthetic image containing 22 lines. There are 3539 image points. When $m = 7$ and $N = 40$, 22 lines were detected. (b) The same image containing noises. 1600 points are uniformly distributed over the image. 22 lines were detected ($m = 7$ and $N = 40$). (c) The same image containing 3200 uniform distributed noise points. 36 lines were reported ($m = 7$ and $N = 40$). If $N = 70$, the result is shown in (d). 21 lines were detected ($m = 7$). The shortest line segment was considered to be noises.

the result were affected by the noises as shown in Fig. 7(c). This problem was eliminated when $N = 70$ (Fig. 7(d)). There are 21 lines detected except the shortest lines.

5. Conclusion and discussion

In this paper, we presented a new algorithm to implement the conventional Hough Transform.

The proposed approach does not require a discrete approach to find the intersection of numerous lines in a parameter space. There are several parameters in the proposed algorithm. However, the proposed algorithm is not sensitive to these parameters if they are reasonably assigned. The following assignments to the parameters worked properly in most of the cases,

$d_1 = 9.5\sqrt{T}$,

$\epsilon = 1.5$,

$m = 9$,

$N = 70$.

The proposed algorithm encountered a problem in our experiment. Since the proposed algorithm tends to find the line passing through the largest number of points first, if there are many parallel short line segments on a row, the algorithm finds lines parallel to the row rather than identifying these short line segments. This problem will be addressed in a future work.

References

Chazelle, B.M., Guibas, L.J., Lee, D.T., 1983. The power of geometric duality. Proc. IEEE 24th Symp. Foundations of Computer Science, 217–225.

Duda, R.O., Hart, P.E., 1972. Use the Hough Transform to detect lines and curves in pictures. Commun. ACM 15 (1), 11–15.

Hough, P.V.C., 1962. Method and means for recognizing complex pattern. U.S. Pattern 06954.

Illingworth, J., Kittler, J., 1987. The adaptive Hough Transform. IEEE Trans. Pattern Anal. Machine Intell. 9 (5), 690–698.

Illingworthe, J., Kittler, J., 1988. A survey of the Hough Transform. Comput. Vision, Graphics, Image Process. 44 (1), 87–116.

Kalviainen, H., Hirvonen, P., Oja, E., 1995. Probabilistic and non-probabilistic Hough Transform: overview and comparisons. Image and Vision Comput. 13 (4), 239–252.

Lam, W.C.Y., Yuen, S.Y., 1996. Efficient technique for circle detection using hypothesis filtering and Hough Transform. IEE Proc. Vision, Images and Signal Process. 143 (5), 292–300.

Lee, D.T., Ching, Y.T., 1985. The power of geometric duality revisited. Information Process. Lett. 21 (3), 117–121.

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

Murphy, L.M., 1986. Linear feature detection and enhancement in noisy images via Random transform. Pattern Recognition Lett. 4 (4), 279–284.

O'Rourke, L., Sloan, K.R., 1984. Dynamic quantization: two adaptive data structures for multidimensional space. IEEE Trans. Pattern Anal. Machine Intell. 6 (3), 266–279.

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