Pattern Anal Applic (2003) 6: 55–64
DOI: 10.1007/s10044-002-0178-2

ORIGINAL ARTICLE

**Q. Ji, Y. Xie**

# Randomised hough transform with error propagation for line and circle detection

Received: 18/10/01 / Accepted: 21/06/02

**Abstract** In this paper, we introduce a new Randomised Hough Transform aimed at improving curve detection accuracy and robustness, as well as computational efficiency. Robustness and accuracy improvement is achieved by analytically propagating the errors with image pixels to the estimated curve parameters. The errors with the curve parameters are then used to determine the contribution of pixels to the accumulator array. The computational efficiency is achieved by mapping a set of points near certain selected seed points to the parameter space at a time. Statistically determined, the seed points are points that are most likely located on the curves and that produce the most accurate curve estimation. Further computational advantage is achieved by performing progressive detection. Examples of detection of lines using the proposed technique are given in the paper. The concept can be extended to non-linear curves such as circles and ellipses.

**Keywords** Curve detection · Error propagation · Hough transform · Line fitting · Voting kernel

## Introduction

Detecting geometric primitives in images is one of the basic tasks of computer vision. The Hough Transform (HT) [1–3] and its extensions constitute a popular and robust method for extracting analytic curves. The principal concept of the HT is to define a mapping between an image space and a parameter space. Each feature point (or a set of feature points) in an image is mapped to the parameter space to vote for the parameters whose associated curves pass through the data point(s). The votes for each curve are accumulated, and after all the points in an image have been considered, local maxima in the accumulator correspond to the parameters of the detected curves. The curves detection in the image space therefore become a peak detection problem in the parameter space. There are two main limitations with the standard HT and its variants. First, the straightforward implementation of the HT is often computationally expensive and memory consuming. This is especially true for curves with several parameters, such as an ellipse. Secondly, the standard HT adopts a top hat strategy to compute the contribution of each point to a hypothesised line. Specifically, the scheme assumes all feature points located within a close range of the hypothesized curve contribute equally to the line. The accumulator is therefore incremented by a unit for all feature points. This scheme is inadequate, in that data points are not all equally reliable. By that, we mean the curve parameters derived from each feature point or a set of feature points may carry different uncertainties due to errors with these feature points. Their contribution to the accumulator array should vary accordingly.

In view of the two limitations with the HT, in this paper, we introduce a new probabilistic HT that is aimed at improving the accuracy and robustness of the HT by explicitly accounting for the errors with the image pixels and the errors with the estimated curves parameters. The proposed method also further improves the computational efficiency of the conventional probabilistic HT by statistically and geometrically determining a subset of seed points most likely located on the curves. These seed points are used with other randomly selected points in its neighbourhood to determine curve parameters.

This paper is arranged as follows. Section 2 discusses recent work on the HT that focuses on addressing the two problems, i.e. accuracy and efficiency. Section 3 discusses our error propagation techniques, which we use for estimating errors with the estimated curve parameters. Section 4 describes the proposed approach for curve detection with effective error propagation. Experimental results that have been performed to test this new approach for line detection are covered in Section 5. Finally, in Section 6

Q. Ji (✉)
Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA.
E-mail: qji@ecse.rpi.edu

Y. Xie
Department of Computer Science, University of Nevada, Reno, NV, USA

a discussion and summary of the proposed approach is given.

## Literature research

Recent research in curve detection using the HT focuses on improving the HT's computational efficiency and its estimation accuracy. The methods for improving computational efficiency can be categorised as non-probabilistic and probabilistic approaches. For the non-probabilistic HT, every image pixel is used to calculate the parameters of the curve. To reduce memory storage and computation time, most approaches focus on using different techniques, such as window technique or postprocessing of the accumulator space, to reduce accumulator size. Gerig and Klein [4,5] considered the problem of rapidly finding correspondences between image points and parameter curve points. They suggested a backmapping which links the image space and the accumulator space. After the accumulation, each pixel in an image can be linked to the most evident location in the accumulator space to obtain the parameters of the curve it is part of. This postprocessing of the accumulator space makes it possible to track different boundaries in the original picture effectively. Since every image point is mapped to one parameter point, memory storage and computation time reduces significantly. Princen et al. [6] introduced the Hierarchical Hough Transform (HHT). In the HHT an image is divided into small subimages (for instance, $16 \times 16$) and the HT is performed on these subimages. The accumulation is carried out curve by curve, and when found each curve is removed from the subimage. Because of the small subimages, the size of the accumulator array needed can be kept small, which will lead to an efficient and robust process. Ben-Tzvi and Sandler [7] presented the Combinatorial Hough Transform. The algorithm uses two pixels of the image to calculate the line parameters. Each pair of two image pixels determine one $(\theta,\rho)$ cell in the $(\theta,\rho)$ accumulator array. To limit the number of pixel pair combinations, the image is segmented (typically in 64 regions), and the voting process is performed segment by segment.

The probabilistic HT uses random sampling for selecting only a small subset of the data. Since this method only uses a small part of image data, it significantly reduces the computation time and memory storage. In the Randomized Hough Transform, Xu et al. [8] proposed a method for extracting curves from binarised edge images. For a curve expressed by an $n$ parameter equation, they selected $n$ pixels at random and mapped them into one point of the parameter space, instead of transforming one pixel into an $n - 1$ dimensional hypersurface in the parameter space as the standard HT and some of its variants do. The Probabilistic Hough Transform (PHT) by Kiryati et al. [9] only uses a small, randomly selected subset of the edge points in the image. This limited poll is used as input for the HT. Since the size of the subset providing a good performance is usually small, the execution time can be reduced considerably. The Dynamic Combinatorial Hough Transform (DCHT) presented by Ben et al. [10] used the $(\theta,\rho)$ line parameterisation. All two-point combinations with the seed point, selected randomly from among feature points, are accumulated into a single value in a $\theta$-histogram. This process is repeated until a redefined threshold is reached with some seed point in the $1 \neq D$ histogram of $\theta$ values. In the next stage, when the threshold is obtained, a detected line is removed, and this sampling procedure is continued until all points have been removed, i.e. lines are detected one by one. If the threshold is not reached, only the seed point is removed from the feature points, and the sampling procedure is continued. Because the detected line is removed when a predefined threshold is reached, only part of image points are considered as seed points. This greatly reduces computation time.

Leavers [11] generalised the DCHT to the Dynamic Generalised Hough Transform (DGHT). In the DGHT, a single connective point is selected and a segment relative to the connective point is determined. A suitable number of points in the segment to calculate the curve parameters are chosen at random, and the solved curve parameters are accumulated. This random sampling is continued until a stopping criterion is satisfied. Finally, once a curve (e.g. a circle or ellipse) has been successfully detected, it is removed from the image, and curve detection is continued. Yuen et al. [12] reported the Connective Hough Transform (ConHT). The algorithm is similar to the Dynamic Combinatorial Hough Transform, for instance, in having a seed point, in this case selected at random, to calculate a one-dimensional $\theta$ -valued accumulator. The accumulation, however, is different. In ConHT the accumulation is performed row by row, to have connectivity between the accumulated points. Since vertical and horizontal lines are considered separately there are, in fact, two accumulators.

Compared to the Standard Hough Transform (SHT), approaches discussed above improved the computational performance of HT to various degrees. Unfortunately, these approaches did not take into account the localisation and discretisation errors, which are present in the image edge pixels and affect the accuracy and robustness of curve detection. Researchers have proposed different schemes to improve the detection performance of the HT in relation to localisation error in the image and discretisation error in both the image and the parameter space.

Stephens [13] formulated a variant of the Hough transform in terms of maximum likelihood estimation. The PHT is defined as the log of the Probability Density Function (PDF) of the output parameters. A PDF for the features is used, which has a uniform component modelling the correspondence errors and a component that falls off as a Gaussian with the distance from the curve to model the measurement errors. For high dimensional Hough space, the proposed method does have some definite advantages comparing to the conventional HT, because the PHT is independent of the size, shape and arrangement of accumulator cells. While this method yields correct propagation of localisation error in terms of a Gaussian error distribution, usually it is computationally expensive.

Kittler and Palmer [14] described a statistical hypothesis testing approach for HT. The proposed technique replaces the commonly used top-hat kernel with a smooth kernel. For each feature point, its contribution to the accumulator array at a hypothesised cell $(\theta,\rho)$ is computed from the smooth kernel function, which is a function of the differences between the estimated model parameters and the hypothesised model parameters. This technique allows localisation error to be propagated into the parameter space.

Breuel [15] described a line detection technique related to the HT that searches hierarchical subdivisions of the parameter space using a bounded error model, and thus avoids some of the problems of the accumulator method. In this technique, the parameter space is divided into cells that are tested to determine whether they can contain a line that passes within the bounded localisation error of a specified number of pixels. If the cell cannot be ruled out, the cell is divided and the procedure is repeated recursively. This continues until the cells become sufficiently small, at which point they are considered to be lines satisfying the output criterion.

Olson [16,18] modified a formal definition of the HT that allows the localisation error to be analysed appropriately. Under this definition, it was shown that the mapping of pixel sets (rather than individual pixels) into the parameter space did not, by itself, improve the accuracy or efficiency of curve detection. He then considered a new method where the HT is decomposed into several constrained subproblems, each of which examines a subset of the parameter space by considering only those pixel sets that include some distinguished set of pixels. The examination of these subproblems allows, first, to propagate the localisation error efficiently and accurately into the parameter space, and secondly, to use randomisation techniques to reduce the complexity of curve detection, while maintaining a low probability of failure. However, the error propagation applied in his work is (a) heuristic, (b) not systematic, and (c) the voting kernel is a top-hat function instead of continuous function.

Ji and Haralick [17] introduced a Bayesian updating scheme that systematically ties the uncertainties computed for each point to its contribution. The contribution of each point to a $(\theta,\rho)$ is proportional to its likelihood. The proposed scheme is based on an analytical propagation of input error. Their results showed that the uncertainty of a feature point depends upon (a) the input perturbation, (b) its relative spatial location to the Hough coordinate system, (c) the edge detector scheme, and (d) the line representation scheme. Their work, however, is computationally intense, since it needs to consider each feature point.

## Error propagation for curve-fitting

Error propagation for a computer vision algorithm is concerned with quantitatively characterising the output perturbation as a function of input perturbation and the algorithm. Error propagation can be performed analytically or numerically or even geometrically [18]. We introduce an analytic approach here.

For a computer vision algorithm, the relationships between its input (either the ideal input vector $X$ or the observed vector $\hat{X}$) and the output quantity it computes (either the ideal parameter vector $\Theta$ or the observed parameter vector $\hat{\Theta}$) can be grouped into three categories: explicit relationship, implicit relationship, and neither explicitly nor implicitly related through an analytic form. In many cases, the relationship is implicit. $\hat{\Theta}$ and $\hat{X}$ are not related through an explicit function but through a nonlinear optimisation function $F$, i.e. $\hat{\Theta}$ is determined by minimising $F(\hat{X}, \hat{\Theta})$.

For most vision algorithms, image pixels that participate in the computation are usually edge points or corner points. Most feature points are detected via a curve-fitting process (e.g. line fitting for corners and ellipse fitting for ellipse points). Least-squares curve fitting refers to determining the free parameters $\Theta$ of an analytical curve $F(x, y, \Theta) = 0$, such that the curve is the best fit to a set of points $(\hat{x}_n, \hat{y}_n)$, where $n = 1, \ldots, N$, in the least-squares sense. A best fit is defined as a fit that minimises the sum of squares of the geometric distances as defined by

$$\epsilon^2 = \sum_{n=1}^{N} \frac{F^2(\hat{x}_n, \hat{y}_n, \Theta)}{\left(\frac{\partial F(\hat{x}_n, \hat{y}_n, \Theta)}{\partial x_n}\right)^2 + \left(\frac{\partial F(\hat{x}_n, \hat{y}_n, \Theta)}{\partial y_n}\right)^2} \tag{1}$$

Error propagation for curve fitting here relates the perturbations of points $(\hat{x}_n, \hat{y}_n)$ to the perturbation of $\hat{\Theta}$, resulted from a least-squares fitting procedure. Let $\Sigma_{\Delta X}$ and $\Sigma_{\Delta\Theta}$ be the covariance matrices of the observed points and the estimated curve parameters. Based on Haralick's covariance propagation theory [19], we have

$$\Sigma_{\Delta\Theta} = \left[\left(\frac{\partial g}{\partial \Theta}\right)^t\right]^{-1} \left(\frac{\partial g}{\partial X}\right)^t \Sigma_{\Delta X} \left(\frac{\partial g}{\partial X}\right) \left(\frac{\partial g}{\partial \Theta}\right)^{-1} \tag{2}$$

where $g(X, \Theta)$ is defined as

$$g = \frac{\partial \epsilon^2}{\partial \Theta} \tag{3}$$

From Eq. (3) and using $F(x_n, y_n, \Theta) = 0$, we have

$$\frac{\partial g}{\partial \Theta} = 2 \sum_{n=1}^{N} \frac{\left(\frac{\partial F_n}{\partial \Theta}\right)\left(\frac{\partial F_n}{\partial \Theta}\right)^t}{\left(\frac{\partial F_n}{\partial x_n}\right)^2 + \left(\frac{\partial F_n}{\partial y_n}\right)^2}$$

Plugging $\frac{\partial g}{\partial \Theta}$ into Eq. (2) yields

$$\Sigma_{\Delta\Theta} = 2\sigma^2 \left[\left(\frac{\partial g}{\partial \Theta}\right)^t\right]^{-1} \tag{4}$$

where $\sigma^2$ represents the image error. Detailed derivations may be found elsewhere [20,21].

## Error propagation for least-squares line fitting

Given a line expressed as

$$F(x_n, y_n, \Theta) = x_n \cos\theta + y_n \sin\theta - \rho \tag{5}$$

a least-squares line fitting amounts to finding the line parameter $\Theta = (\theta, \rho)$ that best fits a set of points $(\hat{x}_n, \hat{y}_n)$, $n = 1, 2, \ldots,$ N. Error propagation is concerned with estimating the perturbation of $\hat{\Theta}$, a least-square estimate of $\Theta$ given the perturbation with $\hat{X}$. From Eq. (1), $\hat{\Theta}$ is obtained by minimising

$$\epsilon^2 = \sum_{n=1}^{N} (\hat{x}_n \cos\hat{\theta} + \hat{y}_n \sin\hat{\theta} - \hat{\rho})^2 \tag{6}$$

Hence, $\frac{\partial g}{\partial \Theta}$ can be computed from Eqs (4) and (5) as follows:

$$\frac{\partial g}{\partial \Theta} = 2 \sum_{n=1}^{N} \begin{pmatrix} k_n^2 & -k_n \\ -k_n & 1 \end{pmatrix}$$

where $k_n = x_n \sin\theta - y_n \cos\theta$. Let

$$u_k = \frac{\sum_{n=1}^{N} k_n}{N} \tag{7}$$

and

$$S_k^2 = \sum_{n=1}^{N} (k_n - u_k)^2 \tag{8}$$

We have [1]

$$\Sigma_{\Delta\Theta} = \sigma^2 \frac{\begin{pmatrix} \mathrm{N} & \Sigma \mathrm{k}_n \\ \Sigma \mathrm{k}_n & \Sigma \mathrm{k}_n^2 \end{pmatrix}}{\mathrm{N}\Sigma \mathrm{k}_n^2 - \Sigma \mathrm{k}_n \Sigma \mathrm{k}_n}$$

$$= \sigma^2 \begin{pmatrix} \frac{1}{\mathrm{S}_k^2} & \frac{\mu_k}{\mathrm{S}_k^2} \\ \frac{\mu_k}{\mathrm{S}_k^2} & \frac{1}{\mathrm{N}} + \frac{\mu_k^2}{\mathrm{S}_k^2} \end{pmatrix} \tag{9}$$

Equation (9) offers insight into the factors that affect the quality of line fitting. Specifically, $k_n$ can be interpreted as the signed distance between a point $(x_n, y_n)$ and the point on the line closest to the origin. Hence, $S_k^2$ represents the spread of points along the line. From Eq. (9), it is clear that with a large $S_k^2$ (i.e. points with large spread along the line) we can obtain better fit as indicated with smaller trace of the covariance matrix. In addition, $\mu_k$ is the mean position of the points along the line. It acts like a moment arm. A large $\mu_k$ (i.e. a longer moment arm) can induce more variance to the estimated $\hat{\rho}$. We can conclude that the error with the estimated line parameters not only depends upon input pixel noise $\sigma^2$, the number of points that participate in the fitting, but also depends upon the spread of points ($S_k^2$) and their centroid $\mu_k$. Of particular interest is the centroid of the points. The closer the points centroid to the origin, the smaller the error is. This implies that we can translate the coordinate system tominimise $\mu_k^2$. In other words, coordinate system matters for the error of $\rho$. Figure 1 shows result of the error of the fitted line parameters as a function of the number points used and the average distance between points. For the same average distance between two sample pixels on the line, the greater is the number of sample points, the less is the error. For the same number of sample points, the greater is the average distance between two sample pixels on the line, the less is the error.

[1] Note $k_n^2 - k_n\mu_k = (k_n - \mu_k)^2$ and $\Sigma \mathrm{k}_n^2 = \mathrm{S}_k^2 + \mathrm{N}\mu_k^2$.

## Error propagation for circle fitting

A circle can be represented by $F(x, y, \Theta) = (x - a)^2 + (y - b)^2 - R^2 = 0$, where $(a, b)$ is the centre of the circle and $R$ is the radius of the circle. Given point scatter $\hat{X} = (\hat{x}_n, \hat{y}_n)$, $n = 1, \ldots, N$, the least squares fitting amounts to estimating the parameter $\hat{\Theta} = (\hat{a}, \hat{b}, \hat{R})$ by minimising the sum of squares of geometric distances as shown in Eq. (1).

Given $F(x, y, \Theta)$ as defined above, we have

$$\frac{\partial F_n}{\partial \Theta} = \begin{pmatrix} \frac{\partial F_n}{\partial a} \\ \frac{\partial F_n}{\partial b} \\ \frac{\partial F_n}{\partial R} \end{pmatrix} = -2 \begin{pmatrix} x_n - a \\ y_n - b \\ R \end{pmatrix}$$

Hence,

$$\frac{\partial g}{\partial \Theta} = \frac{2}{R^2} \sum_{n=1}^{N} \begin{pmatrix} w_n^2 & w_n z_n & R w_n \\ w_n z_n & z_n^2 & R z_n \\ R w_n & R z_n & R^2 \end{pmatrix} \tag{10}$$

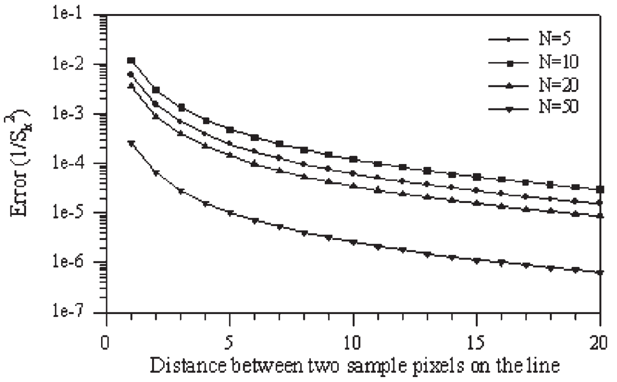


**Fig. 1** Errors of the fitted line parameters as a function of the number of pixels ($N$) and the average distance between them

where $w_n = x_n - a$ and $z_n = y_n - b$. In the polar coordinate system, a circle is represented as

$$x_n = a + R\cos\alpha_n$$

$$y_n = b + R\sin\alpha_n$$

where $\alpha_n$ is the direction from circle centre to point $(x_n, y_n)$. $\frac{\partial g}{\partial \Theta}$ can therefore be re-expressed as

$$\frac{\partial g}{\partial \Theta} = 2\begin{pmatrix} \Sigma\cos^2\alpha_n & \Sigma\sin\alpha_n\cos\alpha_n & \Sigma\cos\alpha_n \\ \Sigma\sin\alpha_n\cos\alpha_n & \Sigma\sin^2\alpha_n & \Sigma\sin\alpha_n \\ \Sigma\cos\alpha_n & \Sigma\sin\alpha_n & N \end{pmatrix} \quad (11)$$

Substituting Eq. (11) into (4) yields the covariance matrix of the circle parameters:

$$\Sigma_{\Delta\Theta} = \sigma^2\begin{pmatrix} \Sigma\cos^2\alpha_n & \Sigma\sin\alpha_n\cos\alpha_n & \Sigma\cos\alpha_n \\ \Sigma\sin\alpha_n\cos\alpha_n & \Sigma\sin^2\alpha_n & \Sigma\sin\alpha_n \\ \Sigma\cos\alpha_n & \Sigma\sin\alpha_n & N \end{pmatrix}^{-1} \quad (12)$$

We can conclude from the above equation that (1) the variances of the estimated circle parameters do not depend upon the circle radius, and (2) the variance of the estimated circle radius depends only upon the number of points used and their spatial distribution (the orientation of each point ($\alpha_n$) and the spread of points) via the first $2 \times 2$ submatrix as illustrated in Fig. 2.

For circle fitting, the error is expressed by the trace of the covariance matrix of Eq. (12). Figure 2 shows the error of the fitted circle parameters with different arc ranges and different pixel numbers. From this figure we find that error decreases with the increase of the number of sample points used for circle fitting. This figure also shows that given the same number of pixels, if sample pixels are selected over a larger arc segment of a circle less error will be obtained. This means that if pixels have a large spread on the circle, we can obtain better fit.

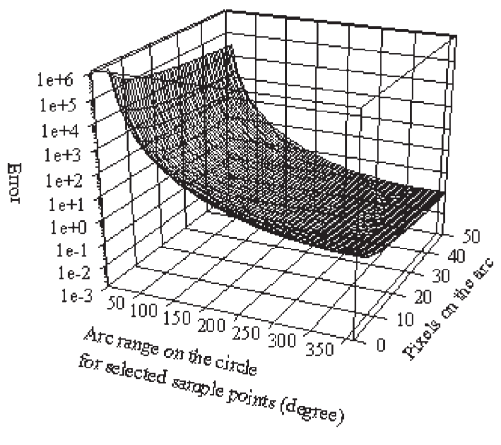


**Fig. 2** Errors of the fitted circle parameters as a function of the number of pixels (*N*) and the range of arc segment occupied by the fitting points

In summary, in this section we introduce an analytical method for propagating input errors to output parameters, estimated via a least-square fitting procedure. We present results from applying the error propagation method to various curve fitting problems. Compared with the existing error propagation methods such as that of Olson [18], ours has the following advantages:

- It is systematic.
- It handles simple and complex curves.
- It provides insights.
- It automatically adapts to image noise characteristics.

In the sections to follow, we will show how the error propagation procedure can be applied to the HT.

## Overview of proposed methods

In this section we describe the improvements we made for curve detection. The proposed error propagation techniques are used both to estimate the uncertainty of the curve parameters and to determine a subset of seed points that are most likely located on the curves.

### Seed point selection

The seeds points are the pixels that are most likely located on the curve and that produce the most accurate curve parameter estimation. By choosing the seed points, only a small part of image pixels near the selected seed points will be examined to determine their contribution to the accumulator array. In the paragraphs that follows, we consider for both line and circles.

#### *Line detection*

The idea of selecting seed points to reduce computation is not new. Several methods [10–12,18] have been proposed in this regard. The seed points in these methods are either chosen randomly or chosen for every pixel. For the image with many pixels, this is time consuming. If we can find the best seed pixel on the line before we calculate the parameters for this line, we do not need to calculate line parameters for every pixel. Instead, we can simply use the most likely seed pixels to calculate the parameters for this curve. This will reduce the computation time greatly. For line detection, a seed point is the one that is most likely located on a line and that yields the most accurate detection. Our study shows that the end points of a line segment should be chosen as the seed points. This is because end points maximise the total range between points on the line segment and the seed points, which leads to smaller errors with the estimated line parameters

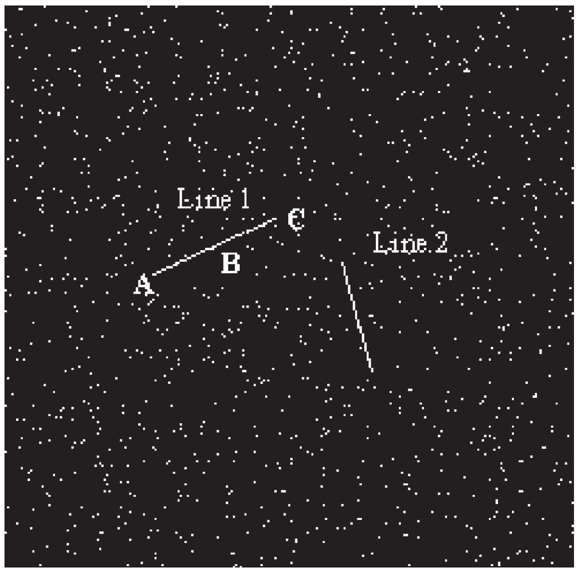


**Fig. 3** Line 1 and line 2 amid 2% of noise

as shown in Eq. (9) and Fig. 1. This is further demonstrated empirically as shown in Fig. 3. Figure 3 shows an example image with two line segments amid 2% random noise. *A* and *C* are two end points of Line 1 and *B* is the midpoint of the line. Figure 4 plots the 1D accumulator for $\theta$ for line 1 in Fig. 3 for different seed points along the line. All peaks correspond to the line segment Line 1 in Fig. 3. For seed pixels located at the ends of the line segment (points A & C), the peaks are much sharper than in the middle part of the segment. In fact, the further a point is from the midpoint of the line, the sharper is the peak in the accumulator. This means that choosing the end points of the line-segment as the seed points will produce more accurate detection. We can determine the end points of a line by checking the connectivity around a pixel. If a pixel connects to only one other pixel or it is an arc point where two line segments meet and form a vertex, we think this pixel probably is an end point of a line.

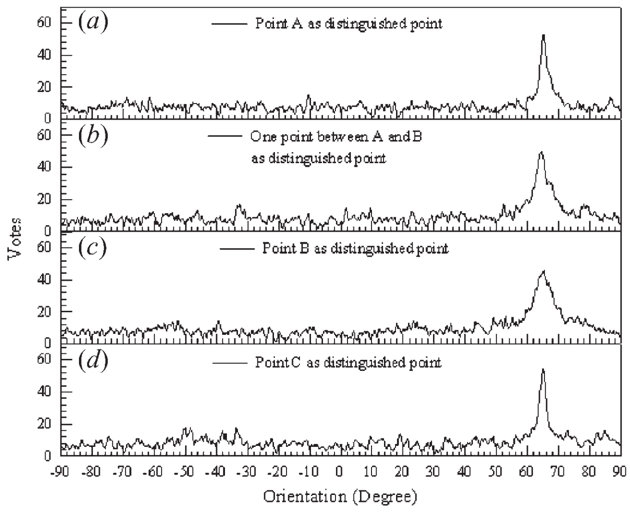


**Fig. 4** One dimensional HT accumulator of line 1 in Fig. 3 for different seed points. Using end points (A and C) as the seed points leads to better line detection (sharper peak). Here, lines detection is performed with the standard HT

## *Circle detection*

For circle detection, the best-seed pixels are are found to be such two pixels that their distance equals to the hypothesised diameter of the circle. The rational remains the same as for seed selection for line detection since the two points that are diametrically point maximise their total range to other points on the circles, which lead to more accuracy with the estimated circle parameters as demonstrated by Eq. (12) and Fig. 2. If a pair of pixels we choose are far enough and have close tangent value, we assume these two pixels are the pixels at the end of the diameter line of the circle and then we use these two pixels as the seed pixels. We can easily determine three parameters of the circle by using these two seed pixels. Then we only need to check if there are enough pixels in a certain range, which depends upon the length and orientation of the line segment connecting the two seed pixels, to support this circle. Statistically, the choice of the two pixels will also lead to better estimation accuracy of the circle parameters as indicated in Eq. (12) and Fig. 2, since the two seed points are maximally separated, and thus increase the average arc segment length.

## Error propagation

The noise in the images may result from a variety of sources. The accumulated effect of this noise induces a locational error to each pixel in the image. In the subsequent vision algorithm, the locational error will be carried over through each vision step up to the final result, so that the output of a vision algorithm is often uncertain. Development of the best curve detection algorithm requires an understanding of how the uncertainty due to perturbation affecting the input images propagates through different algorithmic steps, and results in a perturbation on the output measurements. This means that we must propagate image error through each intermediate vision step up to the final output to characterise the performance of the vision algorithm. The following subsections describe how to use the error propagation technique described in Section 3 to calculate the errors for line fitting and circle fitting.

### *Line detection*

After we select the seed pixel we only need to pick another pixel to calculate the parameters of the line that passes through the pair. Conventionally, the parameters of the line receive the same vote for any two points. Here, we want to tie the weight to the uncertainty of the parameters of the line. Assuming $(x_0, y_0)$ is the seed pixel and $(x_1, y_1)$ the other pixel, the errors with $(x_0, y_0)$ and $(x_1, y_1)$ will propagate to the line parameters they determine. From Eqs (7)–(9) we have

$$\mu_k = (k_0 + k_1)/2$$

$$S_k^2 = (k_0 - \mu_k)^2 + (k_1 - \mu_k)^2 = (k_0 - k_1)^2/2$$

$$\Sigma_{\Delta\Theta} = \sigma^2 \begin{pmatrix} \frac{2}{(k_0 - k_1)^2} & \frac{k_0 + k_1}{(k_0 - k_1)^2} \\ \frac{k_0 + k_1}{(k_0 - k_1)^2} & \frac{1}{2} + \frac{(k_0 + k_1)^2}{2(k_0 - k_1)^2} \end{pmatrix} \tag{13}$$

where $k_0 = x_0 \sin\theta - y_0 \cos\theta$ and $k_1 = x_1 \sin\theta - y_1 \cos\theta$. $k_0$, $k_1$, $\mu_k$ and $S_k^2$ are defined in Section 3.1. Substituting those parameters into Eq. (9) yields the covariance matrix $\Sigma_{\Delta\Theta}$ of the line parameters. Figure 5 shows a 1D accumulator array for $\theta$ for the lines in Fig. 3 with the error propagation. Figure 5(a) shows a case where a noise pixel was used as the seed pixel, Fig. 5(b) shows a case where a pixel on Line 1 was used as the seed pixel, and Fig. 5(c) shows a case where a pixel on Line 2 was used as the seed pixel. Note that peaks are present where appropriate, but that no peak is present when a noise pixel was used as the seed pixel. Comparing Fig. 5(b) with Fig. 4(a) (the same seed pixel), it can be seen that the peak is sharper when error propagation applied. This demonstrates improvement brought out by error propagation procedure.

### Circle detection

Given a pair of seed pixels $(x_0, y_0)$ and $(x_1, y_1)$ and randomly select another one $(x_2, y_2)$, we can use the following formula to calculate the errors with the parameters of the circle determined by the three pixels:

$$\cos(\alpha_n) = \frac{x_n - a}{R}$$

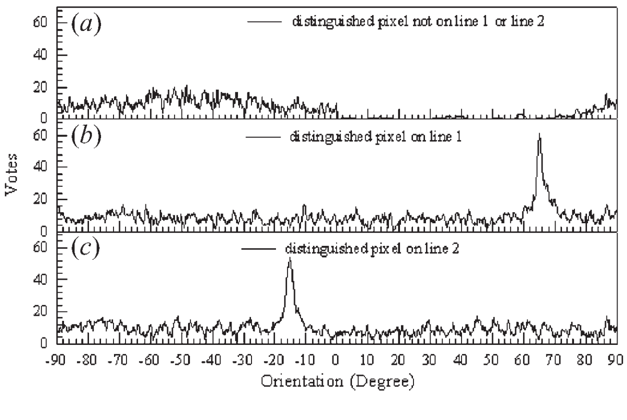


**Fig. 5** One dimensional HT accumulator for line 1 in Fig. 3 for different seed points. Here, lines detection is performed with the new HT that incorporates error propagation

$$\sin(\alpha_n) = \frac{y_n - b}{R} \tag{14}$$

$$\Sigma_{\Delta\Theta} = \sigma^2 \begin{pmatrix} \Sigma\cos^2\alpha_n & \Sigma\sin\alpha_n \cos\alpha_n & \Sigma\cos\alpha_n \\ \Sigma\sin\alpha_n \cos\alpha_n & \Sigma\sin^2\alpha_n & \Sigma\sin\alpha_n \\ \Sigma\cos\alpha_n & \Sigma\sin\alpha_n & 3 \end{pmatrix}^{-1}$$

where $n = 0, 1, 2$.

### Voting kernel

Like many other researchers, Olson [18] used a top-hat voting kernel in his research work on curve detection. Specifically, the top-hat scheme assumes all feature points located within a close range of the hypothesised line contribute equally to the line. The accumulator is therefore incremented by a unit for all feature points. This scheme is inadequate in that data points are not all equally reliable. In this paper, we used a voting kernel that is a smooth function of differences in parameter values for updating accumulator. Kittler and Palmer [14] have shown that the Hough algorithm works better if the voting kernel is a smooth function of differences in parameter values. However, unlike their scheme, which is rather complex and computationally intense, our scheme is much simpler and still robust. Assume $\hat{\Theta}$, a curve parameter vector estimated from a set of feature points, is distributed as $\hat{\Theta} \sim N(\Theta, \Sigma_{\Delta\Theta})$, where $\Theta$ is a quantised parameter vector. Given $\hat{\Theta}$ and the covariance matrix of the estimated curve parameters $\Sigma_{\Delta\Theta}$, which can be calculated from Eqs (9) or (12) for line and circle, respectively, we use the following function to calculate voting kernel $P(\hat{\Theta}|\Theta)$, where $\Theta$ is a quantised parameter vector. With this definition, the contributions of the pixels that give rise to $\hat{\Theta}$ to $\Theta$ can be determined from its likelihood, i.e.

$$P(\hat{\Theta}|\Theta) = \exp^{-0.5(\hat{\Theta}-\Theta)^t\Sigma_{\Delta\Theta}^{-1}(\hat{\Theta}-\Theta)} \tag{15}$$

It is clear from Eq. (15) that, given each $\hat{\Theta}$ (estimated from a set of points) and its covariance matrix $\Sigma_{\Delta\Theta}$ estimated via our error propagation, the bin for a $\Theta$ is updated based on $P(\hat{\Theta}|\Theta)$, i.e. its likelihood. The further away $\Theta$ from $\hat{\Theta}$, the smaller the likelihood is and the less contribution $\Theta$ receives from the point as shown in Fig. 6.

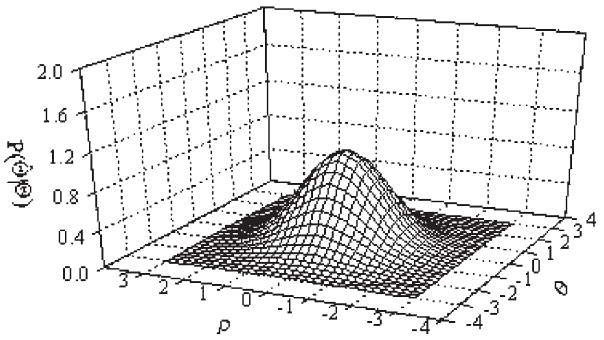


**Fig. 6** Illustration of the voting kernel function

Deterministic algorithms

Based on the discussion above, the algorithms of curve detection technique for a line and circle can be described as follows. The input image in this research is edge image, in which black pixels represent background and white pixels are image pixels.

*Line detection*

1. Detect a seed pixel from the input edge pixels.
2. Randomly find another edge pixel and pair it with the seed pixel.
3. Calculate the line parameters $\hat{\Theta}$ (determined by the pair of pixels) and determine the contribution of the pair of pixels to other quantised line parameters $\Theta$ using Eq. (3), with the covariance matrix determined by Eq. (1), and update the accumulator accordingly.
4. Repeat until all edge pixels have been used.
5. Search the accumulator array to identify the parameters with the maximum vote. If the maximum vote exceeds a predefined threshold, a line primitive has been detected.
6. Remove the pixels associated with the detected line segment from input image.
7. Start over again, with the remaining image.

*Circle detection*

1. Detect two seed pixels from the input edge pixels, that have close tangent values.
2. Randomly select another edge pixel whose distance to the centre of the two seed points is approximately half the distance between the two seed points.
3. Calculate the circle parameters ($a$, $b$, $R$) determined by the three pixels.
4. Determine the contribution of the three pixels to the circle parameters using Eqs (14) and (15), and update the accumulator accordingly.
5. Repeat until all edge pixels have been used.
6. Search the accumulator array to identify the circle parameters with the maximum vote. If the maximum vote exceeds a threshold, a circle has been detected. Add it to the output list.
7. Remove the pixels associated with the detected circle from input image.
8. Start over again, with the remaining image.

## Experimental results

The proposed scheme has been applied to synthetic and real images for line detection to test its performance. All experiments were carried out with the following settings. A 900-bin accumulator was used for all images. We also consider other quantisation levels. Considering both computational time and accuracy, 900-bin accumulator is the best choice. Pixels within a three-pixel-wide corridor were assigned to a line. For each line that surpasses the detection threshold in each subproblem, only the parameters at which the most votes occurred were kept. The detected line segment had no gaps larger than three pixels long. The minimum accepted line length was chosen to reduce the false positives based on the noise level. The higher the noise level is, the larger the minimum accepted line length. Figure 7 shows two synthetic images that were used to test the line detection algorithm. The resolution of both images was 256, both horizontally and vertically. In Fig. 7(a), two lines were amid 5% noise, while Fig. 7(b) contains many line segments of different orientations and lengths amid 2% of noise. Figure 8 shows line segments that were detected. All of the lines were found in the image. Figure 9 shows some real world images that were used for line detection. Those images have different size (640 × 480, 640 × 480, 320 × 240, 531 × 352, 250 × 250 and 531 × 354). All of the long lines in the images were found. However, some short lines and/or curve lines were not found. The reason for this is that we set the minimum accepted line length to a proper value so that we did not have too many false positives. Figure 10 show results of applying our algorithm for circle detection.

For comparison, we also apply our technique and the standard HT to some real images of different qualities. The results are summarized in Figs 11 and 12. We can conclude from these figures that when the quality of the image is good as shown in Fig. 11, two techniques have comparable detection performance, both detecting most of

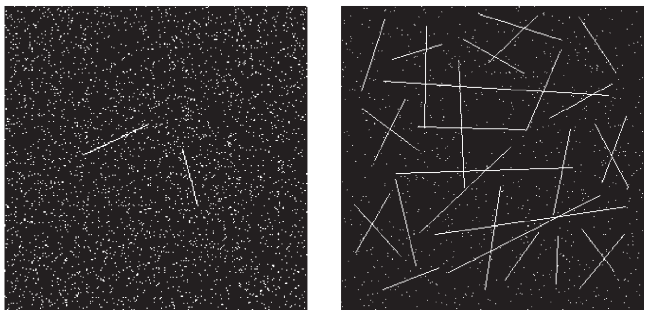


**Fig. 7** Synthetic images of line segments contaminated with image noise

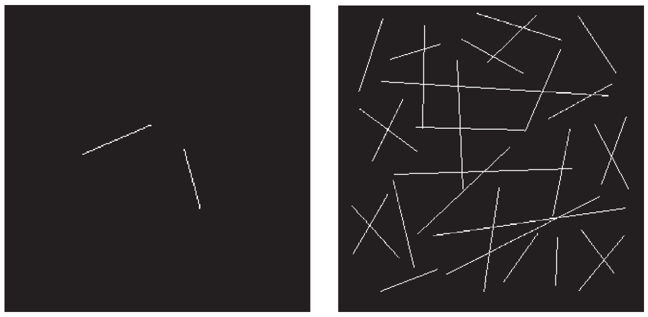


**Fig. 8** Detected line segments from images in Fig.7

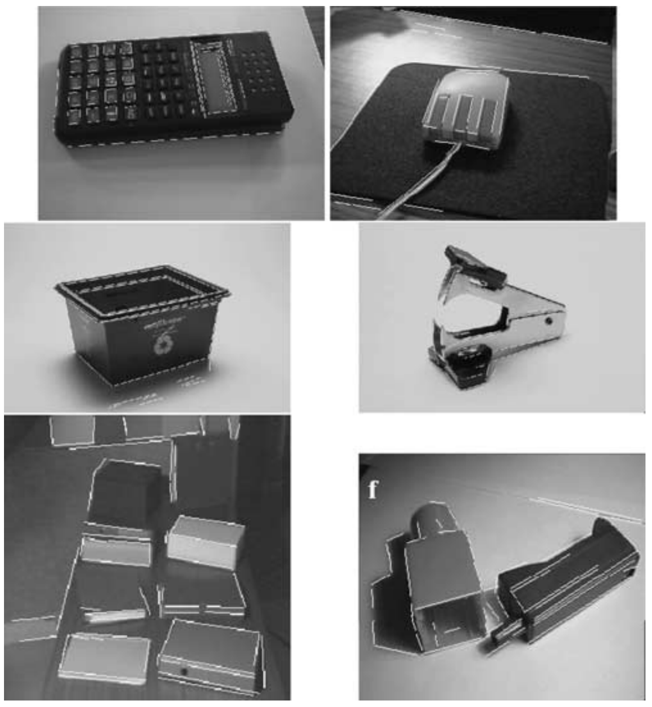


Fig. 9 Various real images with the detected line segments superimposed on the original images

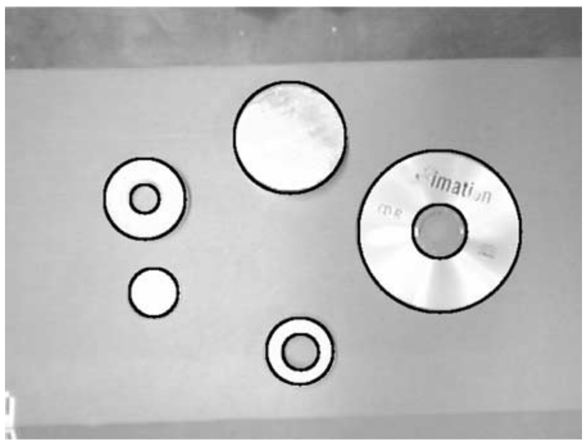


Fig. 10 A real image with the detected circles superimposed on the original images

the major lines. However, ours appear to produce more accurate detection. However, when the image quality is poor as shown in Fig. 12, our technique outperforms the basic HT in both detection accuracy and robustness. The standard HT failed to detect some lines and the detected lines are not as accurate. This is because image errors are explicitly modeled and accounted for by our technique.

## Conclusions

In this paper, we present a new curve detection method aimed at improving accuracy and robustness as well as computational efficiency. The essence of our approach lies in the proposed analytic error propagation technique. It allows to quantitatively determine the contribution of feature points to the curve parameters, and to statistically select seed pixels that most likely are located on the curves and that produce the best curve estimation.

Fig. 11 Results of our algorithm (a) versus that of the standard HT (b) for a good quality image. Ours appears to produce more accurate line detection than the basic HT

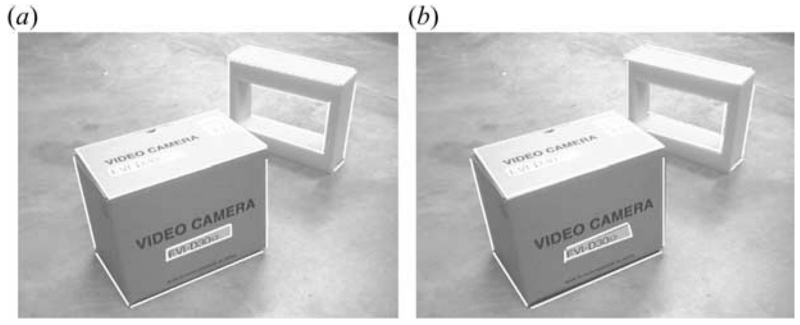
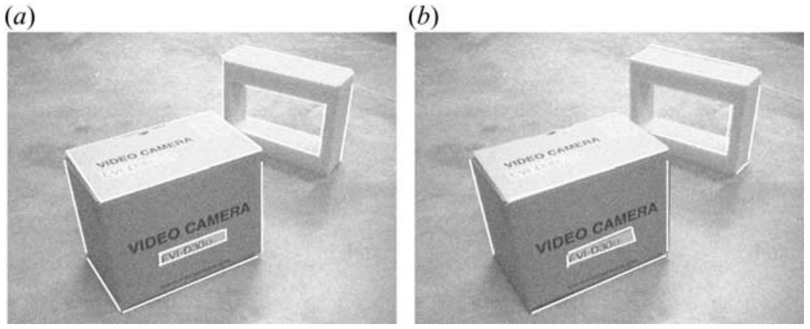


Fig. 12 Results of our algorithm (a) versus that of the standard HT (b) for a low quality image. The standard HT failed to detect some lines and the detected lines are not as accurate



Specifically, robustness and accuracy improvement is achieved by analytically propagating the errors with image pixels to the estimated curve parameters. The errors with the curve parameters are then used to determine the contribution of pixels to the accumulator array. The computational efficiency is achieved by choosing seed pixels, and by performing progressive detection. We have presented detailed step-by-step implementation of our method for line and circle detection. The performance of our method is experimentally studied for the detection of lines and circles. Despite the use of the seed points and random sampling feature points, our technique is still rather computationally involved, due to the need to performing error propagation for each feature point. The experimental results, on the other hand, show that our technique is especially robust with image that has low signal to noise ratio. The concept also can be applied for other curves, such as ellipse.

## References

1 Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. Patt. recogn 13(2):111–122
2 Duda RO, Hart PE (1972) Use of the Hough transform to detect lines and curves in pictures. Comm. ACM 15(1):11–15
3 Hough PVC (1962) Method and means for recognizing complex patterns. U.S. Patent 3069654
4 Gerig G, Klein F (1986) Fast contour identification through efficient hough transform and simplified interpretation strategy. Proceedings 8th Int. Conf. on Pattern Recognition, Paris, France 498–500
5 Gerig G (1987) Linking image space and accumulator space: A new approach for object recognition. Proceedings First Int. Conf. on Computer Vision, London, UK 112–117
6 Princen J, Illingworth J, Kittler J (1990) A hierarchical approach to line extraction based on the hough transform. Comput Vision Graph Image Process 52(1):57–77
7 Ben D, Sandler MB (1990) A combinatorial hough transform. Pattern Recog Lett 11(3):167–174
8 Xu L, Oja E, Kultanen P (1990) A new curve detection method: Randomized hough transform (rht). Patt Recog Lett 11:331–338
9 Kiryati N, Eldar Y, Bruckstein AM (1991) A probabilistic hough transform Patt Recog 24(4):303–316
10 Ben D, Leavers VF, Sandler MB (1989) A dynamic combinatorial hough transform. Proceedings 5th Int. Conf. on Image Analysis and Processing, Positano, Italy 20–22:152–159
11 Leavers VF (1992) The dynamic generalized hough transform: Its relationship to the probabilistic hough transforms and an application to the concurrent detection of circles and ellipses. CVGIP: Image Understanding 56(3):381–398
12 Yuen SYK, Lam TSL, Leung NKD (1993) Connective hough transform. Image Vision Comput 11(5):295–301
13 Stephens RS (1991) Probabilistic approach to the hough transform. Image Vision Comput 9(1):66–71
14 Kittler J, Pamler PL (1994) Robust and statistically efficient detection of parametric curves in 2d images. NSF/ARPA Workshop on Performance v. Methodology in Computer Vision, Seattle, WA
15 Breuel TM (1996) Finding lines under bounder error. Pattern Recog. 29(1):167–178
16 Olson CF (1996) Decomposition of the hough transform: Curve detection with efficient error propagation. Proceedings European Conf on Computer Vision 1:263–272
17 Ji Q, Haralick RM (1998) Breakpoint detection using covariance propagation. IEEE Trans. Patt Anal Mach Intell 20(8)
18 Olson CF (1999) Constrained hough transforms for curve detection. Comput Vision Image Understand 73(3):329–345
19 Haralick RM (1994) Propagating covariance in computer vision. Proceedings 12th ICPR 493–498
20 Ji Q, and Haralick RM (1999) Error propagation for computer vision performance characterization. IEEE Int Conf on Imaging Science, Systems, and Technology, Las Vegas
21 Ji Q, Haralick RM (1999) An optimal bayesian hough transform for line detection. IEEE Int Conf on Image Processing, Kobe, Japan

**Qiang Ji** received a MS degree in electrical engineering from the University of Arizona in 1993 and a PhD in electrical engineering from the University of Washington in 1998. He is currently an Assistant Professor with the Department of Electrical, Computer and Systems Engineering at Rensselaer Polytechnic Institute. Previously, he was an Assistant Professor with the Department of Computer Science at the University of Nevada, Reno. Dr Ji's research areas include computer vision, image processing, pattern recognition, information fusion and robotics. Dr Ji has published more than 50 papers in refereed journals and conferences in these areas. His research has been funded by local and federal agencies such as NIH, AFOSR and ARO, and by private companies including Boeing and Honda.

**Yonghong Xie** received a MS degree in computer science from the University of Nevada at Reno. He is currently working for the Supreme Court of Nevada as a systems analyst.

**Originality and contribution**

This paper introduces a new HT technique for curve detection. The originality of the research lies in the analytic method for error propagation for curve-fitting and the use of the error propagation technique to improve the HT accuracy, robustness, as well as computational efficiency. Specifically, the positional errors with the feature points are explicitly accounted for by analytically propagating the errors to the estimated curve parameters, which are used, in turn, to determine the contribution of feature points to curve parameters. The errors with the curve parameters are also used to determine best seed points to select. Robust, accurate, and efficient curve detection is important for many computer vision and photogrammetry applications.