Computer Vision and Image Understanding **78,** 119–137 (2000)
doi:10.1006/cviu.1999.0831, available online at http://www.idealibrary.com on IDEAL®

Robust Detection of Lines Using the Progressive Probabilistic Hough Transform

J. Matas

Centre for Machine Perception, Czech Technical University, Karlovo náměstí 13, 12135 Prague, Czech Republic; and CVSSP, University of Surrey, Guildford, Surrey GU2 5XH, United Kingdom
E-mail: G.Matas@ee.surrey.ac.uk

and

C. Galambos and J. Kittler

CVSSP, University of Surrey, Guildford, Surrey GU2 5XH, United Kingdom
E-mail: C.Galambos@ee.surrey.ac.uk, J.Kittler@ee.surrey.ac.uk

Received February 12, 1999; accepted November 5, 1999

In the paper we present the *progressive probabilistic Hough transform* (PPHT). Unlike the probabilistic HT, where the standard HT is performed on a preselected fraction of input points, the PPHT minimizes the amount of computation needed to detect lines by *exploiting the difference in the fraction of votes needed to reliably detect lines with different numbers of supporting points.* The fraction of points used for voting need not be specified ad hoc or using a priori knowledge, as in the probabilistic HT; it is a function of the inherent complexity of data. The algorithm is ideally suited for real-time applications with a fixed amount of available processing time, since voting and line detection are interleaved. The most salient features are likely to be detected first. While retaining its robustness, experiments show that the PPHT has, in many circumstances, advantages over the standard HT.

1. INTRODUCTION

The Hough transform (HT) is a popular robust statistical method of extraction of geometric primitives. In literally hundreds of papers every aspect of the transform has been scrutinized—parameterization, accumulator design, voting patterns, peak detection, to name but a few [6]. The introduction of the randomized version of the Hough transform is one of the most important recent developments in the field [8, 16]. The approach proposed in this paper falls into the probabilistic (or Monte Carlo) class of Hough transform algorithms, the objective of which is to minimize the proportion of points that are used in voting while

1077-3142/00 $35.00

maintaining false negative and false positive detection rates almost at the level achieved by the standard Hough transform [14]. Unlike the randomized HT class of algorithms (see [7] for an overview of randomized Hough transforms), the probabilistic HT and the standard Hough transform share the same one-to-many voting pattern and the same representation of the accumulator array.

In the original paper on probabilistic Hough transform [8], Kiryati *et al.* show that it is *often* possible to obtain results identical to those of the standard Hough transform if only a fraction p of input points are used in the voting process. In the first step of the probabilistic Hough transform a random subset of points is selected and a standard Hough transform is subsequently performed on the subset. Successful experiments with p as low as 2% are presented. The poll size is a parameter critically influencing the probabilistic Hough transform performance. The authors analyze the problem for the case of a single line immersed in noise. Unfortunately the derived formulas require *a priori* knowledge of the number of points belonging to the line, which is rare in practice. In [2], Bergen and Shvaytser show that the probabilistic Hough transform can be formulated as a Monte Carlo estimation of the Hough transform. The number of votes necessary to achieve a desired error rate is derived using the theory of Monte Carlo evaluation. Nevertheless, the poll size remains independent of the data and is based on *a priori* knowledge.[1] If little is known about the detected objects, a conservative approach (much larger than necessary poll size) must be adopted, diminishing the computational advantage of the probabilistic Hough transform.

The need to pre-select a poll size can by bypassed by an adaptive scheme [15, 18, 19]. In the adaptive probabilistic Hough transform the termination of voting is based on monitoring the polling process. The criterion suggested by Yla-Jaaski and Kiryati [18] is based on a measure of stability of the ranks of the highest peaks. The goal is formulated as "to obtain the same maximum as if the Hough transform had been accumulated and analyzed." In our opinion, such a formulation is unrealistic, since in almost all applications the detection of multiple instances of lines (circles, etc.) is required, whereas the given stopping rule depends on the prominence of the most significant feature only. For instance, if the input contains any number of lines of equal length, it will not be possible to detect a stable maximum regardless of the percentage of input points used for voting. In [19] the termination rule is based on stable ranks of k largest peaks. The formulation allows detection of multiple instances of geometric primitives, but k has to be known a priori or a threshold has to be set for the stability criterion.

In the paper we present a new form of an adaptive probabilistic Hough transform. Unlike the above-mentioned work, we attempt to minimize the amount of computation needed to detect lines (or in general geometric features) by *exploiting the difference in the fraction of votes needed to reliably detect lines (features) with different numbers of supporting points.* It is intuitively obvious (and it directly follows, e.g., from Kiryati's analysis in [8]) that for lines with strong support (long lines) only a small fraction of the supporting points have to vote before the corresponding accumulator bin reaches a count that is nonaccidental. For shorter lines a much higher proportion of supporting points must vote. For lines with support size close to counts due to noise a full transform must be performed.

This is achieved by dynamically controlling the line acceptance threshold as a function of votes cast. The threshold schedule is derived from theoretical considerations. The line detection algorithm based on the theory is extensively validated experimentally. A comparison

[1] Perhaps it is more appropriate to speak about assumptions rather than knowledge.

with benchmark algorithms including the standard Hough transform shows that it is computationally efficient and has other attractive properties.

The proposed algorithm is sufficiently close to the SHT transform that many of the methods of improving or modifying its performance can be easily adapted to work with it. Examples range from simply detecting other two-parameter shapes such as circles [12] to techniques that change the voting scheme from the one-to-many of the SHT to one-to-one as found in the RHT [16] or window-based schemes such as [3]. The PPHT does not rely on particular aspects of the input data structure such as connectivity (unlike [9]), though it borrows the idea of removing labelled structures from the input data. The algorithm is also applicable to finding objects with a higher dimensional parameterization, as it intrinsically keeps the number of votes in the accumulator space low. In Section 7 we show that the PPHT is well suited for sparse representations of the accumulator space commonly used in RHT implementations [17].

This method is not so easily adapted to techniques that do repeated accumulation and peak detection, such as the adaptive Hough transform [5] and the multiresolution Hough transform [1], since the PPHT relies on being able to identify the pixels belonging to a line as they are detected. Algorithms that use two passes to gain information about the image content before doing a full transform such as [11] could be adapted, but with more difficulty.

This paper is organized as follows. In the next section the new algorithm with its underlying theory and properties is presented. Section 3 describes the experimental setup and the adopted performance criteria for output quality and computational efficiency. Sections 4 and 5 evaluates PPHT performance on synthetic imagery. A comparative evaluation of the advocated algorithm on a selection of real images is given in Section 6. In Section 7 we investigate the accumulator occupancy of the proposed algorithm and its dynamics during the voting process. The paper is drawn to conclusion in Section 8.

2. THE ALGORITHM

To minimize the computation requirements the proposed algorithm, which we call the *progressive probabilistic Hough transform* (PPHT), proceeds as follows. Repeatedly, a new random point is selected for voting. After casting a vote, we test the following hypothesis: "could the count be due to random noise?" or more formally, "having sampled m out of N points, does any bin count exceed the threshold of l points which would be expected from random noise?" The test requires a single comparison with a threshold per bin update. Of course, the threshold l changes as votes are cast. When a line is detected the supporting points retract their votes. Remaining points supporting the line are removed from the set of points that have not yet voted and the process continues with another random selection.

Such algorithm possesses a number of attractive properties. First, a feature is detected as soon as the contents of the accumulator allows a decision. The progressive probabilistic algorithm is an *anytime algorithm*. It can be interrupted and still output useful results, in particular salient features that could be detected in the allowed time. The algorithm does not require a stopping rule. The computation stops when all the points either have voted or have been assigned to a feature. This does not mean that a full Hough transform has been performed. Depending on the data, only a small fraction of points could have voted, the rest being removed as supporting evidence for the detected features. If constraints are given, e.g., in the form of minimum line length, a stopping rule can be tested before a point for voting is selected.

Without a stopping rule the PPHT and the standard Hough transform differ mainly in the number of false positives. False negatives—missed features with respect to the standard Hough transform (SHT)—should not pose a significant problem, because if a feature is detectable by the SHT it will be detected by the PPHT at the latest when the voting is finished. False negatives are therefore caused only by incorrect assignment of edge points to previously detected lines.

In our experiments a straightforward one-to-many voting scheme was used. The proposed method reduces computation by minimizing the number of points that participate in the voting process, so common improvements that reduce the number of votes cast (e.g., by exploiting the gradient information, if it is available) do not interfere with the benefits of the method.

In setting the decision threshold we assume that all points are due to noise. This is a worst-case assumption, but if many lines are present in the image the assumption is almost valid, since only a fraction of points belong to any single line.

Let us denote the number of bins for angle as N_θ and the number of bins for distance from origin as N_ρ (the standard ρ–θ representation of the accumulator space [4] is assumed). We adopt the following model of the voting process. Every randomly selected pixel votes in all N_θ bins. A pixel can either belong to no line (a noise point), belong to a single line, or lie on an intersection of lines. In the first case all N_θ votes cast add noise to bin counts. We assume that, for every θ, a random bin is incremented; i.e., each bin is equally likely to be incremented with probability $1/N_\rho$. If a point lies on a line, one vote is cast into the bin corresponding to this line and the remaining $N_\theta - 1$ votes are assumed to fall into random bins, 1 vote for each angle θ. For points on line intersections we assume $N_\theta - 2$ votes fall in random bins. Since $N_\theta \gg 1$, $N_\theta \approx N_\theta - 1 \approx N_\theta - 2$, we do not (and we cannot not) distinguish between the three cases and assume that always random N_θ bins are incremented.

Clearly, the counts in individual bins are not independent. All counts for bins with a fixed θ must add to the number of edge points that have voted. Moreover, counts in bins with similar θ and ρ are not statistically independent either, due to the cosine voting pattern of a single point. Nevertheless, to keep computation tractable, we will assume that the count in any single bin is an independent random variable with binomial distribution $B(N, p)$, where N is the number of edge pixels that have voted so far and $p = 1/N_\rho$ is the probability of selecting a particular bin with a given θ. In our voting model, the distribution of votes in the N_ρ bins for a given θ follows the multidimensional hypergeometric distribution (not multinomial distribution—the sampling is without replacement). We adopted the $B(N, p)$ simplification because we could not find a practical (efficient) testing procedure for the hypergeometric distribution. The hypothesis that is being tested after every bin is incremented is the following:

Is the count C in bin (ρ, θ) higher than a value likely to occur if $C(\rho, \theta)$ was a realization of a random variable with binomial distribution $B(N, p)$?

We would like to set the threshold so that

$$P(C(\rho, \theta) > \mathrm{thr}(N)) < l \tag{1}$$

Significance level l is a user parameter that will, if the model is accurate, indicate the number of false positives.[2] For a binomial distribution it is easy but laborious to compute

[2] In case of no post processing.

the threshold for a given N, since we have to evaluate the sum

$$\sum_{i=0}^{j} P(C(\rho, \theta) = i) \tag{2}$$

for all j till $1 - l$ is reached. Since $Np \approx Np(1-p) > 1$ as in a typical image $N \gg 1$ at the beginning of the voting process, the binomial distribution can be well approximated by a Poisson distribution. This approximation makes computation of (1) much easier (no need to compute $\binom{n}{i}$), but the summation for computing $P(C(\rho,\ \theta) > \text{thr})$ is still needed. For efficiency reasons we therefore adopt a less accurate approximation for $P(C(\rho,\ \theta))$ and replace the binomial distribution with Gaussian with mean Np and standard deviation $\sqrt{Np(1-p)}$. With this simplification, probability $P(C(\rho,\ \theta) > \text{thr})$ can be obtained using a single look-up in the standard error function.

The approximation will influence the result mostly for small N, since for $Np > 1$ the approximation with a Gaussian can be justified. If, for small values of N, a better model was needed, the values of the threshold could be precomputed. Alternatively, the binomial model could be used directly, since an approximation to the binomial sum (2) by incomplete beta function can be computed relatively efficiently [13]. We believe that even for modest values of N, the effects of various approximations of the binomial sum on the output of the line detector are negligible. Efficiency consideration, typically hardware and software specific, should therefore determine the most suitable approximation for a particular implementation.

It is interesting to note that the analysis is independent of the angular resolution of the accumulator N_θ. For small N_θ the algorithm would confuse instances of lines with similar orientations within the same bin. The recovered models could even compound to give virtual lines. As N_θ increases the recoverability of the lines will increase without affecting the voting strategy. One might argue that if the algorithm was based on incrementing parameter N_ρ and voting into N_θ a certain symmetry should be maintained. This would suggest that the resolution N_θ and N_ρ of the accumulator should be of comparable size.

2.1. Algorithm Outline

1. Check the input image; if it is empty then finish.
2. Update the accumulator with a single pixel randomly selected from the input image.
3. Remove the selected pixel from input image.
4. Check if the highest peak in the accumulator that was modified by the new pixel is higher than threshold thr(N). If not then goto 1.
5. Look along a corridor specified by the peak in the accumulator, and find the longest segment that either is continuous or exhibits a gap not exceeding a given threshold.
6. Remove the pixels in the segment from input image.
7. "Unvote" from the accumulator all the pixels from the line that have previously voted.
8. If the line segment is longer than the minimum length add it into the output list.
9. Goto 1.

Notes. Steps 1 to 3 are carried out in constant time, since all edge pixels are stored in a 1D array. Step 4 is implemented with a single if statement in the voting procedure and is also $O(1)$. Step 5 turns a line detector into a line segment detector. Step 6 can be implemented either by a scan of all pixels still in the input array (with complexity $O(N)$) or by searching

a corridor in the 2D image. The step is carried out only N_l times (N_l is the number of detected lines); only a fraction of the run time is spent during execution of step 6.

3. PERFORMANCE EVALUATION OF THE PPHT

Performance evaluation of a line detection algorithm has many aspects. In synthetic images the *correctness* of the output can be measured by the error rate, i.e., the number of false positives (spurious detected features) and false negatives (misdetected lines). In real images the definition of what should and should not be detected as a line is subjective or application-dependent. Since the PPHT is not designed for a specific application we illustrate its performance on real imagery by comparing it with the standard Hough transform [14] and the randomized Hough transform (Section 6). Experiments designed to assess the correctness (quality) of the PPHT on synthetic imagery are presented in Section 4. We do not present any results on the correctness of pixel assignment and the precision of recovered line parameters. We do not consider precision of the PPHT to be an important characteristic of the method—standard precision can by achieved by a (robust) least-squares fit performed on the assigned points. Computational efficiency of the PPHT on synthetic data is tested in Section 5.

3.1. Experimental Setup

To assess the relative merit of the PPHT, the quality of its output was compared to that of the standard Hough transform. The experiments were designed to minimize any differences between the SHT and PPHT implementations. The implementations were kept as simple as possible, with minimal post- and preprocessing (use of gradient information, connectivity, etc.). Most standard enhancements of the SHT are directly applicable to the PPHT and do not therefore change the relative merits of the methods. In the implementation, the standard ρ, θ line parameterization was used. All experiments were carried out with the following settings. Resolution of the accumulator space was 0.01 radians for θ and 1 pixel for ρ. Pixels within a 3-pixel-wide corridor were assigned to a line. Since the Hough transform detects peaks corresponding to straight lines, but we want to detect finite line segments, a single postprocessing step was implemented to separate collinear lines. From the pixels supporting a particular bin, the longest segment was chosen which had no gaps bigger than 6 pixels long. The minimum accepted line length was 4 pixels. Unless stated otherwise, the value used for the PPHT-specific parameter for the significance threshold l was 10^{-5}.

If the assumptions made are true this will lead to a 1 in 10,000 chance of random voting leading to a false line hypothesis in any single bin. Since the resolution of θ is 0.01, in the simple form of the algorithm 314 votes are made for each point, giving a 1 in 32 chance of getting an incorrect line hypothesis for each point analyzed. Experimentally it can be seen the observed number of false positives generated is lower than predicted by theory. It also should be noted that the algorithm will reject lines whose point density is too low.

4. CORRECTNESS OF LINE DETECTION

The correctness of the PPHT output was measured by the error rate. Definition of what constitutes a false positive and a false negative is not as straightforward as it may first appear. This is particularly a problem where postprocessing is used to obtain line segments

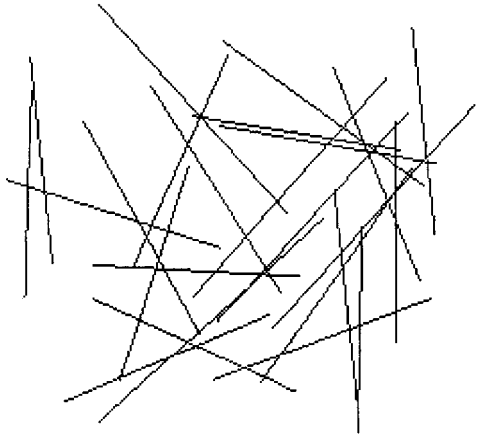


FIG. 1. Example synthetic image.

for the lines found by the Hough transform. If edge pixels are allowed to be assigned only to a single line, incorrect assignment of pixels (e.g., at intersections of lines) can lead to a split of a correctly extracted line. A pair of collinear lines almost covering a model line are not, in our opinion, identical to a pair of false positives and a false negative. Similarly, it has to be decided at what level of assignment errors a feature is declared not detected.

We decided to use the following criteria for determining the error statistics. False positives are detected lines that cover less than 80% of any single ground-truth line in the image. False negatives are those lines in the model which are covered by less than 80% by detected lines, excluding those counted as false positives.

Experiment 1. Hundred images containing a fixed number of randomly positioned lines were synthetically generated. The only source of noise in the synthetic images is introduced by the digitisation of the lines. Figure 1 shows a typical image used in the experiment. Image resolution was 256 both horizontally and vertically; the lines were 100 pixels long. The number of lines varied between 2 and 20. Both the SHT and the PPHT were run on all images. The false negatives and positives were counted as described above. The results of the experiment are summarized in Table 1. Note that the distribution of false positives and false negatives is highly asymmetric. The values of σ are larger than the mean due to the tail on one side of the distribution (FP and FN are positive quantities).

The results in Table 1 show that the PPHT outperforms the Standard Hough Transform in all the cases. Both the number of false positives and the number of false negatives for the PPHT were lower than those given by the SHT. The PPHT uses a fraction of the SHT votes, but this should reduce, in a controlled way, the average correctness of the output. The test results show the advantages of clearing the accumulator space of clutter from explained pixels as soon as the hypothesis they are associated with becomes almost certain.

Experiment 2. The parameter l of the PPHT, the significance level at which lines are accepted, controls the trade-off between false positives and the speed of the method. The influence of l is shown in Table 2. The experiments were again performed on a set of 100 synthetic images of 256×256 pixels each with five lines of 100 pixels. An increase in the significance level for line detection leads to a decrease in the number of false positives at the cost of increasing the number of voting operations required, which is proportional to run

TABLE 1
Effect of Image Clutter on False Positives (FP) and Negatives (FN) (Averages and Standard Deviations (σ) over 100 Runs Are Shown)

	SHT				PPHT			
Lines	FP	σ	FN	σ	FP	σ	FN	σ
2	0.1	0.3	0.1	0.3	0.0	0.1	0.0	0.0
4	0.4	0.7	0.4	0.7	0.1	0.5	0.1	0.2
6	1.1	1.2	1.1	1.2	0.4	0.8	0.2	0.4
8	2.6	1.7	2.6	1.7	1.4	1.3	0.9	1.0
10	4.0	1.8	4.0	1.8	2.3	1.9	1.5	1.5
12	6.4	2.1	6.4	2.1	3.8	2.2	2.8	2.0
14	8.1	2.1	8.0	2.1	5.9	2.2	4.5	2.5
16	10.9	2.1	10.9	2.0	8.4	2.2	6.7	3.1
18	13.4	2.0	13.3	2.1	9.9	2.8	8.2	3.5
20	16.1	1.9	16.0	1.9	12.7	2.4	11.6	4.1

time, as will be shown in the next section. The observed values for the standard deviation of the number of votes required to process an image are relatively large because the order of pixels used in voting is random. If the first few points selected are from a single line, it and all its supporting pixels will be removed and not considered any further. This reduces clutter in the accumulator and increases the chances that any two pixels will be from the same line.

5. COMPUTATIONAL EFFICIENCY

We have argued that the advantage of the PPHT lies in its ability to minimize the number of voting operations while almost retaining the quality of the SHT output. The run time of the algorithm depends on implementation details, the compiler as well as on the machine it is run on. To measure the run time speed we would prefer to use a number related directly to the algorithm itself rather than the measured run time. Since the activity which dominates the computation is voting and "unvoting" (vote retracting) in the Hough space, we will use the number of such operations as a measure of the amount of computation required.

TABLE 2
Influence of Significance Level on False Positives (FP), False Negatives (FN), and Run Time (Averages and Standard Deviations over 100 Tests Are Shown)

l	FP	σ	FN	σ	Votes	σ
0.1	0.34	0.73	0.39	0.82	46.88	9.70
0.01	0.25	0.70	0.36	0.76	46.68	9.80
10^{-3}	0.13	0.34	0.27	0.68	55.47	8.89
10^{-4}	0.17	0.55	0.25	0.59	57.27	8.80
10^{-5}	0.17	0.43	0.36	0.70	72.73	12.15
10^{-6}	0.11	0.31	0.25	0.61	84.63	13.19
10^{-7}	0.16	0.42	0.39	0.79	93.06	15.20
10^{-8}	0.07	0.26	0.33	0.65	108.62	12.18

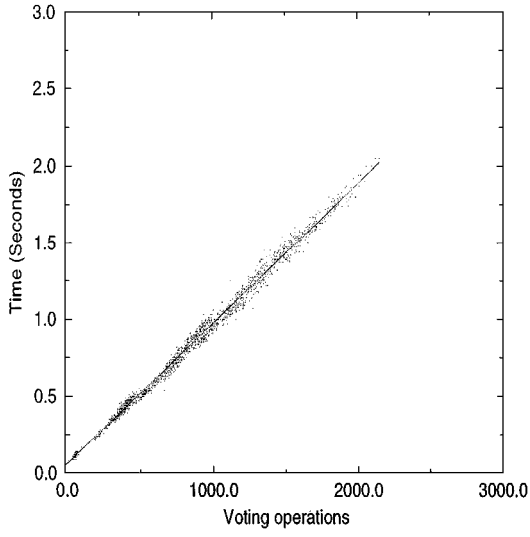


FIG. 2. Run time as a function of voting operations. The correlation coefficient is 0.998.

Experiment 3. To check the validity of the assumption, the run time and number of voting operations were measured on a large set of test images. The results in Fig. 2 show a very high correlation between the number of voting operations and the program run time (correlation coefficient 0.998).

Experiment 4. Since lines are removed as they are found, the number of voting operations required to find a single line is nearly independent of its length. Figure 3 shows the number of voting operations required to find a line in a simple image with only a few lines and no noise. The experiment was repeated 50 times, and the error bars show one standard

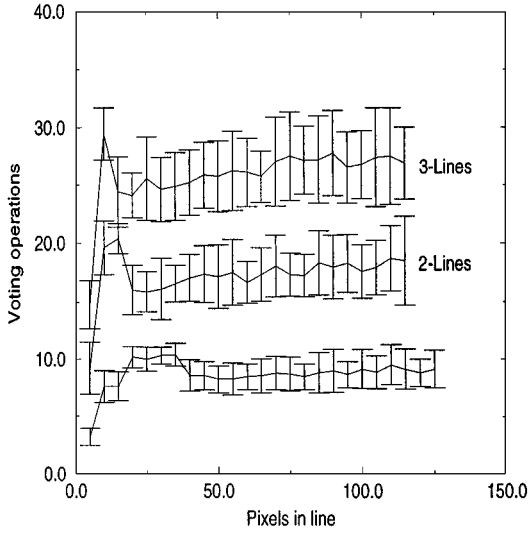


FIG. 3. Votes needed to find a small number of lines.

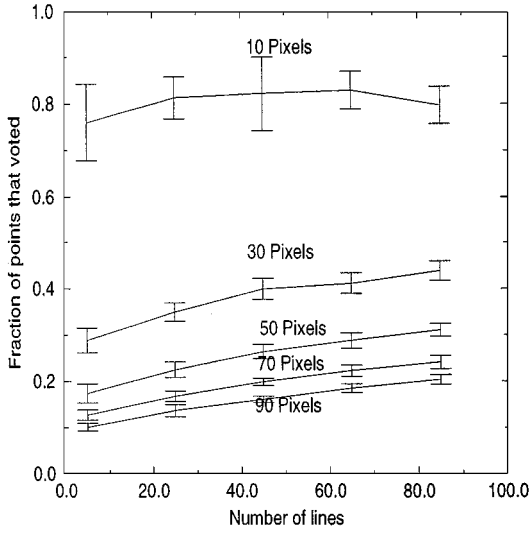


FIG. 4. Effect of line length on number of voting operations.

deviation. Once the line length exceeds a small threshold value, the number of votes required becomes constant. The actual number of voting operations needed to process any image is effectively proportional to the number of lines in the image. This is particularly true if noise points are considered to be small lines in there own right, which is reasonable if the output comes from an edge detector.

Figure 4 shows the fraction of the image points that voted as a function of the number of lines in an image. The slight positive gradient of these lines is due to the accumulator threshold rising as the number of votes it contains increases.

6. EXPERIMENTS ON REAL IMAGES

The aim of the experiments performed on real images was to compare the quality of the output produced by PPHT with that of the standard Hough transform (SHT) and the randomized Hough transform (RHT). The standard Hough transform has been chosen to provide a baseline representing the achievable quality of line detection, whereas the RHT algorithm is representative of voting schemes optimized for computational efficiency. We have adopted public domain implementations of the reference algorithms to ensure that all three algorithms tested would have benefited from a comparable degree of tuning and refinement. The algorithms were run with parameter settings described in Section 3.1.

6.1. Comparison of the Plain PPHT and the SHT

Most SHT implementations (e.g., [11]) exploit gradient direction to increase efficiency of the voting process. The majority of our experiments were carried out using this information; the results are presented in the following section. In this section we compare performance of the plain PPHT and the SHT, letting each edge pixel to vote for all direction. The performance is demonstrated on the HOUSE image (Fig. 12a) which is as a standard benchmark. Results of a number of algorithms applied to it can be found in the literature.

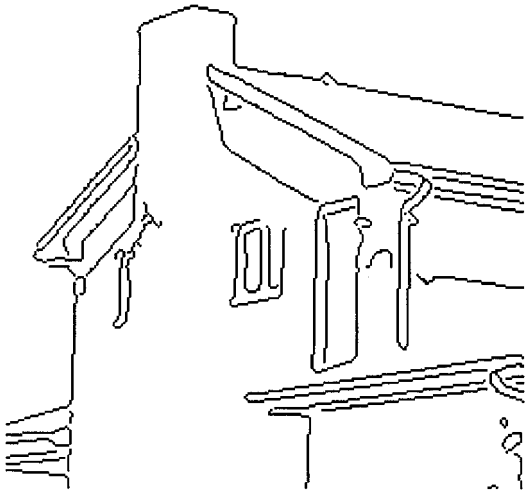


FIG. 5. Input edge image.

Experiment 5. The edge image shown in Fig. 5 is passed as input to both the PPHT and SHT algorithms. Figure 6 depicts the results of the SHT. In Figs. 7 and 8 the results of the PPHT algorithm are shown with low and high values of l, respectively. The number of voting operations used to process each of these images is shown in Table 3.

The main difference that can be seen between these images is due to the combination of two factors. The first is the use of the greedy pixel allocation strategy by these algorithms. The second factor is that the order in which lines are found in the PPHT is much less well defined than that in the SHT. Noise in the accumulator space can mean that shorter lines may be found before longer ones, either because of background pixels, or because chance in the sampling process. This in itself is not incorrect, but because the greedy pixels allocation takes pixels from either end of the shorter line it can interfere with the detection of other,

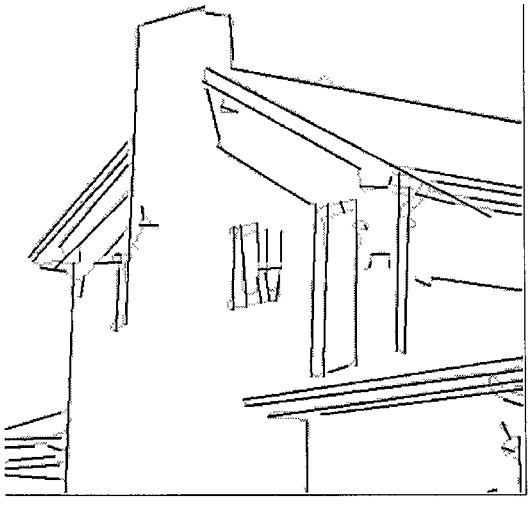


FIG. 6. Results of the standard Hough transform.

TABLE 3
Voting Operations for HOUSE Image

l	Operations
SHT	3120
10^{-9}	1897
10^{-5}	1042

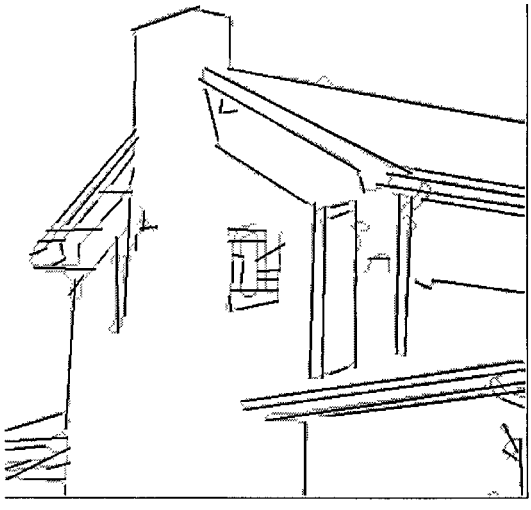


FIG. 7. Results of the PPHT, with a low value for l (10^{-5}).

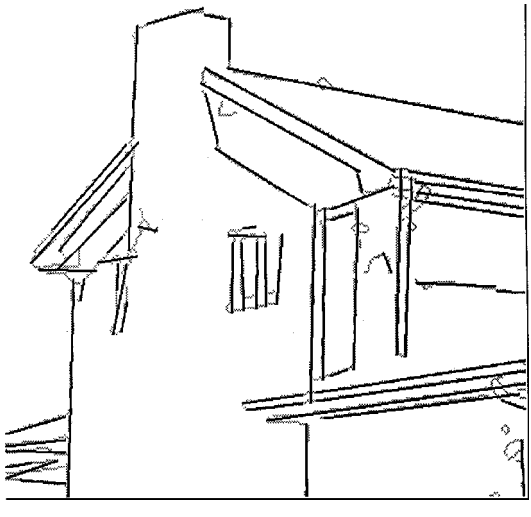


FIG. 8. Results of the PPHT, with a high value for l (10^{-9}).

longer lines which are found by the SHT. This problem is less serious with higher thresholds because the order in which peaks are extracted becomes more deterministic and similar to that found in the SHT.

The other problems inherent in the PPHT with a low false positive threshold can be seen in the lower left corner of Fig. 7. Here false positive lines are found diagonally crossing the actual lines in the image. This problem can also be seen in the output of the SHT, the window in the center of Fig. 6. This problem can be solved by exploiting gradient information.

6.2. Comparison of the RHT, SHT, and PPHT Using Gradient Information

Gradient information can be used in two ways to improve the results of the PPHT. The first is to restrict the range of lines voted for. This reduces the amount of computation required in voting and the amount of clutter in the accumulator space, and hence the number of false positives. Second, gradient information can be used to filter the pixels that are found in the corridor search for each detected line. Only those pixels whose gradient is consistent with the detected line direction are attributed to it. This significantly reduces the variation in the results caused by the undefined order in which lines are recovered using the PPHT. In the experiments described below both options were used. A detailed analysis of the influence of gradient information on the PPHT is given in [10]. Since gradient information was not available on all the test data used, the information was estimated by taking the moments of all the set pixels within a radius of 2.5 grid squares.

Experiment 6. The advocated and the reference algorithms have been tested on five images representing different complexity of line structure and image content (see Fig. 12). The HOUSE image has been presented in the previous section. The ARCH and MACHINE PARTS are images of medium complexity, containing not only straight line segments of different length but also curves. The most testing structures are presented by sets of parallel lines which are separated by a gap of a few pixels only. The OFFICE image is an example of highly complex image where the background structure constitutes a dense clutter. The SPANNER image contains a very simple line structure where the clutter is formed only by the curved segments of the spanner head.

Two criteria were used for the comparison: (i) quality of the extracted lines and (ii) computational efficiency. While the quality of the detected structures was evaluated subjectively, computational efficiency was measured objectively, using different measures. The PPHT and the SHT were compared using the number of votes cast by each method. This is an unbiased measure for these two algorithms, as they perform the same type of operations. Unfortunately, we could not adopt the same approach to measure the computational complexity of the RHT method, as the nature of the operations performed by the algorithm is completely different. For this reasons the RHT's computational complexity was measured in terms of the CPU time and this was related to the same of the PPHT algorithm.

The results on the HOUSE image are shown in Figs. 9, 10, and 11. Comparing the line output produced by the SHT algorithm shown in Fig. 9 with the PPHT output in Fig. 10, we note that they are comparable in quality. It seems that the PPHT algorithm is slightly superior, as it does not tend to favor long lines over short ones. This tendency of the SHT leads to the detection of long virtual lines which are supported by unrelated structures. PPHT has also coped better with the parallel line structures in the bottom right quadrant of the image. It is interesting to note the points used in voting by the PPHT algorithm shown in Fig. 11. The dark points correspond to edge pixels which failed to be interpreted on

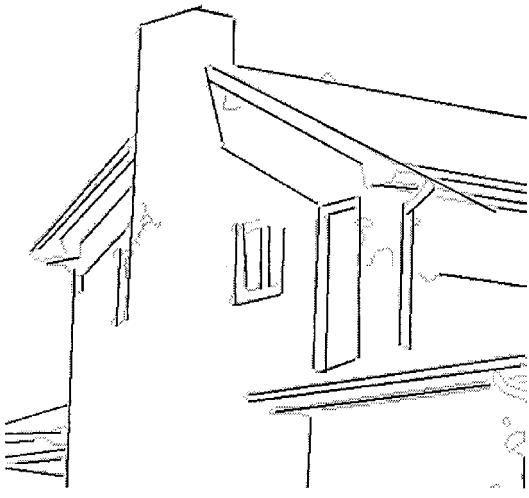


FIG. 9. Results of the SHT. Gradient direction used.

termination of the algorithm when all the pixels were either used in voting or removed from the edge list by virtue of being consistent with the lines already detected. The light gray points are the voting pixels providing evidence for the detected structure. It is apparent, that the number of voting points is considerably lower than the total number of edge pixels used by the SHT. Moreover, the distribution of voting pixels is not uniform. The density of voting pixels for shorter lines is much higher than for longer lines. This shows clearly that the subsampling approach of the probabilistic Hough transform will use an unnecessarily large number of voting points to detect long lines, whereas short lines may end up undetected.

The results obtained on the ARCH, MACHINE PARTS, and OFFICE images (see Fig. 12) clearly demonstrate that in more complex data, the randomized Hough transform tends to detect unacceptable large amount of spurious structure. The PPHT and SHT results are of

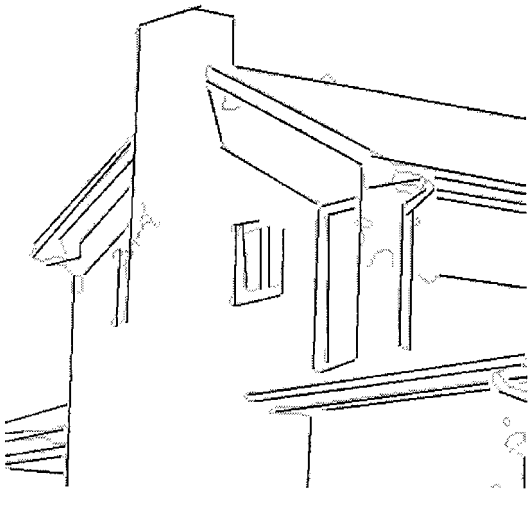


FIG. 10. Results of the PPHT. Gradient direction used.

FIG. 11. Points used in PPHT voting.

similar quality. In some case the tendency of the SHT to form long lines leads to better results, as in Fig. 13, where the two quasi-horizontal lines at the top of the image are detected as single structures, whereas the PPHT breaks the lines up into several models. This tendency works against the SHT in Fig. 14 when modelling the base of the larger machine part. The SHT detects one virtual single line for the base at an angle which allows edge pixels from the two parallel structures to be subsumed by the same model. Even for the simple spanner image (Fig. 16) the RHT did not produce a high quality output. The orientation of the lines detected is inaccurate.

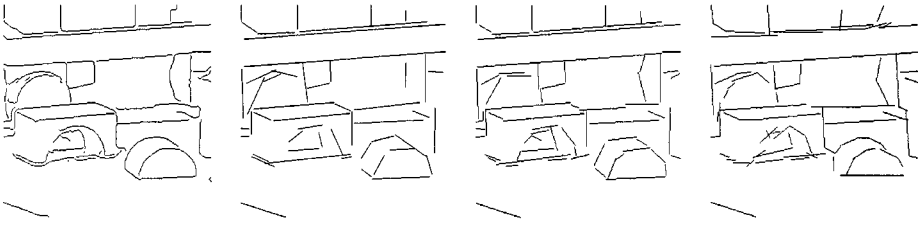
The computational efficiency of the advocated method is illustrated in Table 4. In this table runtimes are used to compare the RHT's performance, as it does not use SHT-style voting. The average savings achieved by the PPHT is about 75% of the time needed to perform the SHT. To a large extent this appears to be independent of image complexity. Table 4 also indicates that the RHT is relatively efficient for simple data. For more complex problems not only does the relative execution time increase but there is also a loss in the quality of its performance. This is particularly visible in Fig. 15, with many of the small features being completely misinterpreted by the RHT. For this image the SHT takes less time to complete the computation than the RHT.

TABLE 4
Computation Requirements for a Variety of Real Images

Image	SHT votes	PPHT votes	Fraction	PPHT time	RHT time
Arch	5345	1186	0.22	0.8	1.6
Part	5347	1439	0.27	1.8	6.2
Spanner	1266	293	0.23	0.2	0.6
Office	36326	8718	0.24	5.0	28.8

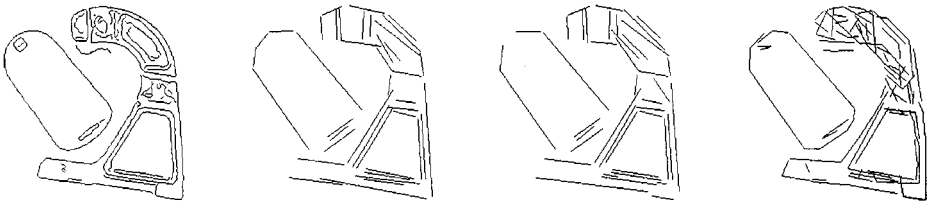
(a) House (b) Arch (c) Machine Parts (d) Spanner (e) Office

FIG. 12. Test images.



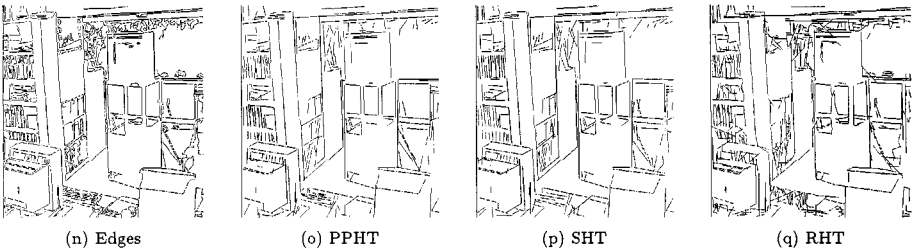
(f) Edges (g) PPHT (h) SHT (i) RHT

FIG. 13. Arch.



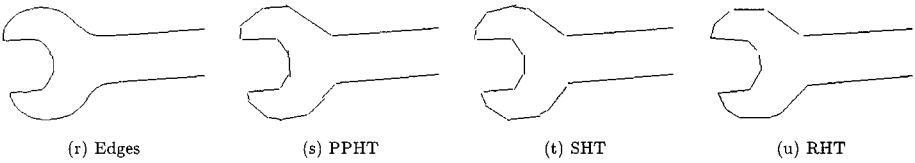
(j) Edges (k) PPHT (l) SHT (m) RHT

FIG. 14. Machine Parts.



(n) Edges (o) PPHT (p) SHT (q) RHT

FIG. 15. Office.



(r) Edges (s) PPHT (t) SHT (u) RHT

FIG. 16. Spanner.

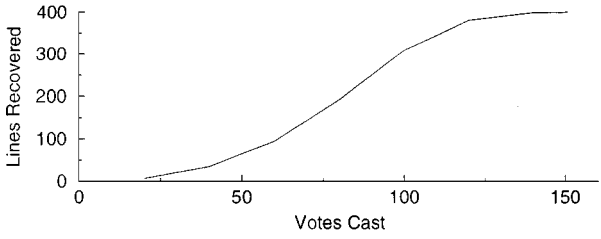


FIG. 17. Histogram of lines found against total number of votes.

7. ACCUMULATOR OCCUPANCY STATISTICS

In this section we investigate the number of votes in the accumulator—accumulator occupancy—and its dynamics during the line detection process. Occupancy becomes important when the accumulator is represented by something other than a fixed size array (e.g., a hash table), which is often the only possibility for higher dimensional problems. In particular, we were interested in exploring the relationship between the total number of votes, the number of lines found, accumulator occupancy, and line length. The experiment was conducted using synthetic images, each composed of 20 lines of random orientation and length between 1 and 100 pixels. The statistics of the above quantities were collected over a sample set of 100 such images. Figure 17 shows the relationship between the number of lines found in this ensemble for a given number of votes per image. As each image contains on average 1000 pixels, it is apparent that the PPHT which detects all the structure with less than 170 votes, about six times faster than the SHT. In 10% of the time taken by the SHT, we detect more than 60% of the structure. Most importantly, from Fig. 18 it follows that this 60% relates to the most salient lines (longest lines). Thus after 10% of the time most of the pixel structure will be interpreted and most important lines segments recovered. The tendency for the PPHT to select long lines first suggests that by terminating the algorithm will always guarantee the recovery of the most salient structure. In fact Fig. 18 shows clearly that short lines start being recovered only after a substantial number of votes have been collected.

The variation of accumulator occupancy with the number of votes cast is shown in Fig. 19. The curve initially increases steeply and linearly with the number of votes. For the SHT this linear relationship would continue until the completion of the accumulation process. In contrast, the PPHT will at some stage start detecting lines and identifying all pixels associated with these lines. The pixels assigned to the recovered lines which have voted will be removed from the accumulator; as a result the accumulator occupancy will

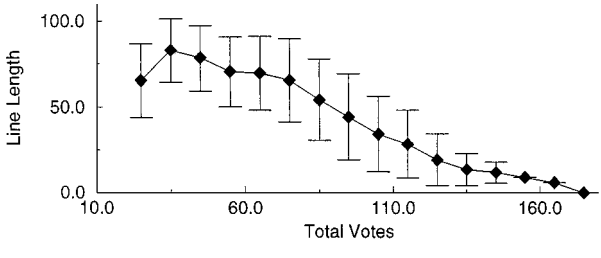


FIG. 18. Votes required to recover lines of different lengths.

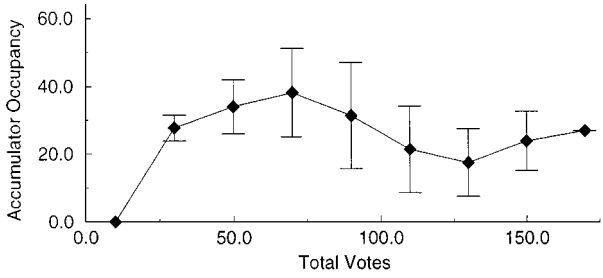


FIG. 19. Changes in accumulator occupancy during the voting process.

be reduced. The pixels that have not yet voted will be removed from the pool of candidate pixels and this is instrumental in accomplishing the claimed computational efficiency gains. Depending on the input data, the accumulator occupancy will rise after votes from noise points and initial votes for a particular line and fall after line detections. The final occupancy will be a function of the saturation value of the line hypothesis test threshold. Thus for the PPHT, with higher confidence levels the threshold will be higher (see Fig. 20) and consequently the terminal accumulator occupancy larger. It is interesting to note that the maximum accumulator occupancy for the tested imagery is less than 5% of that required by the standard Hough transform. This augurs well for the applicability of the proposed algorithm for detecting other parametric curves.

8. CONCLUSIONS

In this paper we presented the *progressive probabilistic Hough transform* (PPHT) algorithm. We showed that unlike the probabilistic HT the proposed algorithm minimizes the amount of computation needed to detect lines by exploiting the difference in the fraction of votes needed to reliably detect lines with different support.

The dependence of the fraction of points used for voting as a function of the inherent complexity of data was studied. We demonstrated on input images containing N lines that the number of votes (speed) of the PPHT is almost independent of line lengths (input points).

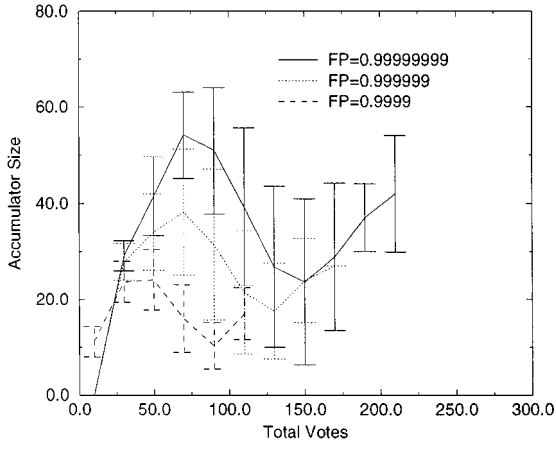


FIG. 20. Effect of varying l on accumulator occupancy as votes are cast.

Though not extensively explored in the paper, the proposed algorithm is sufficiently close to the SHT transform so that many of the methods of improving its performance can be easily adapted to work with it. Use of gradient information was tested as an example. It was shown that a significant improvement can be made in the performance of the algorithm.

The algorithm is ideally suited for real-time applications with a fixed amount of available processing time, since voting and line detection are interleaved. The most salient features are likely to be detected first. Experiments show that the PPHT has, in many circumstances, advantages over the standard HT.

REFERENCES

1. M. Atiquzzaman, Multiresolution Hough transform—An efficient method of detecting patterns in images, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(11), 1992, 1090–1095.
2. J. R. Bergen and H. Shvaytser, A probabilistic algorithm for computing Hough transforms, *J. Algorithms* **12**(4), 1991, 639–656.
3. A. Califano and R. M. Bolle, The multiple window parameter transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(12), 1992, 1157–1170.
4. R. E. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
5. J. Illingworth and J. Kittler, The adaptive Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **9**(5), 1987, 690–698.
6. J. Illingworth and J. Kittler, A survey of the Hough transform, *Comput. Vision Graphics Image Process.* **44**, 1988, 87–116.
7. H. Kalviainen, P. Hirvonen, L. Xu, and E. Oja, Probabilistic and nonprobabilistic Hough transforms—Overview and comparisons, *Image Vision Comput.* **13**(4), 1995, 239–252.
8. N. Kiryati, Y. Eldar, and A. M. Bruckstein, A probabilistic Hough transform, *Pattern Recognition* **24**(4), 1991, 303–316.
9. V. F. Leavers, 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**, 1992, 381–398.
10. J. Matas, C. Galambos, and J. Kittler, *Robust Detection of Lines Using Progressive Probabilistic Hough Transform*, Technical Report VSSP-TR-2/99, University of Surrey, 1999, *available at* ftp://ftp.ee.surrey.ac.uk/pub/vision/papers/matas-TR-VSSP-2-99.ps.Z.
11. P. L. Palmer, J. Kittler, and M. Petrou, Using focus of attention with the Hough transform for accurate line parameter estimation, *Pattern Recognition* **27**(9), 1994, 1127–1133.
12. Soo-Chang Pei and Ji-Hwei Horng, Circular arc detection based on Hough transform, *Pattern Recognition Lett.* **16**, 1995, 615–625.
13. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, Cambridge Univ. Press, Cambridge, UK, 1992.
14. J. Princen, J. Illingworth, and J. Kittler, Hypothesis-testing—A framework for analyzing and optimizing Hough transform performance, *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(4), 1994, 329–341.
15. D. Shaked, O. Yaron, and N. Kiryati, Deriving stopping rules for the probabilistic Hough transform by sequential-analysis, *Comput. Vision Image Understanding* **63**, 1996, 512–526.
16. L. Xu and E. Oja, Randomized Hough transform (RHT)—Basic mechanisms, algorithms, and computational complexities, *CVGIP—Image Understanding* **57**, 1993, 131–154.
17. L. Xu, E. Oja, and P. Kultanen, A new curve detection method: Randomized Hough transform (RHT), *Pattern Recognition Lett.* **11**, 1990, 331–338.
18. A. Yla-Jaaski and N. Kiryati, Automatic termination rules for probabilistic Hough algorithms, in *8th Scandinavian Conference on Image Analysis*, 1993, pp. 121–128.
19. A. Yla-Jaaski and N. Kiryati, Adaptive termination of voting in the probabilistic circular Hough transform, *IEEE Trans. Pattern Anal. Mach. Intell.* **16**(9), 1994, 911–915.