Progressive Probabilistic Hough Transform

J. Matas†‡, C. Galambos† and J. Kittler†

† CVSSP,
University of Surrey,
Guildford, Surrey GU2 5XH,
United Kingdom

‡ Centre for Machine Perception,
Czech Technical University,
Karlovo náměstí 13, 12135 Praha,
Czech Republic
g.matas@ee.surrey.ac.uk

Abstract

In the paper we present the *Progressive Probabilistic Hough Transform* (PPHT). Unlike the Probabilistic Hough Transform [4] where Standard Hough Transform is performed on a pre-selected fraction of input points, PPHT minimises 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 Hough Transform; 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 is interleaved. The most salient features are likely to be detected first. Experiments show PPHT has, in many circumstances, advantages over the Standard Hough Transform.

1 Introduction

The Hough Transform (HT) is a popular method for the extraction of geometric primitives. In literally hundreds of papers every aspect of the transform has been scrutinised - parameterisation, accumulator design, voting patterns, peak detection - to name but a few [2]. The introduction of the randomised version of the Hough Transform is one of the most important recent developments [7], [4] in the field. The approach proposed in the paper falls in the 'probabilistic' (or Monte Carlo) class of Hough Transform algorithms, the objective of which is to minimise the proportion of points that are used in voting while maintaining false negative and false positive detection rates almost at the level achieved by the Standard Hough Transform [5]. Unlike the Randomised Hough Transform (RHT) class of algorithms (see [3] for an overview of the Hough Transform), the Probabilistic Hough Transform (PHT) and the Standard Hough Transform (SHT) share the same one-to-many voting pattern and the representation of the accumulator array.

In the original paper on the Probabilistic Hough Transform [4], Kiryati *et al* . show that it is *often* possible to obtain results identical to SHT if only a fraction p of input points is used in the voting process. In the first step of PHT a random subset of points is selected and a Standard Hough Transform is performed on the subset. Successful experiments with p as low as 2% are presented. The poll size is a parameter critically influencing the PHT performance. The authors analyse the problem for the case of a single line immersed in noise. Unfortunately the derived formulae require the knowledge of the number of points belonging to the line *a priori*, which is rare in practice. In [1], (Bergen and Schvaytzer) it is shown 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 be bypassed by an adaptive scheme [9, 8, 6]. 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 [9] is based on a measure of stability of the ranks of the highest peaks. Shaked *et al.* [6] propose a sequential rule. Both methods formulate their goal so as to "obtain the same maximum as if the Hough Transform had been accumulated and analysed". In our opinion, such 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 the paper we present a new form of an Adaptive Probabilistic Hough Transform. Unlike the above-mentioned work we attempt to minimise 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 [4]) that for lines with strong support (long lines) only a small fraction of its supporting points have to vote before the corresponding accumulator bin reaches a count that is non-accidental. 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.

2 The Algorithm

To minimise the computation requirements the proposed algorithm which we call *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:

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

'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 s points which would be expected from random noise?'. The test requires a single comparison with a threshold per bin update. Of course, the threshold s 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 an algorithm possesses a number of attractive properties. Firstly, a feature is detected as soon as the contents of the accumulator allows a decision. The PPHT 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 have either 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 selecting a point for voting.

Without a stopping rule the PPHT and the Standard Hough Transform differ only in the number of false positives. False negatives – missed features with respect to the Standard Hough Transform – should not pose a problem, because if the feature is detectable by SHT it will be detected by PPHT at the latest when the voting finished when the corresponding bin counts of PPHT and SHT are identical. PPHT and SHT performance differs from the point of view of false positives, only where assignment of points to lines is ambiguous, i.e. where points are in the neighbourhood of more than one line.

In our experiments a straightforward one-to-many voting scheme was used. The proposed method reduces computation by minimising 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 available) do not interfere with the benefits of the method.

In setting the decision threshold we assume that all points are due to noise. It 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.

Since every pixel votes into one bin with a given value of θ, we can focus on the analysis of vote distribution along the ρ axis of the accumulator. If we adopt the assumption that a vote into any bin is equally likely[2], then the distribution of points in the m bins is multinomial. If we used the multinomial distribution to make a detection decision, we would have to monitor all m bins, which is computationally impractical. We therefore assume that counts in the ρ bins are independent, with the probability of a vote falling into a given bin being $1/m$. Again, this is clearly an approximation, since counts in all bins must add to N, the number of votes cast so far. Under the simplifying assumptions the number of votes in a single bin will follow the binomial distribution. Since $Np = N/m$ is of the order of 1 and $p = 1/m$ much less than 1 the binomial distribution is well approximated by a Poisson distribution.

Here is an outline of the algorithm used:

1. Check the input image, if it is empty then finish.

[2]We are aware of the fact that this assumption is most likely not correct, since it corresponds to spatially non-uniform distribution of noise points

Method	SHT				PPHT			
Lines	FP	σ	FN	σ	FP	σ	FN	σ
2	0.08	0.27	0.08	0.27	0.01	0.10	0.00	0.00
4	0.36	0.67	0.36	0.67	0.12	0.46	0.06	0.24
6	1.07	1.22	1.06	1.20	0.37	0.81	0.19	0.44
8	2.56	1.68	2.56	1.68	1.38	1.30	0.90	1.01
10	4.00	1.84	3.98	1.83	2.28	1.90	1.52	1.48
12	6.44	2.07	6.40	2.06	3.79	2.15	2.78	1.99
14	8.13	2.15	8.03	2.11	5.94	2.21	4.48	2.49
16	10.90	2.06	10.86	2.02	8.35	2.23	6.66	3.07
18	13.36	2.03	13.32	2.07	9.86	2.77	8.16	3.51
20	16.12	1.93	16.04	1.88	12.74	2.37	11.56	4.09

Table 1: Effect of image clutter on false positives (FP) and negatives (FN). Averages and standard deviations (σ) over 100 runs are shown.

2. Update the accumulator with a single pixel randomly selected from the input image.
3. Remove pixel from input image.
4. Check if the highest peak in the accumulator that was modified by the new pixel is higher than threshold l. If not then goto 1.
5. Look along a corridor specified by the peak in the accumulator, and find the longest segment of pixels either continuous or exhibiting 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.

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 (mis-detected lines). In real images the definition of what should and should not be detected as a line is subjective or application-dependent. Since PPHT is not designed for a specific application we illustrate its performance on real imagery by comparing it with Standard Hough Transform [5] (Section 3.4). The tests designed to assess the correctness (quality) of PPHT are presented in section 3.2. In the limited space 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. The rest of the experiments test the computational efficiency of the PPHT (section 3.3).

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 minimise any differences between the SHT and PPHT implementations. The implementations were kept as simple as possible, with minimal post and pre-processing (use of gradient information, connectivity etc.). Most standard enhancements of SHT are directly applicable to PPHT and do not therefore change the relative merits of the methods. In the implementation, the standard ρ, θ line parameterisation 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 infinite straight lines, but we want to detect finite line segments, a single post-processing 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 0.99999.

3.2 Correctness of line detection using PPHT.

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 post processing is used to obtain line segments for the infinite lines found by the Hough Transform. If edge pixels are allowed to be assigned 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 percent by detected lines, excluding those counted as false positives.

Experiment 1. A hundred images containing a fixed number of randomly positioned lines were syntheticaly generated, the only source of noise being in the digitisation of the lines themselves. Figure 4 shows a typical image used in the experiment, the image resolution was 256 both horizontally and vertically, the lines were hundred pixels long. The number of lines varied between 2 and 20. Both SHT and PPHT were run on all images, the results of this experiment are summarised in table 1. These measurements of false negatives and positives where counted as described at the beginning of this section.

The results in table 1 show that PPHT outperforms the Standard Hough Transform in all the cases. Both the number of false positives and the number of false negatives for PPHT were lower than those given by SHT. This is an unexpected result. PPHT uses a fraction of SHT votes, but this should reduce, in controlled way, the average correctness of 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.

l	FP	σ	FN	σ	votes	σ
0.9	0.34	0.73	0.39	0.82	46.88	9.70
0.99	0.25	0.70	0.36	0.76	46.68	9.80
0.999	0.13	0.34	0.27	0.68	55.47	8.89
0.9999	0.17	0.55	0.25	0.59	57.27	8.80
0.99999	0.17	0.43	0.36	0.70	72.73	12.15
0.999999	0.11	0.31	0.25	0.61	84.63	13.19
0.9999999	0.16	0.42	0.39	0.79	93.06	15.20
0.99999999	0.07	0.26	0.33	0.65	108.62	12.18

Table 2: Influence of significance level on error rate and run time. Averages and standard deviations over 100 tests are shown.

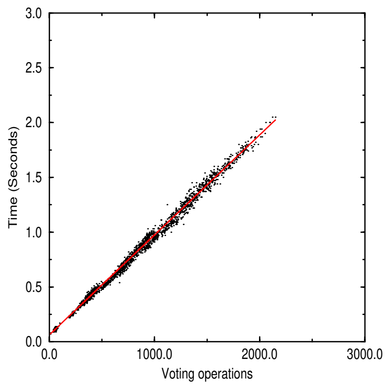

Figure 1: Run time as a function of voting operations. The correlation coefficient is 0.998.

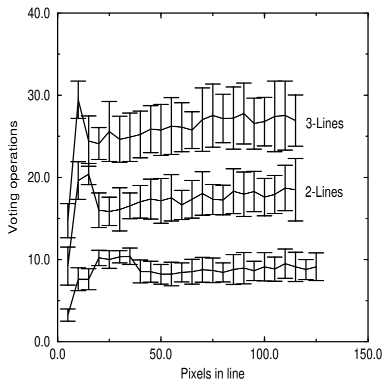

Figure 2: Votes needed to find a small number of lines

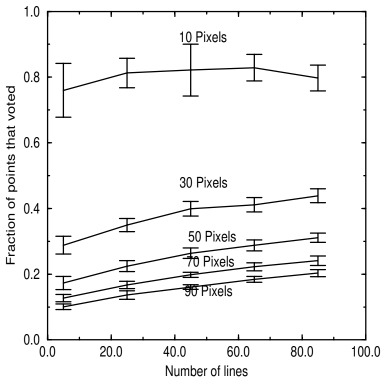

Figure 3: Effect of line length on number of voting operations.

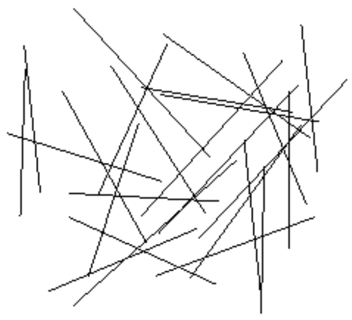

Figure 4: Example synthetic image.

Experiment 2. The parameter l of 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 256x256 pixels each with 5 lines of 100 pixels. Increase in the significance level for line detection leads to a decrease in number of false positives at the cost of increasing the number of voting operations required, which as will be shown in the next section, is proportional to run time.

3.3 Computational efficiency

We have argued that the advantage of PPHT lies in its ability to minimise 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.
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 figure 1 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 2 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 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 their own right, which

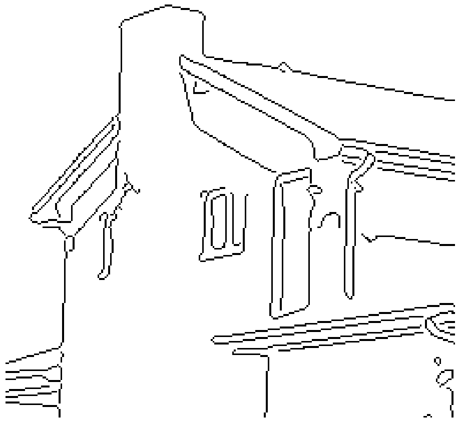
Figure 5: Input edge image.

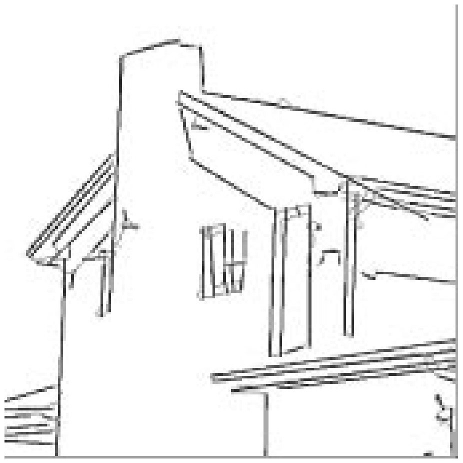
Figure 6: Results of Standard Hough Transform.

is reasonable if the output comes from an edge detector.

Figure 3 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.

3.4 Experiments on real images

Finally we looked at the results of processing real images. The starting point is the edge image shown in figure 5. In figure 6 the results of the SHT are shown. In figures 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 routine used by these programs and the second factor is that the order in which lines are found in PPHT is much less well defined than in SHT. This means short lines may be found before longer ones. 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, longer lines which are found by SHT. This problem is less serious with higher values of l because the

l	Voting operations
SHT	3120
0.999999999	1897
0.99999	1042

Table 3: Voting operations for house image.

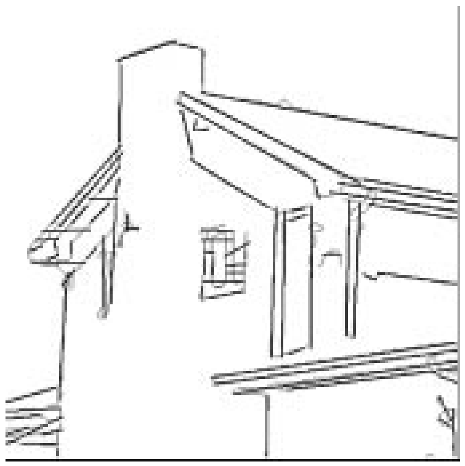
Figure 7: Results of PPHT, with a low value for l (0.99999).

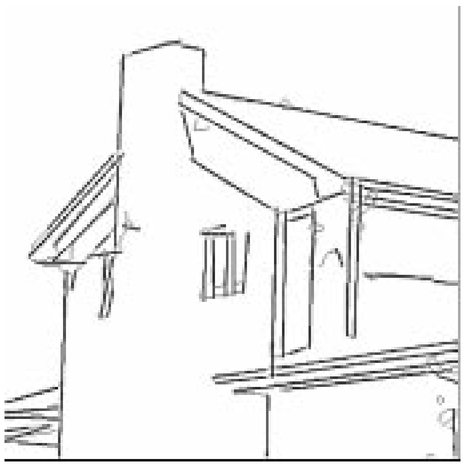
Figure 8: Results of PPHT, with a high value for l (0.999999999).

number of votes required to find a line increases, and so the order of detection becomes more deterministic.

The other problems inherent to PPHT with a low false positive threshold can be seen in the lower left corner of figure 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 centre of figure 6. The problem shall be solved by exploiting gradient information, a common technique used to enhance SHT.

4 Conclusions

In the paper we presented the *Progressive Probabilistic Hough Transform* algorithm. We showed that unlike the Probabilistic Hough Transform the proposed algorithm minimises 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 is a function of the inherent complexity of data. We demonstrated on input images containing N lines that the number of votes (speed) of PPHT is almost independent of line lengths (input points).

The post-processing used for the experiments presented in this paper is essentially the same as that used for SHT. The ordering of recovered lines in PPHT is not as well defined as in case of SHT. Although this loss of ordering will lead to some undesirable results around line intersections when using standard SHT post processing, it is easy to conceive effective schemes which would eradicate this problem. One such scheme could use edge pixel gradients to choose when to add a pixel into a line. Using gradient information when accumulating would also speed up the voting process, and serve to reduce further the clutter in the accumulator space.

References

[1] JR Bergen and H Shvaytser. A probabilistic algorithm for computing Hough Transforms. *Journal Of Algorithms*, 12(4):639–656, 1991.

[2] J. Illingworth and J. Kittler. A survey of the Hough Transform. *Computer Vision, Graphics and Image Processing*, 44:87–116, 1988.

[3] H Kalviainen, P Hirvonen, L Xu, and E Oja. Probabilistic and nonprobabilistic Hough Transforms - overview and comparisons. *Image And Vision Computing*, 13(4):239–252, 1995.

[4] N Kiryati, Y Eldar, and AM Bruckstein. A probabilistic Hough Transform. *Pattern Recognition*, 24(4):303–316, 1991.

[5] J Princen, J Illingworth, and J Kittler. Hypothesis-testing - a framework for analyzing and optimizing Hough Transform performance. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 16(4):329–341, 1994.

[6] D Shaked, O Yaron, and N Kiryati. Deriving stopping rules for the Probabilistic Hough Transform by sequential-analysis. *Computer Vision And Image Understanding*, 63(3):512–526, 1996.

[7] L Xu and E Oja. Randomized Hough Transform - basic mechanisms, algorithms, and computational complexities. *CVGIP-image Understanding*, 57(2):131–154, 1993.

[8] Antti Yla-Jaaski and Nahum Kiryati. Automatic termination rules for probabilistic hough algorithms. In *8th Scandinavian Conference on Image Analysis*, pages 121–128, 1993.

[9] A Ylajaaski and N Kiryati. Adaptive termination of voting in the probabilistic circular hough transform. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 16(9):911–915, 1994.