Technical Report - Randomized Hough Transform: Improved Ellipse Detection with Comparison

Robert A. McLaughlin

***Abstract*— We describe an algorithm for the detection of ellipse shapes in images, using the Randomized Hough Transform. The algorithm is compared to a standard implementation of the Hough Transform, and the Probabilistic Hough Transform. Tests are performed using both noise-free and noisy images, and several real-world images. The algorithm was found to give improvements in accuracy, and a reduction in computation time, memory requirements and the number of false alarms detected.**

Keywords— **Randomized Hough Transform, Hough Transform, Probabilistic Hough Transform, Ellipse Detection.**

I. Introduction

Image processing programs are frequently required to detect ellipses. Examples include cell counting in the detection of breast cancer, and particle classification in an industrial setting. A standard algorithm for this detection is the Hough Transform [1], or its variant the Probabilistic Hough Transform [2] [3]. Both of these algorithms are slow, memory intensive and have a limited accuracy as the number of ellipses in the image increases.

These algorithms have also been used for the detection of lines and circles. Here they suffer the same flaws but to a lesser degree. To alleviate these problems, the Randomized Hough Transform (RHT) [4] [5] [6] [7] was introduced. Unfortunately, RHT becomes impractical when the object to be detected is defined as a non-linear equation of parameters. Ellipses are one such object.

This paper presents an algorithm which reduces this to a linear problem. This allows the use of RHT for ellipse detection. We test the algorithm against the standard Hough Transform and the Probabilistic Hough Transform, with both synthetic and real-world images.

In this paper we work with black images on a white background. The term 'pixel' refers to a black pixel.

II. Randomized Hough Transform

RHT was originally described in [4]. For ease of reference, we briefly explain the algorithm here.

Let us begin by considering RHT as used to detect lines in an image. We shall later generalise this to ellipse detection. As with the standard Hough Transform, we define a parameter space for lines, such that each line in the image maps to a single point in the parameter space. We shall generate a histogram over this parameter space so that maxima of the histogram will correspond to lines in the image. The difference between RHT and the standard Hough Transform lies in how this histogram is generated.

Robert A. McLaughlin is with the Centre for Intelligent Information Processing Systems, The University of Western Australia, Nedlands W.A. 6907, Australia. E-mail: ram@ee.uwa.edu.au

In the standard Hough Transform [8] [9], a pixel in the image generates a curve in the parameter space. A curve is generated for every pixel and these curves are discretised and recorded in the histogram. In the Probabilistic Hough Transform, the image is sub-sampled and curves are generated for only a proportion of the pixels in the image.

In the RHT, we begin with an empty histogram. We randomly choose two pixels in the image and compute the unique line passing through them. The parameters of this line are then recorded as an entry in the histogram. We repeat this process, randomly choosing a new pair of pixels. This is iterated some preset number of times, where the number of iterations is much less than the number of pixel pairs in the image. In this way, we accumulate entries in the histogram.

Applying RHT to circle detection requires that we find the unique circle passing through a triplet of pixels. A circle is defined as the solution to the equation

$$(x-p)^2 + (y-q)^2 = r^2$$

with the restriction that $r > 0$. This is non-linear with respect to the circle parameters (p, q, r). Finding the unique circle that passes through three given pixels will require that we solve a system of three simultaneous non-linear equations. To quote Xu et al. [4, pp 333–334]

> ... for curves expressed by equations which are nonlinear with respect to the parameters, the RHT cannot be directly used.

This can be reduced to a linear problem by making use of a feature of the geometry of circles. The details are in [4].

An ellipse is defined as the solution to the equation

$$a(x-p)^2 + 2b(x-p)(y-q) + c(y-q)^2 = 1 \qquad (1)$$

with the restriction that $ac - b^2 > 0$. This is non-linear with respect to the parameters (p, q, a, b, c). To find the unique ellipse passing through a 5-tuple of pixels in the image would require that we solve a system of 5 simultaneous non-linear equations. Having to solve this system of non-linear equations for each 5-tuple of pixels chosen will slow the RHT and render it impractical.

III. Algorithm

Given an n-tuple of pixels in the image, we need to compute the parameters of the ellipse passing through these pixels. For RHT to be practical, it is important that we reduce this to a linear problem. The following algorithm achieves this.

Choose three pixels from the image and obtain an estimate of the tangent at each pixel. This is done by defining

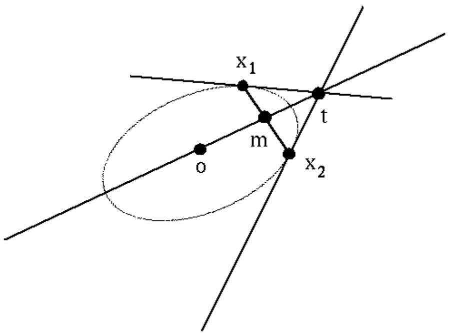

Fig. 1. Ellipse geometry. o = ellipse centre.

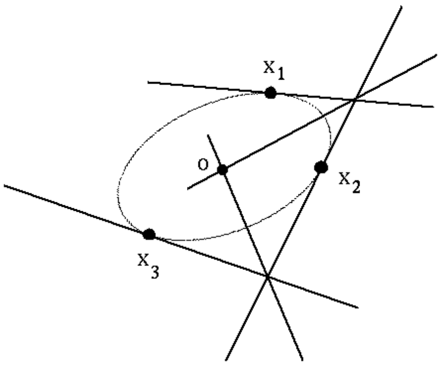

Fig. 2. Estimate of ellipse centre. o = ellipse centre.

a small neighbourhood around the pixel and finding the line of best fit to those pixels within the neighbourhood (using least squares method). The three pixels $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$ and the estimates of tangent form the input to the algorithm. Computing parameters of the corresponding ellipse will be organised into two sections: finding the centre, and finding the remaining three parameters.

To find the ellipse centre we use a feature of ellipse geometry noted by Yuen et al. [1]. Take two points on an ellipse and find their midpoint $\mathbf{m}$, and the intersection of their tangents $\mathbf{t}$. The centre of the ellipse will lie on the line $\overline{\mathbf{tm}}$ (Fig. 1). To understand this, first consider it for the simple case of a circle. Then note that the transformation mapping a circle onto an ellipse is an affine transformation, which maps lines to lines, tangents to tangents and intersections to intersections.

We compute this line using pixels $\mathbf{x_1}$ and $\mathbf{x_2}$, then again with pixels $\mathbf{x_2}$ and $\mathbf{x_3}$ (Fig. 2). The intersection of the two lines gives us an estimate of the ellipse centre.

We next estimate the remaining three ellipse parameters. First translate the ellipse to be centred on the origin. This reduces equation (1), the general equation of an ellipse, to the following:

$$ax^2 + 2bxy + cy^2 = 1$$

Substituting in the coordinates of the three pixels ($\mathbf{x_1} = (x_1, y_1), \mathbf{x_2} = (x_2, y_2), \mathbf{x_3} = (x_3, y_3)$), we obtain the following system of three simultaneous linear equations:

$$\begin{bmatrix} x_1^2 & 2x_1y_1 & y_1^2 \\ x_2^2 & 2x_2y_2 & y_2^2 \\ x_3^2 & 2x_3y_3 & y_3^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Solving this for a, b, c gives the remaining three ellipse parameters.

Next, we check the inequality $ac - b^2 > 0$ which is true if the parameters represent a valid ellipse. If this inequality is false, it implies that either the three pixels do not lie on the same ellipse, or that the estimates of tangent were inaccurate. In either case, we discard the parameters and choose a new pixel triplet.

Finally, we convert from the parameters (p, q, a, b, c) to the parameters (p, q, r_1, r_2, θ) where r_1, r_2 are the major and minor radii of the ellipse (respectively) and θ is the angle of the major axis. From experiment, we found this parameter space to be better behaved than (p, q, a, b, c). That is, the points representing a range of ellipses were spread more evenly throughout the parameter space.

Note that we only require a triplet of pixels to define an ellipse, and not a 5-tuple. The required extra information is contained in the estimates of tangent.

IV. Implementation Details

In this section, we describe certain implementation details of RHT. These were introduced in [4].

The histogram typically found in Hough algorithms is replaced with a linked list structure. Let us refer to the elements of the linked list as cells. Each cell comprises two parts:

- parameter values for a particular ellipse.
- a count.

At the commencement of RHT, the linked list is initialised to be empty. After each n-tuple of pixels is chosen, and parameters of the associated ellipse have been computed, the linked list is searched for a cell whose parameters are the same to within some tolerance. If such a cell is found, then its count is incremented by one. We also update the cell's parameters as a weighted average of its old parameters (weighted by the cell's count) and the parameters of the current ellipse (weighted as 1). If no such cell is found, a new cell is added to the linked list. This cell contains the parameters of the current ellipse and a count initialised to one.

To improve search time, we have organised the linked list into a tree structure. This tree structure is illustrated in Fig. 3. Recall that each cell contains the five parameters of an ellipse. Call them parameters #1, #2, #3, #4 and #5. The cells are ordered upon the value of parameter #1. If two cells contain the same value for parameter #1 (to within some coarse quantisation), then these cells form a branch off from the linked list. This branch is a new linked list, ordered upon the values of parameter #2. If two elements of this branch have the same value for parameter #2 (to within some coarse quantisation), they in turn form

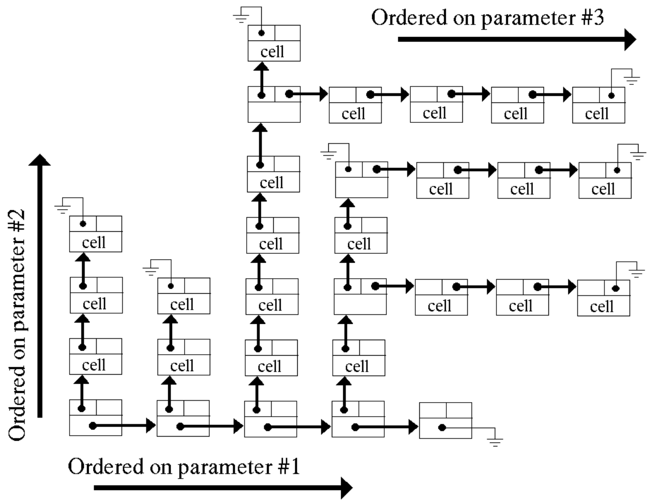

Fig. 3. Tree structure.

a new branch off the new linked list, which is itself a linked list ordered on the values of parameter #3. This continues, until we have a branch ordered on the values of parameter #5.

This tree structure has proven to be more efficient than the 5-D histogram which would otherwise be required. The tree structure contains only entries for those regions of the ellipse parameter space which are used by the current image. In contrast, a histogram would contain many entries that are set to zero. Also, when using a histogram, it is necessary to decide ahead of time which region of the parameter space the histogram will cover. The larger this region, the more memory that is required. The smaller this region, the less likely it is to include those ellipses present in the image. In comparison, implicit in the tree structure is that it can accept entries from anywhere in the parameter space.

Because RHT is much faster than both the standard Hough Transform and the Probabilistic Hough Transform, it is practical to apply it several times to a given image. Each time we shall extract a single ellipse from the image. Xu et al. [4] suggested the use of an 'epoch' to implement this. In one epoch, we choose a preset number of triplets of pixels and map each to a point in the ellipse parameter space. These are recorded in the tree structure. At the completion of each epoch, we find the cell in the tree structure with the maximum count. Having thus found the parameters of an ellipse, we apply a simple form of post-processing validation. This involves counting the number of pixels lying on (or near) the ellipse. This number is normalised by dividing it by the ellipse's perimeter. If the resulting fraction is above some threshold value, the ellipse is judged to exist in the image. We then remove from the image those pixels lying near this ellipse. This will simplify the image. The tree structure is re-initialised to be empty and RHT is subsequently applied to the simplified image. If some preset number of epochs pass without successfully finding an ellipse, then the process is halted.

V. Standard Hough Transform

For the purposes of comparison, we have implemented the standard Hough Transform (SHT) for ellipse detection, as described by Yuen et al. [1], with minor modifications.

As is done with RHT, Yuen et al. organised ellipse detection into two stages:

- find ellipse centres.
- find the other three parameters.

To find ellipse centres, the algorithm used the feature of ellipse geometry that was illustrated in Fig. 1. Two pixels were chosen and local estimates of tangent were found. The midpoint $\mathbf{m}$ of the pixels and the intersection of the tangents $\mathbf{t}$ were then computed. A 2-D histogram was defined over the x-y plane of the image. The line $\overline{\mathbf{tm}}$ was recorded in this histogram by incrementing each histogram element that the line passed through. This process was then repeated with every pair of pixels in the image (with the exception of those pairs whose tangents were near parallel or near perpendicular). Local maxima of this histogram were identified as possible ellipse centres. It should be noted that the number of lines to be entered in the histogram increases approximately as a square of the number of pixels in the image. Thus computing this histogram consumed a majority of the overall computation time of this ellipse finding algorithm.

In Hough algorithms, the method used to find local maxima has a very important impact upon the algorithm's speed, accuracy and reliability. The following method was used in this implementation.

Begin by averaging the 2-D histogram. Define an $n \times n$ window, which we shall gradually move over the entire histogram. Our aim is to decide whether the central value of the window in a local maxima. Call this central value $value_c$. The window size n specifies both the minimum distance between local maxima, and the algorithm's tolerance to noise. A value of $n = 40$ was found to work well for an image size of 800×800 (width $\times$ height). Recall that for the current application, an 800×800 image size implies an 800×800 histogram.

Exhaustively testing $value_c$ against every histogram entry within the $n \times n$ window would be very computationally intensive, especially as the window is moved pixel by pixel over the entire histogram. Thus we first apply a series of tests in which we sub-sample the histogram entries within the window. This allows us to quickly discard most entries which are not local maxima.

We first test whether $value_c$ is greater than some minimum threshold value. Next we sub-sample those histogram values that are within the window. We found that sampling $\frac{1}{10}$ of the values in the window was sufficient. We then tested whether $value_c$ was greater than each of the sampled values. These sampled values were subsequently used to form an estimate of the average histogram value and it was tested whether $value_c$ was notably larger than this average value (we set a threshold of twice the estimated average in experiments 1, 2 and 4, and 1.2 times the estimated average in experiment 3.). If all of these tests succeeded, we finally exhaustively compared $value_c$ to all other histogram values within the window. If $value_c$ was found to be the maximum value in the window, it was recorded as a local maxima. All other entries in the window were then marked as ineligible to become local maxima and were not searched. This algorithm was found to work very effectively. The computational demands were negligible compared to the computational demands of computing the 2-D Hough Transform.

To determine the remaining three ellipse parameters, we have used a 3-D Adaptive Hough Transform [10]. The three ellipse parameters used were r_1, r_2, θ, where r_1, r_2 are the major and minor radii of the ellipse (respectively), and θ is the angle of the ellipse's major axis. This is a different parametrisation to that used by Yuen et al. [1], but we judge this to be a minor modification.

At the completion of the Adaptive Hough Transform, we will have the parameters of possible ellipses in the image. For each ellipse, we count the number of pixels near the ellipse and normalise this by dividing by the ellipse's perimeter. If the resulting fraction is above a threshold value, then the ellipse is judged to exist, otherwise it is discarded.

VI. Probabilistic Hough Transform

The Probabilistic Hough Transform (PHT) is detailed in [2] [3]. In contrast to the standard Hough Transform, which applies a Hough transform to every pixel in the image, PHT does so for only a proportion of the pixels in the image. Let us call this proportion α where $0 < \alpha < 1$. In this paper, we have applied PHT to the algorithm described in Section V.

The work of Kiryati et al. [2] suggested that there exists a threshold value for α at which the histogram produced by PHT is very similar to that produced by the standard Hough Transform. Thus above this threshold the accuracy of PHT is approximately that of the standard Hough Transform. Below this threshold, the accuracy of PHT decreases rapidly. Their results suggested this threshold to be in the range $2\% < \alpha < 15\%$, although it was noted that this was very dependent upon the specific application. In this implementation, we have set $\alpha = 15\%$.

Recall that the method for finding ellipse centres that was described in Section V required that we apply a Hough Transform to pairs of pixels. PHT will reduce the computational intensity of this stage by approximately $\frac{1}{\alpha^2} \approx$ 45 times. Finding the other three parameters involved computing a Hough Transform for individual pixels, thus PHT has a computational advantage of $\frac{1}{\alpha} \approx 7$ times.

VII. Experiment

In this section, we describe the experiments performed to compare the different Hough algorithms. We compare these algorithms for accuracy, computation time, memory requirements and tolerance to noise. Four different experiments were performed.

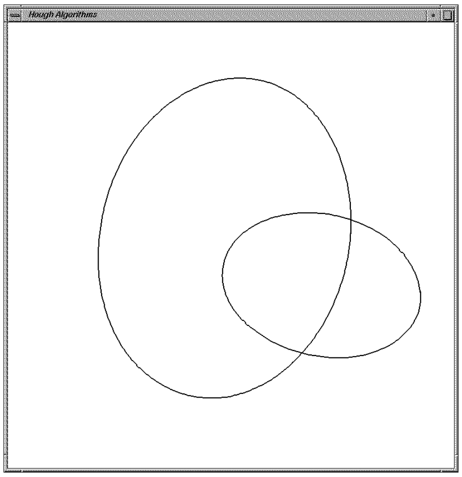

Fig. 4. Image containing 2 ellipses.

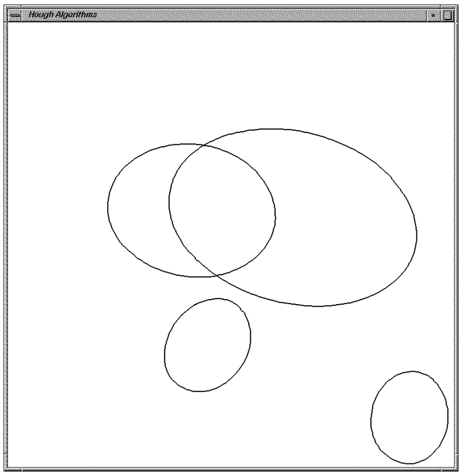

Fig. 5. Image containing 4 ellipses.

A. Experiment 1: Multiple Ellipses.

For the first experiment, a simple drawing program was written, allowing a person to draw ellipses on the screen using a mouse. A collection of two hundred images were produced, twenty of which contained one ellipse, twenty with two ellipses, etc. up to twenty images with ten ellipses. Typical images are shown in Fig. 4 - 8. The ellipses often intersected. We recorded:

- the number of ellipses correctly found,
- the number of false alarms (when the algorithm found a non-existent ellipse),
- the processing time,
- the memory usage

as a function of the number of ellipses in the image.

B. Experiment 2: Additional Curves.

In the second experiment, we tested the tolerance of the algorithms to curve noise. The same drawing program was used to produce a new collection of two hundred images, each containing a single ellipse and a varying number of free-hand drawn curves. Twenty of the images contained

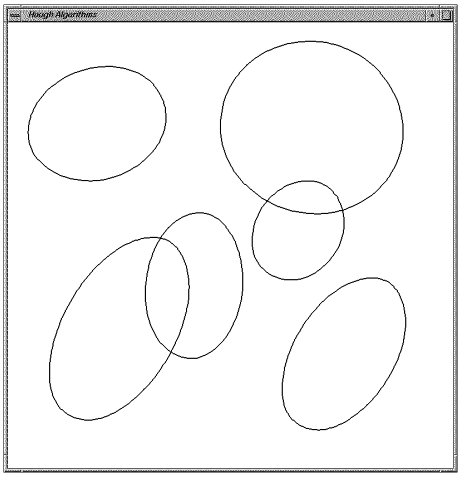

Fig. 6. Image containing 6 ellipses.

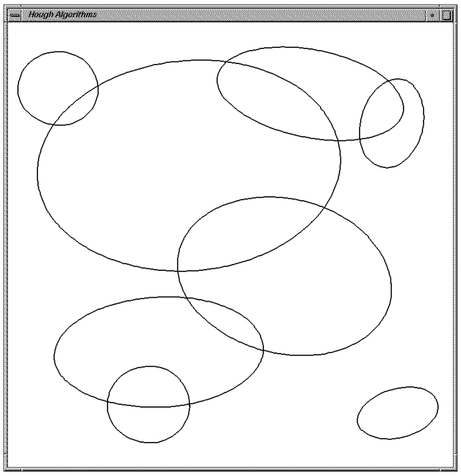

Fig. 7. Image containing 8 ellipses.

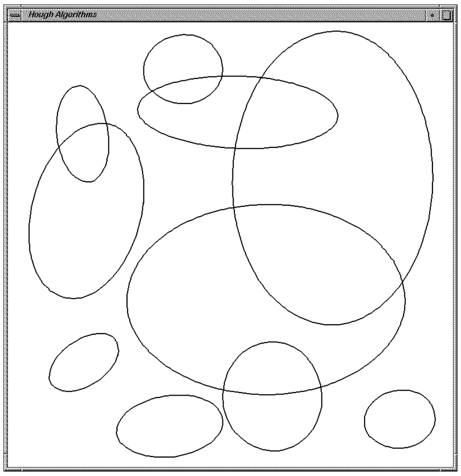

Fig. 8. Image containing 10 ellipses.

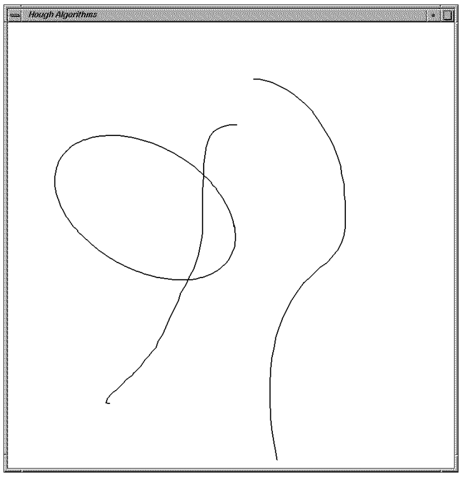

Fig. 9. Image containing 1 ellipse and 2 additional curves.

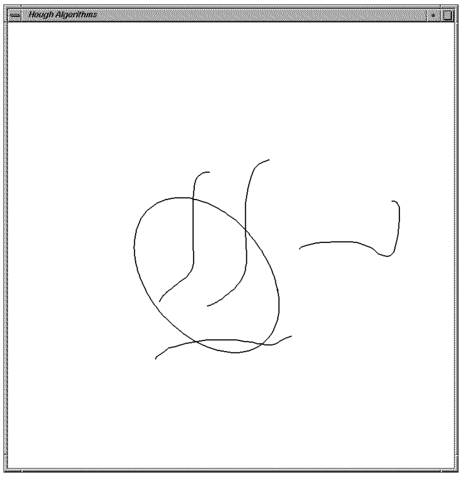

Fig. 10. Image containing 1 ellipse and 4 additional curves.

a single curve, twenty contained two curves, etc. up to twenty images with ten curves. Typical images are shown in Fig. 9 - 13. We recorded:

- how often the single ellipse was found,
- the number of false alarms

as a function of the number of additional curves present in the image.

The drawing program and the Hough algorithms used in Experiment 1 and Experiment 2 are available on the WWW from:

```
http://ciips.ee.uwa.edu.au
  /Papers/Technical_Reports/1997/01/Index.html
```

Note that the code which has been made available on the WWW is a later version than that used in the experiments detailed here.

In both Experiments 1 and 2, an images size of 800×800 pixels (width $\times$ height) was used.

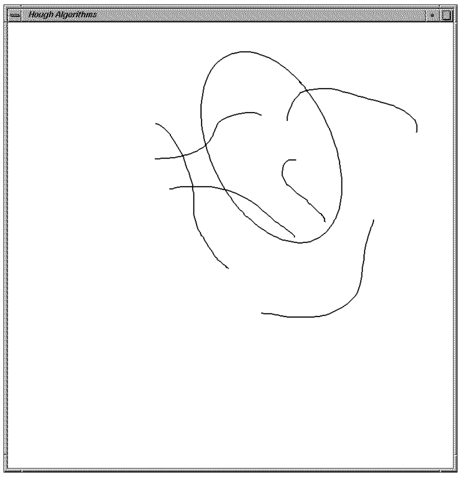

Fig. 11. Image containing 1 ellipse and 6 additional curves.

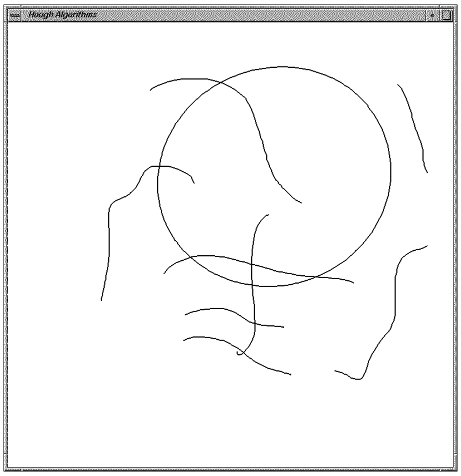

Fig. 12. Image containing 1 ellipse and 8 additional curves.

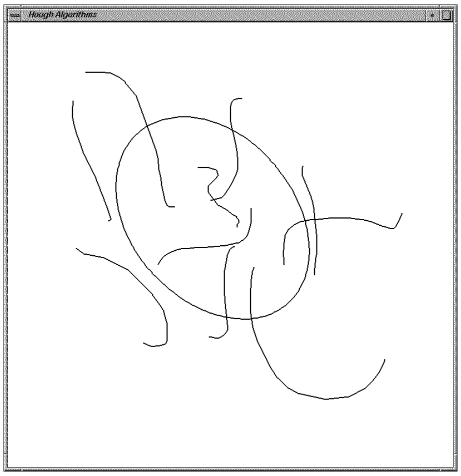

Fig. 13. Image containing 1 ellipse and 10 additional curves.

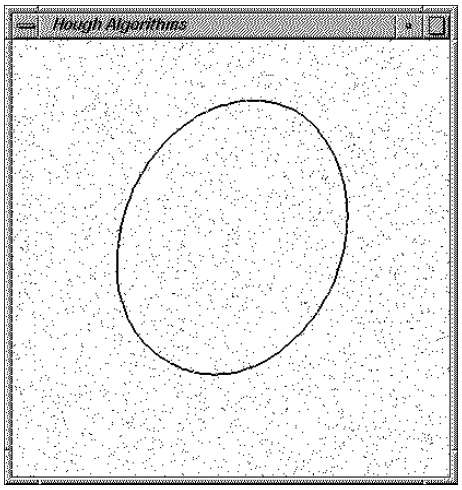

Fig. 14. Image containing 1 ellipse and 2% speckle noise.

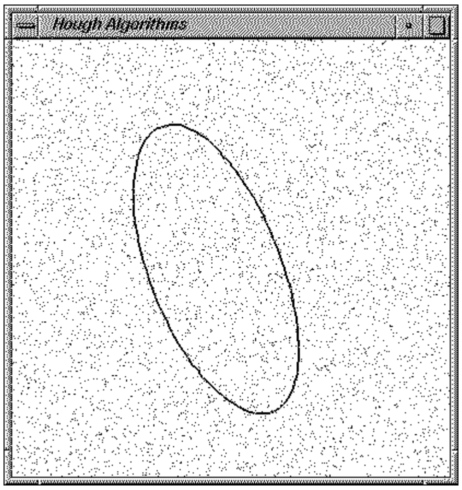

Fig. 15. Image containing 1 ellipse and 4% speckle noise.

C. Experiment 3: Speckle Noise.

Next we tested the tolerance of the algorithms to uniform, speckle noise. Twenty images, each containing a single ellipse were produced. We then randomly selected 1% of the pixels and inverted them (changing black to white, white to black). This was repeated with 2%, 3%, ...up to 8% of the pixels. Note that the same twenty original images were used in each case. Typical noise levels are shown in Fig. 14 - 17. For this experiment, an image size of 400×400 pixels was used.

D. Experiment 4: Real World Images.

Finally, we applied the algorithms to three real-world images of cells in a breast cancer sample (Fig. 27, 30, 33). The images were thresholded and a Sobel edge detector was applied to each image. As the original images were very small, the resulting edge images were notably jagged. To improve this, some smoothing was applied to the edge images. The three Hough algorithms were applied to the resulting smoothed, edge images. The ellipses found by each algorithm were recorded, along with the processing

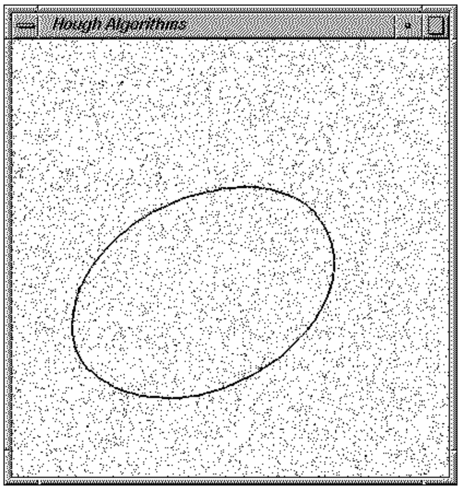

Fig. 16. Image containing 1 ellipse and 6% speckle noise.

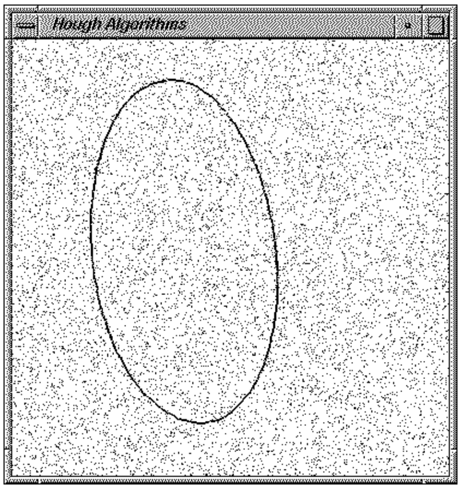

Fig. 17. Image containing 1 ellipse and 8% speckle noise.

time.

VIII. Results

A. Experiment 1: Multiple Ellipses.

The graph of Fig. 18 shows the percentage of ellipses correctly found, plotted against the number of ellipses in the image. All three algorithms begin with 95% for a single ellipse, as each algorithm failed with one image from the testing set. Note that, as predicted in [2], SHT and PHT produce very similar results, with SHT generally slightly more accurate.

Figures 19 - 20 plot the average computation time (in seconds) against the number of ellipses in the image. Because the computation time for SHT was so much greater than that of the other two algorithms, we have presented this as two graphs. If all three algorithms were plotted on the same graph, the differences between RHT and PHT would become obscured.

Fig. 21 plots the total number of false alarms found in each set of images. A false alarm is recorded when a non-existent ellipse is incorrectly identified as existing. RHT

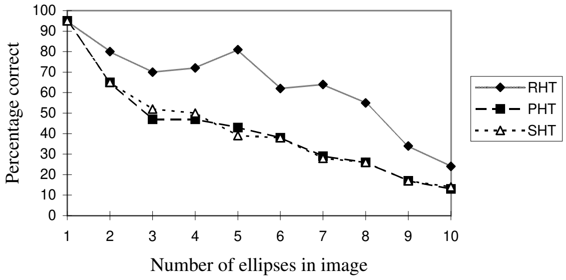

Fig. 18. Percentage correct vs Number of ellipses.

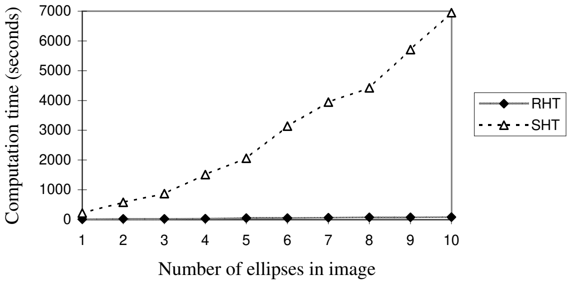

Fig. 19. Computation time vs Number of ellipses. For RHT and SHT.

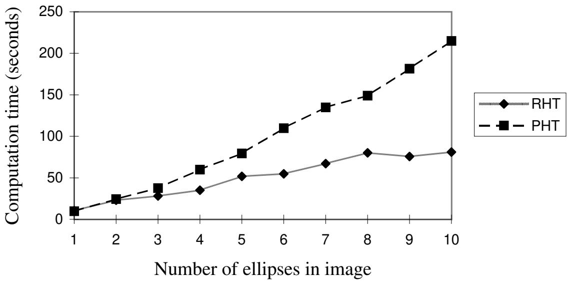

Fig. 20. Computation time vs Number of ellipses. For RHT and PHT.

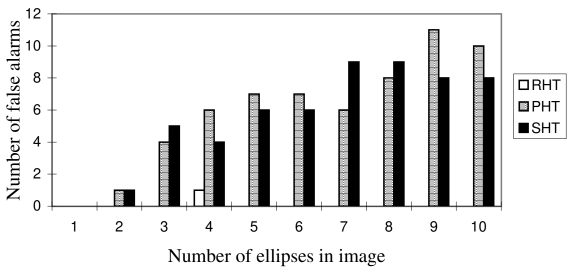

Fig. 21. Number of false alarms vs Number of ellipses.

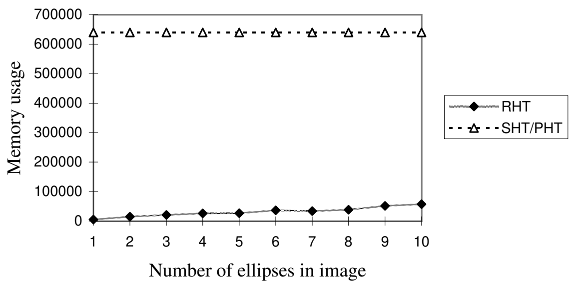

Fig. 22. Memory Usage vs Number of ellipses.

suffered a single false alarm over the entire testing set.

Estimated memory usage is plotted in Fig. 22. The memory requirements for SHT and PHT were identical. For these algorithms, most of the required memory is used by the 2-D histogram employed for estimating ellipse centres. This memory may later be re-used by the 3-D histogram utilised for estimating the remaining three ellipse parameters. Thus we have approximated the memory usage of these two algorithms as that required by the 2-D histogram. The histogram is of the same dimensions as the image (800×800). Each element of the histogram contains a single integer, which we assume requires a single unit of memory.

For RHT, most of the memory requirement is consumed by the tree structure, described in Section IV. Each cell of the tree structure contains:

- a count, requiring a single unit of memory.
- the five parameters of an ellipse. Each parameter stores several decimal places, and so is assumed to require four units of memory, giving a total of $(5 \times 4 =)$ 20 units of memory.
- three memory pointers used to connect to other cells in the tree. Each is assumed to require a single unit of memory.

Thus each cell requires a total of 24 units of memory. The memory usage for RHT was obtained by finding the average tree size (number of cells) and multiplying by the memory usage for a single cell.

B. Experiment 2: Additional Curves.

The graph of Fig. 23 plots the percentage of times that the single ellipse in the image was found, against the number of additional curves in the image. The graph shows all three algorithms as scoring 95% when no additional curves are present, as each algorithm failed for one image in the testing set.

Fig. 24 plots the total number of false alarms found, against the number of additional curves in the images.

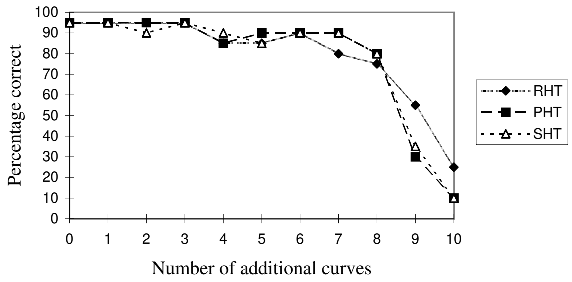

Fig. 23. Percentage correct vs Number of additional curves.

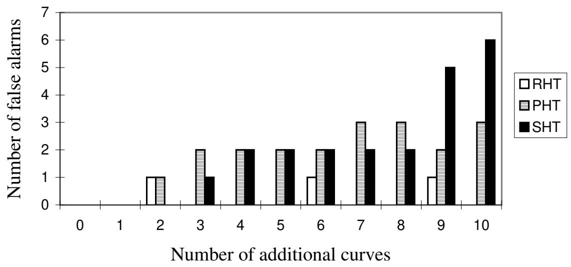

Fig. 24. Number of false alarms vs Number of additional curves.

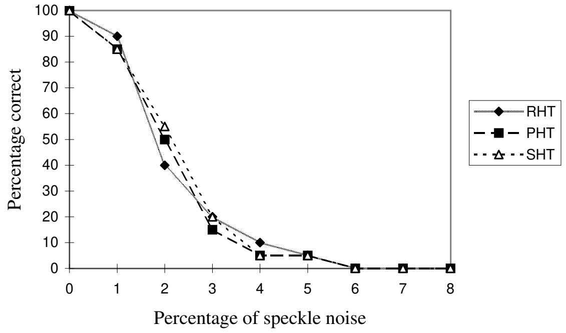

Fig. 25. Percentage correct vs Percentage of speckle noise.

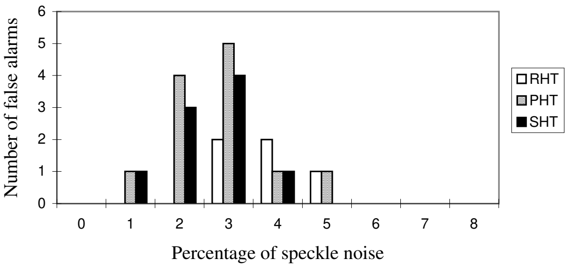

Fig. 26. Number of false alarms vs Percentage of speckle noise.

C. Experiment 3: Speckle Noise.

Fig. 25 graphs the percentage of ellipses correctly found, against the percentage of speckle noise present in the images.

In Fig. 26, we plot the total number of false alarms, against the percentage of speckle noise present.

D. Experiment 4: Real World Images.

Fig. 27, 30 and 33 show three images of cells in breast cancer samples. The results of applying a Sobel edge detector and some smoothing are shown in Fig. 28, 31 and 34. The ellipses identified by RHT are shown in Fig. 29, 32 and 35. The respective computation times for RHT for these images were 122 seconds, 247 seconds and 155 seconds. Both SHT and PHT failed to identify any ellipses in the images.

IX. Discussion

We begin this section by summarising the experimental results of the three Hough algorithms, showing that RHT generally gives superior performance over both SHT and PHT. We explain these results by highlighting the different mechanisms at work in these algorithms. RHT may be

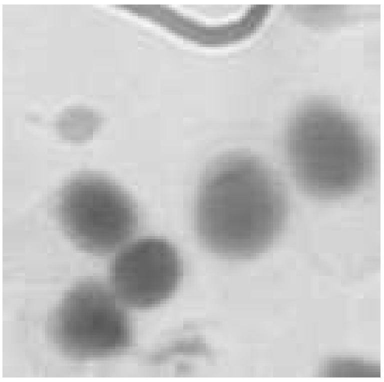

Fig. 27. Breast cell sample 1.

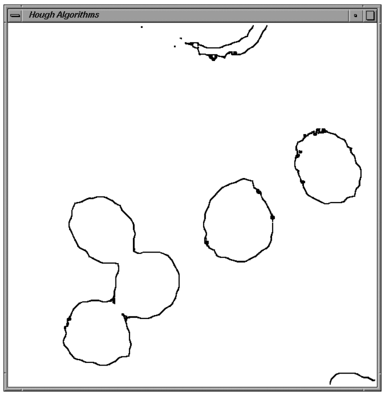

Fig. 28. Edge detected image of Breast cell sample 1.

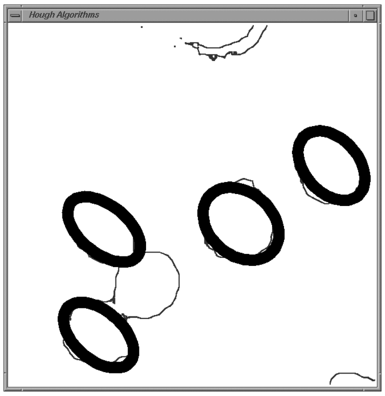

Fig. 29. Results of RHT for Breast cell sample 1.

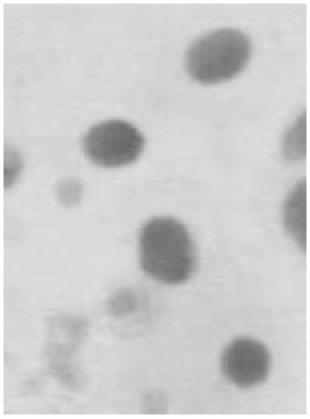

Fig. 30. Breast cell sample 2.

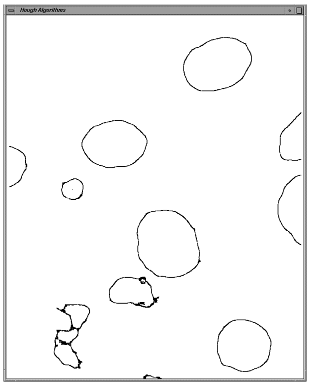

Fig. 31. Edge detected image of Breast cell sample 2.

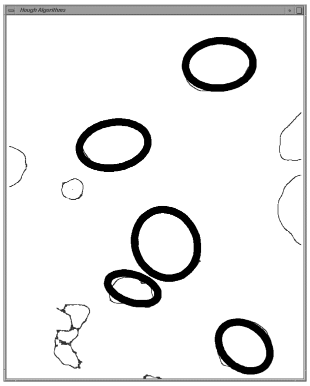

Fig. 32. Results of RHT for Breast cell sample 2.

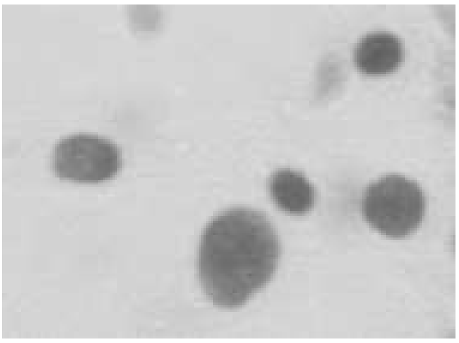

Fig. 33. Breast cell sample 3.

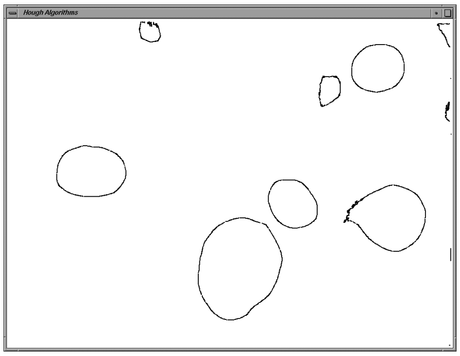

Fig. 34. Edge detected image of Breast cell sample 3.

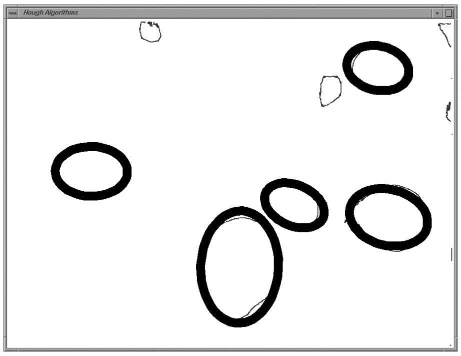

Fig. 35. Results of RHT for Breast cell sample 3.

thought of as an implementation of a 5-D Hough Transform where the parameter space has been sub-sampled. In contrast, SHT and PHT organise the recognition into two simpler Hough Transforms. This results in a reduction in computational and memory requirements, but degrades accuracy under certain conditions.

RHT was found to have higher accuracy than both SHT and PHT, in noise-free images with multiple ellipses (Fig. 18). It was found to be less subject to false alarms (Fig. 21, 24, 26) in both noise-free and noisy images. Moreover, RHT proved to be faster than either SHT or PHT (Fig. 19, 20). Specifically, when compared with SHT, RHT was several hundred times faster. As a final comparison, RHT was found to require substantially less memory (Fig. 22).

To explain these results, let us consider the different mechanisms at work in SHT and PHT, as compared to RHT. First, consider that directly implementing the Hough Transform for ellipse detection would require a 5-D Hough Transform. Traditionally, each pixel in the image generates a curve in the parameter space. In this case, each pixel would generate a 4-D curve in the 5-D parameter space, where the curve describes all possible ellipses which could have generated that pixel. These curves will typically be recorded as entries in a 5-D histogram. The high dimension of the histogram renders this implementation impractical. This is due to both the memory requirements in recording the histogram, and the computational demands in identifying local maxima of the histogram.

A standard approach to overcome these problems is to organise the recognition into several simpler stages, involving lower dimensional Hough Transforms. This is the approach adopted in both SHT and PHT. In the first stage, a 2-D Hough Transform is used to produce a set of candidate ellipse centres. This collection will include spurious points incorrectly identified as ellipse centres. It will also include inaccuracies in the estimates of the position of the ellipse centres.

The collection of candidate ellipse centres is passed as input to a second stage of processing. In this stage, an estimate is made of the remaining three ellipse parameters, for each candidate ellipse centre. This estimate will be nonsensical where the candidate ellipse centre was an incorrectly chosen spurious point. Note that knowledge gained in estimating the remaining three ellipse parameters is not used to refine the position of the ellipse centre. That is, information from this stage is not fed back to the first stage of processing. Thus, if the candidate ellipse centre was inaccurately positioned, this will lead to inaccuracies in estimates of the remaining three parameters. At completion, the second stage produces a collection of ellipses, one for each candidate ellipse centre.

In a final stage of processing, a validation test is applied to each ellipse, in order to discard those which are inaccurate or do not exist in the image. In this implementation, the validation test involved counting the number of black pixels in the image which lie close to the ellipse, and testing whether this is greater than some threshold of the ellipse's perimeter.

This approach, of organising the recognition into stages, has the important advantages of reducing both computational and memory requirements, and hence rendering the algorithm practical. However, as alluded to above, there are two serious disadvantages:

- The Hough Transform of the first stage tends to contain numerous spurious local maxima, which do not correspond to the centre of any ellipse in the image. In the second stage, no attempt is made to evaluate the validity of the ellipse centres. Thus the algorithm will blithely produce estimates of nonsensical ellipses, relying on the simple-minded validation of the third stage to identify these.
- In the second stage, no attempt is made to refine the estimates of ellipse centres from the first stage. We found from experiment that small inaccuracies in the position of an ellipse were magnified in estimates of the remaining three parameters.

These problems can be avoided by use of a single stage, 5-D Hough Transform. The computational and memory requirements of such an algorithm can be made practical by introducing a method of sub-sampling the parameter space. This is the approach used in RHT.

When sampling from a population (in this case, points in the parameter space), the sample is typically drawn according to a uniform probability density function (pdf) defined over the population. With RHT, we are implementing what could be considered a more intelligent form of sampling. By 'intelligent', we mean that our sampling is guided by knowledge of the object to be recognised (an ellipse).

We do not explicitly define a pdf over the parameter space. Instead, we define a uniform pdf over the collection of black pixels in the image. Three distinct pixels are randomly chosen, and these are then mapped to a point in the parameter space (by finding the unique ellipse passing through the three pixels, obeying local estimates of tangent). In this way, we implicitly define a pdf over the parameter space.

For an image containing few ellipses and little noise, this pdf will favour points in the parameter space which describe the ellipses in the image. Because of this, a small sample is sufficient to generate the appropriate clusters in the parameter space. Note that since such a small sample is taken, we are able to use the tree structure described in Section IV, rather than a 5-D histogram. This enables a reduction in memory demands, highlighted in the graph of Fig. 22.

As with any sampling technique, under certain conditions the sample will be poorly chosen. For a particular sampled point, this occurs when the three pixels do not come from the same ellipse. Consider an image containing n ellipses of equal size, each consisting of p pixels. Let event A be defined as "a 3-tuple of randomly chosen pixels lie on a particular ellipse". Then the probability of A is given by:

$$P[A] = \frac{p(p-1)(p-2)}{np(np-1)(np-2)} \approx \frac{1}{n^3}$$

Let event B be defined as "a 3-tuple of randomly chosen pixels come from a single ellipse, for any ellipse". Then the probability of B is given by:

$$P[B] = n\frac{p(p-1)(p-2)}{np(np-1)(np-2)} \approx \frac{1}{n^2}$$

As we increase n, the number of ellipses in the image, less of the 3-tuples of pixels will come from a single ellipse. Thus our sample of the parameter space will consist increasingly of spurious points, with less points located near the parameters of ellipses in the image. In *Experiment 1: Multiple Ellipses*, this manifested itself as a decreased accuracy as the number of ellipses in the image increased (Fig. 18).

The same mechanisms are at work when additional curves are present in the image, as in *Experiment 2: Additional Curves*. Increasing the number of additional curves decreases the probability of choosing a 3-tuple of pixels from the same ellipse. The resulting decrease in accuracy is shown in the graph of Fig. 23.

Uniform speckle noise gives rise to the same problems. In *Experiment 3: Speckle Noise*, speckle noise was generated uniformly over the entire image. As only a very small proportion of the total pixels in the image formed the ellipse, even a small percentage of noise led to a rapid decrease in accuracy (Fig. 25). At very low noise levels (0%, 1%), clusters were evident in those points sampled from the parameter space. At medium noise levels (3%, 4%), sufficient spurious points were present to obscure the clusters, giving rise to false alarms (Fig. 26). At high noise levels (6%–8%), no clusters were discernible in the sample, resulting in no ellipses being found (neither correct ellipses nor false alarms).

X. Conclusions

In this paper, we have proposed an algorithm for ellipse detection using the Randomized Hough Transform (RHT). We considered the task of finding the unique ellipse passing through an n-tuple of pixels in the image. This was reduced to a linear problem by making use of a feature of the geometry of ellipses.

We then compared the resulting algorithm against both the standard Hough Transform (SHT) and the Probabilistic Hough Transform (PHT). The comparison included noise-free images containing multiple ellipses, images containing additional curves, images containing uniform speckle noise, and several real-world images. RHT was found to have a higher accuracy than SHT or PHT with noise-free images containing multiple ellipses. In was also less subject to false alarms. RHT gave considerable improvements in computation time, especially when compared against SHT. The memory requirements were also considerably less than those of either SHT or PHT.

The accuracies of all three algorithms were comparable in the presence of noise, both with additional curves and uniform speckle noise. However, RHT yielded less false alarms.

Finally, the algorithms were tested with three images of breast cancer cells. RHT identified most cells present in the images. SHT and PHT failed to identify any cells.

The difference in the results of the three algorithms can be understood when one considers the mechanisms at work in each. We have described RHT as an implementation of a 5-D Hough Transform with sub-sampling of the parameter space. The sampling method is guided by knowledge of the geometry of ellipses. We also demonstrated why the sampling degrades as the number of ellipses present in the image increases. In contrast, SHT and PHT organise the recognition task into several stages requiring lower dimensional Hough Transforms. The resulting improvements in computation time and memory requirements are offset by a reduced accuracy under certain conditions.

References

[1] H. K. Yuen, J. Illingworth, and J. Kittler, "Detecting partially occluded ellipses using the Hough transform," *Image and Vision Computing*, vol. 7, no. 1, pp. 31–37, Feb 1989.

[2] N. Kiryati, Y Eldar, and A. M. Bruckstein, "A probabilistic Hough transform," *Pattern Recognition*, vol. 24, no. 4, pp. 303–316, 1991.

[3] James R. Bergen and Haim Shvaytser (Schweitzer), "A probabilistic algorithm for computing Hough transforms," *Journal of Algorithms*, vol. 12, pp. 639–656, 1991.

[4] Lei Xu, Erkki Oja, and Pekka Kultanen, "A new curve detection method: Randomized hough transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, May 1990.

[5] Lei Xu and Erkki Oja, "Further developments on RHT: Basic mechanisms, algorithms and computational complexities," in *Proceedings. 11th IAPR International Conference on Pattern Recognition. Vol.1. Conference A: Computer Vision and Applications*, Los Alamitos, CA, USA, Aug 1992, Int. Assoc. Pattern Recognition, vol. 1, pp. 125–128, IEEE Comput. Soc. Press.

[6] Lei Xu and Erkki Oja, "Randomized Hough Transform (RHT): Basic mechanisms, algorithms, and computational complexities," *CVGIP: Image Understanding*, vol. 57, no. 2, pp. 131–154, Mar 1993.

[7] A. R. Hare and M. B. Sandler, "Improved-performance 'randomised' Hough transform," *Electronics Letters*, vol. 28, no. 18, pp. 1678–1680, Aug 1992.

[8] J. R. Parker, *Practical Computer Vision Using C*, John Wiley & Sons, Inc., New York, U.S.A., 1994.

[9] E. R. Davies, *Machine Vision: Theory, Algorithms, Practicalities*, Academic Press Ltd, 24/28 Oval Road, London NW1 7DX, United Kingdom, 1990.

[10] J. Illingworth and J. Kittler, "The adaptive Hough transform," *IEEE Trans. Pattern Analysis & Machine Intelligence*, vol. 9, no. 5, pp. 690–697, 1987.

Robert A. McLaughlin received his B.Eng. (Hons) in Information Technology from The University of Western Australia in 1991. He is currently undertaking a PhD (E. & E. Eng.) at The University of Western Australia, studying general methods of computer vision.