

Circle detection by image analysis using the Dynamic Generalized Hough Transform

Henrik Birk, Lise Grevenkop-Castenskiold, Jacob Lind,
Thomas Baltzer Moeslund, Karina Møller, Klaus Ingemann Pedersen

Institute of Electronic Systems, Aalborg University, Denmark

January 17, 1995

Abstract. The Standard Hough Transform (SHT) in general is a powerful tool in image analysis, e.g. circle detection which is the subject of this paper. However, the method is a time-consuming process and it requires memory out of proportion to what a normal computer offers.

In attempt to solve these problems the Dynamic Generalized Hough Transform (DGHT) is introduced. The algorithm is presented, tested and compared with a modified SHT.

The DGHT samples just as many times as required in order to locate the circles. This dynamic feature reduces the processing time in relation to the modified SHT.

In addition, the memory requirements are reduced in the DGHT algorithm, since it uses three one-dimensional accumulators instead of the one three-dimensional accumulator the SHT uses.

The result is that the DGHT is 75 times faster in average than the modified SHT. However, the price for the faster algorithm is a lower detection rate and a higher sensitivity towards noise.

1 Introduction

The problem in this paper is detection of multiple circles with the possibility of overlap using a Hough Transform (HT). The principle of a HT is a transformation of the image to a parameter space, called the Hough space.

The known Standard Hough Transform (SHT) [1] is a robust technique for detecting any analytically defined shape, but it requires a considerable amount of memory and has a high computational cost. The SHT requires a three-dimensional ac-

cumulator for the detection of the circle parameters (x_0, y_0, r) . In addition, it is difficult to find the local maxima in the Hough space due to the unknown number of peaks, and the possibility of false peaks. In regard to this, many versions of the SHT have been introduced.

A modification of the SHT for circle detection using a two-dimensional array is presented in [2]. This algorithm is aimed at improving the efficiency and reducing the size (dimension) of the Hough space. Unfortunately the method tends to fail unless more than half of the circle arc is visible, because the method is based on locating parallel tangents.

Another method is the new parallel Hough Transform for circles presented in [3]. This algorithm uses a pair of two-dimensional accumulators to reduce memory and computational cost. It is capable of discriminating multiple (including concentric) circles in a complex real life image with a detection rate of 95-100 %.

The Transformation of Inversion (TOI) for circle detection is presented in [4]. In the TOI the circle under detection is first transformed to a straight line so the detection problem is reduced to the detection of lines. This reduces the dimension of the Hough space to a plane and simultaneously improves the computational cost. In addition, the interpretation of the local maxima in the Hough space is simplified.

The methods mentioned above all have one thing in common. The way the algorithms process an image does not change from image to image, and this makes them static. A new family of the Hough Transform has been introduced, the Probabilistic Hough Transform (PHT). The Dynamic Generalized Hough Transform (DGHT) [5] belongs to this family. The DGHT algorithm is preferable in situations where the surroundings change, because it adapts to the given assign-

ment. It samples the region of interest only as many times as required to locate the circles. This dynamic feature gives a significant reduction in the number of calculations compared to the SHT.

In this paper a DGHT algorithm will be presented and tested through a case study. The DGHT algorithm will be tested on digitalized images of mushrooms containing multiple circles. The DGHT algorithm will also be compared to a modified SHT. The detection rate and processing time of the two algorithms will be compared, testing whether a dynamic algorithm is preferable in this particular situation.

2 The Dynamic Generalized Hough Transform

This section deals with a description of the DGHT algorithm. First a brief general description is given, then a more detailed description will follow. The DGHT algorithm is based on the flowchart shown in fig. 1.

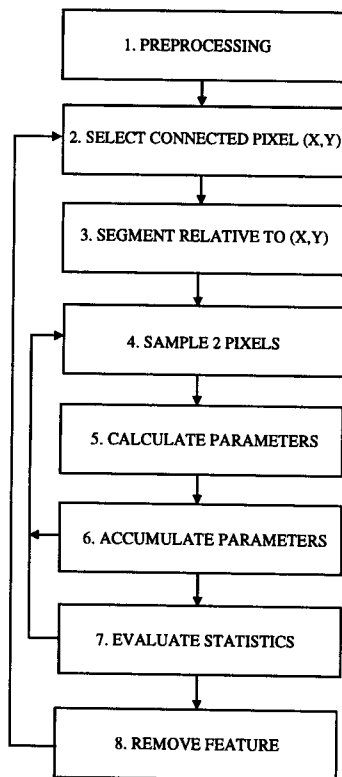


Figure 1: Flowchart for the DGHT algorithm

The first step is to preprocess the input image so it only contains edges without any concavities from overlapping circles. The image is then scanned for a connected edge pixel (cf. section 2.2). Once a connected edge pixel is deter-

mined, its location is used to segment a part of the image (region of interest). All the edge pixels relative connected to the connected edge pixel are then stored in a list. The list is sampled a number of times, each time randomly picking two pixels. These two pixels together with the connected pixel are used to calculate the circle parameters which then are stored in three separate one-dimensional accumulators. The sampling and calculation is continued until the statistics of the accumulators fulfil a predefined condition. When this occurs, the maxima in the accumulators are determined. These maxima correspond to the parameters of the circle under detection. The detected circle is subsequently removed from the image, and the algorithm starts over again by scanning for a new connected pixel. This is continued until the image is completely processed.

2.1 Preprocessing

In this step of the DGHT algorithm the image is preprocessed, that is, edge detection as well as removing the concavities from any overlapping circles.

To execute the preprocessing two images are used, an input image and an output image. First a threshold is applied to the input image in order to make it binary. Doing this the objects (circles) are separated from the background. The thresholding is carried out as a comparison between the grey-level of the actual pixel and a threshold value, th . If the grey-level of the pixel under consideration is smaller than th then the pixel is made black, and otherwise it is made white.

The threshold value th is determined from a grey-level histogram of the image. The histogram in fig. 2 shows two local maxima. The th value is chosen to be the minimum value between these two maxima. In some cases the histogram does not look as simple as in fig. 2. This makes it more difficult to find the threshold value th so in that case another approach is needed.

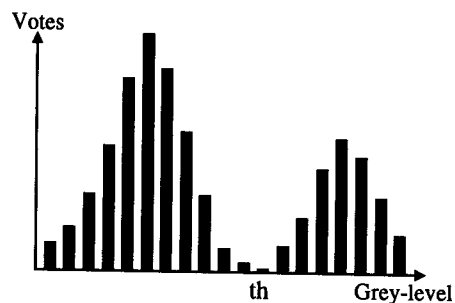


Figure 2: Grey-level histogram

After thresholding, the input image is edge detected.

The edge detection is made by differentiating all pixels in the thresholded image. This is done by calculating the absolute value of the gradient. An approximation to this is given in (1).

$$\nabla f = \frac{\partial F}{\partial x} \mathbf{i} + \frac{\partial F}{\partial y} \mathbf{j}$$

$$|\nabla f| \approx |F(x+1, y) - F(x-1, y)| + |F(x, y+1) - F(x, y-1)| \quad (1)$$

The function F indicates the grey-level of a pixel, in this case 0 (black) or 255 (white). If ∇f of the pixel (x, y) in the input image is larger than 0, the pixel (x, y) in the output image is an edge corresponding to the value 255, otherwise it is given the value 0. After this computation the input image is still the thresholded image whereas the output image is the edge detected image.

Finally the concavities from overlapping circles are removed from the output image. This is done by imposing the 9×9 mask shown in fig. 3 on the pixels (x, y) in the thresholded input image corresponding to the edge pixels (x, y) in the output image. The pixel under consideration is in the center of the mask.

0	0	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	0	0

Figure 3: 9×9 mask for concavity detection

A concavity is detected when the mask holds more than 38 white pixels, corresponding to an angle θ less than 180 degrees. In fig. 4 the concavity angle θ is shown. The number of white pixels to be counted corresponds to the shaded area. The mask is circular in order to make the number of white pixels in the shaded area independent of the location of the concavity angle.

A detected concavity is removed by making the actual edge pixel in the output image black. After the edge detection as well as the removal of the concavities the preprocessed image only contains isolated line segments, each belonging to one circle.

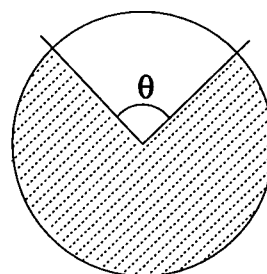


Figure 4: The principle of concavity detection

2.2 Select connected pixel (x, y)

When the preprocessing is completed the DGHT algorithm seeks for the first connected pixel (x, y) in the image, because it is needed for the following segmentation. A pixel is considered to be connected if it holds at least three edge pixels in a 3×3 mask, see fig. 5.

1	1	1
1	1	1
1	1	1

Figure 5: 3×3 mask for connectivity detection

The connectivity is checked by imposing the 3×3 mask containing all ones to each pixel in the image. This is carried out in rows from left to right within top to bottom.

2.3 Segment relative to (x, y)

The next step in the DGHT algorithm is a segmentation relative to the connected pixel. This is done by creating a list with all the edge pixels which are relative connected to the connected pixel. The edge pixels are removed from the image as they are stored in the list. The motivation for using segmentation is that it ensures that only one circle is under detection at a time.

2.4 Sample 2 pixels

Subsequently to the segmentation the algorithm selects three edge pixels p_1, p_2 and p_3 to calculate the circle parameters.

The first edge pixel is chosen to be the connected pixel whereas the two remaining edge pixels are sampled from the segment list. The sampling is carried out as a random (uniform distributed) process and it is checked that the two pixels are different.

2.5 Calculate parameters

Given the three edge pixels on a circle it is possible to calculate the circle parameters. The parameters which will be calculated are the center (x_0, y_0) and the radius r of the circle. The calculation will be based on the knowledge of the circle-equation (2), and the three edge pixels $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$ and $p_3 = (x_3, y_3)$.

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \quad (2)$$

Combining (2) with p_1 , p_2 and p_3 results in three equations with the three unknowns x_0 , y_0 and r . These equations are difficult to solve because (2) contains nonlinear terms, so it is rewritten to

$$x^2 + y^2 = Ax + By + C \quad (3)$$

where

$$\begin{aligned} A &= 2x_0 \\ B &= 2y_0 \\ C &= r^2 - y_0^2 - x_0^2 \end{aligned}$$

The circle equation in (2) has now been transformed to a linear equation (3) in A , B and C . Solving this for A , B and C is the same as solving (2) for x_0 , y_0 and r , because the circle parameters can be calculated based on A , B and C . To solve for A , B and C a system of equations is formulated based on (3) and p_1 , p_2 and p_3 .

$$\begin{bmatrix} (x_1^2 + y_1^2) \\ (x_2^2 + y_2^2) \\ (x_3^2 + y_3^2) \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \quad (4)$$

This equation can easily be solved by multiplying (4) with the inverse of the 3×3 matrix. The inverse of the 3×3 matrix can be calculated based on cofactors.

A condition for using this method is that the determinant is different from zero, which is always the case unless the three pixels are placed on a line. If the determinant is zero two new edge pixels are sampled from the list.

2.6 Accumulate parameters

After the parameters x_0 , y_0 and r have been calculated they are mapped to the Hough space. The DGHT reduces the Hough space from one three-dimensional accumulator, used in the SHT, to three one-dimensional accumulators. This is accomplished by projecting the pixels in the three-dimensional Hough space onto the three axes.

The result of this is a significant reduction of the memory requirements.

If the resolution in x_0 and y_0 is denoted α and the resolution in r is denoted β , then the memory requirements are reduced from $\alpha^2\beta$ to $2\alpha + \beta$. Since α usually is in the hundreds, a significant reduction is obtained. In addition, the reduced Hough space also makes it faster to detect the maxima in the Hough space, corresponding to the parameters of the circle under detection.

It is important that the accumulators are able to store *all* the calculated parameters so information is not truncated. This means that if the image contains circle segments which correspond to circles with centres outside the image, then the x - and y -accumulators should be able to store values outside the range of the image.

It is not necessary to evaluate the statistics (step 7) for the accumulators after each sample, because they do not change much after one sample. Step 4, 5 and 6 are therefore run a preset number of times before executing step 7.

Before turning to the next step the variance of each accumulator is estimated using equation (5).

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (z_i - \bar{z})^2 \quad (5)$$

The variances are calculated because they are needed in order to decide when to stop sampling.

2.7 Evaluate statistics

The next step in the algorithm is to evaluate the statistical information associated with the parameters under detection. The statistical information is used to decide when to stop sampling the list of edge pixels.

The sampling of the list is stopped when the variance of each accumulator has become stable, see fig. 6.

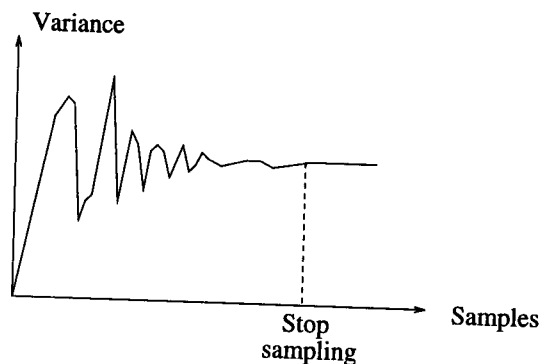


Figure 6: The principle of the stop criterion

If the sampling is proceeded after having reached stability the list will not give any new information. The degree of stability can be estimated by examining the deviation between the actual variance and the mean variance of the last n samples. The mean variance of the last n samples is the best estimator of the value that the variance converges to. The mean variance is calculated using (6).

$$\bar{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n \sigma_i^2 \quad (6)$$

The sum of the variance deviation from the mean variance for n samples is called the residual error and denoted e . This is calculated using (7).

$$e = \sum_{i=1}^n |\sigma_i^2 - \bar{\sigma}^2| \quad (7)$$

The residual error is an expression of the stability where a small error corresponds to a high degree of stability. Based on this, the sampling can be stopped when (8) holds for all three accumulators.

$$K \cdot \bar{\sigma}^2 > e \quad (8)$$

The K-factor is introduced so the precision of the stop criterion can be set individually to a given application.

It is the statistical stop criterion that makes the DGHT dynamic. This means that data of a perfect circle is sampled less times than data of a less perfect circle. The result of this is a faster algorithm compared to the SHT.

To accept the object as a circle the standard deviation, σ , of the accumulators after stabilizing has to be below a predefined level. This corresponds to a well defined narrow peak in the accumulators because of the approximate normal distribution, see fig. 11.

When the sampling is stopped the algorithm continues to step 8.

2.8 Remove feature

If the object is accepted as a circle then the parameters are calculated and the circle is removed from the image, otherwise the algorithm resets the accumulators and returns to step 2. After the circle is removed the algorithm returns to step 2 until the image is completely processed.

Since the objects seldom are perfect circles, it is necessary to remove a band around the detected circle in order to remove all the pixels belonging

to this. The circle band which will be removed is shown in fig. 7.

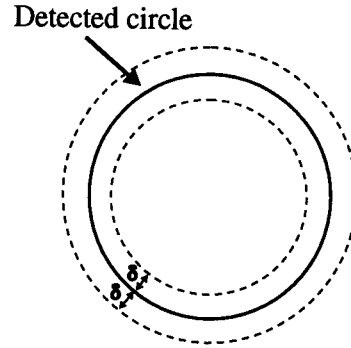


Figure 7: The removal of a detected circle

The width of the circle band is determined by the standard deviation of the estimated parameters. In the case of a normal distribution the width of the curve can be calculated from the approximation in (9).

$$\delta \approx 2 \frac{\sigma_x + \sigma_y}{2} + 2\sigma_r \quad (9)$$

σ_x , σ_y and σ_r are the standard deviations of the x-, y- and r-accumulators, respectively. The use of (9) guarantees that 95% of the votes in the accumulators are included in the circle band given by $r \pm \delta$.

3 A case study

In the following case study it is intended to test the DGHT against a modified SHT. This will be done to illustrate the consequences of reducing the number of calculations in the SHT. A description of the modified SHT can be found in appendix A. In the case study the DGHT and the modified SHT were implemented in the programming language ANSI C.

The DGHT algorithm was implemented with the following features:

- Only circles with radii [20; 70] pixels were accepted.
- Only segments containing more than 10 edge pixels were accepted.
- Step 4, 5 and 6 were carried out 50 times before evaluating statistics.
- In step 6 the variance of the accumulators were estimated when 5 samples had been executed.

- The x and y accumulators were able to store circle parameters positioned 100 pixels outside the image.
- The K-factor defined in equation (8) was set to 0.9.

Setting the K-factor to 0.9 optimized the DGHT algorithm with respect to the total number of errors. Errors are categorized in type 1 (failed to detect a circle) and type 2 (false detection of a circle). This means that the distance from origo to the curve in fig. 8 is minimized. This was also done for the modified SHT.

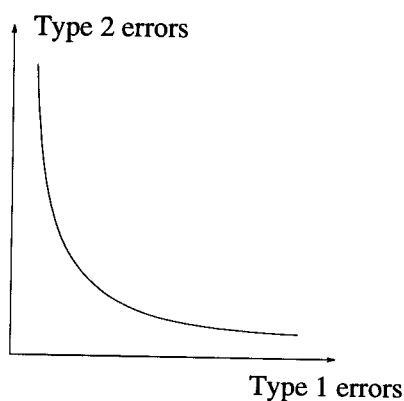


Figure 8: Principle of the relation between type 1 and type 2 errors

The images used for testing originated from video taken of mushrooms, which contained noise and multiple circles overlapping. The tests were carried out on a work station with a 100MHz RISC processor. The test images were digitalized from a black and white video with a resolution of 512×512 pixels and 256 grey-level values.

The first image contained 31 mushrooms in different sizes, see fig. 9.

First the test image was preprocessed to detect the edges and remove the concavities. In order to do this the threshold for the image had to be selected. This was done by examining the grey-level histogram which gave the threshold value 150. The preprocessed test image is shown in fig. 10.

Next the preprocessed image was scanned for a connected pixel, segmented and sampled a number of times. It was observed that the variance of the accumulators stabilized after 50 - 1000 samples depending on the segment under processing. In comparison, the DGHT algorithm was also tested on a generated test image containing perfect circles, which only required 50 samples.

To illustrate the contents of the three accumulators x, y and r after the variances have stabi-

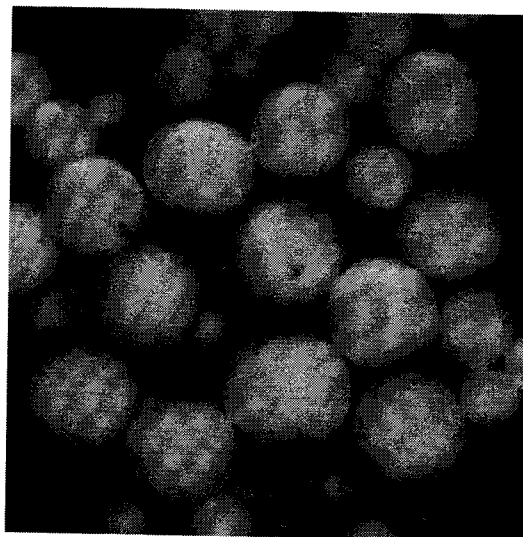


Figure 9: Test image

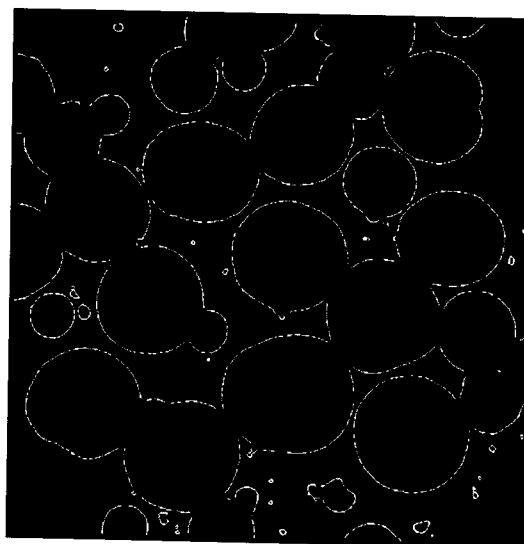


Figure 10: Preprocessed test image

lized, a plot of the accumulators for one of the circles in the test image is shown in fig. 11. The plotted accumulators stabilized after 500 samples.

It can be seen from the plots that there is a well defined peak in each accumulator. The peaks correspond to a circle with center (169,64) and a radius of 32 pixels.

It was assumed that the calculated data in the accumulators were normal distributed. To test this hypothesis a goodness of fit χ^2 -test was carried out at a 5% significance level. The test rejected the hypothesis. However, a further examination showed that the accumulator curves were steeper than the corresponding Gauss curve. The accumulator curves were placed inside the corresponding Gauss curves, except the peaks which were higher. Based on this observation the stan-

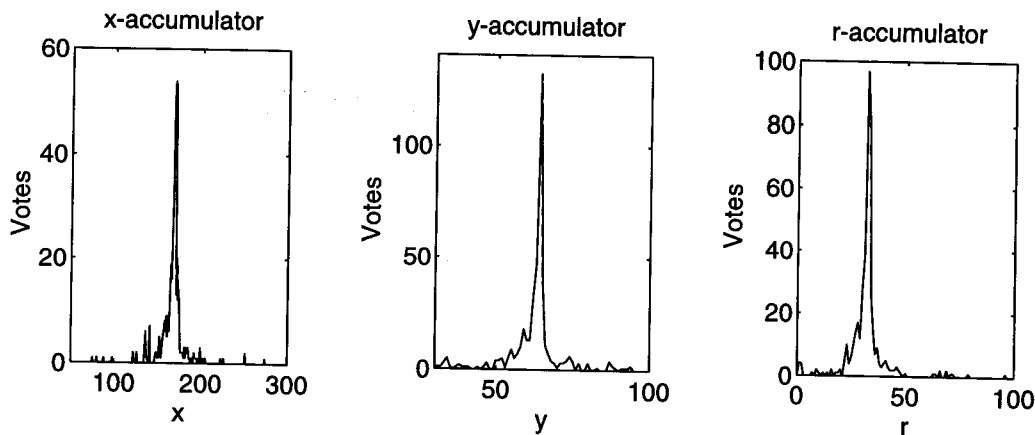


Figure 11: Accumulator plots

standard deviation can still be used as a measure of the stability of the accumulators, so the stop criterion and equation (9) still holds.

Finally all the detected circles were drawn and superimposed on the input image, see fig 12.

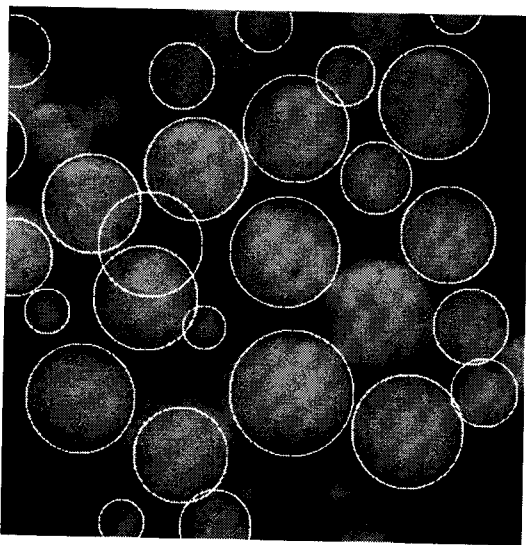


Figure 12: Detected circles using the DGHT

It can be seen that the DGHT found 23 true circles and 3 false circle and failed to detect 8 circles. The processing time was 21.01 seconds.

A postprocessing was added in order to minimize the detection of false circles. This post-processing included two new functions. The first function was able to combine two detected circles that actually corresponded to one. The second function calculated the ratio between white and black pixels in the detected circle, based on the thresholded input image. If the circle contained less than 80% white pixels, it was considered to be false and then rejected.

The postprocessing was implemented and the test repeated. The detected circles were again drawn and superimposed on the input image, see fig. 13.

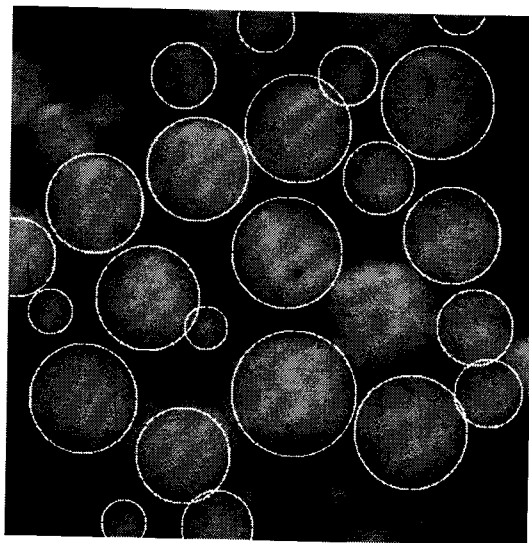


Figure 13: Detected circles using the DGHT and postprocessing

It can be seen that the number of false circles has been reduced from 3 to 0. The processing time was 21.26 seconds.

A similar test was performed on the modified SHT algorithm, see fig. 14. The result was that the modified SHT found 27 true circles, 1 false circle and failed to detect 4 circles. The processing time was 29 minutes.

The same test was performed on 5 different images of mushrooms. During the test the total number of circles, the number of detected true circles, the number of type 1 and type 2 errors and the processing time was observed. The results were averaged and are shown for the DGHT as well as the modified SHT in table 1.

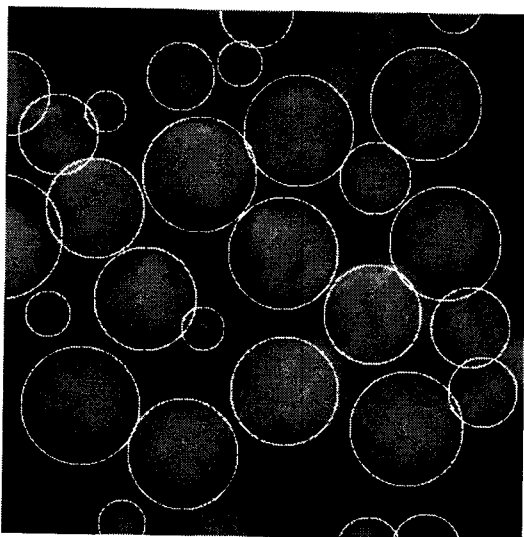


Figure 14: Detected circles using the modified SHT

	DGHT	SHT
Number of circles	23.6	23.6
True circles	13.4	18.4
Type 1 errors	10.2	5.2
Type 2 errors	1.0	3.2*
Processing time	20.33 sec	25.22 min

Table 1: Average results, in circles, of the DGHT versus the modified SHT based on 5 images, * see discussion

In the test it was observed that both the DGHT and the SHT detected most of the big mushrooms, but had difficulties detecting small mushrooms as well as mushrooms with a high percentage of overlap.

Finally a test was carried out to test how robust the DGHT was against noise compared to the modified SHT. The test was performed four times on the image in fig 9, with noise added in different levels (5%, 10%, 15% and 20%). The noise was added to the input image by randomly choosing x% pixels and randomly giving them a new grey-level value.

The results of the tests were plotted in graphs mapping errors of type 1 and errors of type 2 as a function of the different noise levels. The results of these tests are shown in fig. 15. It was observed that the processing time increased with the amount of noise added.

4 Discussion

The DGHT algorithm was tested against the modified SHT on 5 different images. The averaged results are shown in table 1. It can be seen that the modified SHT in average had a detection rate of 78% where the DGHT had a detection rate of 57% in average. This shows clearly that the SHT is more accurate than the DGHT. The test showed that both the SHT and the DGHT had difficulties detecting small circles and circles with a high percentage of overlap. The difficulties of detecting small mushrooms arises due to the fact that the relative resolution in the image is poor for small mushrooms. The problem of detecting mushrooms with a high percentage of overlap can be solved if the preprocessing is able to extract more edge pixels.

The number of type 2 errors for the DGHT was a third of what it was for the modified SHT. (*) The difference in type 2 errors appeared because the DGHT was added a postprocessing and the modified SHT was not.

On the average the DGHT was 75 times faster than the modified SHT.

In a test for robustness towards noise the DGHT was tested against the modified SHT. The test was performed four times on the image in fig. 9, with noise added in different levels. It can be seen in fig. 15 that the number of type 1 errors increased from 4 to 9 for the modified SHT, while the number of type 1 errors increased from 8 to 28 for the DGHT. This showed that the DGHT was very sensitive to noise with respect to type 1 errors, in contrast to the modified SHT which performed better.

In some applications the consequence of making a type 1 error might be higher than a type 2 error or vice versa. In these applications the DGHT algorithm can be optimized with respect to type 1 errors at the expense of type 2 errors or vice versa. The fundamental relation between type 1 and type 2 errors is shown in fig. 8.

In all, the tests showed that the price paid for the very fast DGHT algorithm compared to the modified SHT is a lower detection rate and a higher sensitivity towards noise.

In applications where it is possible to detect and remove some circles and process a new image of the same area again, it is possible to increase the detection rate for the DGHT. The increase is obtained because the new image will contain less circles so more edge pixels can be found. This improvement can be used in the particular case with the mushrooms and still the DGHT will be considerably faster than the modified SHT.

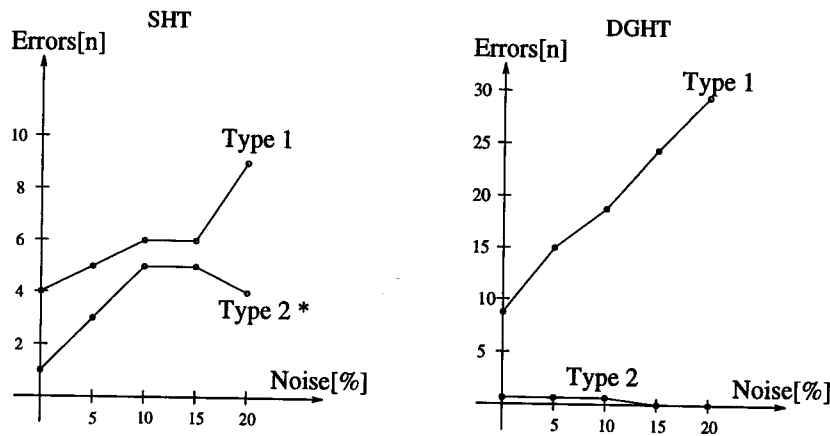


Figure 15: Errors for the modified SHT (left) and DGHT (right), * see discussion

It is possible to decrease the processing time of the DGHT if it is implemented in parallel. The DGHT is well suited for this because several segments can be processed at the same time.

The low memory requirements of the DGHT makes it attractive for implementation on a stand alone system which is often the case in industrial applications.

5 Conclusion

The problem in this paper was detection of multiple circles with the possibility of overlap. For this purpose the DGHT was proposed. The DGHT algorithm was described in details and tested on images of mushrooms in comparison to a modified version of the SHT.

The algorithms were tested on 5 different images. The average result was that in general the modified SHT had a higher detection rate than the DGHT, but on the other hand the DGHT was about 75 times faster than the modified SHT.

The algorithms were also tested on an image with noise added in different levels. This was done in order to conclude on the robustness of the DGHT. It was found that the DGHT algorithm was very sensitive to noise, while the modified SHT performed better.

The conclusion is that the price for the very fast DGHT algorithm with low memory requirements is a lower detection rate and a higher sensitivity towards noise.

6 References

- [1] William K. Pratt: *Digital image processing*, second edition. Wiley-interscience publication, ISBN 0-471-85766-1. pp. 613-622.
- [2] Raymond K.K. Yip, Peter K.S. Tam and Dennis N.K. Leung: *Modification of Hough Transform for circles and ellipses detection using a 2-dimensional array.* "Pattern Recognition" Volume 25, No 9, September 1992.
- [3] R. Chan and W.C. Siu: *New parallel Hough Transform for circles.* IEEE Proceedings-E, Volume 138, No 5, September 1991.
- [4] Wilson C.Y.Lam, Lam T.S. Lam and Dennis N.K. Leung: *Circle Detection By Transformation Of Inversion.* ICIP 92, Proceedings of the 2nd Singapore International conference on image processing.
- [5] Leavers, V.F. *The Dynamic Generalized Hough Transform: Its relationship to the Probabilistic Hough Transforms and an application to the concurrent detection of circles and ellipses.* IMAGE UNDERSTANDING 1992; 56: 381-398.

Appendix A

The modified SHT

This appendix is a brief description of the modified SHT algorithm implemented and used for testing. A focus is made on the modifications.

In fig. A.1 the flowchart of the modified SHT algorithm is shown. There are 2 major changes in the modified SHT algorithm compared to the SHT algorithm.

The first change is that the concavities are removed from the edge detected image. This is done in order to get isolated circles or parts of circles before the transformation. By doing this, step 7 (which is usually the difficult step in the SHT algorithm) becomes a much easier task. In order to transform all circles in the image the outer loop is added.

The second change is a reduction of the memory from a three-dimensional Hough space in the SHT algorithm to a Hough plane in the modified SHT algorithm. This is done by re-using the Hough plane, illustrated by the inner loop. The search size of the Hough plane used in step 5 is subsequently reduced by only searching in an area corresponding to the diameter of the circle that is wished to be detected.

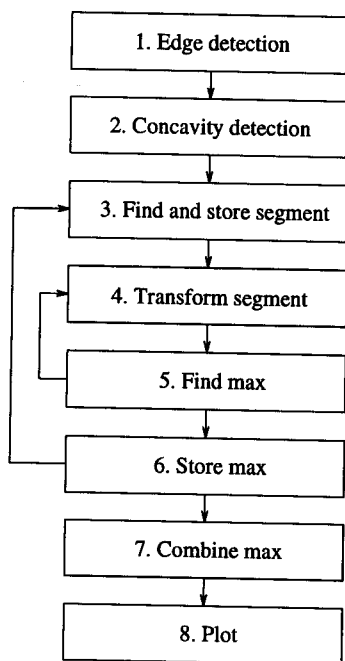


Figure A.1: Flowchart for the modified SHT.

The steps in fig. A.1 will now be explained.

(1) & (2) cf. section 2.1 in the main article.

(3) In this step the algorithm searches the preprocessed image, until it finds an edge pixel. Once an edge pixel is found all the edge pixels continuously connected to it is segmented. This is done by creating a list containing all these connected edge pixels. If the number of edge pixels in the segment list is less than a preset number, the segment is rejected as a part of a circle and the same step is run again, else the algorithm continues.

(4) In this step the edge pixels in the segment list are transformed to the Hough plane. Offset values are calculated to make sure that the first pixel in the segment list ends up in the middle of the Hough plane. By doing this the size of the Hough plane can be reduced to $2(2 \times R_{max})$. For every edge pixel in the segment list circles are drawn in the Hough plane with the current radius level being examined.

(5) In this step the current Hough plane is searched for the cell with the highest number of votes and compared to the previous maximum. If the current radius plane being examined is less than the highest accepted, the radius is incremented and the Hough plane is reset (all cells set to 0) before it loops back to the 'Transform segment' step. If however the current radius is the maximum radius then the algorithm continues to the next step.

(6) This step stores the detected maximum in a list sorted after radius (smallest first) if it is allowed to. Whether or not it should be saved is determined from the radius plane where it is found. If the maximum is found in a radius plane that is considered a critical region $[R_{min}; (R_{min} + 10)]$, then the Hough planes $[1; R_{min}]$ are examined. When the highest number of votes from these planes exceeds the previous number of votes found in the step 'Find max' the radius of the segment is considered to be too small and therefore it is rejected as a circle. As the last thing the segment list is erased before either moving on to a new segment, meaning looping back to the step 'Find and store segment', or to the next step in the algorithm if all segments have been processed.

(7) This step combines the detected maxima of the segments that originate from the same circle. This is done by calculating a weighted average of the maxima. Finally the results are stored in a textfile.

(8) This step uses the textfile to draw the detected circles in the output image.