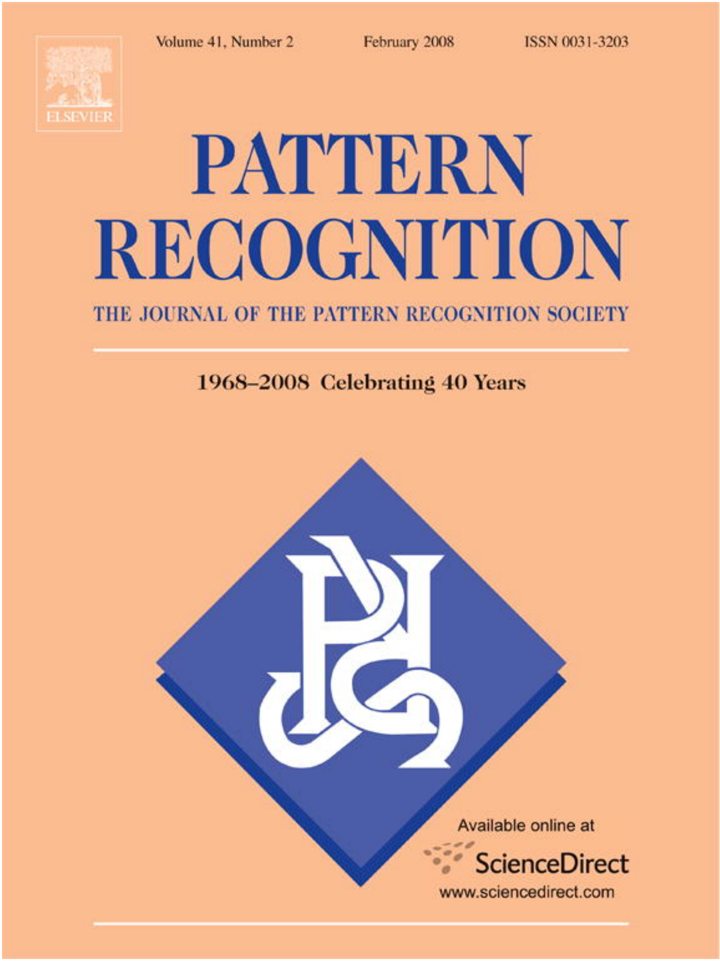
Volume 41, Number 2
February 2008
ISSN 0031-3203
ELSEVIER
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
1968–2008 Celebrating 40 Years
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Pattern Recognition 41 (2008) 567–574
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Optimization of an Hough transform algorithm for the search of a center

Julien Cauchie[a], Valérie Fiolet[b], Didier Villers[a,*]

[a]Laboratoire de Physicochimie des Polymères, Université de Mons-Hainaut, 20 Place du Parc, B-7000 Mons, Belgium
[b]Service d'informatique, Université de Mons-Hainaut, 20 Place du Parc, B-7000 Mons, Belgium

Received 28 March 2006; received in revised form 29 May 2007; accepted 2 July 2007

Abstract

We present improvements of an adaptative Hough transform algorithm applied to the search of a common center of circular or partially circular components present in an image. The efficiency has been considerably optimized by a continuous update of a list of voting points, in conjunction with the evolution of the accumulator size and position. The method was implemented as a plugin for the scientific open source image processing package ImageJ. Although initially designed for X-ray diffraction analysis, numerous other applications are quoted in different other scientific field, in image measurement techniques, industrial vision, and biometry, i.e. for iris localization.

Keywords: Hough transform; Center detection; JAVA ImageJ plugin

1. Introduction

Center detection is a fundamental task in some image processing applications. For example, X-ray diffraction, a powerful technique largely used in condensed matter studies (this is the case in our laboratory), yields two-dimensional patterns which must be quantitatively analyzed. As a prerequisite condition, the position of the stopped incident beam on such recorded images must be determined with a high precision. Due to the experimental configuration, this point is also the center of what crystallographers call Debye–Scherrer rings or arcs, which are sometimes only partially recorded. In order to obtain the coordinates of this point, we experienced simple geometrical methods, using well known triangles and circles properties, and find out a lack of accuracy in many cases. Moreover, all these simple methods require a long, eventually tedious intervention of the operator. We have thus developed a sophisticated method to automatically determine the position of the center of the diffraction pattern [1], using the principle of the Hough transform (HT).

Hough's concept was introduced in the field of pattern recognition as a powerful tool to detect alignments [2,3]. The idea of the HT is that points of an image (defined as the real space) produces points or trajectories in a so-called Hough (transformed) space, which describes the feature of the geometrical searched shape (i.e. slope and intercept for a line). Then the identification of peaks in the parameter space provides the features of detected patterns in the image. To avoid difficulties to determine these maxima, a variation on the initial model has been introduced, the so-called adaptive HT [4], which is an iterative, coarse to fine, accumulation technique.

The HT algorithm has also been generalized for the extraction of many types of shapes, more complex than straight lines, like circles and ellipses [6–9]. This is useful in applications like automated industrial inspection. These searched shapes have more features (i.e. center coordinates, radius, eccentricity, oblicity) which increase the dimensionality of the accumulator parameter space. As we are interested by a related pattern recognition problem somewhat restricted (the detection of only one common center), it has been possible to reduce the parameter space to a two-dimensional space which has the advantage to drastically decrease the needed amount of computational resources (concerning both time and memory size).

Recently Hendriks et al. [10] studied the detection of multiple hollow hyper-spheres in D-dimensional space and addressed the accuracy issue from a sampling perspective. They proposed an approach to reduce the memory requirements due to the higher

* Corresponding author. Tel.: +32 65373820; fax: +32 65373054.
E-mail address: didier.villers@umh.ac.be (D. Villers).

0031-3203/$30.00 © 2007 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2007.07.001

dimension of the parameter space, where they made the assumption that there is no concentric spheres or circles. Roughly speaking, their paper focuses on the detection of multiple well separated circles, whereas this study intends to find the unique center of multiple concentric rings.

In the present paper we propose important additional improvements of the previous algorithm [1]. In order to reduce the computation time, the number of voting points is optimized at each iteration. It is far and away the most interesting practical result of this paper since time improvements by a factor two or three has been observed in some applications of the algorithm. The evolution of the accumulator has also been improved. During the process, it happens that the rectangular accumulator must sometimes be translated instead of reduced, leading to the possibility of a cycling behavior. The detection of the occurrence of such cycle has been implemented, as well as a way to avoid infinite looping. Additionally, to ensure an easy communication and free use of the algorithm in the scientific community, the method has been implemented as a Java plugin for the open source image processing package ImageJ [11]. Both executable and source files can be easily downloaded from a publicly accessible internet server [12].

Clearly, it is also interesting to extend the use of this algorithm beyond the initial context. Circle detection is a more general related problem, important in many image processing applications, for example in industrial vision [13], iris recognition in biometry [14,15], and even ball recognition in soccer [16,17]. We will thus finally present some possible fields of application.

2. Proposed method

To a first approximation, a typical diffraction image (see Fig. 1) can be viewed as the sum of some more or less large and intense concentric rings or arcs superimposed on a continuous background of low level intensity. Mathematically, the ideal diffraction circles can be modelized by the following standard equations:

$$(x - x_0)^2 + (y - y_0)^2 = r_i^2. \quad (1)$$

Since we are dealing with digitized images, including noise and spreading of the signal, a direct recognition of circles and the determination of their center and radii are obviously impossible. On the other hand, HT is a standard method to carry out such task (see e.g. Ref. [6]). Its advantages are robustness to noise, distortions, and missing part of searched shapes. As its computational requirements strongly depend on the dimensionality of the parameter space, it is important to restrict the problem to a two-dimensional HT since all subsequent usual treatments of the image intensities only require the two coordinates of the center (x_0, y_0), but not the values of the radii r_i. Let us note that under these conditions the parameter space becomes congruent with the image space. This reduction can be done using a gradient-based method owing to the fact that the intensity gradients are essentially directed to the center of the diffraction pattern. We have introduced such a method in an adaptative coarse to fine HT algorithm [1] that we will

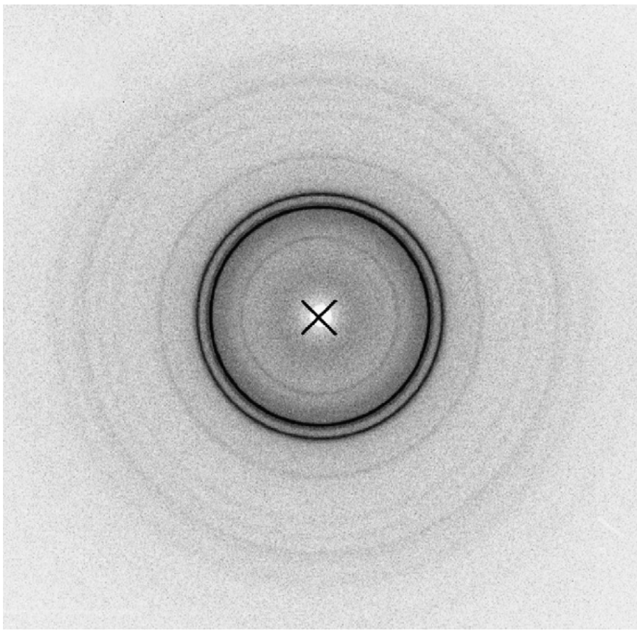

Fig. 1. Diffraction image characterized by isotropic circles (Debye–Scherrer rings). This image is used as the first test image in this paper. The definition is $590 * 588$ pixels. The cross indicates the position of the center found by the algorithm.

summarize here before introducing improvements in the next section.

Firstly, only a given percentage of the more intense points are taken in considerations as they produce the highest gradients in diffraction patterns. Then for these points the absolute value and direction of the gradient are computed using the Canny matricial operator [18] which presents the advantage to simultaneously perform a filtering of noise and the extraction of the gradient. This operator consists of two masks G_x and G_y used for convolution of the image. We have i.e. for x direction:

$$G_x(x, y, \sigma) = \frac{-x}{\sigma^2\sqrt{2\pi\sigma}} \exp\left[\frac{-(x^2 + y^2)}{2\sigma^2}\right]. \quad (2)$$

The standard deviation σ is determined as a function of the filter dimension so that the size of the convolution matrix include 3σ on both sides of the Gaussian distribution.

As it is quite long to directly calculate the intersections between all gradients, we use the so-called adaptive HT [4]. A given part of the parameter space is subdivided into cells. This matrix is initialized to zero and acts like an accumulator as the image is scanned. The accumulator cells which are crossed by the gradient lines are incremented by the absolute value of the gradient. The next step roughly consist in the search of the cell of higher value. A threshold is applied to the accumulator and the center of gravity of the accumulator is calculated. Applying the threshold means eliminating all the cells of smaller values than the accumulator thresholding ratio. It is useful to eliminate contributions of bad gradients (e.g. as produced by a beam-stop, nonradial gradients in anisotropic patterns, etc.). The coarse to fine principle is then applied by defining a new finer accumulator which has the size of the maximum cell. The procedure is repeated until the cell dimension reaches the required precision.

The transformation thus needs the five following parameters: the accumulator dimension, the filter dimension, the percentage of used pixels, the required precision, and the accumulator thresholding ratio.

3. Details and optimization of the algorithm

3.1. Use of the accumulator

This step is in fact more complex than mentioned in the previous section and can be split into three separate tasks.

Update of the accumulator: After initialization of the accumulator array and all temporary variables, the program scans the voting points in order to examine the intersects of gradient lines with accumulator cells. As this job must be done many times, we use an intermediary Boolean array, and the cells which are crossed by the gradient are ticked using an adaptation of the Bresenham line-drawing algorithm [5]. All accumulator cells corresponding to true Boolean cells are then incremented by the absolute gradient intensity.

Search for the center: In fact, due to the noise, it is not sufficient to consider the accumulator cell of maximum value. It is better to look for the multicellular rectangular region which maximizes the probability to find the center. This is done by this way:

(1) The accumulator is scanned to find the maximum value.
(2) A threshold, which is a user-defined percentage applied on the maximum value, allows to select the most important cells.
(3) A temporary Boolean array is defined where these cells are labeled.
(4) The connected components are identified.
(5) We determine the smaller rectangles including each connected components and compute the mean values of corresponding accumulator cells.
(6) We retain the rectangle of maximum mean value, which have the highest probability to contain the center.

Changing size and position of the accumulator: Considering the rectangle obtained in the previous step, there are different cases to carry out this task. The most obvious is when the rectangle is not adjacent to any of the accumulator boundaries. Then the new accumulator takes size and position of the rectangle. This method cannot be applied when the rectangle is against accumulator boundaries. Indeed some data normally useful to determine the position of the center could be outside the current accumulator. In order to optimize the accuracy of the process, we operate a translation of the accumulator to center it on the rectangle.

3.2. Optimized control of the voting points

In order to reduce the number of computations, the gradients are computed only once and memorized. Furthermore, the overall performance of the transform is strongly influenced by the number of voting points because at each time the accumulator is refreshed, it is necessary to find the cells which are crossed by gradient lines. It is thus highly advisable to manage this number as the accumulator size is reduced. This is done using linked list objects implemented in standard programming languages.

Initially, a linked list is created for voting points according to a given percentage provided by the user: only these more intense points are used and list items contain the coordinates of the point together with the angle and intensity of the gradient.

At a given step of the algorithm, we use a second linked list which contains the indexes of the voting points which are capable of being suppressed, i.e. for which the associate gradient direction does not cross the accumulator cells. If the accumulator size is reduced, all gradient lines outside the preceding accumulator will avoid even more so all later accumulators and the voting points of the second linked lists can be removed from the main linked list.

In the case where the accumulator retains its dimension and is translated, we cannot strictly apply the above-mentioned procedure. Indeed, as shown in Fig. 2, some gradient lines cross the new accumulator but not the preceding one. These lines are important for the determination of the center and cannot be neglected. This special situation is taken into account in the algorithm by adding two Boolean variables in the structure of the linked list of voting points capable of being suppressed, indicating the relative position and direction of the gradient lines around the accumulator (i.e. rising on the right). Then, following the direction of the translation of the accumulator, only the voting points with gradient lines in the opposite direction are suppressed.

3.3. Avoiding infinite loops

In bad circumstances, it could be possible that the accumulator undergoes successive translations and reverts to an already examined position, inducing a possibly infinite looping behavior. To detect it, a small linked list stores the upper left coordinates of successive accumulators while their sizes remains unchanged. After each translation, the new position is compared to preceding ones and in the case of an exact matching the accumulator is reduced by a half cell width close to its edges. On the other hand, the linked list is reinitialized each time the accumulator is reduced.

3.4. Initial and optimized algorithms

The initial and optimized algorithms can be schematically summarized as follows:

Algorithm (*optimized part are indicated with boldface characters*).

1. Reading of the image, width, and height.
2. Creation of the Canny convolution masks.
3. Calculation of intensity acceptability limits.

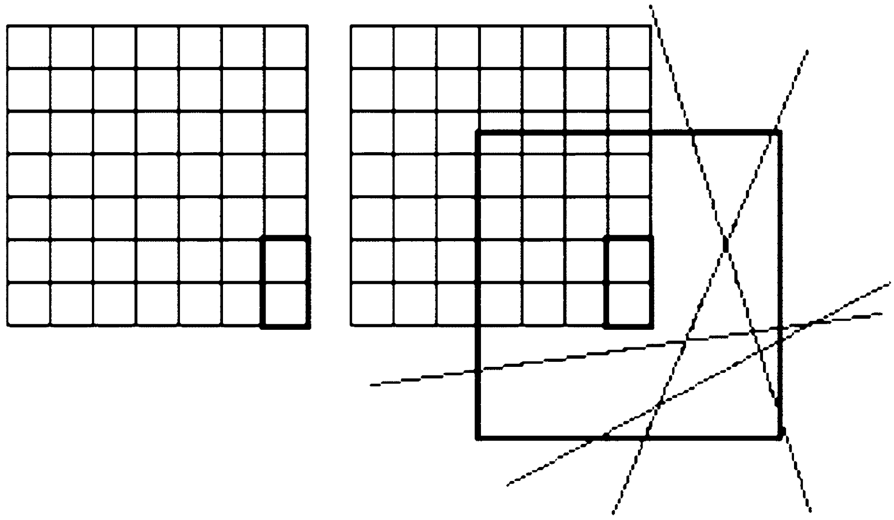

Fig. 2. Case where the accumulator retains its dimension and is translated, for which we need to consider in further step gradient lines which did not contribute to the accumulator during the previous step.

4. Setting up of the list of voting points.
5. While accumulator width is greater than the required precision:
   5.1 Initialization of the accumulator and temporary variables.
   5.2 Calculation of the size of the list of voting points.
   5.3 For each point in the list:
      5.3.1 Calculation of intersections with the accumulator.
      5.3.2 Update of the accumulator.
   5.4 Analysis of the accumulator to obtain the best connected component.
   5.5 **Detection of cycles in accumulator translations**.
   5.6 **Optimized control of the voting points** .
6. Calculation of the center.
7. Restitution of the results.
8. End.

3.5. Computer implementation

The method presented above was implemented as a plugin for the scientific open source image processing package ImageJ [11]. This package was developed using the Java programming language, which allows to run it on many operating system with a Java virtual machine installed. ImageJ can be easily extended by different existing plugins customized to specific image applications.

The "`Hough_Center`" plugin and its source can be freely downloaded by Internet from a page of our University website [12]. This page provides also detailed instructions for installation and a sample image file. As many specific Java objects has been created, the Javadoc tool has been used to generate an HTML documentation, also available, in order to allow easy later use of the source code.

As already mentioned, we used Java linked lists instead of arrays to store the list of voting points and the list of voting points which are capable of being suppressed, owing to one of the specific advantages of this data type: due to the internal use of pointers, the deletion of an item does not need to shift all the next ones, saving thus time and memory. To reach the expected time complexity $O(n)$, the standard Java object has, however, been modified (see also Ref. [12]).

Table 1
Definition of variables used for the expression of time or memory complexity

li	Image width (pixels)
hi	Image height (pixels)
la	Accumulator width (number of cells)
lf	Width of convolution masks (number of cells)
nv	Initial number of voting points
it	Number of iterations to reach the required precision
ts	Number of voting points which are capable of being supressed
tc	Number of coordinates in the history of accumulator translations

4. Theoretical and experimental computational efficiency

The time and storage complexities of the proposed method are analyzed on the base of the algorithm presented in Section 3.4. The theoretical time complexity (*TC*) can be split in as many terms as successive steps in the full algorithm. Considering the optimized steps, we obtain the following formula (see Table 1 for the signification of the used variables):

$$TC = O(li * hi) + O(nv) * O(lf^2) + O(it) * [O(nv) * O(la^2) + O(ts) + O(tc)]. \quad (3)$$

In first view, there is no significant change in the case of the optimized algorithm (the additional contribution concerns the two terms $O(ts) + O(tc)$). The complexity is even slightly greater due to the fact that it must be calculated at the worst. In general cases, due to the inherent feature of the adaptative HT, the accumulator width is usually a small constant (i.e. 3 or 5), and the iterative steps are the main part of the algorithm, so that we can simplify TC to $O(it) * O(nv)$.

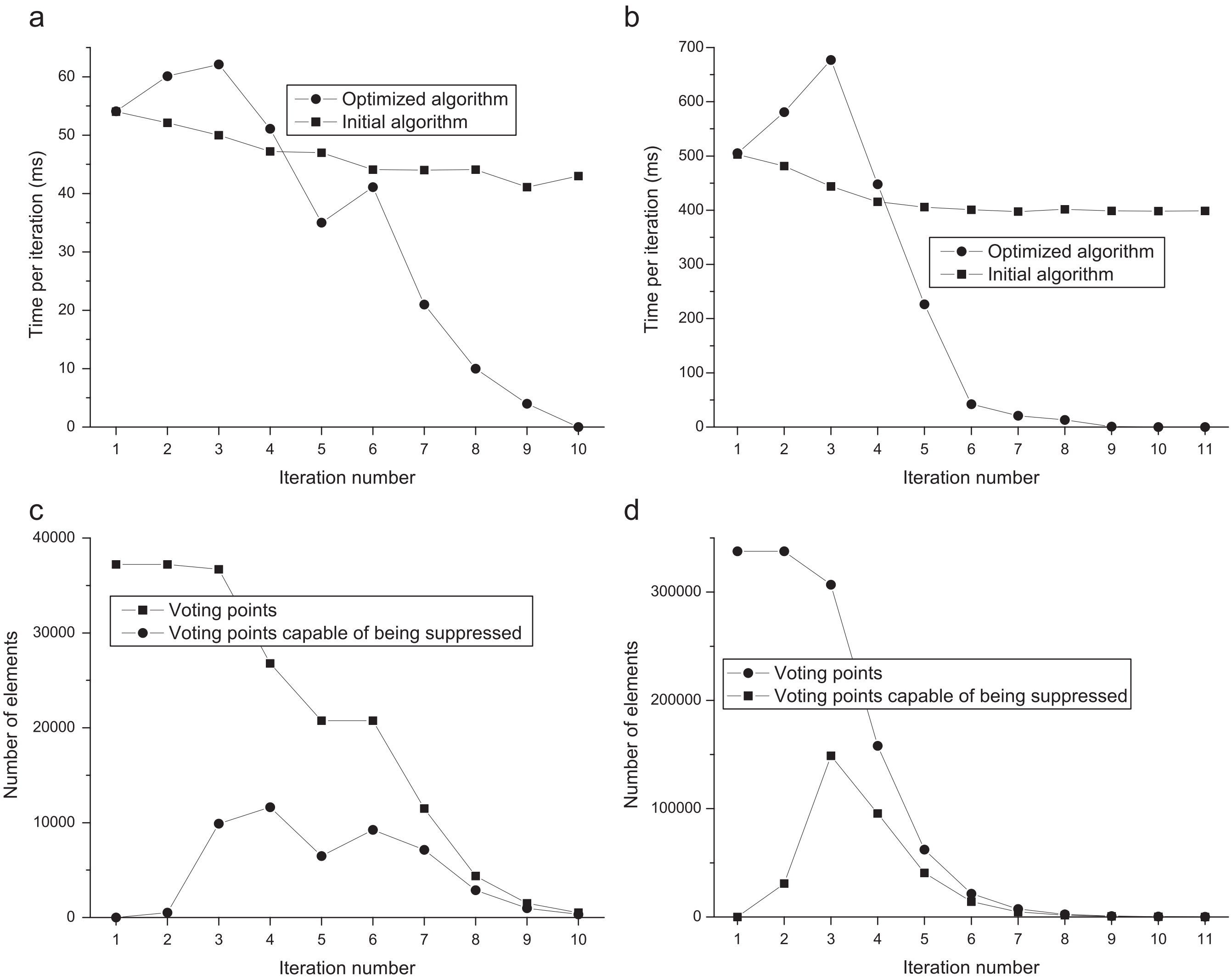

Fig. 4. Effective time complexity on two test images of small definition (see Fig. 1) and large definition (see Fig. 3). Time per iteration vs iteration number (a) for the small image, (b) for the large image. Number of elements in the linked lists vs iteration number (c) for the small image, (d) for the large image.

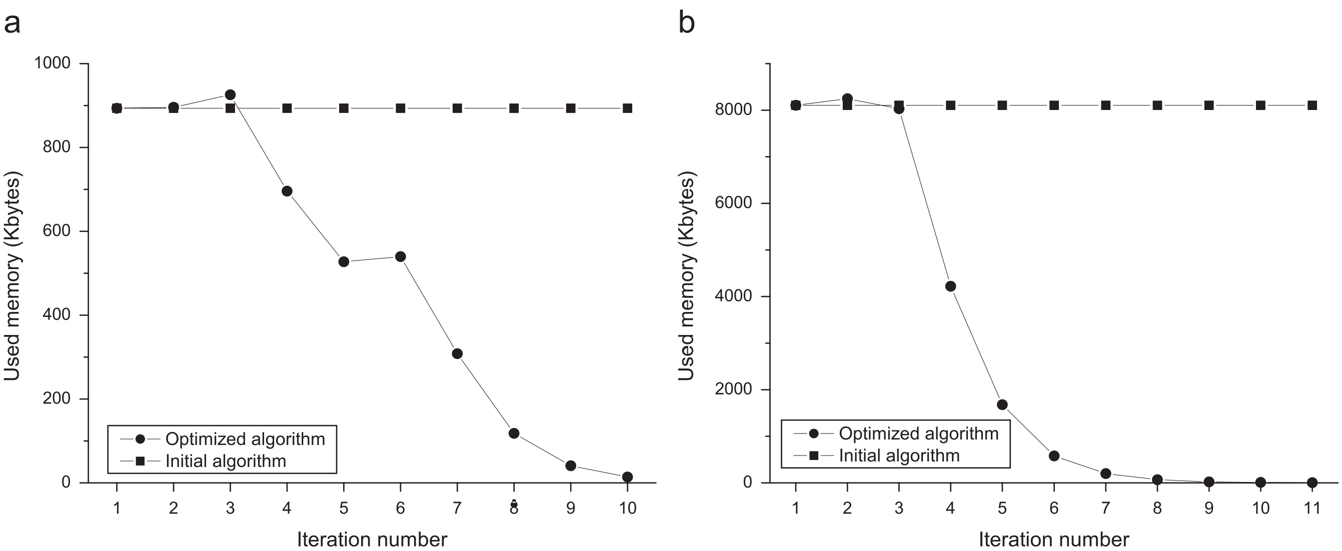

Fig. 5. Effective memory complexity on two test images of small definition (see Fig. 1) and large definition (see Fig. 3). Used memory vs iteration number (a) for the small image, (b) for the large image.

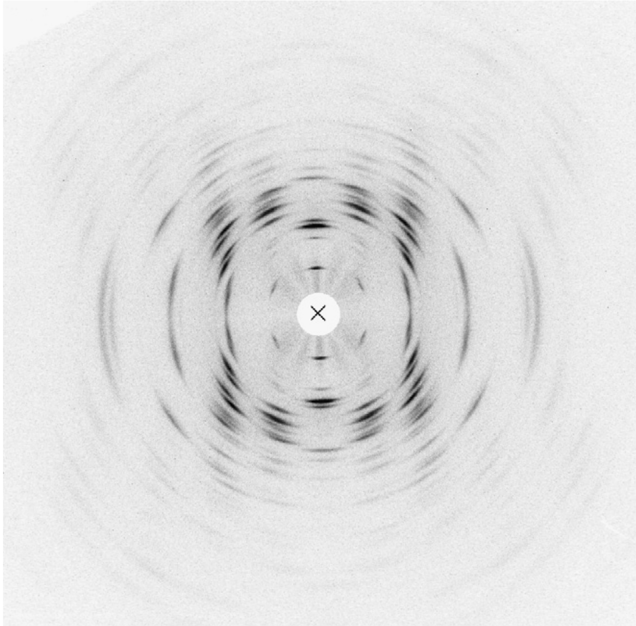

Fig. 3. Diffraction image characterized by arcs. This image is used as the second test image in this paper (2464 ∗ 2464 pixels).

In concrete terms, a lot of images for which we were looking for centers have allowed to drastically decrease the number of voting points as the optimized algorithm proceeded, compared to the initial algorithm. This had in turn an important and interesting effect by reducing the computation time. This has been validated on two test images. The first one (see Fig. 1) is characterized by isotropic diffraction circles (Debye–Scherrer rings) and have a quite small size (590 ∗ 588 pixels), whereas the second one (see Fig. 3) is bigger (2464 ∗ 2464 pixels) and characterized by diffraction arcs. An accumulator size of 9 cells has been used to insure a great number of iterations. For both images, Fig. 4(a) and (b) shows that during the four first iterations of the optimized algorithm, the execution time per iteration slightly exceeds those of the reference algorithm. However, during the subsequent steps, this overcharge is balanced and the total execution time is largely improved when using the optimized algorithm, even if an accumulator translation occurs during the process (as illustrated by the small shoulder of Fig. 4(a)). This is clearly explained by the reduction of the number of elements in the linked lists, as it is represented in both Figs. 4(c) and (d).

The theoretical storage complexity (*SC*) has also been evaluated and is given by the following equation:

$$SC = O(la^2) + O(lf^2) + O(nv) + O(ts) + O(tc). \qquad (4)$$

The same arguments as previously stated lead to consider that *SC* is roughly proportional to the initial number of voting points: $SC = O(nv)$. The effectively used memory has also been recorded during the search of center on the two test images. Fig. 5(a) and (b) shows that the memory required by the optimized algorithm strongly decreases with the number of voting points as the algorithm proceeds and even vanishes for the last steps. The little overrun during the first iterations remains negligible when compared to the constant storage requirement of the initial algorithm.

The accuracy of the method has been assessed by using synthetic test images of size 1000 × 1000 pixels. Circles characterized by a Gaussian intensity profile of 7 pixels width at half maxima have been generated with variable radius ranging from 25 to 250 pixels. The difference between the true center and the position find by the Hough method is always less than 2% of the radius. Let us note that the presence of many concentric rings as in X-ray diffraction images contributes to lower this difference. The precision has also been estimated by using the image presented on Fig. 1, adding a uniformly distributed noise. The standard deviation (SD) of the center position has been determined from sets of 17 random samples varying signal to noise ratio (SNR). For SNR = 10, 5, and 3, the obtained SD was, respectively, equal to 2.47, 3.81, and 5.12, suggesting that the corresponding variance is almost inversely linearly proportional to SNR.

5. Applications

Obviously, X-ray diffraction is not the only scientific or technical field producing images characterized by a common center. The present algorithm was thus tested and applied on various other documents. The quite satisfactory results are shown in Fig. 6. The first application concerns the analysis of small angle light scattering experiments, a technique quite close to diffraction. In this precise case (see Fig. 6(a)), the algorithm found the center despite the fact that the main circle was not fully recorded. The micrography of Fig. 6(b) show an instability in a thin polymer film, resulting in the growth of a circular dewetting front from an initial perturbation. The program allows to find the position of the initial hole in the film. This application could be interesting in the field of defect detection and analysis. The present work was also successfully applied to astronomical images, i.e. on a solar eclipse photography (Fig. 6(c)), and for the detection of the center of a spiral galaxy (Fig. 6(d)).

As already mentioned in the introduction, the circle detection is an essential task for the eye localization and further iris recognition in biometry, a very important technique for people identification. The use of circular HT has already been proposed for iris localization. Wildes [15] uses the HT algorithm with a three-dimensional parametric space defined by the iris boundary contour (the coordinate of the center together with the radius of this circle). On the other hand, Benn et al. [19] proposed to use a gradient decomposed HT, allowing thus the use of two-dimensional accumulator, but without the adaptative principle. The present algorithm could thus greatly improve the speed of this localization task. Fig. 6(e) shows the application of the program to find the center of an iris in a close high resolution face photography in the vicinity of an eye.

Finally, we present the application of the algorithm to a picture of a car wheel (Fig. 6(f)). Despite the asymmetric illuminating conditions, the program yields a correct value for the center of the wheel. There is obviously no specific limitation to the present program, the only requirement being the existence

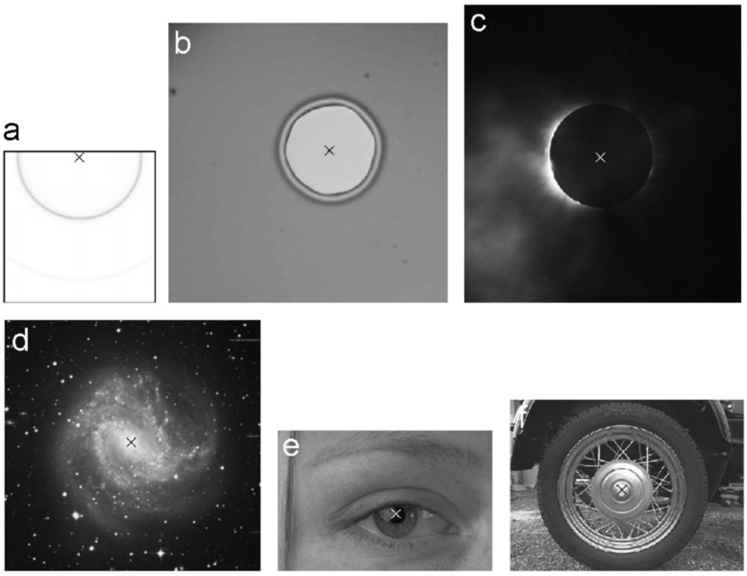

Fig. 6. Application of the present Hough transform algorithm to some different scientific and technical fields. (a) SALS, (b) growing circular hole obtained during dewetting experiments, (c) solar eclipse, (d) astronomical image of a spiral Galaxy, (e) eye and iris, (f) car wheel. A cross in each image shows the position of the center, as found by the computer program.

of intensity gradients directed to one common center, at least in a given part of the image.

6. Conclusion

We presented improvements of a HT algorithm applied to the search of a common center of circular or partially circular components present in an image. This was initially developed to allow the automatic retrieval of the center of X-ray diffraction patterns, for which it is important to determine the position of the stopped incident beam used for all subsequent digital signal analysis. In addition to the use of the classical adaptative coarse to fine version of the method, the accumulator Hough space is reduced to a minimal two-dimensional size and the efficiency has been considerably optimized by a continuous update of the list of voting points, in conjunction with the evolution of the accumulator size and position. In order to make easier the use and spreading of the method, it was implemented as a plugin for the scientific open source image processing package ImageJ. The user can adjust some parameters for a specific application. Although the initial field of application was restricted to X-ray diffraction analysis, numerous other applications are possible and quoted in different other scientific field, in image measurement techniques, industrial vision, and biometry, i.e. for iris localization. This general procedure could virtually be applied on any digital image, in conjunction with segmentation and other digital image processing pre-treatments.

Acknowledgment

The authors thank Véronique Bruyère for valuable comments concerning the expression of the theoretical computational efficiency.

References

[1] C. Dammer, P. Leleux, D. Villers, M. Dosière, Use of the Hough transform to determine the center of digitized X-ray diffraction patterns, Nucl. Instrum. Methods B 132 (1997) 214–220.
[2] P.V.C. Hough, Method and means for recognizing complex patterns, U.S. Patent 3069654, 1962.
[3] J.C. Russ, The Image Processing Handbook, fourth ed., CRC Press, Boca Raton, FL, 2002.
[4] J. Illingworth, J. Kittler, The adaptive Hough transform, IEEE Trans. Pattern Anal. Mach. Intell. 9 (5) (1987) 690–698.
[5] J.E. Bresenham, Algorithm for computer control of a digital plotter, IBM Syst. J. 4 (1) (1965) 25–30.
[6] E.R. Davies, A modified Hough scheme for general circle location, Pattern Recognition Lett. 7 (1987) 37–43.
[7] E.R. Davies, Finding ellipses using the generalised Hough transform, Pattern Recognition Lett. 9 (1989) 87–96.
[8] C.L. Huang, Elliptical feature extraction via an improved Hough transform, Pattern Recognition Lett. 10 (1989) 93–100.
[9] V.F. Leavers, The dynamic generalized Hough transform: its relationship to the probabilistic Hough transforms and an application to the concurrent detection of circles and ellipses, CVGIP Image Understanding 56 (3) (1992) 381–398.
[10] C.L.L. Hendriks, M. van Ginkel, P.W. Verbeek, L.J. van Vliet, The generalized Radon transform: sampling, accuracy and memory considerations, Pattern Recognition 38 (2005) 2494–2505.
[11] W.S. Rasband, ImageJ, U.S. National Institutes of Health, Bethesda, Maryland, USA, ⟨http://rsb.info.nih.gov/ij/⟩, 1997–2006.
[12] D. Villers, Professional web page at the Université de Mons-Hainaut, Belgium, ⟨http://staff.umh.ac.be/Villers.Didier/Hough/Hough_Center.html⟩, 2006.
[13] H.S. Kim, J.H. Kim, A two-step circle detection algorithm from the intersecting chords, Pattern Recognition Lett. 22 (2001) 787–798.
[14] J. Daugman, How iris recognition works, IEEE Trans. Circuits Syst. Video Technol. 14 (1) (2001) 21–30.
[15] R.P. Wildes, Iris recognition: an emerging biometric technology, Proc. IEEE 85 (9) (1997) 1348–1363.
[16] D. Ioannou, W. Huda, A. Laine, Circle recognition through a 2D Hough transform and radius histogramming, Image Vision Comput. 17 (1999) 15–26.

[17] T. D'Orazio, C. Guaragnella, M. Leo, A. Distante, A new algorithm for ball recognition using circle Hough transform and neural classifier, Pattern Recognition 37 (2004) 393–408.

[18] J. Canny, A computational approach to edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1986) 679–698.

[19] D.E. Benn, M.S. Nixon, J.N. Carter, Robust eye centre extraction using the Hough transform, in: J. Bigun, G. Chollet, G. Borgefors (Eds.), Proceedings of the First International Conference on Audio- and Video-Based Biometric Person Authentication, 1997, pp. 3–9.

About the Author—JULIEN CAUCHIE received the Master degree in Computer Science from the University of Mons-Hainaut in 2004, presenting a master thesis dedicated to the subject of the present paper. He is currently working in a software company.

About the Author—VALERIE FIOLET received the Master degree in Computer Science from the University of Lille in 2001, and will present her Ph.D. in few months. She is currently an assistant professor in the computer science department of the University of Mons-Hainaut. Her main research interests include GRID and Distributed Computings especially High Performance and Data Mining Computings and JAVA programming.

About the Author—DIDIER VILLERS received the Ph.D. degree in Chemistry from the University of Mons-Hainaut in 1989. He is now a professor and has his current research interests in physical chemistry of polymer.