To: ICS 2002, Workshop on Computer Systems

A Parallel Hardware and Associated Algorithms for Fast Detection of Circles and Symmetric Patterns

Ming-Yang Chern

Department of Electrical Engineering
National Chung-Cheng University
Min-Hsiung, Chia-Yi, Taiwan
Tel: 886-5-2720411~ex23273 Fax: 886-5-2720862
E-Mail: ieemyc@ccunix.ccu.edu.tw

Contact Author: Ming-Yang Chern

Abstract

A VLSI array processor, consisting of two sets of data-shifters and an array of AND-gates, is proposed to implement the hardwired midpoint finding and to support the fast locating of possible symmetric axes of image patterns. By further analyzing the line segments formed by these midpoints, the radii and centers of circles in the image can be found. The circuit of this design is simple and the operation is efficient. This detection scheme can be extended to the more general cases of ellipses and patterns with vertical (or horizontal) symmetric axis.

Keywords: array processor, circle detection, parallel processing, VLSI, symmetry

1. Introduction

In computer vision applications, the circle detection is often needed. An example of this is the detection of oil tanks or circular installations from the satellite pictures, while the recognition of circular-shaped parts on the industrial production line is another example. For circle detection, a popular approach is to use the Hough transform [1-4] to find out each circle's center and radius statistically from its parameter space. The Hough-transform method for circle detection is simple and robust (under noises). Nevertheless, it is time-consuming for its voting process, which requires the computation of all the candidate circle centers' addresses for each edge point and the increment of all the referred accumulator memory locations. Given an image of size n x n, the time complexity for circle detection Hough transform would be of $O(n^4)$.

To reduce the processing time, two approaches are generally adopted. One is to modify the Hough transform (or to use other circle detection schemes), the other is to apply parallel processing. Quite a few modified versions of Hough transform have been proposed [5-10][12]. Many of them use edge directions while some combines the use of other properties of the circle to reduce the complexity of Hough transform's voting process. In general, these methods take advantage of certain properties of circles to reduce the processing time of the image. Nevertheless, with the more specific analysis on part of extracted image features to reduce the algorithm's complexity, it becomes more difficult to perform parallel processing further on these algorithms. And the chance for further improvement of the processing speed is usually limited. The paper [11] takes the advantage of symmetry to detect circles and ellipses. Yet its method is suitable for an image with only few simple patterns. For pictures with many objects, the locating of all midpoints would take too much time (unless it is running on specialized parallel hardware) and the finding of symmetric axes from the "midpoint map" is somewhat complicated.

In this research, we are interested in finding the circle detection scheme, other than the Hough transform, that can possibly be performed by parallel processing. For the scheme found, we are also interested in designing VLSI parallel processor for its implementation, and for understanding how simple and how efficient its hardware realization can be.

In this paper, we present the result of the above investigation under which a design of one-dimensional array processor for locating the centers of circles in the image is proposed. The proposed parallel hardware processes the image data row by row (or column by column). The design of this array circuit is quite simple and suitable to be implemented either as an independent VLSI chip or on the same VLSI chip of the image memory. By joining appropriate circle verifying algorithm or verifier hardware, the circles in the given image can be detected in short time. The performance of this parallel processor and its supporting algorithms is analyzed in this paper, and is found to have the time complexity within two orders of magnitude n. Some simulated results of detection are also presented to show the correctness and efficacy of our image processing scheme.

2. The Proposed Approach

Observing the geometry of a circle, the line of perpendicular bisector of any chord must pass through the circle center. Based on this property, the intersection of two perpendicular-bisector lines with respect to two arbitrary non-parallel chords must be the circle center. In another word, the circle center can be located by finding the intersection point of such two perpendicular-bisectors. Yet a problem exists that before the circle is found, we do not know which two edge points in the image form a chord. Nor we know which two chords belong to the same circle. Thus we can not use two chords to find a circle center directly.

To solve the problem, we take a statistical approach similar to that of Hough

transform. For a circle of radius R, there will be about as many as 6*R edge points on the circle. From the chords formed by the pairing of these edges points, all the intersection points of their perpendicular-bisectors would fall (near and centered) at the center of this specific circle. For the random pairing of other chords (formed by the random pairing of edge points other than from the same circle), their intersection points would be scattered rather than concentrated at a few circle centers. Thus by counting the intersection points (of pairs of perpendicular-bisectors for arbitrary chords) falling at different locations, the locations of circle centers would accumulate much higher count. Then by locating all the above-threshold local peaks in the accumulator data array, the circle centers can be found.

To implement the above scheme, we choose to use only the horizontal chords and vertical chords instead of the arbitrary pairing of all edge points. There are several important reasons for doing this. The reasons are listed as follows. (1) The calculation of the perpendicular-bisector of the horizontal chord or vertical chord is much simpler and faster than that of an arbitrary chord. (2) With such restriction, the total number of chords is much reduced. It is reduced by an order of magnitude. (3) The perpendicular-bisectors of the horizontal chords and vertical chords are vertical and horizontal, respectively. The calculation of the intersection for horizontal and vertical lines is much simpler. (4) The calculation of the intersection for horizontal and vertical lines is more accurate than the general case of arbitrary chords. This is especially apparent when compared with the case of two chords near parallel. (5) With the number of chords much reduced and the fact that parallel lines do not intersect, the total number of perpendicular-bisector's intersection points due to the pairing of the horizontal chords and vertical chords is greatly reduced. It is reduced by two orders of magnitude comparing with respect to the case of the arbitrary chords, yet it is still large enough to form a good statistical figure for finding the circle centers.

With the constraint of using only the horizontal and vertical chords, the edge-point pairing process can be much simplified. Yet we still need some efficient algorithm or scheme to obtain all these possible pairings. On the other hand, the calculation of the perpendicular-bisectors takes advantage of the horizontal and vertical chords; for that we only need to calculate the midpoint of each chord to express the line of its perpendicular bisector. Based on the technique often used in computer graphics [14], the midpoint can be located by adding the coordinates of two end points and then shift the resultant value to the right for one-bit (to avoid the division operation).

In our research, an ingenious way to acquire the midpoint coordinates without using the adder is proposed. Taking advantage of the fact that all chords are either horizontal or vertical, one of the coordinates of the midpoint has been known in advance. By shifting two edge points in the same row (or same column) toward each other, their midpoint position can be detected at the place where these two edge points meet. We also notice that for all other edge points in the same row of the image, the above edge-point shifting can be performed (in parallel) at the same time, if they reside in one row of data buffer and their shifting direction is the same. Thus by having two copies of the edge-point data in a row, each copy in a set of buffer registers, we may shift the data in two opposite directions such that any pair of edge points in the same row can meet to reveal the position of their midpoint.

Each revealed midpoint of an edge-point pair in a row indicates a perpendicular-bisector of a horizontal chord. Accumulating the count of these revealed midpoints at each x-coordinate position for all the rows in the image, it gives the circle centers a profile of their distribution on the x-axis of the image. A similar reasoning holds for collecting the midpoint count column-wise in the vertical direction. The accumulated count at each y-coordinate position gives the profile of the circle centers' distribution on the y-axis of the image. And from the intersection of these (horizontal and vertical)

profiles' peak coordinates, we can locate the center points of all circles in a small set of possible positions.

To estimate the radius for each circle and to pinpoint the position of each circle center more accurately, we take the approach of analyzing the midpoint map in this research. Though further process is needed, it would take only little time relative to the usual circle detection. Besides, some special parallel hardware can be designed to facilitate the verification of circles centered at these limited positions. The fast acquisition of midpoints and the fast locating of circle centers based on our proposed scheme, speed up the detection of circles much. Its hardware design, the one-dimensional processor array, is to be illuminated in the next sections.

3. Parallel Hardware for Midpoints Extraction

As mentioned, we propose to use two shift registers to process the image (binary edge pixel) data in a row. Given an image of n x n pixels, the two shift registers must have n bits each. While the 'preset data' inputs of both shift registers are connected to a buffer register of n bits used to hold a row (or a column) of data output from the source edge-map image. For each row (or column) of the image data waiting to be processed, they must be read into the buffer register. At the beginning of a new row-processing cycle, the buffered image data will be copied into both shift registers. Soon after this step, the data in the two registers will be synchronously shifted, one position per clock pulse, in opposite directions.

The detection of the midpoint of an edge-point pair, (thus the perpendicular bisector of this chord,) depends on the encounter of the two edge points in our shifting scheme. To reveal such encounter at the middle position, we use the two-input AND gate to check the edge data in the coincident (and near-coincident) registers, assuming that the edge point is represented as 1. Note that for two neighboring pixels, their midpoint is half way in between the pixels. Hence, for image data of n bits, we will

need as many as 2n – 3 AND gates for the 2n – 3 possible positions of the midpoint. The circuits of the n-bit row data buffer, two shift registers and the required AND gates all together form a processor array. The circuit configuration is depicted in Figure 1.

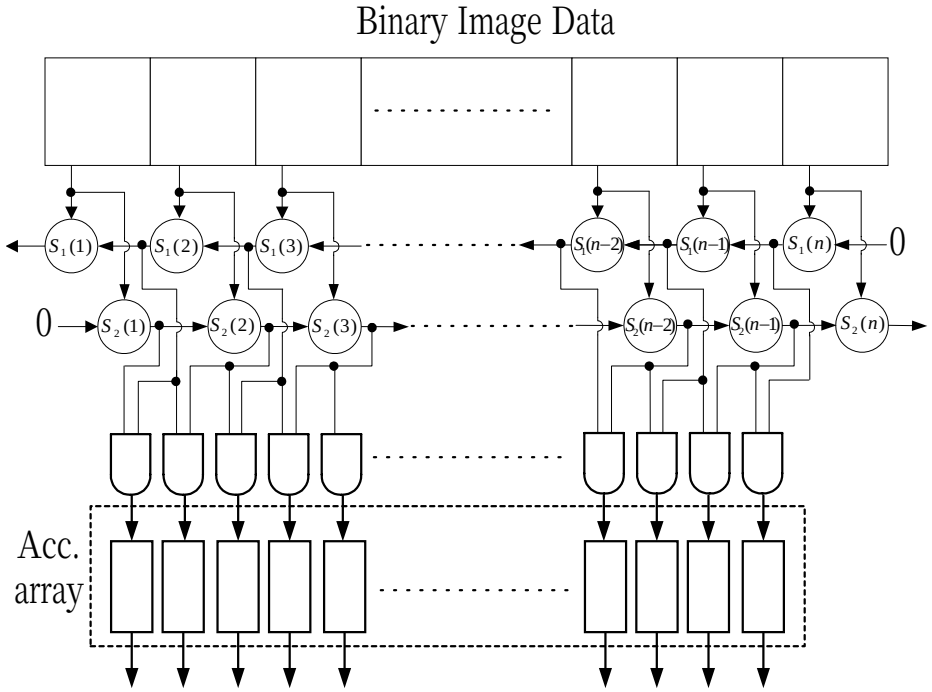

Figure 1. The circuit array of the proposed parallel processor

In this proposed one-dimensional array processor, the output of the bit position $S_1(k)$ of the left-shifter is designed to AND with the outputs of $S_2(k-1)$ and $S_2(k)$, respectively. In another word, the output of the bit position $S_2(k)$ of the right-shifter is to AND with the outputs of $S_1(k)$ and $S_1(k+1)$, respectively. The AND-gate having inputs from $S_1(k)$ and $S_2(k)$ reveals the bisector detection at the pixel coordinate of k. While the AND gate joining $S_1(k)$ and $S_2(k-1)$ is for pixel coordinate of k-0.5 , and the gate for the pair $S_1(k+1)$ and $S_2(k)$ stands for the k+0.5 position. Note that in Figure 1, the shift register bits $S_1(1)$ and $S_2(n)$ are actually unused.

At the output of each AND gate, there is a block of supporting circuits. The accumulator of the supporting circuits counts the number of occurrence of the

midpoint detection. And this implements the perpendicular-bisectors' statistics collection over all the rows (or columns) of the whole image. The 2n – 3 blocks of supporting circuits are also linked to a memory called "midpoint map" that is of size n x (2n-3) with each cell of single bit. This memory collects the midpoint bits detected in each row (or column) and forms a map of horizontal (or vertical) midpoint distribution for the given image.

Beside the midpoint locating, the edge pairing is another issue we have to address. For any given edge point in a row (or column), we have to find out all its possible pairing with other edge points in the row (or column). Our proposed dual shifter scheme not only reveals the midpoint positions, but also performs a comprehensive edge-point pairing for all edge points in the row (or column) at the same time. The proof is as follows.

Given an arbitrary edge pixel in a row, it will be duplicated as two copies and stored at the same position in each of the two shift registers. With the latter shifting operations, the edge-point copy stored in the left-shifter would pair with all the edge pixels to its left, meeting them at each pair's middle position. Similarly, the edge-point copy stored in the right-shifter would pair with all the edge pixels to its right. Thus for any one edge point, its pairing (to form a horizontal chord) with all other edge pixels in the same row will be realized under our dual shifter scheme, and all the perpendicular-bisectors' positions will be revealed through the attached AND array. That means, our proposed parallel processor and the shifting operations would reveal all the perpendicular-bisectors' positions for all the possible horizontal chords in the given row. When operated for the image data in a column, the same argument holds for the perpendicular bisectors' positions of all possible vertical chords in the given column.

As the above two n-bit data shifters are synchronously shifted in opposite directions, for the n-bit data, it takes less than n/2 shifts for any edge pixel to pass by

all other pixel data. To be more precise to say, it takes (n-1)/2 shifts for one row (or one column) of data, if n is an odd number. For an even number of n, then n/2 – 1 shifts are required. If the radii of circles are known to be less than r_{max}, then for locating the circle centers only r_{max} shifts rather than n shifts are needed in the midpoints finding process for each row (or column).

4.Supporting Hardware for Symmetric Axis Detection

As mentioned, the output of each AND gate connects to a supporting block that contains an accumulator circuit. Every time after the shift registers are shifted, a new set of midpoint detection signal will appear at the AND gate outputs, which must be recorded or accumulated in some way. To design such accumulator circuit, a simple way is to use only a counter, as shown in Figure 2.

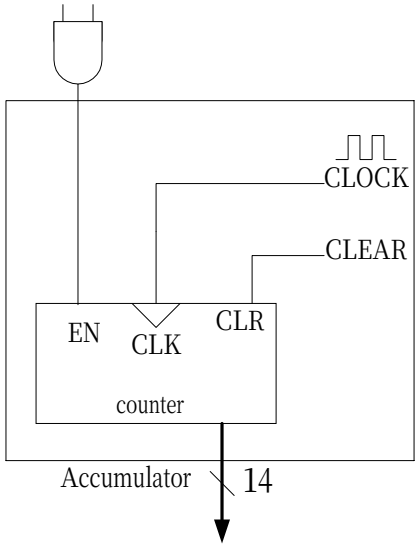

Figure 2. The accumulator circuit based only on a counter

In the design of Figure 2, the output of AND gate is connected to the ‘enable’ control of counter. While the clock pulse to signal the counter circuit is synchronized with the clock of the shifter registers (perhaps with some proper phase delay). When the AND gate output is 1, the counter will be incremented by the oncoming clock

pulse; while for the case of 0, the counter value will remain unchanged. The counter can be reset by the CLEAR signal, which is usually issued at the beginning of a new image frame processing.

Based on the above accumulator circuit, three operations must be accomplished in each clock cycle of the shifting step. To ensure reliable result, the period of the shifter's clock pulse must be longer than the total time required for the data shifting, the AND-gate response plus the delay of the counter response. Another issue that concerns us is whether we desire to count the total number of midpoints detected for the same row of shifter data at the same coordinate position or not. The advantage of counting every occurrence of midpoint detection is that the multiple bisectors of concentric circles will be shown in the accumulation. Nevertheless, hazard may occur when a horizontal line is encountered (continuous edge-points appear as the shifter data). Accumulating every occurrence of bisector detection, in this special case, would give a large number of misleading counts. This may obscure the peaks in the latter peak extraction process, which in turn may largely increase the difficulty of the circle center detection.

To remedy the problem, we propose to add a latch to the accumulator circuit. The new circuit, as shown in Figure 3, has the output of the AND gate connected to the latch, while the output of the NOR-gate formed latch is connected to the 'enable' terminal of the counter. Every time a new row of data is loaded into the two shift registers, a 'clear' signal will issued to reset all the latches in the accumulator circuits array. At the end of the row processing (after performing all the necessary data shifting for the row), a clock pulse will be given to counter, which in turn will increment the counter value by one if the latch output is 1. And as the former design, the counter can be reset by the CLEAR signal, which is usually issued at the beginning of a new image frame processing.

With the new circuit and operation scheme, the counter is incremented for the

midpoint detection only once per row of the shifter data. After processing all rows (or columns) of the image frame, the memory of “midpoint map” indicates the horizontal (or vertical) midpoint distribution, while the values in the accumulator array give the horizontal (or vertical) profile of the midpoint distribution. The peaks in the horizontal (or vertical) profile indicate the x-coordinates (or y-coordinates) of the most probable vertical (or horizontal) symmetric axes.

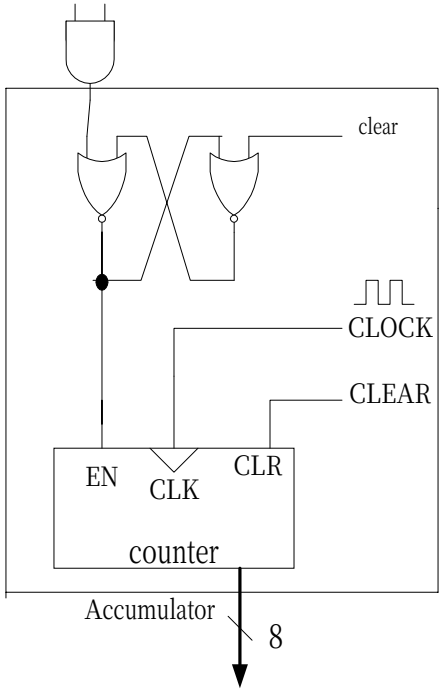

Figure 3. The accumulator circuit with latch-buffered detection

5. Circle Center Estimation and Circle verification

Based on the proposed array processor, the midpoint map of the given edge image can be generated. The horizontal midpoint map can be obtained by connecting the row output of the image memory to the array’s input data buffer, while the vertical one can be obtained from the same hardware as well by connecting the input to the column output of the image memory. An example of the midpoint map after processing the image of Figure 4(a) is shown. In Figure 4(b), the red ink shows the

horizontal midpoints while the blue one indicates the vertical midpoints. The histogram profiles of the two midpoint maps, obtained from the data of the accumulator array, give the possible locations of the vertical and horizontal symmetric axes, respectively. Figure 4(c) shows only those midpoints that contribute to the peaks in these two profiles.

The locations of peaks in the profiles indicate the coordinates of the horizontal and vertical symmetric axes detected from the patterns in the given image. For a circle, the circle center must be at the cross point of a horizontal and a vertical symmetric axis. Thus to find circles, we check only the cross points between the two sets of peak-coordinates of the midpoint profiles. If a cross point is within both a vertical symmetric axis in the horizontal midpoint map and a horizontal symmetric axis in the vertical midpoint map, then this cross point can be a candidate circle center.

To detect the circle, we also need to find its radius and verify the existence of this circle in the image. There are two schemes adopted to estimate the radius in our design. With the first scheme, we trace the symmetric axes in four directions starting from the cross point, and record the length of each traced line segment. To avoid the missing points caused by noise, the gap of one or two pixels is allowed in the line tracing. From the lengths of the four line segments, we take the most consistent value (with more than one occurrence) as the radius to verify the circle in the image. In another case when all the four lengths are different, then we take the smallest two values as two candidate radii of the circle. It may also happen that two length values both occur twice. In this case, both values will be adopted as the candidate radius.

In order to save processing time, our algorithm verifies the above candidate circles by first checking the existence of edge points at dozen specific angles along the circle locus. If the edge points appear at the majority of these positions, then all the remaining locus points will be checked. Through the detailed checking, as the candidate circle locus shows a large percentage of edge-point count, the existence of

the circle with the specific center and radius can be confirmed.

Yet chances exist that some circles remain undetected. For example, the smaller concentric circle in Figure 4(a) is still hidden from the above detection. By removing all the edge points of the confirmed circles from the original image, two new midpoint maps can be generated using the same parallel hardware (i.e., our proposed array processor). The new midpoints taken at the new profiles' peak coordinates are marked in yellow ink as shown in Figure 4(d). Repeating the previous process for the new midpoint maps, the remaining circles can be detected further.

For applications that have many circles enclosing or overlapping each other, we may adopt another scheme in determining the radius. The algorithm checks the possible radius of a circle by first tracing a few selected rays out of the cross point. The rays are selected at specific angles such as 0º, 30º, 45º, 60º, 90º et al. so that the x- and y-coordinates of the checking points can be easily calculated for specific radius. The checking process goes on with the incremental radius until the maximum possible radius or the length of its symmetric axis is reached. During the process, whenever the checking points for a specific radius can find enough matching edge points in the image, the circle of that radius will be checked in details to verify its existence. In this way, no circles can be missed in the detection.

To verify the existence of a circle in the image, we count the number of edge points along the calculated locus of the circle. When the total number of counts exceeds a certain percentage of the maximum possible counts, the circle is confirmed. The threshold depends on the noise level we allow and the completeness of the circle desired. As to the fast addressing of all checking points based on the given circle center and radius, the Bresenham's circle algorithm can be used [14]. Considering the small round-off errors in the image processing and the determination of radius and circle center, the verification process can be proceeded by varying the center location and radius size slightly in order to have more accurate detection. Figure 4(e) shows

the detection of circles based on the first radius selection scheme. For the ellipses in Figure 4, the radius based on the short axis (horizontal axis) is tested; and as a correct result, no circle is detected.

6. Detection of Ellipses and Symmetric Patterns

The detection of ellipses having symmetric axis parallel to the coordinate axis is similar to that of circles except that one more parameter is added. From the horizontal and the vertical symmetric axis analysis, the cross point is used as the center of ellipse while the length of the long axis and the short axis may be revealed. Using the "midpoint ellipse algorithm", the locus of the ellipse to be verified can be calculated efficiently. And counting the number of edge points along the calculated locus gives the percentage of locus matching. This, in turn, indicates if the ellipse exists or not. Figure 4(f) shows the detection of the two ellipses.

For the more general detection of arbitrary ellipse, we may base on the two midpoint maps (produced by our array processor) to find the ellipse's two symmetric axes (not necessary horizontal or vertical in their directions). Following the method in paper [11], the ellipse can be detected.

For the general patterns with vertical or horizontal symmetric axis, the midpoint map produced by our array processor provides the clue. The clue of vertical or horizontal line segment in the midpoint map indicates the probable existence of such patterns. We may acquire all the edge-point pairs that produce the line segment by back tracing and then keep only those edge points that form continuous shape. The back tracing can be performed either by software or by specialized parallel hardware (by adding some provision to our array processor). If some group of edge points remains after the above filtering, the symmetric pattern(s) are detected.

7. Simulated Results and Performance Analysis

In addition to experimental result of Figure 4, we have also tested our scheme on the computer-generated patterns of Figure 5(a) and real image of coins in Figure 6(a). Figure 5(d) shows successful detection of the overlapping circles as well as the enclosed circle. With the support of hue differentiation, Sobel operation, and thinning, all the circles in the coin's color image can be detected. Note that there is a pair of concentric circles and one of the coins occludes another coin. Though the circle of the occluded coin is not complete, it can still be detected by our symmetry scheme.

The advantage of our proposed array processor is that no matter how complex the image is, the generation of two midpoint maps takes only about n^2 shifts or less (if we know the maximum size of circles in advance). In the current VLSI technology, a shift-AND cycle of 20 nanoseconds can be easily achieved. Thus for an image of 512 x 512, it takes about 0.5 ms to generate the two midpoint maps for an image. The remaining steps depend on how many objects are in the image. Taking the test of Figure 5 as an example, there are 36 cross points checked yet only 7 of them are required to find the radius. Based on our first radius-selection scheme, only seven (center, radius) pairs are invoked to verify the circles. And all the seven circles are confirmed (detected). For the test on coins, there are 16 cross points checked and only 4 of them are required to find the radius. Based on our first radius-selection scheme, only four (center, radius) pairs are invoked and four circles are confirmed. And from our priori knowledge that there may be concentric circles, we regenerate the two midpoint maps (costing 0.5 ms) after removing the edge points of the four confirmed circles. With one cross point and only one (center, radius) pair to check, the last circle is verified. All the five circles are detected correctly.

With the support of our proposed array processor, the time spent for circle detection is quite limited. The time spent in tracing the real circle's locus in our experiment is unavoidable and the time overhead spent in roughly checking the wrong

circle locus is only a small percentage. While the time required by each pair of midpoint maps is only 0.5 ms. It takes total time within several milliseconds for most of the multiple circle and ellipse detection.

8. Conclusion

This paper presents a design of VLSI array processor that facilitates the generation of the midpoint maps and the fast locating of the symmetric axes for the given image patterns. Based on the extracted vertical and horizontal symmetric axes, most circles in the image can be detected. With the formation of midpoint line segments, the ellipses can be detected as well unless the image is too complicated. From the midpoint maps, the symmetric patterns with vertical or horizontal symmetric axis can also be revealed. The algorithms we apply here share a common ground of using the horizontal and vertical midpoint maps, while our parallel hardware design offers such a basis with very high efficiency in processing speed. And this is the major contribution of this paper.

So far, our proposed array processor can not support the detection of symmetric patterns with arbitrary symmetric axis. And this can be a topic for future research.

Acknowledgement

This work was supported by the National Science Council, ROC, under grant NSC 90-2215-E-194-010.

References

[1] P.V.C. Hough, “Method and means for recognizing complex patterns,” U.S. Patent 3069654, 1962.

[2] R.O. Duda and P.E. Hart, “Use of Hough transformation to detect lines and curves in pictures,” *Comm. ACM*, vol. 15, pp. 11-15, 1975.

[3] C. Kimme, D.H. Ballard, and J. Sklansky, “Finding circles by an array of accumulators,” *Comm. ACM*, vol. 18, pp. 120-122, 1975.

[4] G. Gerig and F. Klein, “Fast contour identification through efficient Hough Transform and simplified interpretation strategy,” *Proc. 8th Int. Joint Conf. Pattern Recognition*, pp. 495-500, 1986.

[5] J. Illingworth, and J. Kittler, “A survey of the Hough transform,” *Computer vision, Graphics and Image Processing*, vol. 44, pp. 87-116, 1988.

[6] E.R. Davies, “A modified Hough scheme for general circle location,” *Pattern Recognition Letters*, vol. 7, pp. 37-43, 1988.

[7] H.K. Yuen, J. Princen, J. Illingworth, and J. Kittler, “Comparative study of Hough Transform methods for circle finding,” *Image and Vision Computing*, vol. 8, pp. 71-78, 1990.

[8] R. Chan and W.C. Siu, “New parallel Hough transform for circles,” *IEE Proceedings-E*, vol. 138, pp. 335-344, 1991.

[9] R.K.K. Yip, P.K.S. Tam, and D.N.K. Leung, “Modification of Hough transform for circles and ellipses detection using a 2-dimensional array,” *Pattern Recognition*, vol. 25, pp. 1007-1022, 1992.

[10] P. Kierkegaard, “A method for detection of circular arcs based on the Hough transform,” *Machine Vision and Applications*, vol. 5, pp. 249-263, 1992.

[11] C.T. Ho, and L.H. Chen, “A fast ellipse/circle detector using geometric symmetry,” *Pattern Recognition*, vol. 28, pp. 117-124, 1995.

[12] N. Guil and E.L. Zapata, “Lower order circle and ellipse Hough transform,” *Pattern Recognition*, vol. 30, pp. 1729-1744, 1997.

[13] S. Kumar, N. Ranganathan, and D. Goldgof, “Parallel algorithms for circle detection in images,” *Pattern Recognition*, vol. 27, pp. 1019-1028, 1994.

[14] D. Hearn, and M.P. Baker, Computer Graphics, 2nd Ed., Prentice-Hall, Chapter 3, 1994.

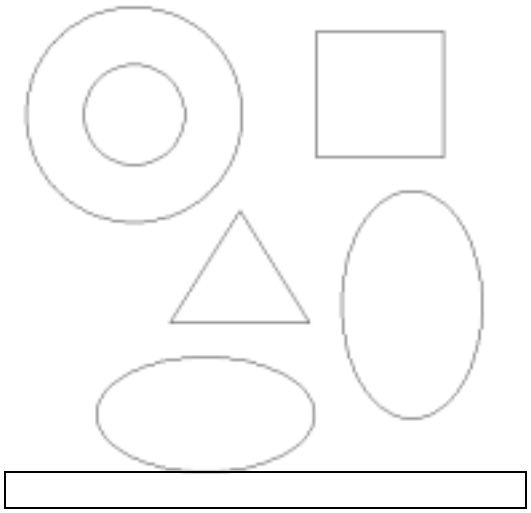

Fig. 4(a)

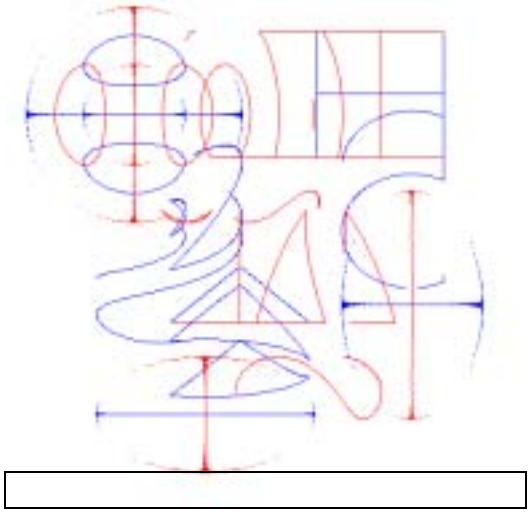

Fig. 4(b)

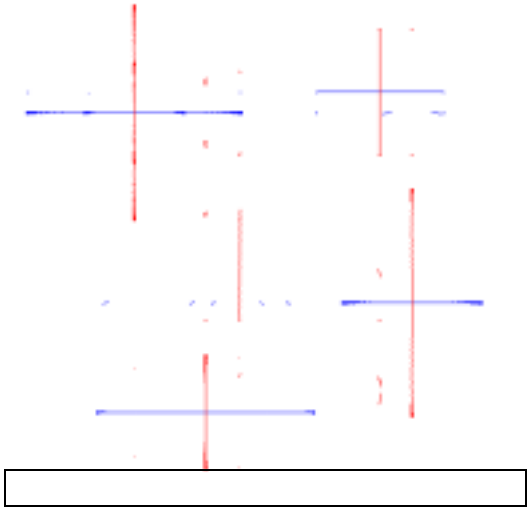

Fig. 4(c)

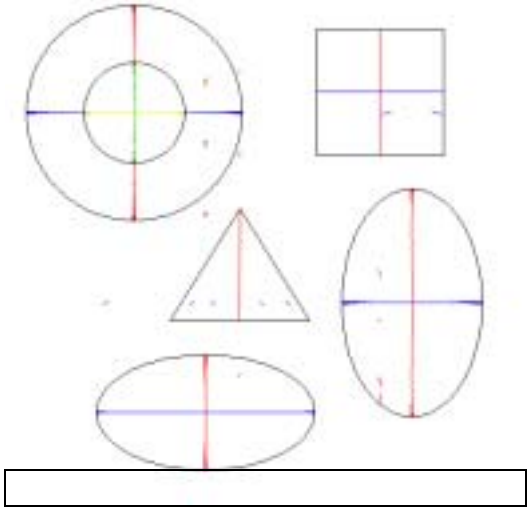

Fig. 4(d)

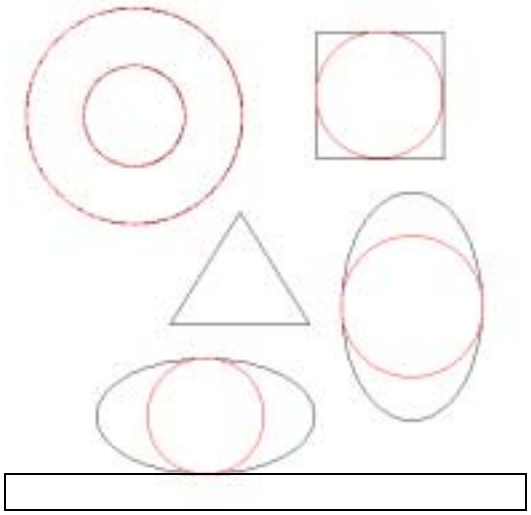

Fig. 4(e)

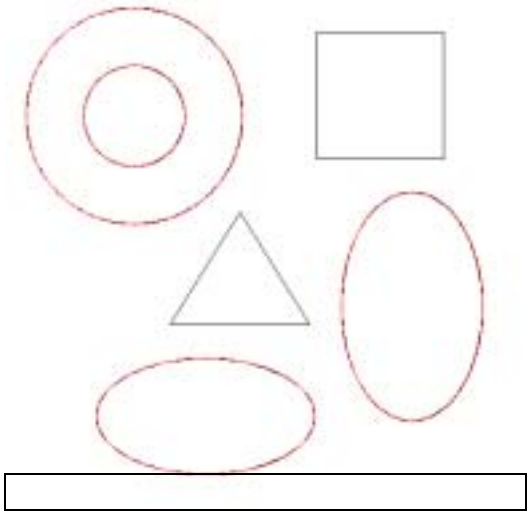

Fig. 4(f)

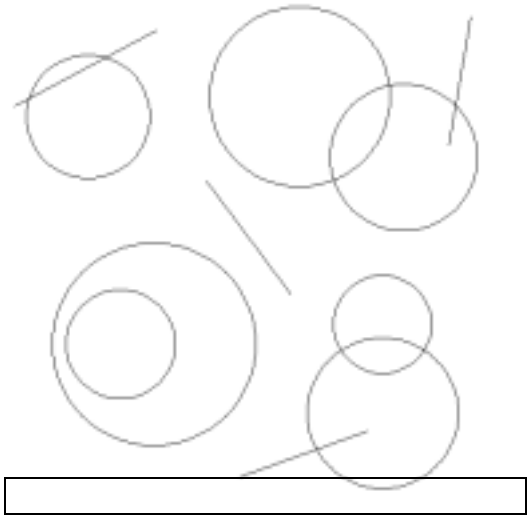

Fig. 5(a)

Fig. 6(a)

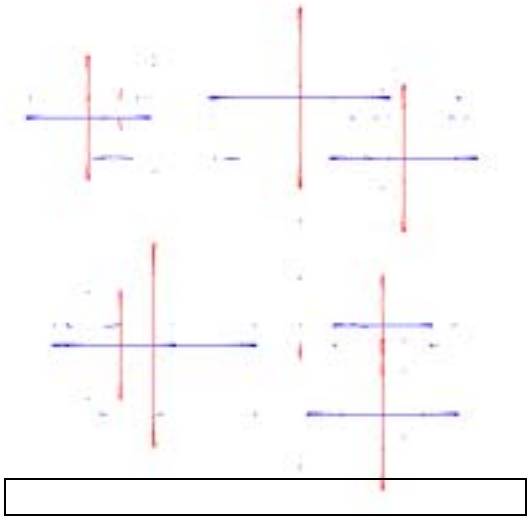

Fig. 5(b)

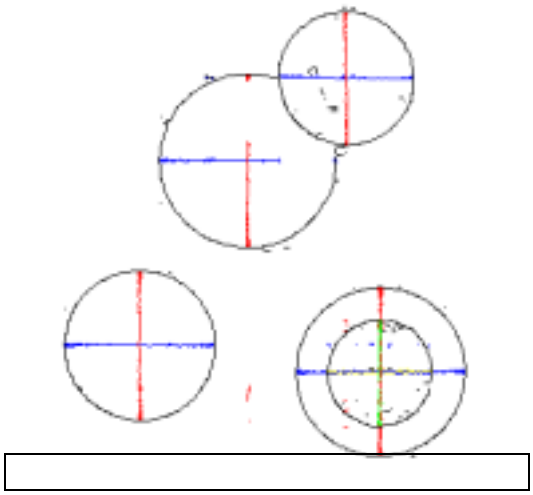

Fig. 6(b)

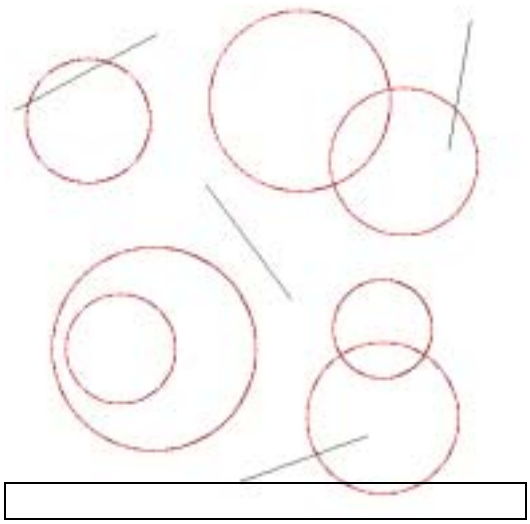

Fig. 5(c)

Fig. 6(c)