Efficient Algorithms for the Hough Transform on Arrays with Reconfigurable Optical Buses*

Sandy Pavel and Selim G. Akl
Department of Computing & Information Science, Queen’s University
Kingston, Ontario, Canada, K7L 3N6

Abstract

This paper examines the possibility of implementing the Hough transform for line and circle detection on arrays with reconfigurable optical buses (AROBs). It is shown that the Hough transform for line and circle detection in an $N \times N$ image can be implemented in a constant number of steps. The costs of the two algorithms are $O(N^2 p)$ and $O(N^2 p^2)$, respectively, where $p$ is the magnitude of one dimension in the parameter space. These values are optimal with respect to the time complexity of the best known sequential algorithms.

1. Introduction

The Hough Transform (HT) was introduced as a method of detecting the shape of object boundaries in binary images [7]. In order to achieve this, a number of parameters are associated to the patterns of interest. The main objective of the HT is to transform the spatially-extended patterns, represented by the edge pixels in the image, so that they produce spatially-compact features in the associated parameter space. The global shape detection problem is converted into a local, less difficult, peak detection problem.

Suppose that we have an $N \times N$ digitized image which contains a set of $n$ edge pixels, $n \leq N^2$, obtained after some preprocessing (edge detection) step. A pixel can be specified by its coordinates $(x, y)$, $0 \leq x, y \leq N - 1$, in the image plane. Herein the origin of the system of coordinates $(0,0)$ is considered to be the bottom left corner of the image plane. Thus, the $n$ edge pixels are represented by the set \{$(x_i, y_i)$ | $0 \leq i \leq n - 1$\} for which pixel $(x_i, y_i)$ is “1”. For line detection, the edge pixel $(x_i, y_i), 0 \leq i \leq n - 1$, is transformed into a sinusoidal curve in the $(\rho, \theta)$-parameter plane which is described by:

$$\rho = |x_i \cos \theta + y_i \sin \theta|$$

(1)

The pair $(\rho, \theta)$ represents a parameterization of a line in the image space where $\rho$ is the normal distance from the origin to that line and $\theta$ is the slope of this normal. In the image space $\rho \leq \sqrt{2N}$. If $p$ values are used to represent $\rho$, then $p \leq \sqrt{2N}$. The slope $\theta$ of the normal is in the interval $[0, \pi]$. Due to the limited resolution and for simplicity the number of distinct values for the $\theta$ parameter is also considered to be $p$, with $p \leq \sqrt{2N}$. Thus, the $(\rho, \theta)$-space is $p \times p$ and has associated a $p \times p$ array, $count(p, \theta)$. For each edge pixel $(x_i, y_i)$ and each $\theta_k$ the value of $\rho_{i,k}$ is computed using equation (1) and the value corresponding to the particular entry of $count[\rho_{i,k}, \theta_k]$ is incremented. Finally, a threshold function can be applied to all the elements of $count$. The time complexity of the sequential algorithm for computing the $count$ array is $O(p n)$. The circle detection algorithm has $O(p^2 n)$ time complexity, due to an increased parameter space. In practice it is possible to have the entire $N \times N$ image as input for the algorithm. As $p = O(N)$ the time complexities for line and circle detection become $O(N^3)$ and $O(N^4)$, respectively. The same orders of magnitude are obtained when the number of edge pixels is $n = O(N^2)$.

Parallel algorithms for line detection using the HT have been designed for different architectures [7]. The algorithms given in [8, 12] have $O(p \log(N/p))$ time complexity and use an $N \times N$ reconfigurable mesh. In [8] the time complexity is improved to $O((p/N) \log N)$ and $O((p/N^{1/2}) \log N)$ by increasing the two-dimensional array to $N \times N^2$ and $N^{1.5} \times N^{1.5}$, respectively. When the input is reduced to the $n$ edge pixels, constant time algorithms are obtained in [9] and [13] using reconfigurable arrays of dimensions $p \times n \times n \times n$ and $n \times \log^2 n \times p \times p$, respectively. As $n$ can be in the order of $O(N^2)$, the two architectures could use $O(p N^6)$ and $O(p^2 N^2 \log^2 N)$ processors, respectively. In [10] three parallel HT algorithms are given for the circle detection. The best of these algorithms runs in $O(N^2)$ time and uses an $N \times N$ mesh architecture.

We propose two algorithms for line and circle detection.
using the Hough transform. Both algorithms take a constant number of steps and are implemented on the AROB model [15].

2. The AROB Model

Consider a linear array of \( n \) processors connected to an optical bus as in Fig. 1. Each processor is connected to the bus through two directional couplers. One is used to write data on the bus and the other to read the data from the bus. Each processor sends data on the upper (transmitting) segment of the bus and reads messages from the lower (receiving) segment of the bus. During a write cycle, the data written by a processor into the bus propagate as indicated with arrows in Fig. 1, and may be read by any subsequent node on the bus. Due to the directionality of the signal propagation and the predictable delay of the signal, the same bus may be used to transmit messages between other nodes in the same time. Let us consider that each message is \( b \) bits long. Each bit is represented by an optical signal of width \( w \) seconds for a binary value of 1. The absence of such a signal represents a 0. Two conditions are essential for this kind of transmission: (1) all transmissions are synchronized and (2) the length of the optical path on the waveguide between any two adjacent nodes, \( d \) in Fig. 1, is larger than or equal to \( b u/w \), where \( c_g \) is the velocity of light in the waveguide. These two conditions ensure that the signals corresponding to different messages, which travel in a waveguide in the same direction at the same time, do not physically overlap at any point on the waveguide, i.e., are space multiplexed. The end-to-end propagation time is \( \sigma_b = 2n \tau \) seconds, where \( n \) is the number of processors in the linear array and \( \tau \), is the time taken for a light pulse to traverse the optical distance \( d \). This system is called linear Array Processors with Pipelined Buses (APPB), and its principles have been introduced in [11, 4].

Two-dimensional arrays with optical pipelined buses have been proposed in order to reduce the number of processors connected to a linear optical pipelined bus [4, 16, 17, 6]. The array with reconfigurable optical buses we propose in [14] uses the basic architectural and functional structure of a classical reconfigurable network, RN, [1], see Fig. 2. The communication system of this network uses optical waveguides and optical switches in order to allow the implementation of the pipelined optical communication mechanisms as they are introduced in [11, 4]. We call this structure Array Processors with Reconfigurable Optical Buses, AROB. It is shown in [14] that the local switch setting provides a mechanism for building arbitrary APPB-like linear structures on the AROB and that an AROB with \( n^2 \)-processors can simulate any \( n^2 \)-processor RN operation in a constant number of steps. For more details on the AROB see [14].

It is shown in [5, 14] that the sum of \( n \) bits can be computed on a linear APPB with \( n \) processors in a constant number of steps.

3. The HT for line detection on AROB

Assume that we have an \( N \times N \times p \) AROB which stores a digitized \( N \times N \) image as the input data. Each edge (feature) pixel \( (i, j) \), for \( i \) and \( j \) in \( \{0, \ldots, N-1\} \), has a value of “1” and it is stored in the base array by processor \( P_{i,j,0} \). All other background pixels are “0”. We associate a sub-array of processors \( P_{*,*\_k} \) with each value of the angle parameter \( \theta_k \), \( 0 \leq k \leq p-1 \). The symbol “*” is used in order to simplify the notation and represents all the values of the index it replaces in the notation. In the first step of the algorithm each processor \( P_{i,j,0} \) in the base plane broadcasts pixel \( (i, j) \) on column \( P_{i,j,*} \). Each sub-array \( P_{*,*\_k} \), \( 0 \leq k \leq p-1 \), gets a copy of the input image. It is assumed that each processor of the \( k \)th sub-array \( P_{*,*\_k} \) knows the value of \( \theta_k \) (otherwise it can compute it in \( O(1) \) time).

The next steps of the algorithm are performed independently and in parallel by all arrays \( P_{*,*\_k} \), \( 0 \leq k \leq p-1 \). For each pixel \( (i, j) \), \( P_{i,j,*\_k} \) computes \( \rho_{i,j} = [i \cos \theta_k + j \sin \theta_k] \). All the edge pixels which have the same value \( \rho_{i,j} = r \) describe a potential line in the image space. This line is represented in the parameter space by the \( (\rho, \theta_k) \) pair. In order to compute the value of the \( \text{count}[\rho, \theta_k] \) accumulator we need to count the number of edge pixels which have associated the same \( (\rho, \theta_k) \) pair. To simplify the presentation we restrict the angle values in the range \([0, \pi/4]\). By using the symmetry properties, all the other values of \( \theta_k, \pi/4 < \theta_k \leq \pi \), can be treated similarly.
In a plane $P_{x,y,k}$ all the processors which have computed and stored the same values for $p$ can be connected in a linear APPB-like structure. For this we rely on a number of results previously obtained in [12]. The following observations are made for all $k$ such that $0 \leq \theta_k \leq \pi/4$: (1) For any $j$, with $0 \leq j \leq N - 1$, the $p$-distances satisfy $\rho_{i,j} \leq \rho_{i+1,j}$ for $0 \leq i \leq N - 2$. It can be shown that no more than two consecutive values of $p$ in row $j$ can be equal. (2) For all $i, j$, $0 \leq j \leq N - 1$ and $0 < i \leq N - 2, 0 \leq \rho_{i+1,j} - \rho_{i,j} \leq 1$. (3) For all values of $i, j, 0 \leq i, j \leq N - 2, \rho_{i,j} \neq \rho_{i+1,j+1}$. (4) If $\rho_{i,j} = \rho_{i,j+2}$ for $0 \leq i \leq N - 1$ and $0 \leq j \leq N - 3$, then $\rho_{i,j} = \rho_{i,j+1} = \rho_{i,j+2}$. Therefore, all processors which have computed the same value of the normal distance $p$, for some angle $\theta_k$, can be connected in a linear APPB-like structure. This operation requires only local communications, each processor changing information with only some of its four neighbors. More specifically, the possible cases are depicted in Fig. 3. No two distinct buses are connected to the same port of a processor, the actual setting of a switching system being in one of the states depicted in Fig. 2b. A possible bus configuration is depicted in Fig. 4 for $\theta_k = \pi/6$. One end of each bus is located in the bottom row or the rightmost column of the array. This is designated as the leader processor of the linear APPB and is marked with a star in Fig. 4. The leader selection requires only a constant number of computations and/or local communication. Next we apply the APPB bit sum algorithm in each linear bus constructed this way, and the values of $\text{count}[\rho, \theta_k]$ are obtained and stored by each leader. This process is done in parallel by $P_{x,y,k}$, for all values of $\theta_k, 0 \leq k \leq p - 1$.

**Claim 1** The Hough transform for line detection in an $N \times N$ image can be implemented in a constant number of steps on an $N \times N \times p$ AROB.

The cost of this algorithm is $O(N^2p)$. This is optimal with respect to the time complexity of the best known sequential algorithm, when the $N \times N$ image is given as input (or when $n = O(N^2)$). As $p = O(N)$ the total number of processors (and the cost) is $O(N^3)$.

4. Circle detection on the AROB

The HT for circles maps each edge point in the input image to a cone in the three-dimensional parameter space. A circle is given by the equation:

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

where $(x_0, y_0)$ is the center of the circle and $r$ is its radius. In the case of circle detection the parameter space is described by $(x_0, y_0, r)$. As the parameters can be unbounded, for practical reasons we limit the center of the circle to lie within the image boundaries, i.e., $x_0$ and $y_0$ take values in the interval $[0, N - 1]$. These constraints are satisfactory for most practical applications [10]. As a consequence any circle satisfying the above requirement has an integer radius of up to $\lceil \sqrt{2}N \rceil$. Therefore, each parameter is assumed to be quantized in $p$ values with $p = O(N)$.

The HT algorithm for circle detection is similar to that for the line. The accumulator $\text{count}$ associated to the parameter space is initialized to zero. Then, for each edge pixel $(x,y)$ in the image, all the parameter entries $\text{count}[x_0, y_0, r]$ for which equation (2) is satisfied are incremented. In the end, the number stored by an entry $\text{count}[x_0, y_0, r]$ represents the number of pixels lying on a circle of radius $r$ and center $(x_0, y_0)$. The overall time complexity of this sequential algorithm is $O(N^4)$. In [10] three parallel algorithms are given for the circle detection. All circles of a given radius are traced in parallel, updating in the same time the parameter accumulator associated to each of these circles. The tracing technique is actually the midpoint circle scan-conversion algorithm, given in [3].

Assume that we have an $N \times N$ image stored on an $N \times N$ AROB, one pixel per processor. The first objective is to connect to the same bus all the processors which lie on the same scan-converted circle. In the next step the number of edge pixels associated to that circle is determined. We trace all the circles with the same center, say $(x_0, y_0)$, and different radii in parallel. For simplicity and without loss of generality we assume that $(x_0, y_0) = (0, 0)$. 

![Figure 3. Possible connections of a processor to the bus associated to $\rho_*$ for $\theta_k \in [0, \pi/4]$.](image3.png)

![Figure 4. Mesh configuration for $\theta_k = \pi/6$.](image4.png)
Claim 2 Let \((x, y)\) be a pixel with \(x \geq 1\) and \(y \geq 1\). There is a circle with center \((x_0, y_0) = (0, 0)\) and integer radius \(r, r \geq 1\), such that \((x, y)\) is a point in that circle’s scan-converted representation if and only if \(z_1 \leq r \leq z_2\), where 
\[ z_1 = \left[ \left( (y - 0.5)^2 + x^2 \right)^{1/2} \right] \quad \text{and} \quad z_2 = \left[ \left( (y + 0.5)^2 + x^2 \right)^{1/2} \right]. \]
If such an \(r\) exists then it is unique and \(r = z_1 = z_2\).

Proof: Using Fig. 5.a the proof of the first part of this statement is immediate. Let \(M_N\) be the point at the middle vertical distance between \((x, y)\) and \((x, y + 1)\) and \(M_S\) the point at the middle vertical distance between \((x, y)\) and \((x, y - 1)\). There is a circle with integer radius \(r\) and center \((0, 0)\) such that \((x, y)\) is a point in its scan-converted representation if and only if that circle intersects the vertical which passes through \((x, y)\) somewhere between \(M_N\) and \(M_S\). This condition can be written as \(y_r, \leq y_r < y_{r2}\), where \(y_{r1}\) and \(y_{r2}\) are real values and are the radii of the two circles, with centers in \((0, 0)\), which pass through \(M_N\) and \(M_S\), respectively. Since \(y_{r1} = y - 0.5\) and \(y_{r2} = y + 0.5\) one can verify that 
\[ y_{r1} \leq y_r < y_{r2} \]
\[ \leq \left( (y - 0.5)^2 + x^2 \right)^{1/2} \leq \left( (y + 0.5)^2 + x^2 \right)^{1/2}. \]
As \(r\) must be an integer, this expression becomes \(z_1 \leq r \leq z_2\). The second statement can be proved by contradiction, assuming that there are two circles of integer radii \(r\) and \(r + 1\) such that the circles intersect the vertical in \(x\) between \(M_N\) and \(M_S\). Let \(y_{r1}\) and \(y_{r1+1}\) be the two points of intersection in the second octant between these circles and the vertical in \(x\). The distance between \(y_{r1}\) and \(y_{r1+1}\) must be strictly greater than 1 and \(y_{r1} \leq y_r < y_{r1+1}\) \(\leq y_{r2}\). This means that the distance between \(y_{r1}\) and \(y_{r2}\) is strictly greater than 1, which is impossible. Thus, if such a circle exists for pixel \((x, y)\) then it is unique.

With the results in Claim 2 we have established a technique which can be used by each processor in the array to determine if its associated pixel is part of any scan-converted circle with center \((x_0, y_0)\) and integer valued radius. Furthermore, if such a circle exists then it is unique and its radius can be determined in constant time. The entire test takes constant time. Each processor which has determined a valid value for \(r\) must connect itself to other processors with the same \(r\) in order to establish the linear bus associated to that circle, if such processor exists. We restrict our discussion to the second octant, \(\pi/4 \leq \theta_x \leq \pi/2\), and describe a processor \(P_{x,y}\) as active if it is connected to the bus associated to the corresponding circle and in the same time its pixel is part of the scan-converted circle. It is also possible for a processor to be connected to a bus without the corresponding pixel being part of the associated circle, see Fig. 6. This is called inactive processor.

Claim 3 Let \((x, y)\) be a pixel for which there is a circle with center \((x_0, y_0) = (0, 0)\) and integer radius \(r\) such that \((x, y)\) is part of that circle scan-converted representation. Then, either \(P_{x+1,y}\) or \(P_{x+1,y+1}\) is the next active processor adjacent to \(P_{x,y}\) in the clockwise scan-converted order on that circle.

Proof: For a proof we can use Fig. 5.b. The two possible extreme cases, when \((x, y)\) is part of some scan-converted circle, are represented by the two arcs of circle which intersect the vertical passing through \((x, y)\) in \(M_N\) and \(M_S\). One can easily verify that these circles intersect the vertical in \(x + 1\) somewhere between points \(A\) and \(B\). Thus, either processor \(E(P_{x+1,y})\) or \(SE(P_{x+1,y+1})\) are the active neighbors of \(P_{x,y}\). An example of parallel bus construction for the first quadrant is depicted in Fig. 6.a. The different processor connections for the second octant are shown in Fig. 6.b. The only exception from these rules is made for the interconnection of the four processors around \(P_{x_0,y_0}\), i.e., \(P_{x_0+1,y_0}\), \(P_{x_0,y_0-1}\), \(P_{x_0-1,y_0}\), and \(P_{x_0,y_0+1}\). For this interconnection \(P_{x_0-1,y_0-1}\), \(P_{x_0-1,y_0+1}\), \(P_{x_0+1,y_0+1}\) and \(P_{x_0+1,y_0-1}\) are used as intermediate, inactive processors.

A bus can correspond to either a full scan-converted circle or to an arc of a scan-converted circle. Examples of circles
with center \((x_0, y_0)\) which have at least one arc contained in the image are depicted in Fig. 7.a. In order to apply the APPB bit sum algorithm a leader processor must be specified for each constructed arc or circle bus. The leader processors are represented with shaded circles in Fig. 7. There are two different cases. In the first case the entire scan-converted circle with center \((x_0, y_0)\) and radius \(r\) is contained in the image. Thus, all associated processors are connected to a linear bus and \(P_{x_0,y_0+r}\) is designated as the leader of this bus. All these leaders are grouped in the rounded box marked with “1” in Fig. 7.b. In the second case, a scan-converted circle is represented by up to four arcs, the first processor in the clockwise order of each arc can be set as the leader. The other four rounded boxes in Fig. 7.b contain the leaders associated to these arcs. The process of leader detection uses only local computations and can be done in a constant number of steps. The partial values of the \(\text{count}[x_0, y_0, r]\) accumulators are obtained by applying the sum algorithm in each linear bus associated to a circle or arc of circle. Each active processor participates with its pixel value in this summation step. The inactive processors use the 0 value. The leader processor participates with its pixel value in this summation.

**Claim 4** The Hough transform for circle detection in an \(N \times N\) images can be implemented in a constant number of steps on an \(N \times N \times p^2\) AROB.

5. Conclusions

In this paper it is shown that the Hough transform can be implemented in \(O(1)\) steps. The costs of the two algorithms for line and circle detection are \(O(N^3)\) and \(O(N^4)\), respectively. These values are optimal with respect to the complexity of the best sequential algorithms and compare favorably to the best known parallel algorithms. In general, the line detection algorithm can be implemented in \(O(t)\) time on an \(N \times N \times \frac{t}{r}\) AROB with \(t \leq p\). The circle algorithm can take \(O(t)\) time on an \(N \times N \times \frac{p^2}{r}\) AROB, \(t \leq p^2\).

**References**


![Figure 7. (a) Relative positions of circles within an image. (b) The leader processors and their grouping.](image)