IMPROVEMENT AND SYSTOLIC IMPLEMENTATION OF THE HOUGH TRANSFORMATION FOR STRAIGHT LINE DETECTION

H. F. Li, Derek Pao, R. Jayakumar

Computer Science Department
Concordia University
Montreal, Quebec

ABSTRACT

Hough Transformation (HT) is an efficient method to detect straight lines in digital pictures. In the conventional HT, pixel contiguity is not accounted for, and this leads to the following drawbacks: (1) actual length of line segments cannot be computed; (2) colinear line segments cannot be distinguished; and (3) very often, false lines are detected and short lines go undetected. This paper proposes a simple and efficient way to perform contiguity check. A systolic architecture that implements this modified transform is presented. The area-time complexity of the proposed architecture was shown to be superior to the conventional sequential algorithm. The modified transform was applied to detect straight edges in a road map and satisfactory results were obtained.

KEYWORDS: Hough Transform, pixel connectivity, systolic array, bucket, area-time complexity.

1. Introduction

Hough Transformation (HT) is an elegant technique for detecting colinear feature points in binary digital pictures [2]. The Hough transform technique has been used successfully in many different applications [3,5-8] that involve detection of straight lines. Using the parameterization proposed by Duda and Hart [2], a straight line in the image plane can be defined by the angle θ of its normal and its algebric distance r from the origin (fig. 1). The equation of the straight line is

$$x \cos\theta + y \sin\theta = r$$

The value of θ is restricted to the interval $[0,\pi)$ and r is restricted to the interval [-R, R], where R is the size of the retina.

In computing the transform, the r-θ space is quantized and represented by a 2-D accumulator array. Each bucket in the r-θ space (r_i,θ_k) corresponds to a bar-shaped window of infinite length and of width Δr in the image (fig. 2). In the conventional HT, each accumulator cell (bucket) records the number of feature points (edge pixels) that fall within the corresponding window. Line segments are located by detecting peaks (local maxima) in the accumulator array.

Three major drawbacks can be be easily observed in the conventional HT:

1. actual length and location of the line segment cannot be obtained;
2. colinear line segments cannot be distinguished; and
3. very often false lines are detected and short lines go undetected.

In applications that require more precise information, such as VLSI photoresist inspection [8], such inaccuraies of the transform are not tolerable. The first two drawbacks are due to the fact that connectivity of the edge pixels is not accounted for in the transformation. The third drawback is due to the inherent shortcoming of the thresholding operation used to detect peaks in the accumulator array.

This paper describes modifications to the conventional HT to account for pixel connectivity and a systolic implementation of the modified transformation. The new transformation will compute the end points of a line segment in addition to the normal parameters r and θ. By checking the contiguity of edge pixels, unrelated pixels (random noise or pixels that actually belong to another line) can be rejected. Each line segment is characterized by its two end points, hence, a bucket can hold multiple colinear lines. Knowing the end points of a line, we can determine the "best" fit to the group of "colinear" edge pixels by counter-checking with the

lines sitting in adjacent buckets. In this way, we can avoid the confusion that usually arises when thresholding is applied.

Section 2 will discuss the modified transform and the systolic architecture in detail. Section 3 will analyze the area-time complexity of the architecture and section 4 discusses the line detection algorithm. Some experimental results are presented in section 5.

2. Modified HT and a Systolic Implementation

Shu et al [8] proposed to use a bit array associated with each accumulator cell to determine the pixel connectivity and line segment length. The bit array can be regarded as another partition in the bar-shaped window. For each pixel falling into the same bucket, it will mark the corresponding bit in the bit array. A continuous string of marked bits in the bit array corresponds to a group of connected points. To confine the size of the bit array, they processed a window of size 31*31 at a time. The window was shifted by 15 pixels to the right or downward in successive iteration, that is, there will be 50% overlap between each pair of adjacent windows. Chuang et al [1] designed a systolic array processor based on this approach. There are several shortcomings in Shu's approach and Chaung's systolic architecture.

1. Most of the pixels except those at the edge of the image will be processed four times because of the 50% overlap between processing windows.
2. The angular resolution is low because of the small window size used.
3. In Chaung's implementation, the edge pixels (represented by the x- and y-coordinates) are sent in one by one. Concurrency is achieved only through extensive pipelining of the processing cells. Moreover, a lot of post-processing, such as sorting and merging, is required.

In our approach, we check the pixel connectivity incrementally. Suppose the image is being scanned from left to right and bottom to top. The edge pixels belonging to the same line segment will be processed and mapped to the same bucket in sorted order. Each bucket can reject unrelated pixels by simply checking the connectivity of the incoming pixel with the last pixel sitting in the bucket. This approach only requires constant amount of memory space to store the start and end points of the line segment in each accumulator cell and a simple comparator for checking connectivity.

When mapping this approach into VLSI architecture, we want to achieve the following three objectives:

1. Minimize external and intercell I/O requirements:
   To reduce the external I/O requirement, we process the bit map of the binary image directly. Each pixel will be represented by a single bit instead of its coordinates. By using a carefully designed local timing scheme, we can compute the location of the pixel based on its arrival time at the accumulator cell. This mean that we don't have to pass around the coordinates of the edge pixels. The number of intercell interconnection wires can be further reduced by using bit-serial architecture.
2. Simple processing cell:
   Hardware multipliers are complicated. In our design, all multiplications are replaced by simple additions. Hence the cell design is extremely simple.
3. Maximize concurrency:
   The algorithm exploits concurrency by both multiprocessing and pipelining of the processing cells. One column or row of pixels are processed per time unit.

Details of the implementation now follow. Let the dimension of the input image be N*N and the r-θ space be represented by a $m_r * m_\theta$ accumulator array. The basic systolic architecture is shown in fig. 3. Suppose we process one column of pixels per time unit. The destination of an edge pixel p_{ij} for a given value of θ, say θ_k is

$$r_{ij} = i \cos\theta_k + j \sin\theta_k.$$

The corresponding destination of the next pixel $p_{i+1,j}$ that enter the j-th compute cell is

$$r_{i+1,j} = (i+1) \cos\theta_k + j \sin\theta_k$$

The different in destination between adjacent pixels that enter the same compute cell is a constant equal to

$$r_{i+1,j} - r_{ij} = \cos\theta_k$$

By preloading the values of $\cos\theta_k$ and $j*\sin\theta_k$ into the j-th compute cell, the computation of the destination of each edge pixel can be replaced by simple addition. The pseudo code of the compute cell funtion is shown in fig. 4. The output of the compute cell can be made always positive by adding an offset of $m_r/2$ to it, i.e. the "destination" register is initialized to $j*\sin\theta_k + m_r/2$.

The next step is to map the N values

of r output from the compute cells to the m_r accumulator cells. In this mapping process, we have to preserve the adjacency relationship between the edge pixels. The mapping function is performed by the routing network (fig. 3). The routing network is composed of a m_r*N array of routing cells. Let the quantization in r be Δr and the number of pixels that can fall into the same bucket be k (fig. 5). If we send in the column of pixels to the array of compute cells with a 45° skewing (fig. 3), the k pixels that belong to the same bucket can be collected successively by a "collector" that traverses the routing network from left to right. The collector consists of a bit-vector of k bits and a "count" of modulo k. The bit-vector and the count are initialized to zero's. The function of the routing cell is shown in fig. 6. The collector starts collecting k consecutive pixels (0 or 1) when it first meets an edge pixel that belongs to the target bucket. The value of count will then be frozen. The accumulator cell will use the value of count to calculate the offset of the first edge pixel in the bit vector with respect to the reference location.

If we scan the image vertically (process one column of pixels per time unit) for $45^\circ \leq \theta \leq 135^\circ$ and horizontally (process one row of pixels per time unit) for $0^\circ \leq \theta < 45^\circ$ and $135^\circ < \theta < 180^\circ$, then the value of k will be bounded by $\sqrt{2} * \Delta r$. If $\Delta r < 3/\sqrt{2}$, then $k \leq 3$. When scanning the image horizontally, the i-th compute cell will be initialized with $i*\cos\theta_k + m_r/2$ and $\sin\theta_k$.

The function of the accumulator cell is shown in fig. 7. The base register stores the location of the reference point (fig. 5). By comparing the values of ref_count and the count of the incoming collector, the offset of the bit-vector can be computed. The bit vector can then be compared with the last bit vector in the bucket to check for the pixel connectivity.

3. Area-Time Complexity

This section analyzes the area-time (AT) complexity of the architecture presented in the previous section. The area costs of the three type of processing cells are first considered.

1. Compute cell
   If the image size is N, then $R = \sqrt{2} * N$. The destination register should have O(logN) bits. It has been shown in [4] that the allowable truncation error in $\cos\theta$ and $\sin\theta$ is $\Delta r/N$. If $\Delta r \geq 1$, then logN bits are required to store the value of $\cos\theta$ or $\sin\theta$. Hence the area cost of the compute cell is $A_c = O(\log N)$.
2. Routing cell
   For values of Δr less than or equal to some constant ρ, the length of the bit vector is bounded. So the area of the routing cell is dominated by the register storing the destination r, which is $A_r = O(\log N)$.
3. Accumulator cell
   In the accumulator cell, there are registers storing the start and end points of a line segment. The bit vector and comparator have constant area. The area cost of the accumulator cell is $A_a = O(\log N)$.

Thus, the total area of the systolic array is:

$$A_s = N*A_c + m_r*N*A_r + m_r*A_a$$
$$= O((N + Nm_r + m_r)\log N)$$

The time required to scan the image once is N. The AT complexity is:

$$AT = O(m\theta * N * (N + Nm_r + m_r)\log N)$$
$$\simeq O(N(N+2)m_\theta m_r \log N)$$

The AT complexity of the conventional sequential algorithm is:

A = storage space of the accumulator array
$= O(m_\theta m_r \log N)$

T = computation time $= O(Pm_\theta)$

where P is the number of edge pixels in the image.

$$AT = O(Pm_\theta{}^2 m_r \log N)$$

If $Pm_\theta > N(N+2)$, then the proposed architecture has better AT performance than the sequential algorithm. Usually, P ranges from 5 to 10 percent of the image size and m_θ has the same order of N.

The area cost of the processing cells in Chuang's deisgn [1] are:

$A_a = O(N\log N)$, and
$A_c = O(\log N + \log N \log\log N)$.

The time required, excluding the post-processing time, is P. The total AT for his design, neglecting the area of compute cells, is $AT = O(PNm_\theta m_r \log N)$. For large N, $P >> N$. Hence the AT complexity of our design is better.

4. Line Selection

Each line segment detected by a bucket is an approximate fitting of a straight line through a group of connected edge pixels within the tolerance specified by Δr. Each line segment is defined by the four-tuple (r,θ,S,E), where S and E are the start and end points of the line, respectively. Obviously, the length of the line segment as compared to the lengths of other line segments detected by adjacent buckets

that correspond to the same group of points is a good measure of the fitting. To determine the "best" fit is equivalent to finding the longest line segment that passes through the same group of points. Assume that the sampling rate for θ is high enough, such that, $\Delta\theta << \phi$, where ϕ is the minimum angle between any two intersecting lines in the image. With this assumption, any two intersecting lines L1 and L2, detected by buckets (θ_k, r) and (θ_{k+1}, r'), respectively, should correspond to different (or same) portions of the same "line" in the image. If there is significant overlap between the two lines, say 75 percent, then the longer line will be the better fitting. Details of the selection algorithm is given in fig. 8.

5. Experimental Results

The architecture presented in section 2 was simulated to evaluate the effect of considering pixel contiguity. The test image (fig. 9a) was extracted from a road map and digitized with a resolution of 200 points per inch. The width of the road is about 8 pixels. Fig. 9b shows the edges extracted from the image after preprocessing. The values of Δr and $\Delta\theta$ are 3 and 2°, respectively. Line segments with length less than 10 units are discarded. The results of the modified transform is shown in fig. 9c and 9d. Only two short edges marked in fig. 9c are missing. If a higher resuolution image is used, they would have been detected. The curved contours in the image are approximated by a number of straight line segments. Out of the 112 selected line segments, about 24 of them are redundant. These redundant lines are mostly short lines with length less than 20 units. They are usually 4 or 5 columns apart from the "best" selected lines and are not removed by the line selection algorithm.

6. Concluding Remarks

A modified Hough Transformation technique that can take into account pixel contiguity has been presented. The systolic architecture described in section 2 is an efficient design judging from the following facts: (1) simple processing cells; (2) minimal I/O requirements; (3) highly concurrent; and (4) better AT complexity than that of the conventional sequential algorithm.

The line selection algorithm described in section 4 involves only local operations. Parallelization of this function should not be difficult.

The modified Hough transform was simulated and the observed performance is satisfactory. The accuracy of the modified transform in describing the original image is remarkable. If the line selection algorithm is modified slightly such that only the overlapping part of the two lines is removed, a linearization of the image can be obtained.

References

1. Henry Y.H. Chuang, C.C. Li, "A Systolic Array Processor for Straight Line Detection by Modified Hough Transform", Proc. Comp. Arch. for Pattern Anal. & Database Management, 1985, pp 300-303.

2. R.O. Duda, P.E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", Comm. of ACM, Vol 15, no 1, Jan 1975, pp 11-15.

3. C.R. Dyer, "Gauge Inspection Using Hough Transform", IEEE Trans. PAMI, Vol 5, no 6, 1983, pp 621-623.

4. K. Hanahara T. Maruyama, T. Uchiyama, "A Real-Time Processor for the Hough Transform", IEEE Trans. PAMI, Vol 10, no 1, Jan 1987, pp 121-125.

5. K.Y. Huang, K.S. Fu, T.H. Sheen, S.W. Cheng, "Image Processing of Seismograms: (A) Hough Transformation for the Detection of Seismic Patterns; (B) Thinning Process in the Seismogram", Pattern Recognition, Vol 18, no 6, 1985, pp 429-440.

6. R.M. Inigo, E.S. McVey, B.J. Berger, M.J. Wirtz, "Machine Vision Applied to Vehicle Guidance", IEEE Trans. PAMI, vol 6, no 6, 1984, pp 820-826.

7. M. Kushnir, K.Abe, K. Matsumoto, "Recognition of Hand-Printed Hebrew Characters Using Features Selected in the Hough Transform Space", Pattern Recognition, Vol 18, no 2, 1985, pp 103-113.

8. D.B. Shu, C.C. Li, F. Mancuso, Y.N. Sun, "A Line Extraction Method for Automated SEM Inspection of VLSI Resist", IEEE Trans. PAMI, Vol 10, no 1, Jan 1987, pp 117-120.

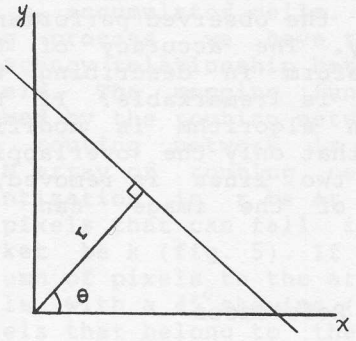


fig. 1 Normal parameters for a line.

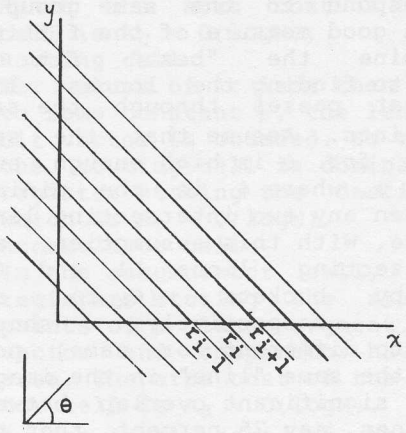


fig. 2 Quantization of the r-θ space divides the the image plane into bar-shaped windows.

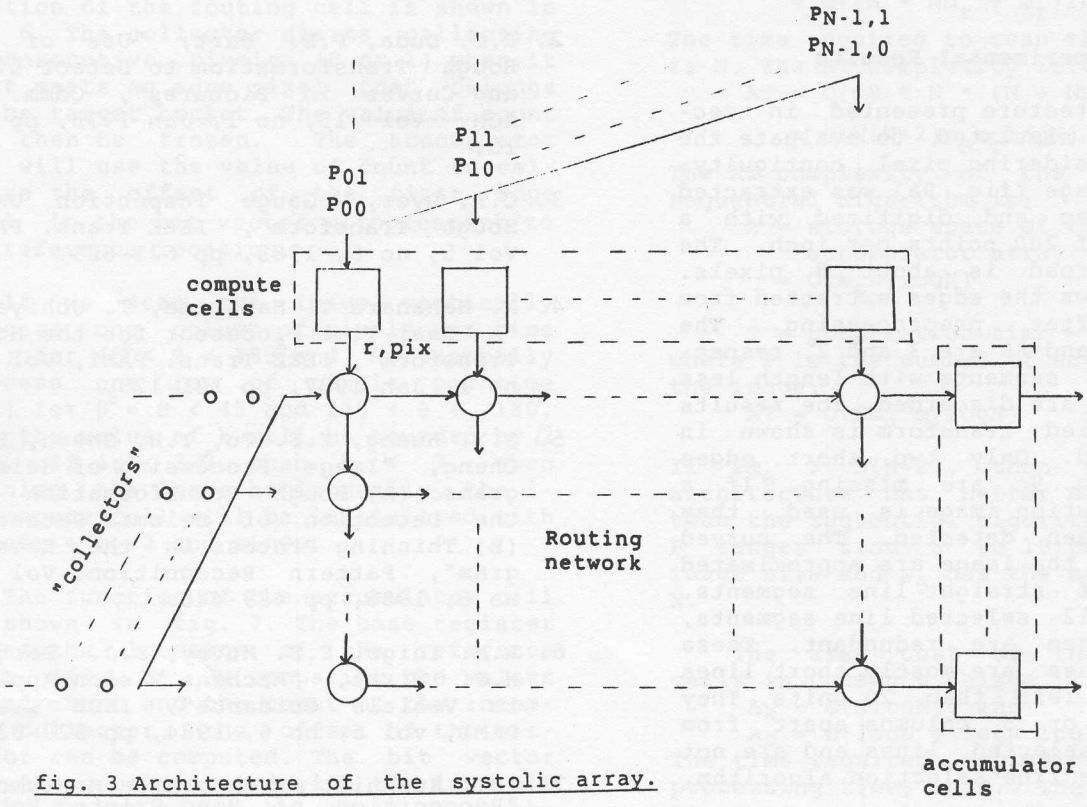


fig. 3 Architecture of the systolic array.

```
preload cosθ
preload destination

pix_out = pix_in;
r_out = destination;
destination = destination + cosθ;
```

fig. 4 Function of the compute cell.

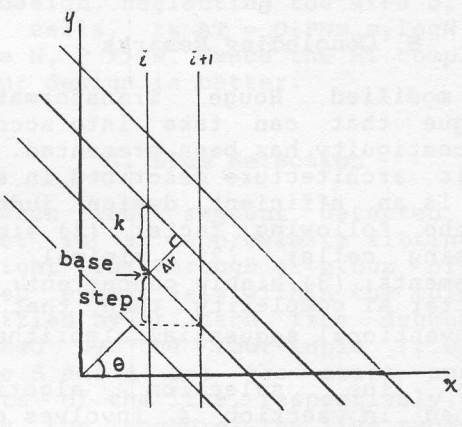


fig. 5 Parameters used in the accumulator cell with vertical scanning: k = Δr/sinθ, step = 1/tanθ.

```
pix = pix_in;
bit_vector = bit_vector_in;
freeze_count = freeze_count_in;
pick_up = pick_up_in;

IF (freeze_count = False)
    count = (count_in + 1) mod k
ELSE
    count = count_in;

r = r_in - 1;
IF (r = 0)
    IF (pix = 1)
        shift pix into bit_vector;
        freeze_count = True;
        pick_up = True;
    ELSE
       IF (pick_up = True)
           shift '0' into bit_vector
    pix_out = 0;
ELSE
    IF (pick_up = True)
        shift '0' into bit_vector;
    pix_out = pix;

IF bit_vector full
    pick_up = False;
r_out = r;
count_out = count;
bit_vector_out = bit_vector;
freeze_count_out = freeze_count;
pick_up_out = pick_up;
```

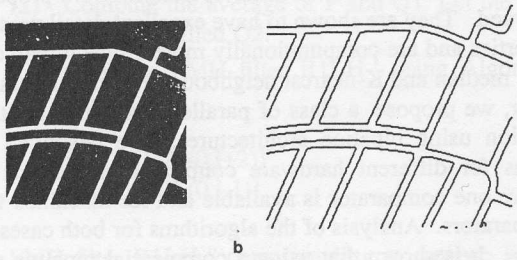
fig. 6 Function of routing cell.

```
/*
    the start and end points of the line are:
    (start, start_cycle) and
    (end, current_cycle-1).
*/

 preload base, ref_count, step;
 initialize current_cycle = 0;
 initialize bucket_empty = True;

offset = compare(ref_count, count_in);
current_bit_vector = align(bit_vector_in, offset);
IF (current_bit_vector is empty)
    IF (bucket_empty = False)
        output line segment;
        bucket_empty = True;
ELSE
    IF (bucket_empty = True)
        start = end = base + offset;
        start_cycle = current_cycle;
        last_bit_vector = current_bit_vector;
        bucket_empty = False;
    ELSE
        check_connect last_bit_vector and
        current_bit_vector;
        IF no pixel connected
            output line segment;
            last_bit_vector = current_bit_vector;
            cycle_start = current_cycle;
            start = end = base + offset;
        ELSE
            last_bit_vector = connected pixels in
                              current_bit_vector;
            end = base + offset of the last
                  connected pixels;

 current_cycle = current_cycle + 1;
 base = base + step;
 ref_count = (ref_count + step) mod k;
```

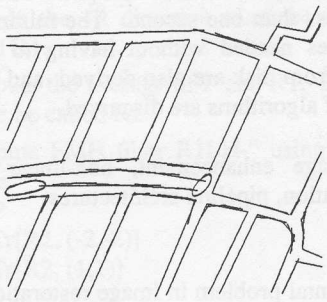
fig. 7 Fuction of the accumulator cell.

```
FOR each line segment L1 = (r,θm,S,E)
    FOR m2 = m-3 to m+3
        rs = S(x) cosθm2 + S(y) sinθm2;
        ry = E(x) cosθm2 + E(y) sinθm2;
        FOR r2 = rs to ry
            FOR all L2 in bucket (r2,θm2)
                check_overlap(L1, L2)
                IF 75% of L2 overlap with L1 and
                length of L2 < length of L1
                    mark L2;

select all line segments not marked
```

fig. 8 Line selection algorithm.



a b



c

total # of line segments detected = 1831
# of line segments selected = 112
longest line segment selected = 145 units
shortest line segment selected = 14 units
# of redundant lines selected = 24

d

fig. 9 (a) Test image extracted from a road map; (b) edge pixels extracted (c) line segments detected; and (d) summary of the results of the HT.