F.Dehne, A.Hasenclover, J.-R. Sack, N.Santoro, "Parallel Visibility on a Mesh-Connected Parallel Computer", Int. Conference on Parallel Processing and Applications, L'Aquila, Italy, September 23-25, 1987, pp. 173-180

PARALLEL VISIBILITY ON A MESH-CONNECTED PARALLEL COMPUTER

F. Dehne, A. Hassenklover, J.-R. Sack, N. Santoro

School of Computer Science, Carleton University, Ottawa, K1S 5B6, Canada

ABSTRACT

Consider a set of images represented on an nxn mesh-connected computer, where each mesh processor represents the center of a unit-area pixel in a nxn grid of pixels. Such a configuration is called a systolic screen. The parallel visibility problem is the problem of determining the visibility from a light source, located at infinity, of a set of images represented on a systolic screen. In this paper, a systolic algorithm is presented which computes the portion of each pixel illuminated by rays of light parallel to a given direction **d**; it is shown that the algorithm requires O(n) time for the entire computation.

1. INTRODUCTION.

Two-dimensional arrays of processors [8] have long been proposed for image processing [6] since images can be naturally mapped on a mesh of processors (MCC). An *image* is commonly defined as a subset of a digitized plane Π of size n^2, where Π is a rectangular array of lattice points (or pixels) with integer coordinates (i,j), where i,j $\in\{1,..,n\}$. The pixels are of unit width and pixel(i,j) covers the square [i-0.5,i+0.5] by [j-0.5,j+0.5]. A *systolic screen* of size n^2 is a nxn MCC where each processing element (PE) P_{ij} represents a pixel centered about the lattice point (i,j), with pixel $p_{1,1}$ occurring at the top left-hand corner of the MCC [2]. Figure 1.1 shows an image in a MCC, where the dark pixels are occupied by a digitized object. For a nxn systolic screen, an equivalent diagram will be employed; the diagram for Figure 1.1 is shown in Figure 1.2.

In image processing, arrays of processors are mostly used for low level local operations such as image restoration, noise removal, computation of connected components and edge detection (cf.[1,3,5,6,9]). Recently, systolic screens have been proposed as a machine model also for Computational Geometry [4,7]. In this context, Miller and Stout [4,7] have presented O(n) time algorithms for computing the distance between two images, determining extremal points, diameter, and smallest enclosing circle of an image, as well for testing convexity and separability of digitized images. Recently, Dehne, Sack and Santoro

[2] have proposed O(n) algorithms for computing the contours of an image, and all k^{th} rectilinear convex hulls of an image stored on a systolic screen.

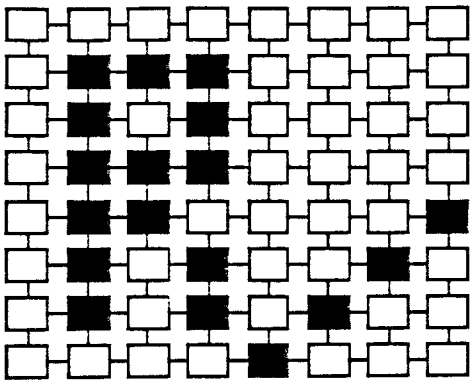


Figure 1.1

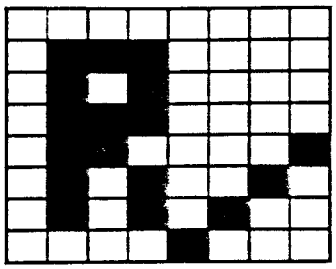


Figure 1.2

In this paper we continue this line of research by solving the following problem (*Parallel Visibility Problem*): given a light source located at infinity that emits rays of light parallel to a specified direction **d**, compute the portions of each object that are illluminated by the light source.

In the following section we discuss a sweep method which will be used to solve the parallel visibility problem. In Section 3, the implementation of the algorithm on a systolic screen is described in some detail, and its time complexity is analyzed.

2. Systolic Parallel Visibility.

Given a direction **d**, the *parallel visibility problem* on the systolic screen is the problem of determining, for each object in the screen, which portion is *visible* in that direction; that is, to determine the region of each pixel illuminated by a light source, located at infinity, emanating rays parallel to the given direction **d**. Pixels that are part of a digitized image are called *occupied* pixels and they may block the light rays.

A *strip* **S** is a region of the systolic screen bounded by two lines parallel to direction **d**. In computing the visible portions of a pixel, the systolic screen will be divided into fixed-size consecutive strips $\mathbf{S_1},\mathbf{S_2},...,\mathbf{S_m}$. For a strip $\mathbf{S_k}$, denote by $\mathbf{s_k}$ and $\mathbf{s_{k+1}}$ the directed lines parallel to **d** forming the *boundaries* of $\mathbf{S_k}$. Since the visibility in a strip is affected only by the pixels intersecting that strip, we will consider only the problem of computing visibility in a strip. The *visibility information* for a strip $\mathbf{S_k}$ is an interval $\mathbf{I_k}$ perpendicular to the direction **d**, and whose length is initially set to the width **w** of the entire strip (see Figure 2.1). The strip will be scanned; during this process, the visibility information will

be updated so that, at any point in time, the interval indicates which parts of the strip are still visible from the light source. The actual value of the width **w** depends on the specific direction **d**; the method for choosing **w** will be discussed later.

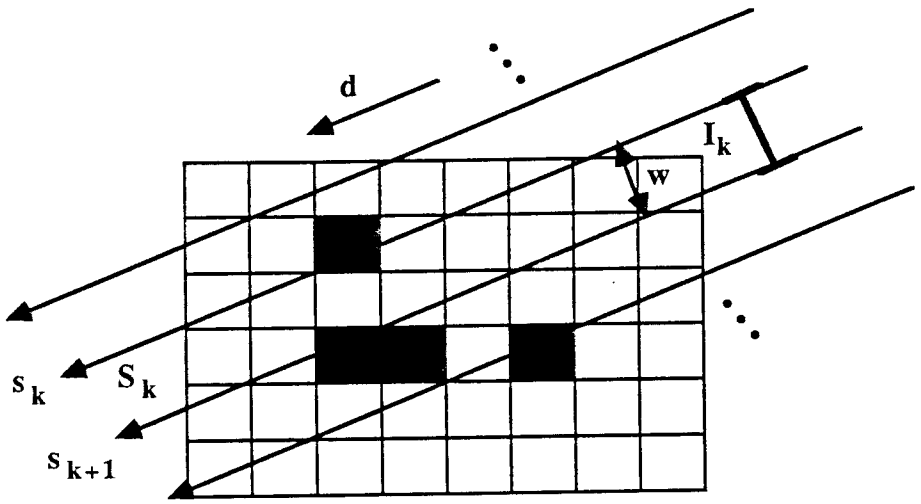


Figure 2.1. A strip S_k, and its associated interval I_k of width w.

Initially, the entire strip is considered visible, and the length of the interval is **w**. The interval is passed along the strip in direction **d**, and is updated whenever an occupied pixel is encountered. For example, in Figure 2.1, the interval I_k initially covers the entire strip S_k. In Figure 2.2, the same strip S_k is shown with only the occupied pixels being indicated. After the first occupied pixel is encountered, I_k is reduced to I'_k; it is further reduced when the next occupied pixel is encountered, resulting in the final interval I''_k.

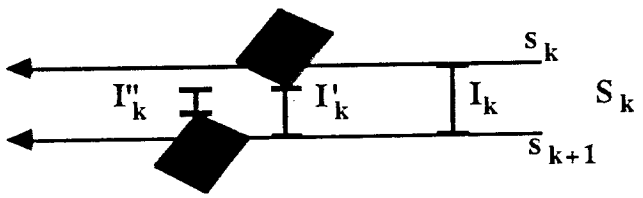


Figure 2.2. Update of interval I_k when scanning strip S_k.

The visibility information I_k must be sent to all pixels intersecting strip S_k. Let $P_k=\{p_1,p_2,\ldots,p_m\}$ be the set of all pixels intersecting S_k. If pixel p_i blocks pixel p_j, to correctly compute the visibility in S_k, it must be ensured that interval I_k reaches p_i (and is updated there) *before* it reaches p_j. A linear ordering $\mu(P_k)$ of the elements of P_k is said to

be *correct* if $p_i <_\mu p_j$ whenever p_i blocks p_j. If interval $\mathbf{I_k}$ travels in strip $\mathbf{S_k}$ according to a correct linear ordering, a correct computation of the visibility is ensured. In particular, a correct ordering of the pixels in $\mathbf{P_k}$ is the ordering $\pi(\mathbf{P_k})$ induced by the projections of the centers of the pixels onto s_k. With respect to $\pi(\mathbf{P_k})$, we may now define the *successor, succ* (PE) [the*predecessor, pred* (PE)], of a PE in a strip $\mathbf{S_k}$ as the pixel that follows [precedes] PE in $\pi(\mathbf{P_k})$.

We shall now consider the choice of the width of a strip. The width **w** must be selected so as to satisfy requirements 1 and 2 below:

Requirement 1: The *size* of the messages sent by each PE is bounded by a small constant.

Note that this requirement is easily met by avoiding fragmentation of $\mathbf{I_k}$; that is, by ensuring that interval $\mathbf{I_k}$ remains continuous.

Requirement 2: The *number* of messages a PE sends or receives at each time step is bounded by a constant.

Proposition 1. *Let the width w equal $\sqrt{2}\cos(45° - \beta)$ for angles $\beta \in [0,90]$ from the positive x-axis to the direction* **d**. *Then requirements 1 and 2 are met.*

Proof: Choosing $\mathbf{w} = \sqrt{2}\cos(45° - \beta)$ ensures that any pixel of the mesh intersects at most two strips (see Figure 2.3). This implies that a PE sends and receives at most two messages; thus, Requirement 2 is met. Fragmentation of interval $\mathbf{I_k}$ can only occur if a pixel is totally contained inside a strip, possibly leaving gaps at either end as shown in Figure 2.4. However, this situation can never occur, for any direction **d**; in fact, the boundaries of a strip form lines of support to a pixel region, which ensures that any pixel of the MCC either blocks an entire strip or one side of a strip; thus, Requirement 1 is met. □

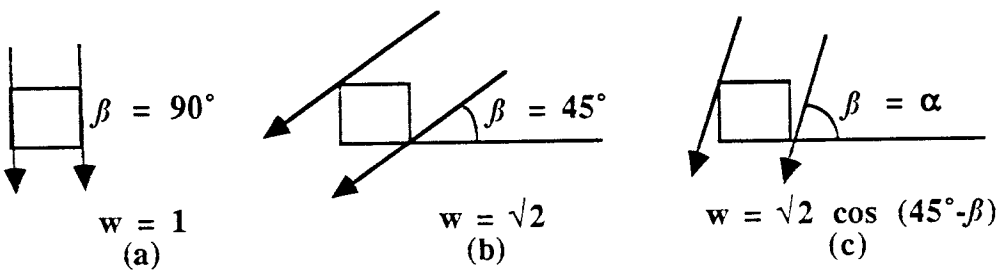


Figure 2.3. Definition of width w of a strip.

Directions of visibility other than those described by $\beta \in [0, 90]$ are analogous and thus omitted here.

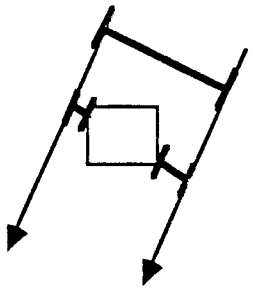


Figure 2.4. Subdivision of an interval.

We shall now describe the general strategy for solving the parallel visibility problem on a systolic screen. In the presentation, the general case of a PE intersecting two strips is considered; the case of a PE intersecting only one strip is implied in the discussion.

General Strategy

1. Given the direction **d** of the light rays, each PE computes the strips in which it lies, and determines for each strip its successor and predecessor pixels in the linear ordering $\pi(\mathbf{P_k})$. (This computation involves the projection of a constant number of pixels onto $\mathbf{s_k}$, as discussed later.) Each PE then computes interval $\mathbf{I_k}$ (set initially to be the entire strip width) for each of the strips it intersects; if the PE is occupied, it transmits to its successor in $\pi(\mathbf{P_k})$ the portion of $\mathbf{I_k}$ which it does not block.

2. Upon receiving an interval from its predecessor in $\pi(\mathbf{P_k})$, a PE uses it to update its own interval, and sends the resulting interval $\mathbf{I'_k}$ to its successor in $\pi(\mathbf{P_k})$.

3. Processing terminates when no more messages are sent in the mesh; as shown later, this occurs after at most O(n) time steps. Upon termination, the final interval (one for each strip) available at a PE represents the portion of the pixel which is illuminated within that strip.

3. Mesh Implementation of Parallel Visibility.

The algorithm assumes that the image has been digitized on the systolic array. The direction **d** of the incoming light is sent to one PE, and then broadcast to the other PE's in the MCC.

When computing its successor in a strip, if PE **x** finds that there are two or more PE's whose projected centers coincide, then the PE with the minimum j-coordinated is selected as **x**'s successor. Should the projected center of a PE **p** coincide with the projected center of **x** *and* its j-coordinate is less than that of **x**, then **p** is not the successor of **x**. A similar approach is used to select the predecessor PE. This ensures that infinite loops do not result in the selection of a successor.

The PEs at the edges of the screen may not have successors or predecessors. To ensure simple and symmetric computation, the border layers of the systolic screen will contain only unoccupied pixels; these PE's perform the algorithm, but are not part of the final result.

The pseudo-code below describes the operation of a PE **x** of the systolic screen. (Note that any message contains identification of both source and destination PE). Subscript i in $succ_i$ (**x**) [$pred_i$ (**x**)] will indicate the successor [predecessor] of **x** in strip S_i, provided that **x** intersects strip S_i.

PARALLEL VISIBILITY ALGORITHM
(as executed by PE **x**)

1. **if** direction **d** is received **then**
   * calculate the width **w**.
   * compute the strips S_k and S_{k+1} (if it exists) intersected.
   * compute $succ_k(x)$, $pred_k(x)$, $succ_{k+1}(x)$, and $pred_{k+1}(x)$ for strips S_k and S_{k+1}.
   * set I_k and I_{k+1} to be of width **w**.
   * **if** **x** is occupied **then**
     * remove from I_k and I_{k+1} the portion blocked by **x**.
     * send the updated intervals I_k and I_{k+1} to $succ_k(x)$ and $succ_{k+1}(x)$, respectively.
     endif
   endif

2. **if** interval **I** is received from $pred_h(x)$ (where h=k or h=k+1) **then**
   * update I_h by intersecting with **I**.
   * send the updated I_h to $succ_h(x)$.
   endif

3. **if** received interval **I** is not for **x** **then** route **I** to its destination **endif**

4. **if** the number of elapsed time steps is equal to the number of steps at completion **then**
   * I_k and I_{k+1} are the regions of **x** visible from the light source.
   endif

END PARALLEL VISIBILITY.

We shall first prove that *succ* (PE) may be computed in constant time. This will also ensure that the number of routing steps required to route an interval from a PE to its successor in a strip is constant.

Proposition 2. *The distance (in the L_1 or Manhattan metric) between a PE* x*, and its successor in* $\pi(P_k)$ *of strip* S_k *is at most 3.*

Proof: Three distinct cases may occur.

Case 1: the boundary line s_k touches PE **x** at the top left-hand corner. This implies, by definition of **w**, that s_{k+1} touches **x** at the bottom right-hand corner. If the direction is

changed continuously from 0° to 90°, the successor of **x** is either the left or bottom neighboring PE (see Figure 3.1).

Case 2: the boundary line s_k touches **x** almost at the bottom right-hand corner. This implies, by definition of **w**, that, if the direction is changed continuously from 0° to 90°, the possible successor of **x** is at distance $d \leq 3$ from **x** (see Figure 3.2).

Case 3: the boundary line s_k lies between the top left-hand corner and the bottom right-hand of **x** corner. This case is easily reduced to the above two cases. []

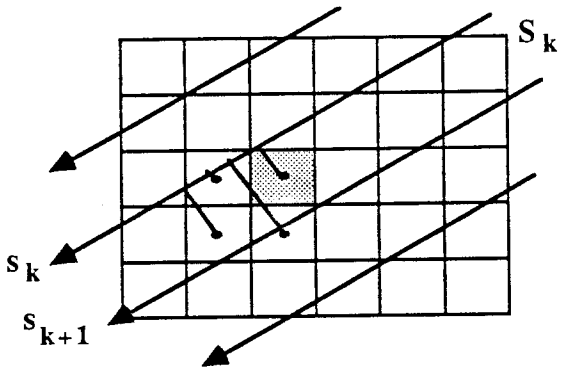


Figure 3.1.

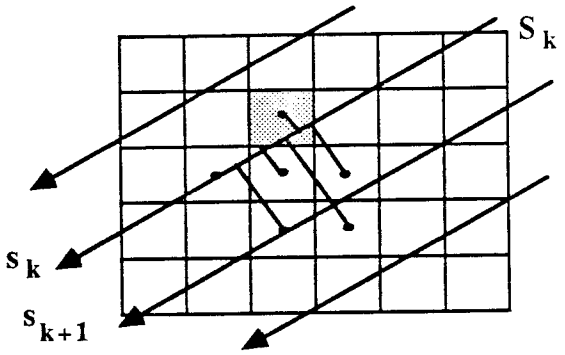


Figure 3.2

We can now prove the overall time complexity of the proposed algorithm.

Proposition 3. *The Parallel Visibility Algorithm requires O(n) time steps on an nxn MCC.*

Proof: Each step (1-3) of the algorithm requires a constant number of time steps. In fact, the successors and predecessors of a PE can be computed in constant time, and the time to route an interval to a successor requires no more than 3 routing steps (by Lemma 3.1). Furthermore, since a PE intersects at most two strips, during each step of the algorithm a constant number of fixed size messages will be sent (by Requirements 1 and 2). Finally, there are at most n PE's intersecting each strip and, in the worst case, an interval travels down the entire strip. []

AKNOWLEDGMENT

This work was supported in part by the Natural Sciences and Engineering Research Council under Grants A0392, A2415, and A9173.

REFERENCES.

[1] Danielson P.E., Levialdi S. *Computer architecture for pictorial information systems.* IEEE Computer, Nov. 1981.

[2] Dehne F., J-R. Sack, N. Santoro. *Computing on a Systolic Screen: Hulls, Contours and Applications.* Conf. on Parallel Architectures and Languages. Eindhoven, The Netherlands, June 15-19,1987.

[3] Klette, R.A. *parallel computer for image processing.* Elektronische Informationsverarbeitung und Kybernetik. EIK 15 (1979) 5/6 pp. 237-263.

[4] Miller R., Stout Q.F. *Computational geometry on a mesh-connected computer.* Proc. Int. Conf. on Parallel Processing, 1984.

[5] Nassimi D., Sahni S. *Finding connected components and connected ones on a mesh-onnected parallel computer.* SIAM J. Comput., Vol. 9, No. 4. Nov. 1980.

[6] Reeves A.P. *Survey, parallel computer architectures for image processing.* Computer Vision, Graphics and Image Processing 25, 1984.

[7] Stout Q.F., Miller R. *Mesh-connected computer algorithms for determining geometric properties of figures.* 7th Int. Conf. on Pattern Recognition, Montreal, Canada, July 30 to August 2, 1984.

[8] Thompson C.D., Kung H.T.*Sorting on a mesh-connected parallel computer.* CACM, Vol. 20, No. 4. April 1977.

[9] Unger S.H. *A computer oriented towards spatial interaction.* Proc. IRE, Vol. 46, pp. 1744-1750, 1958.

INTERNATIONAL
CONFERENCE
ON

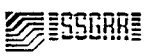
PARALLEL PROCESSING AND APPLICATIONS

L'Aquila, Italy, September 23 - 25, 1987

Sponsored by:

IEEE
North Italy Section
Middle - South Italy Section

AEI



Scuola Superiore
G. Reiss Romoli

UNIVERSITY OF L'AQUILA

CNR
National Research Council