CONCURRENCY: PRACTICE AND EXPERIENCE, VOL. 5(5), 379–406 (AUGUST 1993)

Region-to-region visibility analysis using data parallel machines

Y. ANSEL TENG, DANIEL DeMENTHON AND LARRY S. DAVIS

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3411, USA

SUMMARY

We propose an algorithm for solving region-to-region visibility problems on digital terrain models using data parallel machines. Since global communication is the bottleneck in this kind of algorithm, the algorithm we propose focuses on the reduction of global communication. The algorithm analyses a strip of the source region at a time and sweeps through the source strip by strip. At most four sweeps are needed for the analysis. By exploring the coherence properties in the processor structure, global communication is minimized and complexity is substantially improved. Furthermore, all global write operations are exclusive and concurrency in global read operations is minimized. Since the problem size is usually large, we also designed rules of decomposition to efficiently handle the cases where the required number of processors is greater than available. The algorithm has been implemented on a Connection Machine CM-2, and results of computational experiments are presented.

1. INTRODUCTION

In this paper we study the following problem: given a source region S and a destination region D on terrain, compute a measure of visibility from the region S to the region D. In terms of discretized geometry, this problem can be restated as computing for every point in S the number of visible points in D. This region-to-region visibility analysis (RRVA) problem occurs in choosing sites with high visibility or invisibility and has environmental, communication and military applications. It also arises in our study of visibility-based stealth terrain navigation systems[1,2]. In such systems, an observation point must be chosen from a reachable area so that an agent can carry out a reconnaissance mission to track the movement of adversaries over time. A measure of visibility is needed for optimizing both the observability of the point and the safety of the path to that point. The analysis has to be fast enough for the whole system to perform real-time path planning. In light of the computation-intensive nature of visibility analysis, an algorithm that exploits the power of parallelism is in order.

Most previous work on visibility analysis is concerned with finding the visible area from a given viewing *point*, and can be divided into two categories based on the underlying terrain models. On polyhedral terrain models, Cole and Sharir investigated several aspects of this problem[3]. Jung[4] made a comparison between three different algorithms. A parallel hidden surface removal algorithm was proposed by Reif and Sen[5] which is distinguished by its output-sensitive complexity. On digital terrain models (DTM), terrain is represented by associating elevation data with regular grids. The regularity facilitates fast solutions to many problems by mapping the terrain to a data parallel machine with regular

1040–3108/93/050379–28$19.00

Received 10 December 1991
Accepted 18 September 1992

connections like the Connection Machine[6]. For example, Blelloch[7] implemented a visibility algorithm with his scan operation model on the Connection Machine CM-2. This approach is adapted in Reference 1 to develop a point-to-region visibility analysis (PRVA) algorithm for the purpose of visibility-based navigation using data parallel machines. An open question in visibility analysis is whether the complexity of a visibility analysis for N viewing points can be less than N times that for a single point. We do not know the answer in general; however, in this paper, we present a method that can achieve this inequality for the discretized RRVA problem.

In the rest of this paper, we assume a data parallel machine model that provides the flexibility to embed a mesh-connected array of any dimension and the efficiency to perform parallel prefix operations along any axis of the array in logarithmic time. A hypercube network is an example of such a model. These features will be used extensively in our algorithms; this kind of parallel prefix operation will be referred to as a *scan* operation. Communication between processors is always a dominating factor in designing practical parallel algorithms. Two types of communication are involved in our algorithm: a processor can obtain data from its neighboring processors by *local* communication, where neighborhood is defined according to the underlying machine model, while by *global* communication a processor can send or receive data to or from any other processor. Local communication is usually supported by direct links between neighboring processors while global communication relies on the routing mechanism of the machine and is usually orders of magnitude slower than local communication on existing machines. Therefore, it is not surprising that global communication is frequently the bottleneck of practical parallel algorithms. Since this is also the case for visibility analysis algorithms, our research focuses on the reduction of global communication by using coherence in the processor structure. We review the basics of visibility analysis and the PRVA algorithm in the following Section and present the RRVA algorithm in Section 3. Some further observations and improvements are made in Section 4. Since the problem size is usually large, we also discuss how to handle efficiently the situation where the required number of processors is greater than available. The algorithm is implemented on a Connection Machine CM-2 and experimental results are shown in Section 5.

2. PRELIMINARIES

2.1. Visibility along a line

Two points on terrain are intervisible if the line joining them lies completely above the terrain; otherwise they are invisible to each other. The basic parallel algorithm for visibility analysis is to compute the visibility with respect to a given viewing point for every grid cell along a line on the plane projection of the terrain. Suppose we have a list of processing elements (PEs) representing this line. Each of them contains the elevation of the corresponding grid cell on the line with the first cell as the viewing point. The visibility of each grid cell from the viewing point can be determined by first computing the elevation angle from the viewing point to each cell and then comparing the angle with the maximal angle among all cells closer to the viewing point. If its angle is greater than this maximal angle, it is visible. These steps are illustrated in Figure 1. The total complexity is dominated by the computation of the maximal angle of the cells closer

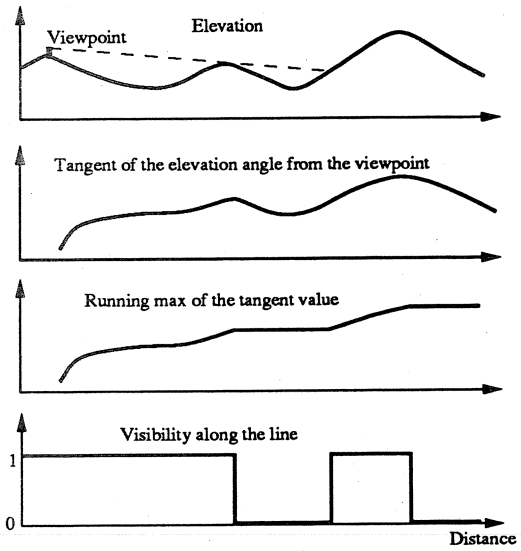


Figure 1. Computing visibility along a line. The X-axis represents the horizontal distance from the viewpoint. The effect of the middle hill is shown by the broken line in the elevation plot

than each cell, which can be done by a scan operation in O(log W) time on the assumed machine models using W processors, where W is the number of grid cells along the line.

2.2. Point-to-region visibility analysis

PRVA algorithm computes the visibility from a viewing point to a region. Since the underlying terrain is represented by an array of processors, each corresponding to a grid cell of the terrain, the result is returned as a *visibility map* in which the visibility of each grid cell is indicated by a flag in the associated PE. For computational efficiency, the region is assumed to be an upright rectangle. For a region of arbitrary shape, we may obtain the maximal and minimal X and Y co-ordinates of the region and use the rectangular box determined by these co-ordinates instead. Our algorithm is a generalization of the one by Blelloch[7]. The idea is to find a minimal set of lines that covers all the grid cells in the region, allocate a list of processing elements to each line, let each processor get the elevation of its corresponding cell, then use the line-visibility algorithm to determine the visibility of each corresponding cell, and send back the result to that grid cell on the terrain.

Here we define and clarify some terminology in our discussion. Geographically, we use the term *ray* to refer to a directed line segment on the plane projection of the terrain. It also refers to a list of PEs from the computational point of view. The processor structure representing a set of rays is called a *ray structure*. We do not distinguish between the two meanings of rays as long as no ambiguity occurs. The same principle applies to the terrain

cells and points. In a DTM, terrain is actually a tessellation of square cells in a regular grid pattern. Each cell has a representing grid point in the center, indexed by the X and Y co-ordinates. We usually use the term 'point' to emphasize a specific location and the term 'cell' to imply a piece of the tessellation with unit area. Computationally, both terms refer to the PE assigned to that grid cell. Two processor structures are used in the analysis: one for the terrain and the other for the ray structure. We usually refer to the processors of the former as terrain cells, and of the latter as PEs of a ray or in a ray structure.

As mentioned above, we assume the region to be an upright rectangle. We define a *far side* as a side of the rectangle such that when drawing a line from the viewing point to any non-end point on that side, the line will pass through the interior of the rectangle. The minimal set of lines covering all grid cells in a rectangular box will be the lines from the viewing point to all grid points on the far sides. Therefore the PRVA can be accomplished by constructing a ray structure for each far side of the rectangle and running the line-visibility algorithm for each ray in the ray structure.

Let L be the length of a side of the rectangle, and W be the maximal number of grid cells on a single ray to this far side. An $L \times W$ 2-D array of processors is allocated for the ray structure, as shown in Figure 2. Each row in the ray structure corresponds to a ray. The key point in building the ray structure is to find the corresponding grid cell in the terrain map for each processor in the ray structure so that each processor can obtain the corresponding elevation. We first broadcast the co-ordinates of the viewing point and the end points of the far side. Using the ray index, each processor can find the end point of its own ray. Then, using the element index, each processor can find its corresponding grid cell by the digital differential analyser (DDA) technique[8]. The time complexity for building the structure is constant.

After the construction of the ray structure, each processor obtains the elevation information from its corresponding terrain cell and the line-visibility algorithm is conducted along all rays simultaneously. The result will be sent back to its corresponding grid cell. If several results are sent back to the same grid cell, they are combined by an OR operation.

The complexity of this parallel algorithm is

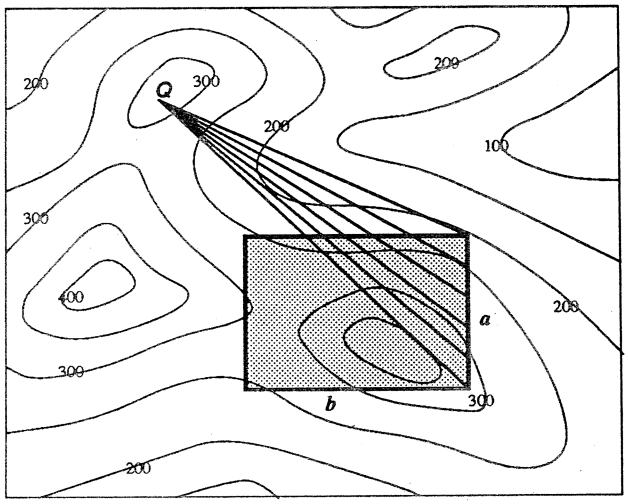
$$O(LW \log W + LW \times Comm) \text{ operations}$$

with minimal time

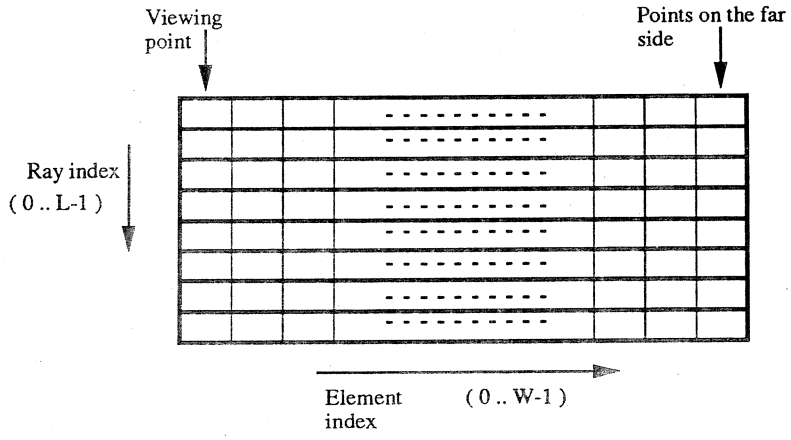
$$O(\log W + Comm)$$

using $L \times W$ processors, where $Comm$ stands for the complexity of a global communication.

As we can see from Figure 2, many PEs in different rays may be mapped to the same terrain cell. Thus concurrent reads/writes occur in the global communication between the two processor structures. It can be shown that all PEs that are mapped to the same terrain cell have the same element index and consecutive ray indices. Using this property, concurrent reads/writes can be eliminated by grouping the PEs according to the terrain cell they are mapped to and allowing only one PE from each group to participate in the global communication. In each group, the data can be distributed to or combined from every member by segmented scan operations. As this operation is conducted along the dimension of size L, and we expect L to be $O(W)$, it will not increase the complexity.



(a)



(b)

Figure 2. Ray and ray structure. (a) the rays are shown by line segments between the viewing point V and side a of the rectangle. The ray structure consists of all rays for this side and is mapped to a triangle on the terrain map. Note that only sides a and b are far sides; (b) a 2D array is allocated as the ray structure

3. REGION-TO-REGION VISIBILITY ANALYSIS

In this Section we present an algorithm for solving the discretized version of the RRVA problem. For simplicity, we assume both the source and the destination are upright rectangles. For the general case, circumscribing rectangles can be used. The source and

the destination regions can be in any relationship, e.g. they can be overlapping or even identical.

From the preceding Section, there apparently exists a brute-force solution by applying the PRVA to each grid cell in the source region. The complexity of this brute-force algorithm is $O(L_S^2 L_D W \log W + L_S^2 L_D W \times Comm)$ operations, using up to $L_S^2 L_D W$ processors, where L_S and L_D are the linear size of the source and the destination, respectively, W is the maximal number of grid cells along a ray, and *Comm* stands for the complexity of a global communication operation.

If we have $L_S^2 L_D W$ processors, this time complexity will become $O(\log W + Comm)$. Judging from the usual problem size, no existing machines can provide this number of processors. Even if such a machine existed, the number of concurrent reads would make this algorithm virtually implausible. Therefore, we have to complete the analysis by several iterations each of which computes a subproblem of the entire analysis. The other extreme is to conduct the analysis point by point, which will be very inefficient in that the global communication will occupy most of the total running time and the acquired data will be duplicated across iterations. Since global communication is the bottleneck in the analysis, the algorithm we propose focuses on the reduction of global communication by using coherence properties.

3.1. Observations and definitions

One important property is observed in Figure 3, which shows that the ray from the source point (x_s,y_s) to the destination point (x_d,y_d) is enclosed by the triangle formed by the rays from $(x_s - 1,y_s)$ and $(x_s - 1,y_s + 1)$[1] to the same destination point. Furthermore,

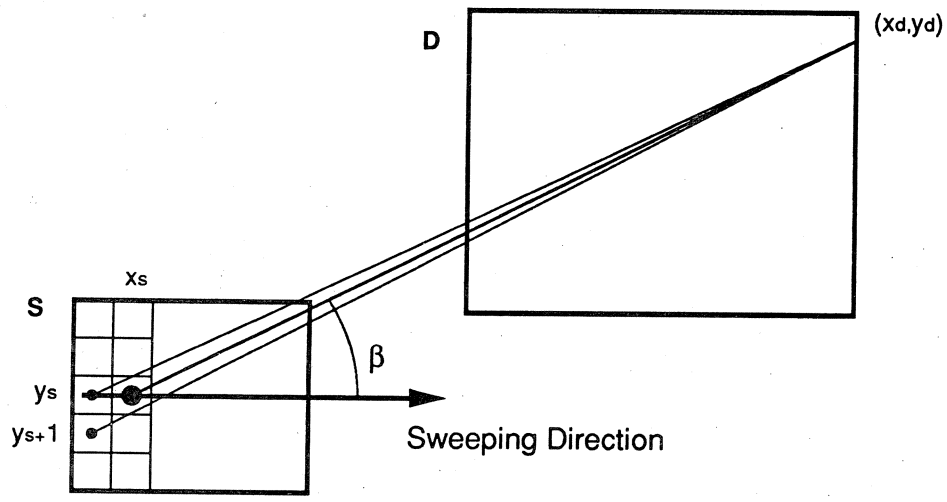


Figure 3. Coherence between rays in consecutive strips

[1] For a better mapping between the geometry in the Figures and the ordering in the data structure, a screen co-ordinate system is adopted in all Figures, i.e. the origin is in the upper left corner and the Y-co-ordinates increase downwards.

the vertical width of the triangle is less than 1 from x_s to x_d, which implies that the grid cells along this ray must also belong to at least one of the two enclosing rays. This observation suggests the idea of a sweeping algorithm, which analyses the visibility for a column of the source cells at a time and sweeps across the source from left to right. With proper arrangement, the elevation data may be rearranged by local communication within the ray structure to reduce global communication between the ray structure and the terrain cells.

However, this enclosing property holds only when the angle between the ray and the sweeping direction is no greater than 45 degrees. This angle is referred to as the *emitting angle* of the ray with respect to this sweeping direction. To ensure that the above property holds during a sweep, all rays are partitioned into four groups according to the position of the end point of a ray with respect to its starting point, as shown in Figure 4. Each group is analysed in a separate sweep. Sweeps are named by their sweeping directions, i.e. east, west, north and south. Generally speaking, each ray is assigned to a group such that the emitting angle with respect to the sweeping direction is minimized. Ties are broken by a priority in the order of east, west, south and north. This partition leads to the following lemma:

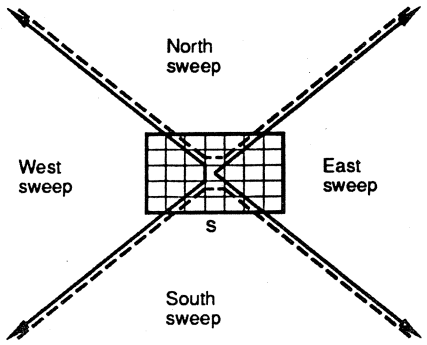


Figure 4. Partition of the rays

Lemma 1: The emitting angle of a ray with respect to its corresponding sweeping direction is less than or equal to 45 degrees

In each sweep, a *strip* of the source region is analysed at a time, where a strip is either a row or a column that is perpendicular to the sweeping direction. The *projected direction* of a ray is the direction of its projection on the strips. A sweeping direction also defines an order among strips. The first strip in a sweep is called the *initial strip*. The idea of our sweeping algorithm is to obtain the elevation data from the terrain cells only for the initial strips and to locally rearrange the data within the ray structure for the subsequent strips, taking advantage of the coherence between consecutive strips. This coherence property is formally defined in the following paragraphs.

Definition 2: The parent *of a ray is the ray starting from the cell that precedes the ray's starting point in the sweeping direction and ending at the same destination point. The* guardian *of a ray is the ray starting from a cell that precedes the parent's starting point in the ray's projected direction and ending at the same destination point*

These definitions are illustrated in Figure 5.

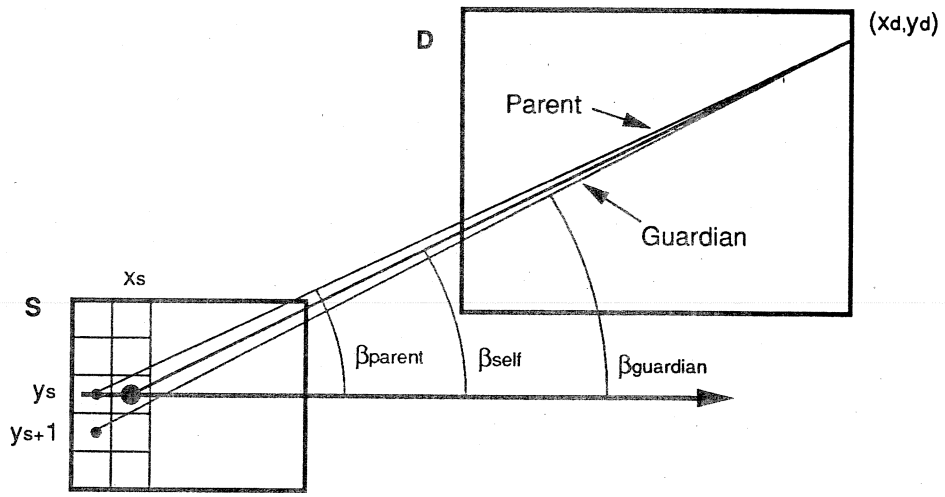


Figure 5. The enclosure of a ray by its parent and its guardian

Lemma 3: Every grid cell of a ray also belongs to its parent or its guardian

Proof

We prove this lemma for the east sweep using Figure 5. Other sweeps can, of course, be treated identically.

(1) The ray is enclosed by the triangle formed by the parent and the guardian. Let

$$dx = |x_d - x_s|$$

and

$$dy = |y_d - y_s|$$

Also let β_{self}, β_{parent}, $\beta_{guardian}$ be the emitting angles of the three rays, respectively. Then

$$\tan\beta_{self} = \frac{dy}{dx}$$

$$\tan\beta_{parent} = \frac{dy}{dx+1}$$

$$\tan\beta_{guardian} = \frac{dy+1}{dx+1}$$

Since the ray belongs to the east sweep, β_{self} is less than or equal to 45°. We have

$$dy \leq dx$$

So

$$\beta_{parent} < \beta_{self} \leq \beta_{guardian} \leq 45°$$

which states the fact that the ray is enclosed by its parent and guardian.

(2) Since all the angles are less than 45°, there will be a single grid cell in each ray for each grid step along the X-axis, according to the principle of the DDA technique. The vertical width of the triangle is always less than one grid cell for the range of X-co-ordinates from x_s to x_d, so the cell of the ray corresponding to each grid step along the X-axis will be identical to the grid cell of either the parent or the guardian.

Lemma 4: A ray, its parent and its guardian belong to the same sweep

Proof

This lemma follows from the relation among the three emitting angles and the above partition rule. For the special cases where $\beta_{guardian}$ is 45°, β_{self} must also be 45°. In fact, the ray will coincide with its guardian in such cases, and thus will be assigned to the same sweep.

Lemmas 3 and 4 imply that the elevation data may be obtained from within the ray structure when the analysis advances from one strip to the next strip. However, parents and guardians may not start from within the source region. For those rays starting from the initial strip of a sweep, their parent and guardian will start from outside the source region, and it is necessary to obtain the elevation data for them by global communication. If a ray starts at a region border parallel to the sweeping direction, its guardian may also start from outside the source region, as shown in Figure 6. It is not advisable to use global communication to obtain elevation data for these rays, since all the processors have to wait for the time-consuming communication of a few processors on a SIMD machine like the Connection Machine. Our solution to this problem is to extend the source region[2] by the side length parallel to the sweeping direction minus one, start each sweep with the *extended initial strip*, and ignore the rays that cannot obtain valid data from inside the ray structure. When the analysis advances to the next strip, only rays starting from the end(s) of the extended strip may fail to obtain valid data. Since each extension shrinks by at most 1 after each iteration, extending the initial strip by $L_S - 1$ will ensure that each ray from the original source region can obtain the required data. See Figure 6 for an illustration.

3.2. The algorithm

The ideas developed in the preceding subsection are combined into the following algorithm:

[2] Later, we present an alternative and discuss how to minimize the overhead.

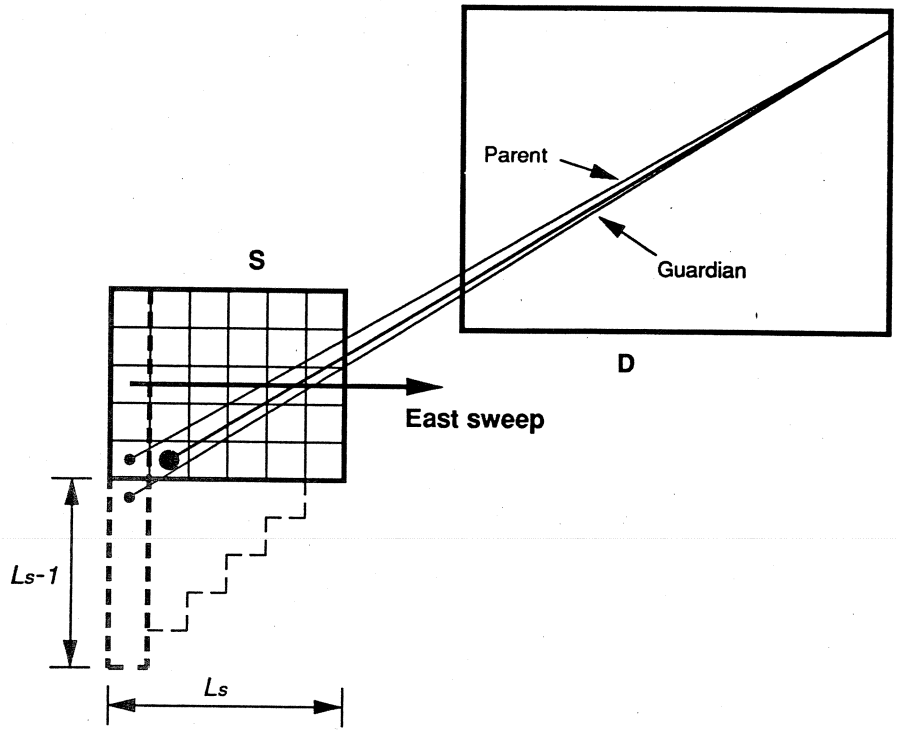


Figure 6. Extended source: the thick broken line shows the extended initial strip, while the thin broken lines show how the portion that contains valid data shrinks during the sweep

```
Algorithm RRVA
begin
  for all grid cells in the source region
    set visible-count to 0;
  for each sweep
    construct the 3D ray structure;
    for all PEs in the ray structure
      compute the corresponding grid cell on the terrain for the initial strip;
      get the elevation data from the terrain map for the initial strip;
      for each strip
        compute the visibility along each ray;
        combine the result in the ray structure;
        update the ray structure for the next strip;
      end for each strip
    send the result back to the source cells on the terrain;
  end for each sweep
end Algorithm
```

Visible-count is the memory location in each processor representing a terrain grid cell that will contain the number of visible grid cells in the destination region. The details of each step are discussed below. All the examples are given for the east sweep and can apply directly to all four sweeps.

3.2.1. Construction of the ray structure

For each sweep, the *active sections* of the far sides are computed. In the RRVA problem, a far side refers to a far side with respect to any cell on the initial strip. An active section of a far side contains all the grid cells that will produce rays belonging to this sweep from the initial strip. This is done by intersecting the far sides with the union of the endpoint locations of all source cells on the initial strip. Figure 7 illustrates this process. If there is no active section, this sweep is omitted. Otherwise, each border parallel to the sweeping direction is checked to see if *any portion* of the destination lies on the same side as the source. If so, the initial strip is extended on that side. Then a ray structure that contains all rays from each source cell on the (possibly extended) initial strip to every grid cell on the active section of far sides is allocated. Since a ray is actually a list of processors, the entire ray structure is a 3D array of processors, which is indexed by three indices, as shown in Figure 8: u is the index for the grid cells along the strip in the source region, v is the index for the grid cells along the far sides of the destination region, and w is the index of grid cells along the ray.

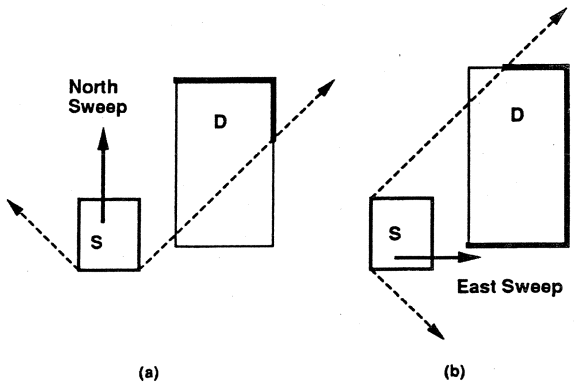


Figure 7. The active sections of the far sides are indicated by the thick line segments along the borders of the destination regions

When the value of one index is fixed, the other two indices specify a 2D slice of the structure. For example, a *v-w slice* refers to the 2D array for a fixed value of u. It corresponds to all rays starting at the same source point and is part of the 2D ray structure for the PRVA of that source point. This correspondence is shown in Figure 9. We also use the terms *u-neighbors*, *v-neighbors* and *w-neighbors* to specify the neighboring elements along each axis.

Two subtle arrangements in Figure 8 should be noted: first, the active sections of the far sides are linked into a continuous border so that two neighboring cells on the sections always have consecutive values of v; second, for all source cells, their corresponding

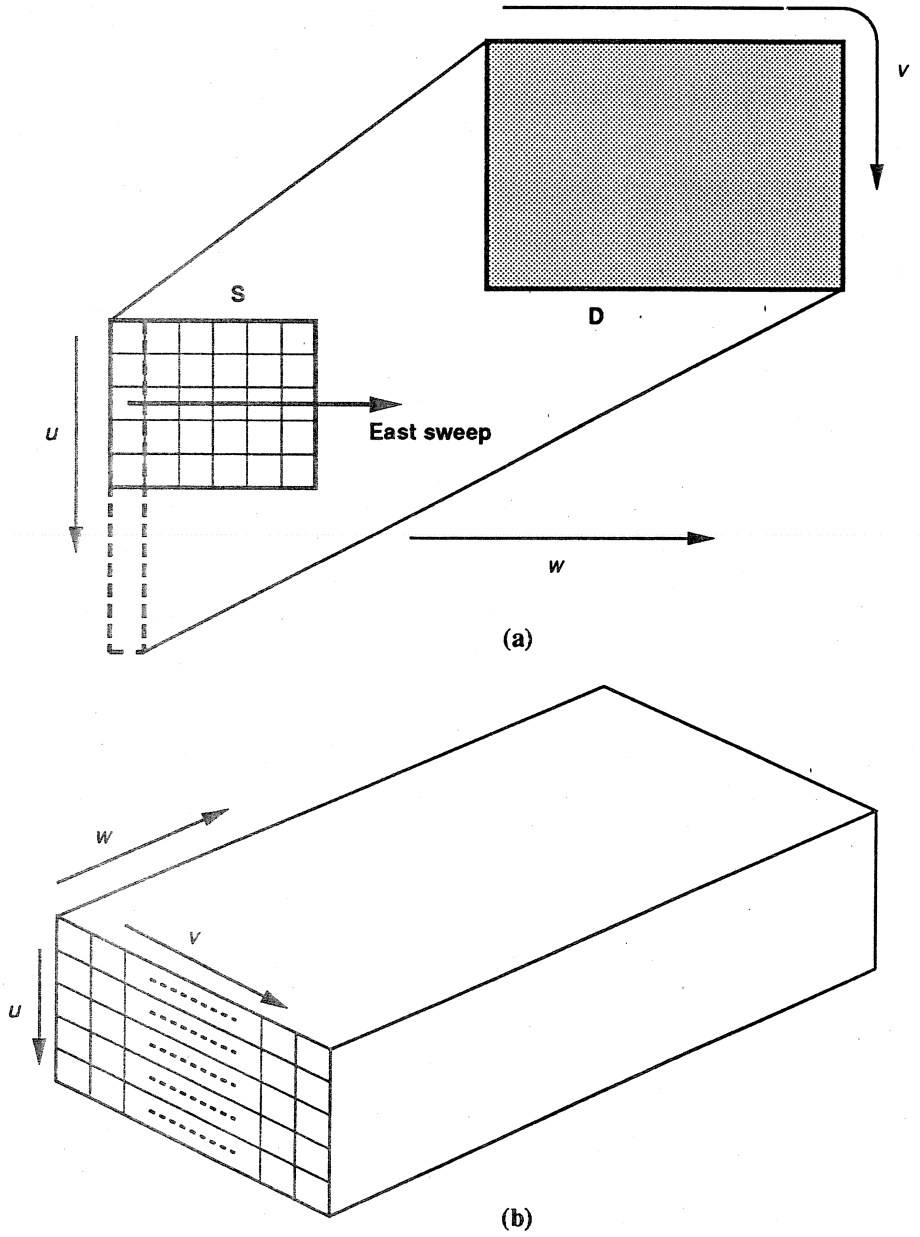


Figure 8. The 3D ray structure: (a) the corresponding dimensions on the terrain; (b) the ray structure

v-w slices are built for the same portion of the far sides. So some of the rays in the ray structure may not be in this sweep. For example, in Figure 9, the active section starts at the upper left corner of the destination since the ray from the top cell of the strip to the corner is in the east sweep, but the ray from the third source cell to that corner does not

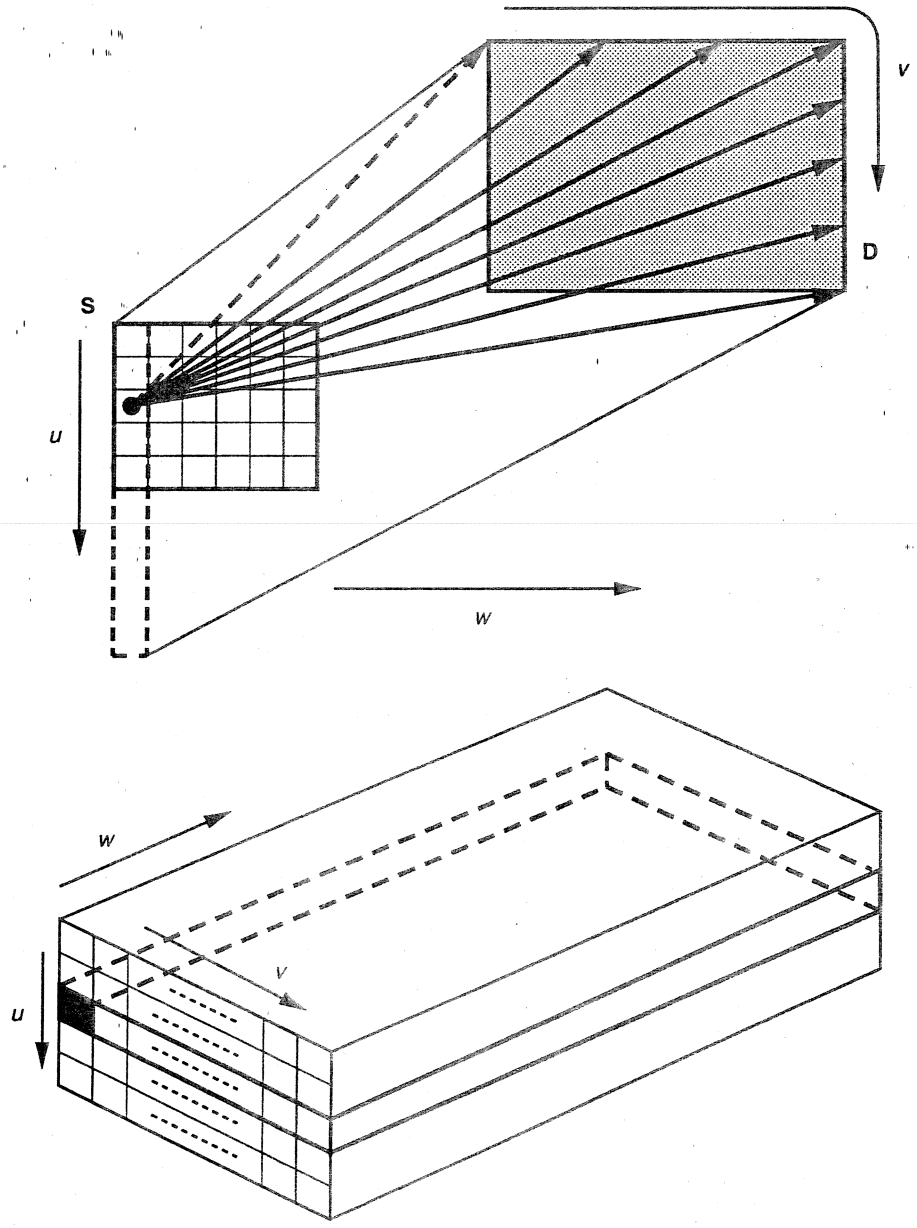


Figure 9. The v-w slice and its corresponding rays

belong to the east sweep. This ray is nevertheless allocated in the ray structure but will be deactivated from the following computation. It is marked by a black cell in the Figure. At the price of a few extra rays, this alignment ensures that all rays with the same value of v will end at the same grid cell and simplifies the computations and communications

in the ray structure. Actually, under these arrangements, a ray and its parent are using the same list of processors in the ray structure, and the guardian of a ray is just one of its two u-neighbors depending on its projected direction.

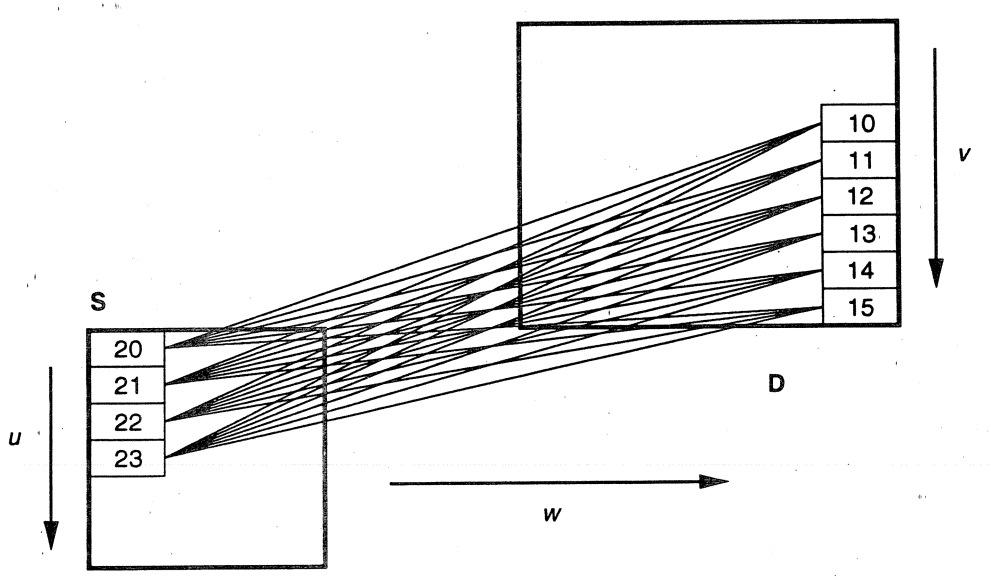
3.2.2. *Obtaining the elevation data for the initial strip*

By broadcasting the co-ordinates of the initial strip and the far sides, each PE in the ray structure can compute its corresponding grid cell on the terrain according to the three indices, in a manner similar to that in the PRVA algorithm. The address of the processor corresponding to that terrain grid cell can be computed and the elevation data can be obtained by global communication. Since many PEs in the ray structures may be mapped to the same grid cell on the terrain, concurrent reads will cause serious congestion. This situation can be alleviated by a grouping technique similar to the one employed in the PRVA algorithm.

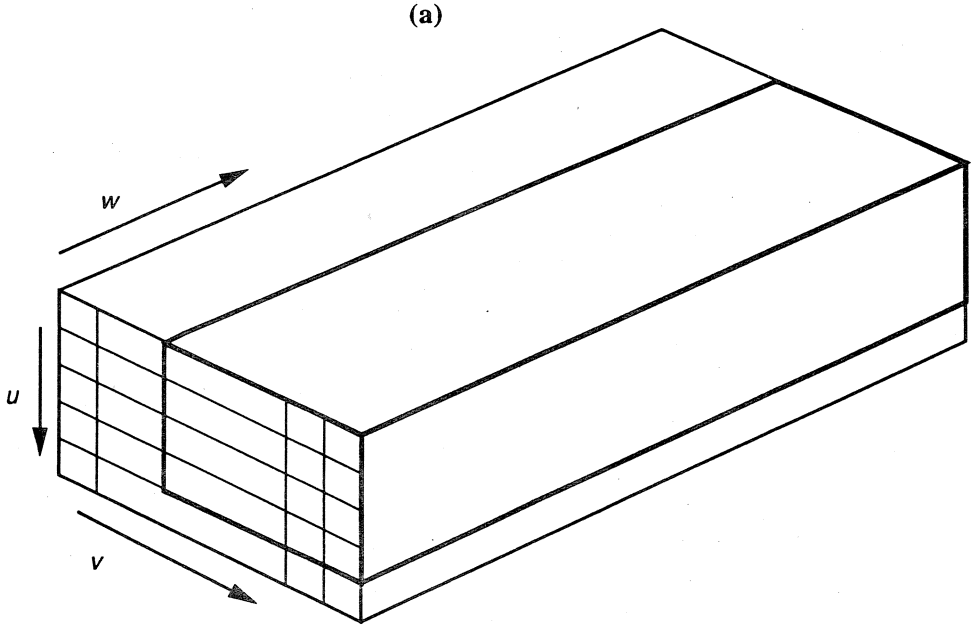
Since w indexes the grid cells along the sweeping direction, two PEs with different values of w will not be mapped into the same grid cell. So duplications will only occur within a u-v slice. We use the east sweep example in Figure 10 to illustrate the duplication in the ray structure. Figure 11 shows portions of several u-v slices for different values of w. The values shown in each slice indicate the Y-co-ordinates of the corresponding terrain grid cells for the PEs. PEs with the same Y-co-ordinate in a slice correspond to the same terrain cell. From the four slices in Figure 11, we can see that the distribution of duplications changes from along the v-axis at $w = 0$, to diagonally across the u and v axes when w equals half of its maximal value, to along the u-axis at the maximal w. We can use this property to reduce concurrent reads in the global communication by grouping the PEs of the same terrain cell and allowing only one of them to participate in the global communication. Since the range of v is usually larger than that of u, we group the PEs of the same terrain cells along the v-axis first, and then merge groups of the same cells along the u-axis if the data can be copied among the groups efficiently.

The algorithm for this operation is given below. Two flags, namely the u-start and the v-start, are allocated in each PE to indicate if the PE starts a group along each axis.

```
Global communication for the elevation data
begin
  for all active PEs in the ray structure
    set the u-start and v-start flags;
    if the higher v-neighbor corresponds to the same terrain cell
      unset the v-start flag;
    for all PEs whose v-start flag is set
      if the lower u-neighbor corresponds to the same terrain cell
          and its v-start flag is set
        unset the u-start flag;
    for all PEs whose u-start flag is set
      read the elevation data from the corresponding terrain cell;
    copy the data along the positive u-axis by a segmented parallel prefix operation
        in which u-start marks the start of a segment;
```



(a)



(b)

Figure 10. An example of the ray structure in detail

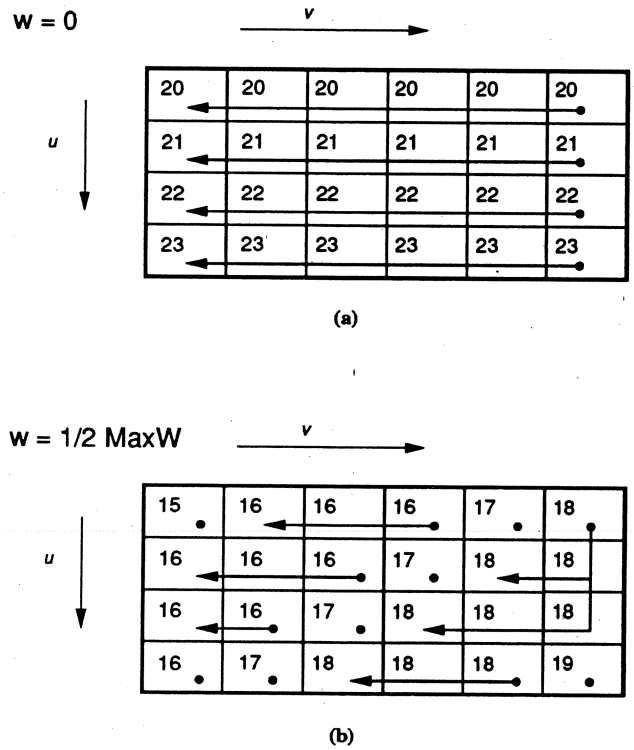


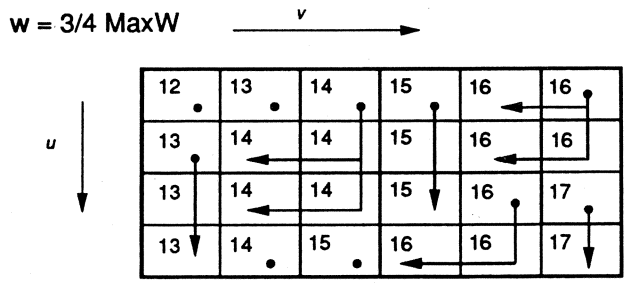
Figure 11. The u-v slices at different values of w

copy the data along the negative v-axis by a segmented parallel prefix operation
in which v-start marks the start of a segment;
end for all
end

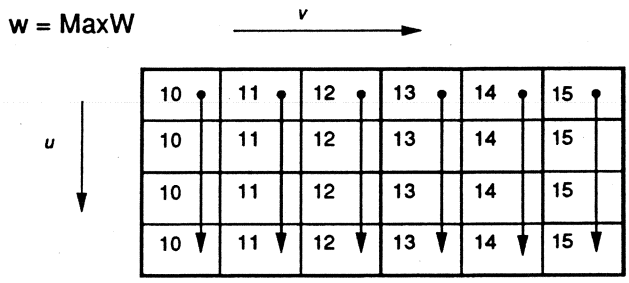
Figure 11 also shows how this algorithm works. The solid dots show which PEs participate in the global communication and the arrowed lines indicate how the data are copied inside the ray structure. This algorithm reduces the concurrent reads but cannot eliminate them completely. In the worst case, like the diagonal distribution in Figure 11(b), the number of concurrent reads is reduced to the cardinality of u.

3.2.3. The strip-wise iterations

After obtaining the elevation data from the terrain map for the initial strip, a for-loop computes the visibility for each strip. In the beginning of the loop, the ray structure should contain all the rays starting at this strip and each ray should have elevation data for the corresponding terrain grid cells. Therefore the visibility analyses are conducted along all



(c)



(d)

Figure 11. continued.

rays in parallel as explained in Section 2. This computation determines the value of a visibility flag in each PE. Since several rays emanating from the same source point may pass through identical terrain cells, these results should be combined in the ray structure before they are sent back to the source cells. From our previous explanation, rays for the same source point form a v-w slice in the ray structure, and the rays passing through the same terrain cell are neighbors along the v-axis, so a segmented OR scan is conducted along the v-axis to combine the results in each group of PEs that are mapped to the same terrain cell.

After this combination, a reduction is conducted in each v-w slice to compute the total number of visible destination cells for that source point. This total is stored in the first active PE of the first active ray for each source cell on the strip. Since the first active PE of each ray will be deactivated in the next step, this value will not change during the remainder of the sweep. Then the results for all source cells will be sent at the same time to reduce the occurrence of global write operations to once per sweep. Proper deactivations should be done to the PEs corresponding to the borders among sweeps to avoid redundant counting of those cells. This is illustrated in Figure 12. Cell A is a source

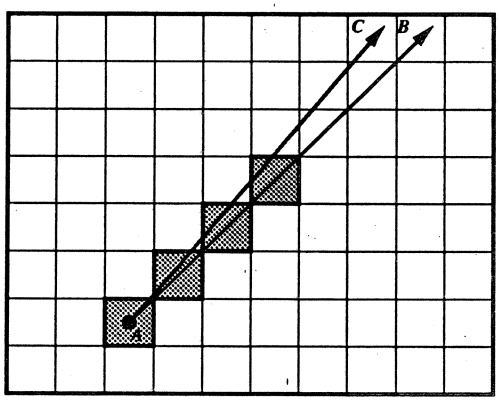


Figure 12. Avoiding redundant counting along sweep borders

cell, and cells B and C are on the far side of the destination region. According to the partition rule, ray AB belongs to the east sweep, while ray AC belongs to the north sweep. Since the first four grid cells along both rays are identical (the shaded cells in the Figure), they should be counted in only one sweep and deactivated in the other sweep to avoid redundant counting. The partition rule in Figure 4 is adopted to determine in which sweep a destination cell will be counted.

Then the ray structure should be updated for the analysis of the next strip; the steps are shown below:

```
Updating the ray structure for the next strip
begin
  de-activate the first active u-v slice;
  for all active PEs in the ray structure
    change the starting point of each ray to the next grid point
      along the sweeping direction;
    compute the new corresponding terrain cell;
    if it is different from the previous one
      if there is a u-neighbor previously corresponding to this terrain cell
      then
        get the elevation data from this u-neighbor;
      else
        de-activate the entire ray the PE is in;
    end if
  end for all
  de-activate rays which emitting angles are greater than 45 degrees;
end
```

After the iterations, the counts of visible destination cells stored in the ray structure are sent back to the source cell on the terrain and are added to the results from other sweeps.

This is an exclusive write operation since only one PE was designated to keep the count for each source cell in each sweep and only these PEs will participate in this communication.

3.3. Improved complexity

The resulting complexity of this RRVA algorithm is

$$O(L_S^2 L_D W \log W + L_S L_D W \times Comm) \text{ operations}$$

with a minimal

$$O(L_S \log W + Comm_{write} + Comm_{read}) \text{ time}$$

using up to $L_S L_D W$ processors.

In comparison to applying PRVA to all source points, global communication in this algorithm is reduced in three ways: the number of occurrences of global communication operations is reduced to only two (one for read and one for write), the total number of such operations is reduced by a factor of L_S, and the congestion due to concurrent read/write operations is minimized; actually the global write operation is exclusive. The computation part is the same as the PRVA algorithm, since the same number of rays are allocated for each point and the computation along each ray remains the same.

4. OVERHEAD AND FURTHER IMPROVEMENTS

The algorithm proposed in the preceding Section alleviates the communication bottleneck, but the use of extended sources may introduce some overhead. The overhead is determined by the shape of the source and the geometric relation between the source and destination. We describe this overhead and further improve our algorithm in this Section:

1. Extended sources are needed at both sides for most cases. Figure 13 shows an east sweep example. To obtain information from the guardians, the extended sources are needed at both sides because the guardians of the two rays shown in the Figure start from outside the rectangle. However, we find that all the terrain grid cells involved in the sweep are already covered by the rays starting from the unextended initial strip. More coherence properties might be found from the ray structure.
2. When the required number of PEs is not available, we can decompose the source region or the destination region to form smaller problems to fit the size of available machines. In general, the problem size is determined by the perimeters of the source and the destination, so a problem has to be decomposed into four subproblems to make each of them half of the original size. Figure 14 illustrates the case of decomposing the source region. In other words, if the original problem requires P_r processors, and only P_a processors are available, the number of subproblems generated by decomposition in the worst case is $(P_r/P_a)^2$.

Both sources of overhead can be avoided if the rays starting from the borders along the sweeping direction do not rely on guardians to obtain elevation data.

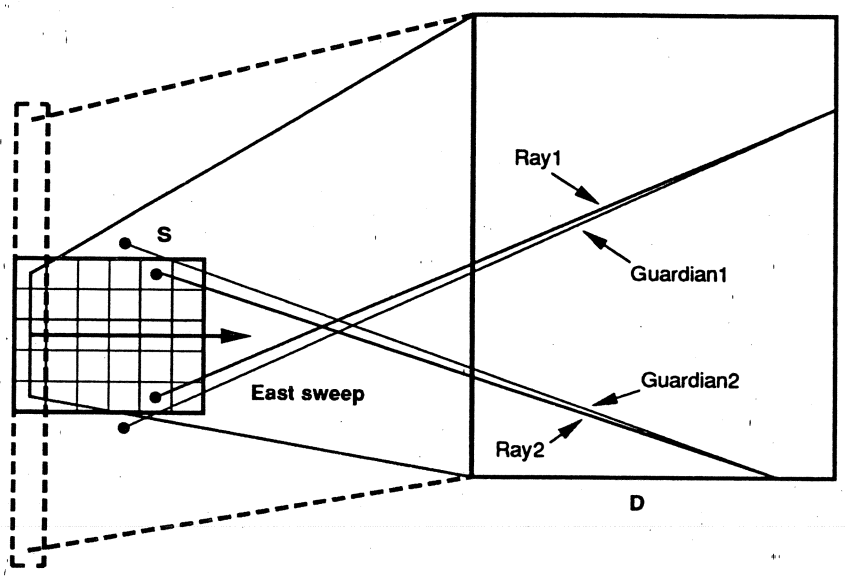


Figure 13. The overhead of using extended sources

Definition 5: The friend *of a ray is the ray starting from the same source point but ending at a neighboring cell of the end point on the far side of the destination such that the emitting angle of the friend is smaller than that of the ray itself*

The definition is illustrated in Figure 15. In terms of the ray structure, the friend of a ray is just one of the two v-neighbors depending on its projected direction.

Lemma 6: Every grid cell of a ray belongs to either its parent or its friend

Lemma 7: A ray and its friend belong to the same sweep if the friend exists

Both lemmas can be proved by the same method used for the guardian. However, in the algorithm, the data in a ray and its friend are updated at the same time. When a PE in a ray fails to obtain the elevation data from its parent and asks for the data from its friend, the corresponding PE in the friend may not have the correct data either; it may also be asking for data from its friend. (That is why we call it a 'friend'.) This recursive relation stops if there is a ray emitting along the sweeping direction; this ray coincides with its parent and hence has all the needed data from its parent. We call this ray the *leader*. If a leader exists, all other rays will be able to obtain the elevation data by propagation. Actually, in our ray structure, this data acquisition can be done by a parallel prefix operation that copies data along the v-axis from those who have the correct data by parents to those who cannot obtain the data from their parents.

What if there is no ray emitting along the sweeping direction? We can choose from two options:

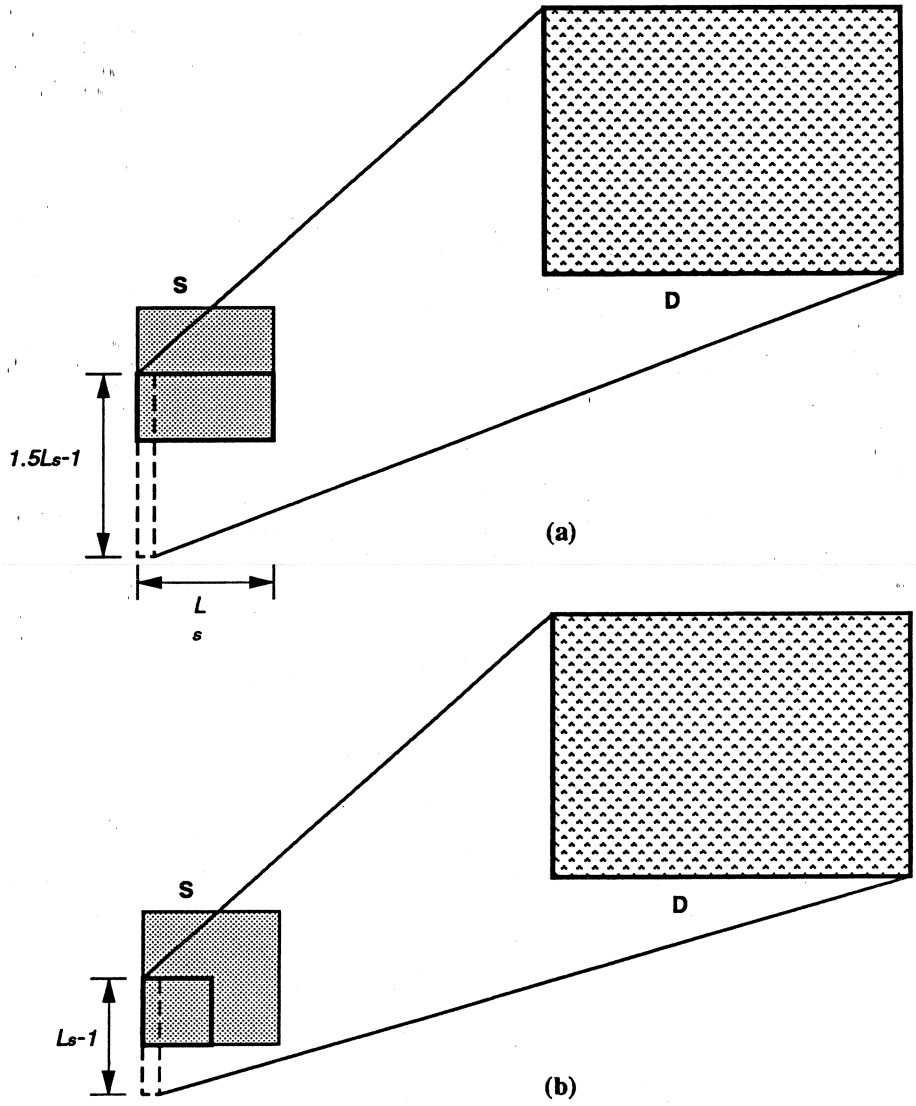


Figure 14. The overhead of decomposition

1. Use an extended source and guardians instead.
2. Use an *extended destination* to ensure the existence of the leader; the destination is extended to be aligned with the border of the source along the sweeping direction. See Figure 16.

We can choose the case that is favorable to the algorithm based on the geometric relation between the source and the destination. The decision should be made in the beginning of the algorithm so that the ray structure is organized appropriately. In general, there are four cases for the relation between the source and the destination, as shown in Figure 17 for the east sweep:

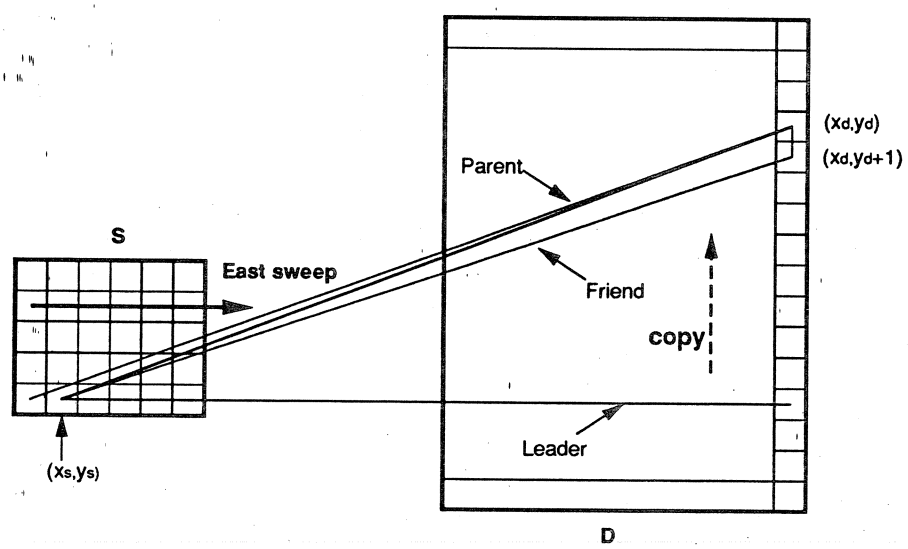


Figure 15. The friend of a ray

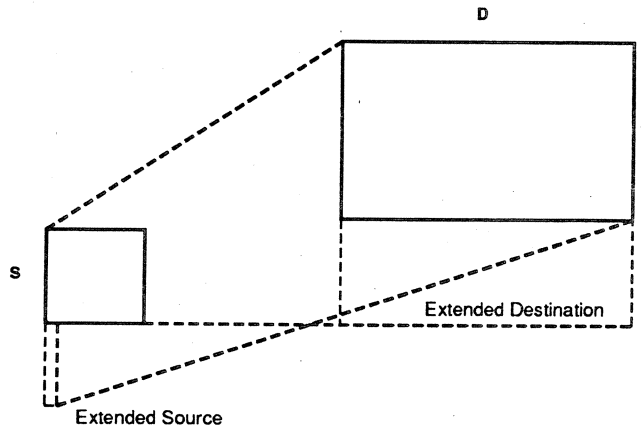


Figure 16. Extended destination against extended source

Case 1: The source strip is *covered* by the destination when projected along the sweeping direction (see Figure 17(a)). Leaders exist for both ends of the strip, so the rays starting from the borders do not rely on their guardians and the extended sources are not necessary. If the problem size is too large, decompose the source region along the sweeping direction into halves. This produces two subproblems each of size one half of the original problem. For a given number of processors, we need only to decompose the problem into (P_r/P_a) subproblems. So there is no overhead in the number of subproblems in this case.

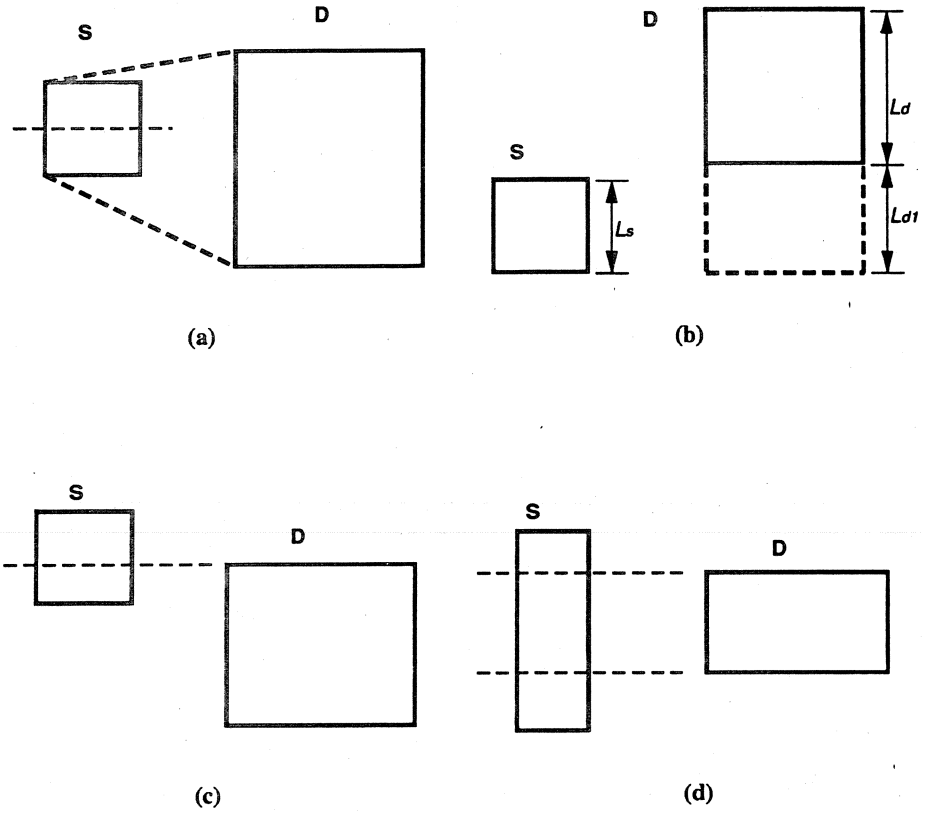


Figure 17. The decomposition rules

Case 2: The source strip is *disjoint* from the destination when projected along the sweeping direction (see Figure 17(b)). An extension of the source or destination is needed. The number of processors required can be computed to determine which strategy is better: assume both the source and the destination are squares with side length L_s and L_d, respectively. Let L_{d1} be the length of the extended destination, and W be the maximal number of grid cells along a ray. The number of processors required for using an extended source is

$$P_{rs} = (2L_s - 1) \times 2L_d \times W \approx 2L_s \times 2L_d \times W$$

and for using an extended destination is

$$P_{rd} = L_s \times (2L_d + L_{d1}) \times W$$

The numbers of subproblems in the two cases are, respectively,

$$(P_{rs}/P_a)^2 \text{ and } (P_{rd}/P_a)$$

With a little derivation, we obtain the result that the number of subproblems favors the use of an extended destination if

$$\frac{L_{d1}}{L_d} < 4\frac{P_{rs}}{P_a} - 2$$

Case 3: The source strip *partially overlaps* the destination when projected along the sweeping direction (see Figure 17(c)). The leader exists in the covered side of the strip while the other side requires an extension of the source or the destination. Usually extending the destination will make the problem size smaller. However, we would rather avoid extending the destination in this case, because the extension of the destination changes the far sides, and hence changes the allocated rays. This may generate a slightly different result from what we obtain by using the PRVA on each source point due to the nature of digital approximation. For a consistent result we prefer using an extended source. If a decomposition is needed, the rule of thumb is to cut the source by the line that extends a border of the destination. This will generate two subproblems which fall into the first two cases and can be treated accordingly.

Case 4: The source strip *covers* the destination when projected along the sweeping direction (see Figure 17(d)). For the same reason as in case 3, we extend the source on both ends. If this turns out to be too costly, we decompose the problem by cutting the source along both borders of the destination.

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

This algorithm has been implemented on a Connection Machine CM-2. In this Section we present some experimental results from our implementations. All experiments were conducted on a Connection Machine CM-2 using 8K or 16K processors. The terrain size is 512 × 512 grid cells. We allocate one virtual processor to each grid cell, so the virtual processor ratio (VPR) is 32 or 16 for the terrain processor set. The size of the ray structure depends on each instance of problem.

Figure 18 shows the experimental results for different geometric relationships between the source and the destination regions. Both the source and the destination regions are of size 64 × 64. Figure 18(a) is a gray-level display of the relevant portion of the terrain in which brighter pixels are higher. Figure 18(b) shows the number of visible destination cells for each cell in the source region by its brightness: brighter pixels have higher visibility to the destination region. The destination region is detached from the source region, and is indicated by a black frame on the terrain. Figure 18(c) shows for the same source region the visibility to a partially overlapping destination region; Figure 18(d) shows the visibility when both regions are identical.

For cases where both regions are identical and square, a timing result for different side lengths is shown in Figure 19. This experiment was done on a CM-2 with 16K processors. Both axes are in logarithmic scale. The turning point at side length 16 indicates where the size of the ray structure reaches the number of physical processors in the machine. According to the complexity analysis in Section 3.3, the computational time complexity is of $O(L \log L)$ before the machine is fully utilized, and is of $O(L^4 \log L/p)$ after the machine is fully utilized, where L is the side length of the square and p is the number

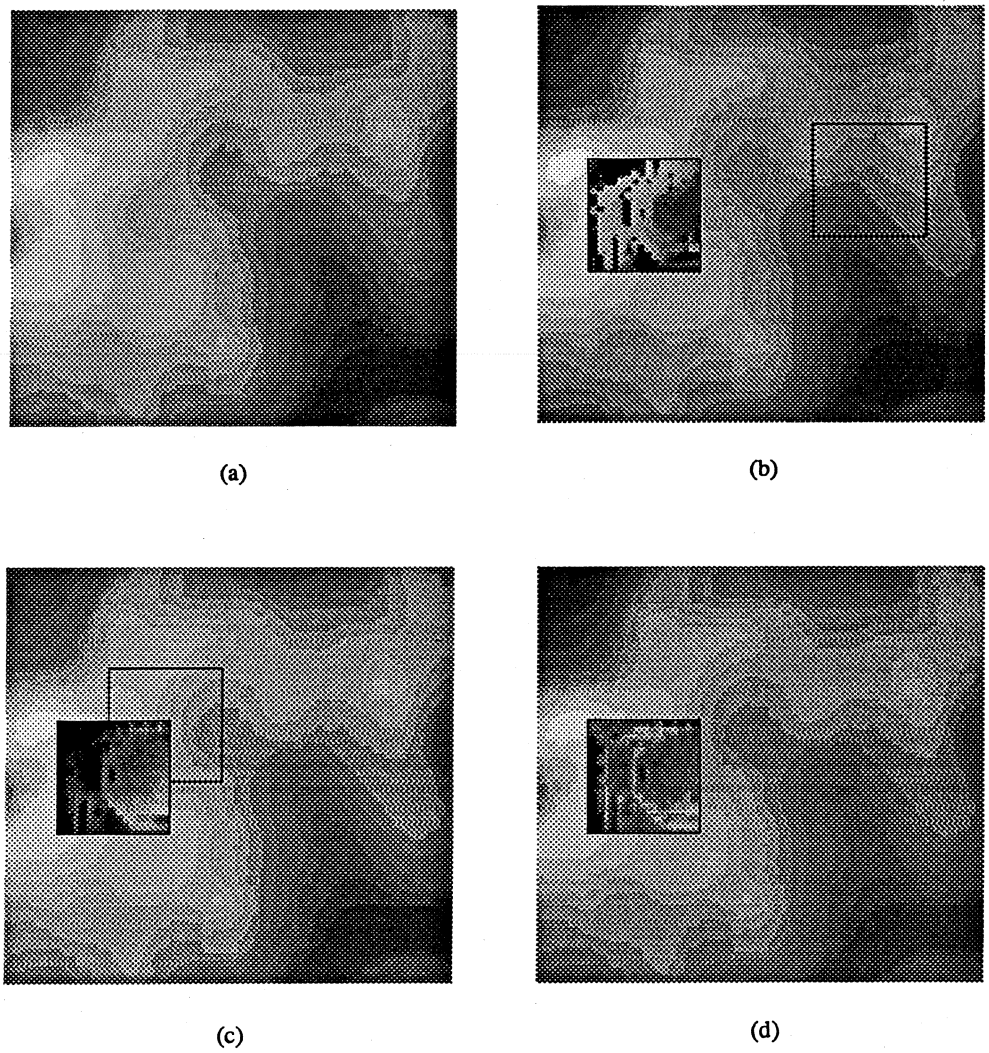


Figure 18. Experimental results for different geometric relationships between the source and the destination regions: (a) the terrain; (b) detached regions; (c) partially overlapping regions; (d) identical regions

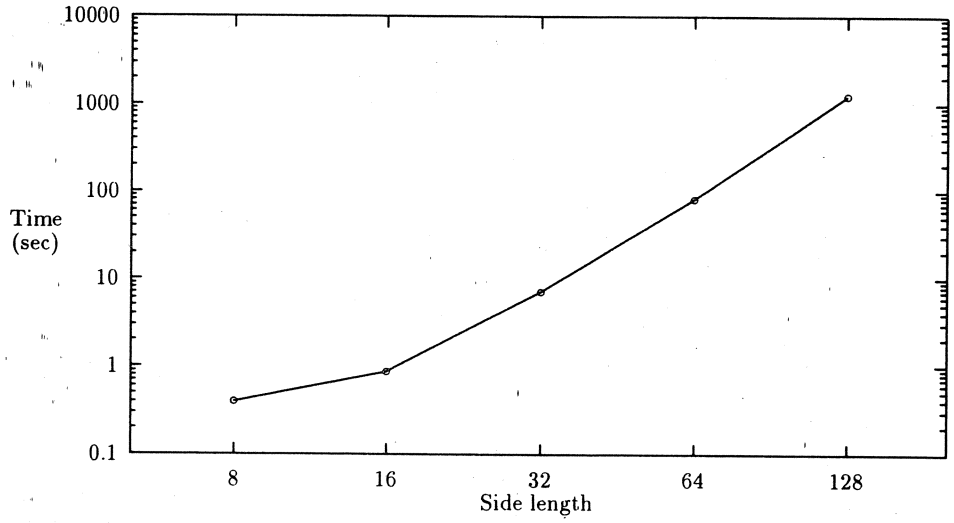


Figure 19. Running time against side length of the square region

of physical processors. The running time curve does reflect these terms, though it does not grow as fast due to the reduced communication and the use of virtual processing.

We also made comparisons on timing results to show the improvements. Table 1 shows the reduction in communication time and its resulting reduction in running time and communication ratio. This experiment was also done on a CM-2 with 16K processors. The size of the source is 16 × 16, and the destination is 64 × 64. The maximal number of grid cells along a ray is 256. The size is chosen such that the VPR is 1 for each analysis from a point to a far side in the PRVA (64 × 256 = 16K), and that the problem is not so large as to require decomposition in RRVA. The relation of the two regions is of case 1 and the destination lies completely in the east sweep. The first column shows the time consumed by applying the PRVA algorithm point by point without any reduction of global communication. The second column is the result of applying PRVA with modifications in the communication that avoids concurrent reads by copying data in the ray structure, and the third column is the result of our RRVA algorithm with virtual processor ratio of 64. It is clear that the communication time is greatly reduced at a very small price, which is shown in the last row as the time of the copying operations. This reduction results in the faster running time as well as the reduction in communication ratio.

Table 2 shows the improvements of the algorithm obtained by using friends and the extended destination instead of the extended source. The two columns on the left present results obtained by using only the extended source, whereas in the two columns on the right-hand side we chose the best strategy between the two options. This experiment was done on a CM-2 with 8K processors. The size of the source is 16 × 16, and the maximal number of grid cells along a ray is 256. The maximal VPR is limited to 8 so that the algorithm starts decomposing the source at the second instance. The geometric relation corresponds to case 2 and the destination lies completely in the east sweep. The columns labeled '#Dcp' show the number of subproblems generated by the decomposition mechanism. The number increases quadratically when using an extended source only and increases linearly after the modifications.

Table 1. Reduction in communication time

Time, s	PRVA	PRVA w/copy	RRVA
Total time	68.46	36.43	11.89
Global read time	49.74	15.90	3.38
Read/total	73%	44%	28%
Copying time		0.79	0.05

Table 2. Improvement on decomposition overhead

Size of D	Time, s	#Dcp	Time, s	#Dcp
8×8	1.95	1	1.94	1
16×16	5.01	4	4.63	2
32×32	11.54	16	9.25	4
64×64	26.00	64	18.03	8

6. CONCLUSION

In this paper we have proposed an algorithm for solving region-to-region visibility problems on digital terrain models using data parallel machines such as the Connection Machine CM-2. This algorithm is an extension of the point-to-region visibility algorithm presented in Reference 1. Since global communication is the bottleneck in this kind of algorithm, the algorithm we propose focuses on the reduction of global communication. The algorithm analyses a strip of the source region at a time and sweeps through the source region strip by strip. At most four sweeps are needed for the analysis. By exploring the coherence properties in the processor structure, global communication is minimized. Furthermore, the global write operations are exclusive and the concurrency in global read operations is minimized. As a result, the complexity of our RRVA algorithm is lower than that of a point-wise application of the PRVA algorithm. The algorithm was implemented on a Connection Machine CM-2. Experiments show that this algorithm does improve the running time substantially. Therefore, we conclude that for the region-to-region visibility problem discussed in this paper, this algorithm gives an affirmative answer to the question whether the cost of a visibility analysis for N viewing points can be less than N times that for a single point.

ACKNOWLEDGEMENTS

The support of the Defense Advanced Research Projects Agency (ARPA order no. 6350) and the US Army Engineer Topographic Laboratories under contract DACA76-88-C-0008 is gratefully acknowledged.

REFERENCES

1. Y. Ansel Teng, Daniel DeMenthon and Larry S. Davis, 'Stealth terrain navigation', Technical Report CAR-TR-532, Center for Automation Research, University of Maryland, 1991; also to be published in *IEEE Trans. Systems, Man, and Cybernetics.*

2. Y. Ansel Teng, Daniel DeMenthon and Larry S. Davis, 'Stealth terrain navigation with bounding overwatch', in *Proc. Image Understanding Workshop*, 1992, pp. 979–989; also published in *Int. J. Pattern Recognit. Artif. Intell.* **6**, 395–416 (1992).
3. Richard Cole and Micha Sharir, 'Visibility problems for polyhedral terrains', *J. Symbolic Comput.*, **7**, 11–30 (1989).
4. Dz-Mou Jung, 'Comparisons of algorithms for terrain visibility', Master's thesis, Rensselaer Polytechnic Institute, Troy, New York, 1989.
5. John Reif and Sandeep Sen, 'An efficient output-sensitive hidden-surface removal algorithm and its parallelization', in *Proc. 4th ACM Symposium on Computational Geometry*, 1988, pp. 193–200.
6. W. Daniel Hillis, *The Connection Machine*, MIT Press, 1985.
7. Guy E. Blelloch and James J. Little, 'Parallel solution to geometric problems on the scan model of computation', in *Proc. of the 1988 International Conference on Parallel Processing*, 1988, pp. 218–222.
8. D. Hearn and M.P. Baker, *Computer Graphics*, Prentice-Hall, 1986.