STEALTH TERRAIN NAVIGATION FOR MULTI-VEHICLE PATH PLANNING

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

In this paper, we propose a method for solving visibility-based terrain path planning problems for groups of vehicles using data parallel machines. The discussion focuses on path planning for two groups of vehicles so that they move in a bounding overwatch manner. Furthermore, the planned paths for the vehicles themselves are subject to intervisibility constraints, configuration constraints, and different terrain traversabilities due to variations in terrain type and slope. A spatial-temporal sampling approach is adopted to discretize the solution space and facilitate fast computation on a data parallel machine. One of the key computations in the planning is the region-to-region visibility analysis, which is computationally expensive but essential to the choice of subgoals to carry out reconnaissance activities. A parallel algorithm for this analysis is developed. By reducing the communication complexity, our algorithm achieves much faster running time than traditional methods. The algorithms are implemented on a Connection Machine CM-2, and the experimental results show that the planning system effectively generates good paths.

Keywords: Terrain navigation, visibility analysis, bounding overwatch, parallel algorithms.

1. INTRODUCTION

This paper describes a path planning method for stealth terrain navigation with bounding overwatch using a data parallel machine. This is an extension of the stealth terrain navigation approach used by Teng, DeMenthon, and Davis.[1] We consider the problem of planning paths for two groups of vehicles, each of which consists of two vehicles, from their common initial location to their common final goal. The terrain through which they must move is "hostile" in the sense that there are adversaries moving through the terrain. The vehicles should remain hidden from the adversaries to the greatest extent possible. Initial information is available concerning the locations and movements of these adversaries, but it is expected that these models will degrade over time, so that the plan developed must support reconnaissance activities. This requirement should be fulfilled in a bounding overwatch manner such that one of the two groups serves as observer while the other group moves along a safe path to a new observation point using the information collected from the observer; the two groups then switch roles. Furthermore, the planned paths for the vehicles themselves are subject to intervisibility constrains for line of sight communication, and configuration constraints such as those requiring that the two vehicles in a group move in parallel. The progress of the vehicles through the terrain is differentially impeded by terrain type and slope. Generally, these problems are instances of path planning in two-dimensional space with time varying constraints. Such problems are known to be

International Journal of Pattern Recognition and Artificial Intelligence Vol. 6 No. 2 & 3 (1992) 395–415

computationally hard.[2,3] This will lead us to the development of heuristic and approximate algorithms that can avoid a direct assault on these combinatorial problems, while at the same time developing demonstrably good solutions to such problems.

Path planning on hostile terrains has also been explored by Mitchell[4] using polyhedral terrain models, and by Metea and Tsai[5] using hierarchical terrain models. However, both plan paths for a single vehicle in a static environment, and it does not seem feasible to extend these approaches to solve our problem. Our basic approach is to represent the problem using discretizations of space and time, and to develop data parallel algorithms for the fundamental underlying computations (e.g. visibility analysis, reachability on terrain). The discretization allows many basic computations to be arranged in a regular pattern, and therefore to be solved in parallel efficiently.

The remainder of this paper is organized as follows: in Sec. 2 we describe our planning scheme and its application to the above problem. Section 3 describes the visibility analysis algorithms which are essential to the choice of path points for maximal safety and observation points for reconnaissance activities. The algorithms are implemented on a Connection Machine CM-2[6] and experimental results are shown in Sec. 4.

2. THE PATH PLANNING SCHEME

In our path planning scheme, the path planning process is divided into stages; the two groups take turns serving as an observer while a path is planned for the other group at each stage. The planning process in one stage is referred to as a *subplanning process*, and the length of a stage, referred to as the *subplan interval*, is usually determined before the subplanning process. Since the entire planning process is done by repeating the subplanning processes, the following discussion focuses on the subplanning process. Since a spatial sampling approach is adopted, the terrain is represented by regular grids with elevation data at each grid cell.

We first describe the subplanning process for a single agent, and then extend the approach to plan for a group of agents. The criteria for choosing subgoals in the bounding overwatch problem is described at the end of this section.

2.1. Subplanning for a Single Agent

In the subplan for a single agent, a subgoal is chosen from the area that is reachable within the subplan interval by a good path. In the case of a single agent, a "good" path usually refers to a path that remains hidden from the adversaries as much as possible by taking advantage of the terrain. This property is referred to as *safety*.

Since it is very difficult to represent the visibility map of a moving object analytically, a temporal-sampling approach is adopted, in which a sampling period is determined in advance, a visibility map is computed for each sampling period, and the safety is defined by these samples. The visibility map consists of a binary value on each grid cell and represents the visible regions from the predicted adversary locations.

It is computed by the point-to-region visibility analysis algorithm which will be described in Sec. 3.2.

These visibility maps are combined with terrain traversability using a dynamic programming paradigm to compute the reachable region and evaluate the safety of the best path to each grid cell in the region. This computation is performed incrementally for each sampling period throughout the subplan interval. The data structure can be described as a path credit table

$$Path\_credit(P, t)$$

where P is the cell index and t is the time index. Each element counts the number of safe points along the safest path from the initial location to the cell P at time t. Initially.

$$Path\_credit(P, 0) = \begin{cases} 1 & \text{if cell } P \text{ is the initial location} \\ 0 & \text{otherwise} \end{cases}$$

The table is computed slice-wise for each sampling period from $t = 0$ to the end of the subplan interval, and the computation for each element depends on the terrain traversability.

In our method, terrain traversability is modeled by discretizing the directions in which the agent can move through a grid cell and associating a traversal cost for each of them as the amount of time needed to pass through the cell in that given direction. The cost can be determined by the slope, the type of terrain, and other factors that may affect the mobility. Using this modeling of terrain traversability, the *Path_credit* table is computed according to the following recurrence relation:

$$\begin{aligned} Path\_credit(P, t) = Max\{&Path\_credit(P, t-1) + Safety(P, t), \\ &Path\_credit(Q, t - Tcost(Q, P)) \\ &+ Safety\_count(Q, t - Tcost(Q, P) + 1, t)\} \end{aligned}$$

where Q is any neighbor of P

$Tcost(Q, P)$ = the traversal cost through Q in the direction to P

$$Safety(P, t) = \begin{cases} 1 & \text{if cell } P \text{ is safe at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$Safety\_count(P, t_1, t_2) = \sum_{t=t_1}^{t_2} Safety(P, t)$$

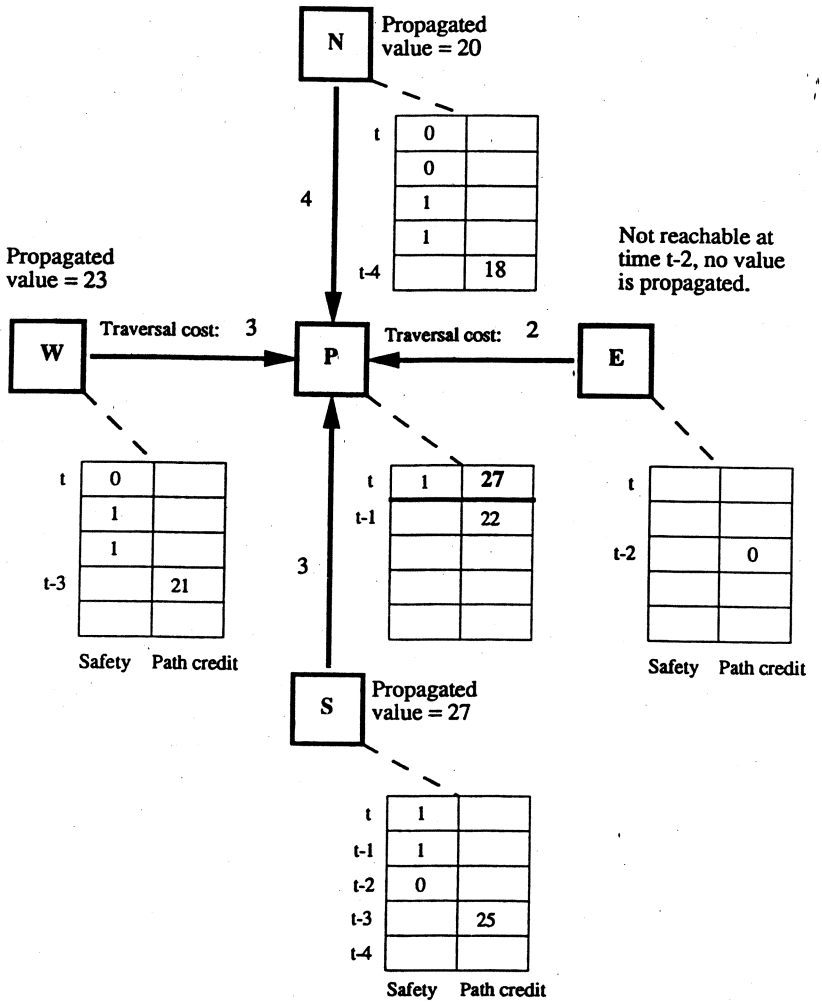


Fig. 1. Construction of the path credit table with the consideration of terrain traversability. For each neighbor of P, the propagated value is the sum of all values shown in the two-column table associated with each cell.

See Fig. 1 for an illustration of this formula, in which four directions are considered for traversing through a cell. One way to interpret this formula is that each grid cell propagates delayed path credits to its neighbors. Each path credit is delayed by an amount of time equal to the traversal cost before it is propagated to the corresponding neighbor and updated with safety information during the delayed period. After receiving the path credits from its neighbors, each cell determines its new path credit by choosing the maximum among the received values and its own previous credit which is also updated by adding the current safety to it.

Clearly, $Path\_credit(P, t) > 0$ if and only if cell P can be reached from the initial location at time t. Thus, the subgoal is chosen only from cells with path credit greater than zero at the end of the interval. After the subgoal is chosen, the path can be extracted from the path credit table by a gradient-following method, or more efficiently, by storing a pointer for each element to the previous cell along the best path during the construction of the table and tracing these pointers back to the initial location.

The following algorithm summarizes the subplanning process for a single agent:

```
Algorithm: subplanning process for a single agent
begin
  for every terrain grid cell P
  begin
    if P is the initial location of the agent then
      Path_credit(P, 0) = 1;
    else
      Path_credit(P, 0) = 0;
    end if
    for t = 0 to end_of_subplan_interval do
    begin
        update Path_credit(P, t) by the recurrence formula;
        Prev(P, t) = the pointer to the cell that contributes to the value of
                     Path_credit(P, t);
    end for
  end for every
  choose a subgoal;
  Path(end_of_subplan_interval) = the subgoal position;
  for t = end_of_subplan_interval to 1 do
    Path(t - 1) = Prev(Path(t), t);
end Algorithm
```

2.2. Subplanning for a Group of Agents

Now we extend the method to plan a path for a group of agents so that they can maintain a given configuration and optimize their safety and mutual visibility during their movement. In our problem, the agents in a group are required to maintain their configuration as a line segment perpendicular to their direction of motion. A line segment can be represented by the location of its center point and its orientation. With the additional time-axis, the dimensionality of the solution space is four. Our framework can be applied directly in this four-dimensional search space, but it is more efficient to make a further reduction of the dimensionality by applying a decomposition.

First we represent the locations of the agents by the center of the segment, and find a good path for the center, where "good" means a high likelihood that a segment

centered at this cell will be safe and intervisible, no matter what orientation the segment is in. This path is actually a corridor for the segment, and the agents are allocated within the corridor using some simple heuristics. Since the corridor has a statistically good evaluation for all the constraints, we should be able to find an acceptable allocation easily. Therefore, we can apply the subplanning method for a single agent to the subplanning for the group center by propagating a path credit defined on both safety and intervisibility.

For a cell to be a potential group center, safety is modeled as the number of safe cells within a circle centered at the cell with a diameter equal to the segment length. This information has to be computed for each temporal sample. For efficiency on regular grids, the circle is approximated by the circumscribing square.

Intervisibility is measured by sampling several line segments centered at a grid cell, and counting the number of pairs of cells that can see each other along these line segments. It needs to be computed only once due to its invariance with time. Currently we sample the horizontal, vertical, and the two diagonal segments to take advantage of the local communication links of the regular grids so that the computation can be done on the terrain efficiently. The intervisibility on each line segment is computed by pipelining the sequential version of the line-visibility algorithm (see Sec. 3.1.).

A weighted sum of safety and intervisibility determines the *point credit* of a cell to be the group center. A threshold on point credit determines the goodness of a cell as a path point, and the path credit is defined as the number of good path points along the best path. The path credit table is constructed the same way as in the case for a single agent. After the path is found, the orientations of the segments are determined by first setting them to the ones perpendicular to the moving direction, and then smoothing them to avoid large change between consecutive samples.

2.3. Choosing the Subgoal

The choice of a subgoal depends upon the specific mission in each problem instance. In the bounding overwatch case, the following criteria are considered:

1. Reachability: it must be reachable at the end of the subplan interval;
2. Configuration: its location should be ahead of the current observer by an adequate distance and within a corridor predetermined for the entire movement so that the bounding overwatch pattern can be maintained;
3. Path quality: it should be reachable by a path that is good in terms of safety and intervisibility;
4. Future safety: it should be safe for the next subplan interval;
5. Observability: it should have good observation points in its vicinity for monitoring the movement of the adversaries in the next stage.

The reachability and configuration criteria are satisfied by considering only cells with path credit greater than zero and ahead of the observers by about half of the maximum distance the agents can travel in a subplan interval. Path quality is evaluated by the path credit table. The future safety and observability criteria require computation of visibility from these candidate cells to the predicted trajectories of the

adversaries in the next stage. As the model of the adversary movements degrades with time, the possible trajectories of the adversaries usually span a fairly large region. Since the visibility computation is expensive and these candidate cells usually cluster in regions, a region-to-region visibility analysis algorithm was developed[7] to achieve much faster computation than applying the point-to-region visibility analysis to each candidate cell. This algorithm is briefly described in the next section. After these computations, the subgoal is chosen from the candidate cells by combining all the criteria using a weighted sum and/or thresholds.

3. ALGORITHMS FOR VISIBILITY ANALYSIS

One of the fundamental computations in our planning system is the visibility analysis, which is also the most expensive computation in the system. Due to its importance in computer graphics, navigation, engineering applications, and geographical information systems, various algorithms have been developed for computing visibility from a given viewing point[8–12]; however, previous visibility analysis from a region or a set of points has been done only by applying to every point an algorithm for a single point.[10,13] In this section, we describe the algorithms we used for the three visibility analyses in our planning system: the line-visibility analysis, the point-to-region visibility analysis (PRVA), and the region-to-region visibility analysis (RRVA). All algorithms are based on digital terrain models and are designed for data parallel hypercube machines. Hypercube machines provide 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. These advantages are used extensively in these algorithms; this kind of parallel prefix operations will be referred to as *scan* operations.

3.1. Visibility Along a Line

Visibility between two points is defined by drawing a line between the two points. The two points are visible to each other if and only if the line lies completely above the terrain. The basic parallel algorithm for visibility analysis is to compute the visibility w.r.t. 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 the maximal angle among the cells before it, then this cell is visible. The steps are illustrated in Fig. 2. The total complexity is dominated by the computation of the maximal angle before each cell, which can be done by a scan operation in $O(\log W)$ time on a hypercube machine using W processors, where W is the number of grid cells along the line.

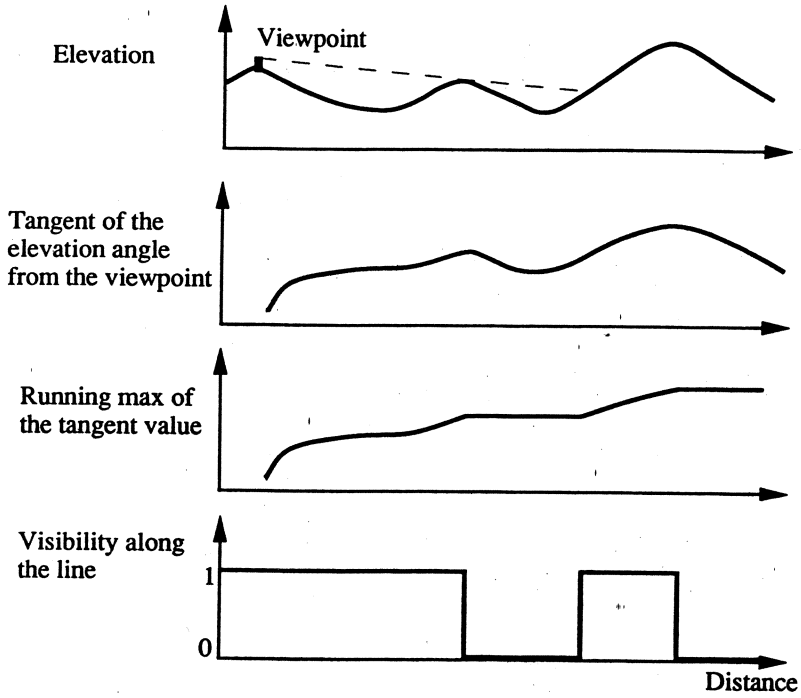


Fig. 2. Computing visibility along a line. The X-axis represents the horizontal distance from the viewpoint. The effect of the middle hill is shown by the dashed line in the elevation plot.

3.2. Point-to-Region Visibility Analysis

The PRVA algorithm computes the visibility from a viewing point to a region. The algorithm we use is based on that in Ref. 8, which was further developed in Ref. 1. 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* by indicating the visibility of each grid cell with 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 coordinates of the region and use the circumscribing upright rectangle.

We use the term *ray* to refer to a line segment with a direction on the plane projection of a terrain geographically, while it refers to a list of PEs computationally. The processor structure respresenting a set of rays is called a *ray structure*. 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 done by constructing a ray structure for each far side of the rectangle and running the line-visibility algorithm for each ray in the ray structures.

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 Fig. 3. Each row in the ray structure corresponds to a ray. After broadcasting the coordinates of the viewing point and the end points of the far side, each processor can find its corresponding grid cell by the digital differential analyzer (DDA) technique,[14] and thus obtain the elevation data. The line-visibility algorithm is then 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

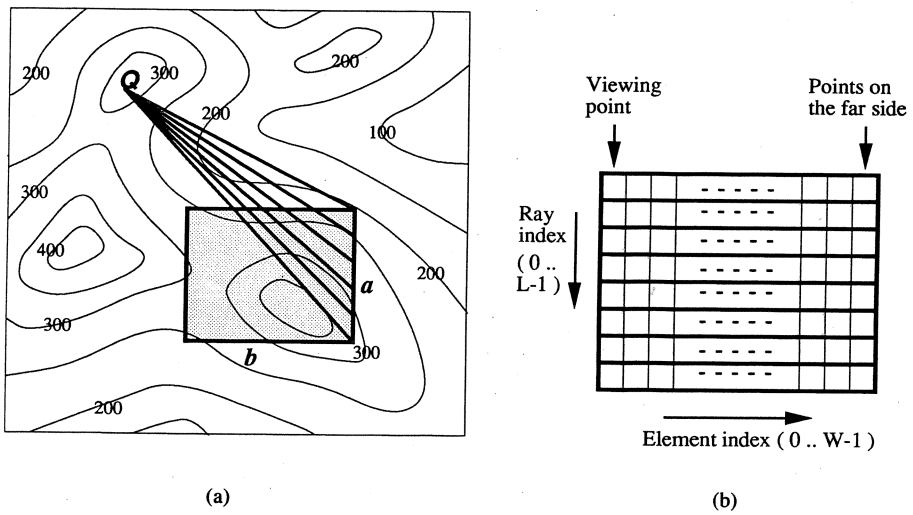
$$O(LW \log W + LW \times Comm) \text{ operations}$$

with minimal time

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

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

As we can see from Fig. 3, 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. An improvement can be made to eliminate the concurrent reads/writes. 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 coherence



(a) (b)

Fig. 3. 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 side a and b are far sides. (b) A 2-D array is allocated as the ray structure.

property, the PEs can be grouped by the terrain cell they are mapped to, and only one from each group will 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.

3.3. Region-to-Region Visibility Analysis

The region-to-region visibility analysis (RRVA) problem is defined as follows: given a source region S and a destination region D on a 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. In the RRVA algorithm, both the source and the destination regions are assumed to be upright rectangles, but they can be in any relationship, e.g. they can be overlapping or even identical. From the previous subsection, 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. Judging from usual problem sizes, 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 stage the analysis in several iterations, each of which computes a subproblem of the entire analysis. Since the communication may be duplicated among iterations, we identify some important coherence properties and explain how they can be used to improve the efficiency of communication.

One important coherence property is depicted in Fig. 4, 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 $(x_s - 1, y_s + 1)$ to the same destination point.[a] The two rays are referred to as the *parent* and the *guardian* respectively, as shown in the figure. Furthermore, the vertical width of the triangle formed by the parent and the guardian is always less than 1 from x_s to x_d. If the visibility analysis is conducted for a column of the source at a time, and *sweeps* from left to right using the same ray structure, then the PEs in this ray should be able to obtain the elevation data of the corresponding terrain cells from one of the two rays for the previous column using only local communications within the ray structure.

This observation suggests the idea of a sweeping algorithm, which sweeps across the source horizontally and/or vertically. Since the above property holds only when the *emitting angle*, β in the figure, is no more than 45 degrees, a partition is defined to divide the rays into four sweeps, namely the East, West, North, and South sweeps, so that the emitting angle angle is always no more than 45 degrees. In each sweep, the analysis is conducted strip by strip, each being a row or a column of cells that is

[a] For a better mapping between the geometry in the figures and the ordering in the data structure, a screen coordinate system is adopted in all figures, i.e. the origin is in the upper left corner and the Y-coordinates increase downward.

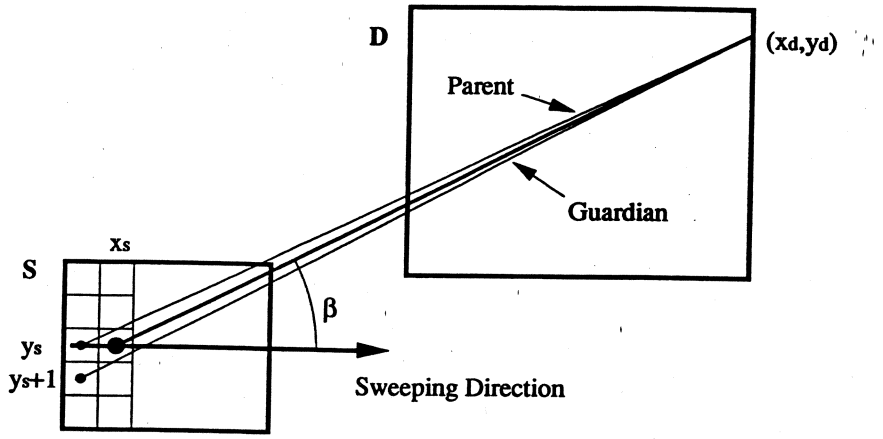


Fig. 4. An observation on the coherence property.

perpendicular to the sweeping direction. The elevation data are obtained by global communication only for the first strip of each sweep and are passed within the ray structure for the subsequent strips. It can be proved that a ray, its parent, and its guardian belong to the same sweep if they exist. In order to ensure the existence of the guardian for all rays, it may be necessary to extend the source region by the side length along the sweeping direction minus one, and to start the sweep with the *extended initial strip*. Figure 5 shows one such situation. Using these properties, the algorithm is outlined as follows:

```
Algorithm PRVA
begin
    for all grid cells in the source region
        set visible-count to 0;
    for each sweep if it is necessary
        construct the 3-D 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 terrains;
    end for each sweep
end Algorithm
```

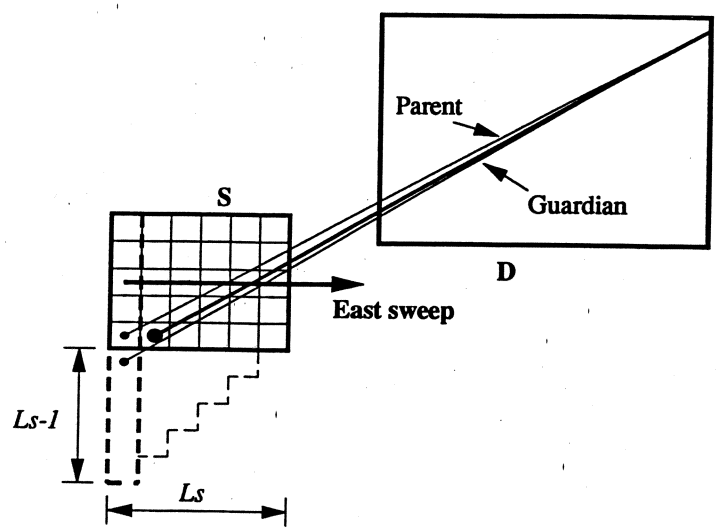


Fig. 5. Extended source: thick dashed lines show the extended initial strip, while the thin dashed lines show how the strip shrinks during the sweep.

Visible-count is the memory location in each terrain cell that returns the number of visible destination cells. The algorithm consists of three major parts:

1. Construction of the ray structure
   The ray structure contains all rays from each source cell on the (possibly extended) initial strip to every grid cell along the far sides of the destination that belongs to the sweep. Since a ray actually contains a list of processors, the entire ray structure is a 3-D array of processors, which is indexed by three indices, as shown in Fig. 6: 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.
   When the value of one index is fixed, the other two indices specify a 2-D slice of the structure. For example, a *v-w slice* refers to the 2-D array for a fixed value of u. It corresponds to all rays starting at the same source point and is actually a part of the 2-D ray structure for the PRVA of that source point. We also use the terms *u-neighbors*, *v-neighbors*, and *w-neighbors* to specify the neighboring elements along each axis.
2. Obtaining the elevation data for the initial strip
   For each PE in the ray structure, its corresponding grid cell on the terrain can be computed in a way similar to that in the PRVA, and the elevation data can then be obtained by global communication. The grouping technique in our PRVA algorithm is applied to alleviate the congestion from concurrent reads. However, it cannot be eliminated completely: in the worst case, the number of concurrent reads is reduced to the range of u.

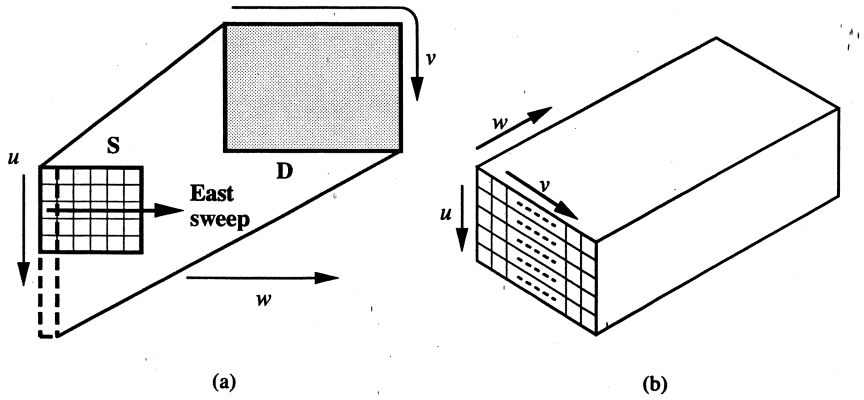


Fig. 6. The ray structure for RRVA: (a) the relevant elements on the terrain map; (b) a 3-D array of processors is allocated as the ray structure.

3. Strip-wise iterations

The visibility analysis is conducted along all rays in parallel in the beginning of the loop. The results are combined into a count of visible destinations cells for each source cell on the strip. Special care should be taken in combining these results to avoid duplicated counting. This count is stored in the first active PE of the first active ray for each source cell. The ray structure is then 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 have emitting angles 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. It is an exclusive write operation since only one PE in the ray structure was designated to keep the count for each source cell and only these PEs will participate in this communication.

The resulting complexity is

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

with a minimal

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

using up to kL_SL_DW processors, where k is a constant determined by the number of far sides and the number of extended sources.

The bottleneck of the computation derives from global communications, and its reduction is achieved 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. 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.

An example of timing results showing the improvements due to the reduction in communication complexity is given in Table 1. This experiment was done on a

Table 1. Reduction on communication radio.

Time in seconds	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

Connection Machine 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 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 gives the results of our RRVA algorithm with virtual processor ratio of 64. It is clear that the ratio between the communication time and the total running time is greatly reduced at a very small price which is shown in the last row as the time of the copying operations. More details and other experimental results can be found in Ref. 7.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section, we present the results from our implementation of the algorithms. All experiments were conducted on a Connection Machine CM-2 using 8K processors. The terrain size is 512×512 grid cells, and the terrain cells are mapped to a two-dimensional array of virtual processors.

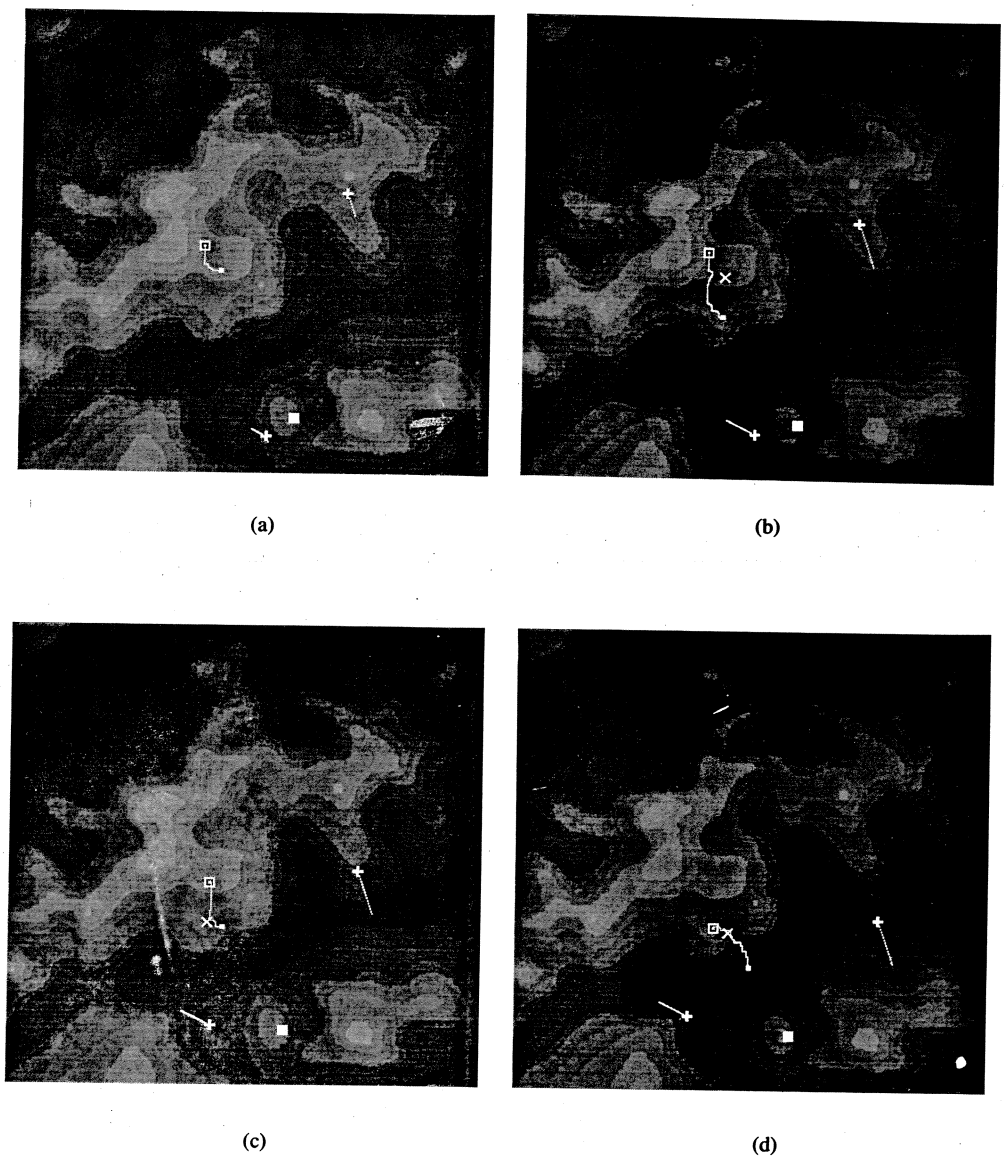
Figure 7 illustrates an example of the planning algorithm, which consists of eight subplans. Each subplan is illustrated by marks on the terrain map indicating the centers of the groups, and the gray level in the background shows the elevation at each pixel: brighter pixels are higher. The observer is indicated by an "X", and the initial location of the group being planned is marked by a square with a dot in its center. The curve connecting the square to a small white dot is the path planned for the center of the group, and the small white dot at the end of the path indicates the subgoal location. The large solid white square near the bottom of each figure shows the location of the final goal. The adversary locations in the beginning of each subplan are indicated by crosses, and the emanating curves illustrate their predicted trajectories. In the figure for the first stage, both groups are located at the same initial position.

After finding the path for the center, the locations of the two agents are determined by the direction of motion and the spacing between agents. Figure 8 enlarges the region around the path of the second stage and shows the configuration of the group as line segments. The agents are located at the end points of each line segment. The safety and intervisibility of each agent is indicated by the brightness and the shape of the marks at the agent locations. A white dot means a path point at which the agent is safe and visible to its partner. A black dot means a path point at which the agent is neither safe nor visible to its partner. A white $\perp$ indicates an agent location that is safe but not visible to its partner, while a black $\perp$ indicates an agent location that is not safe but visible to its partner.

The key computation in the selection of the subgoal is the region-to-region visibility analysis. Figure 9 shows the regions considered in the region-to-region visibility analysis in the third stage. The dark area around the moving agent is the region that can be reached within this subplan interval. A set of candidate cells for the subgoal are chosen from this region using the subgoal criteria except the future visbility. These cells form the source region of the analysis, as indicated by the small set of white points to the right of the observer (the "X" mark). The destination region is the region spanned by the possible adversary trajectories of the next stage: a circumscribing upright rectangle is used. A subgoal is then chosen using the visibility information. The visibility from the vicinity of the subgoal to the destination region is shown in Fig. 10 by the white area, in which the subgoal is marked by a black square. The experimental result shows that the planning system effectively returns a fairly good path.

5. CONCLUSION

In this paper, we have proposed methods for solving visibility-based terrain path planning problem for groups of vehicles using data parallel machines. Our discussion

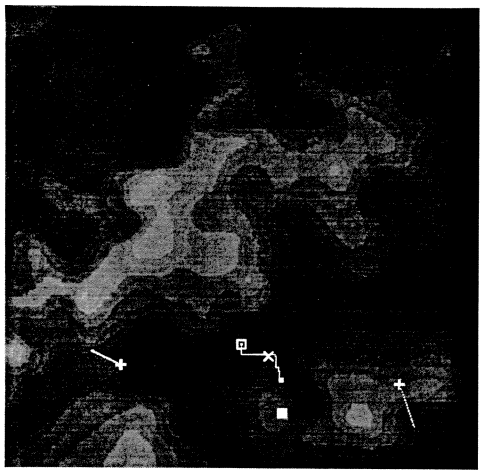


(a) (b)

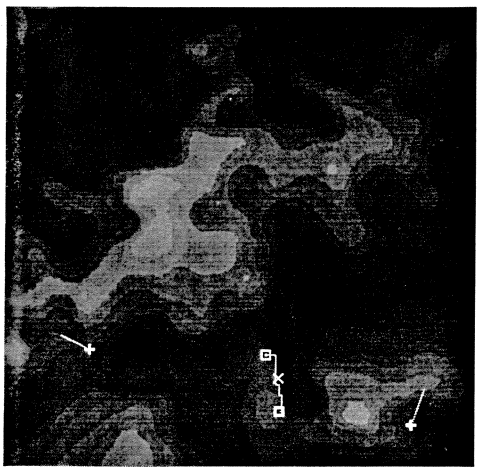
(c) (d)

Fig. 7. The path of each stage.

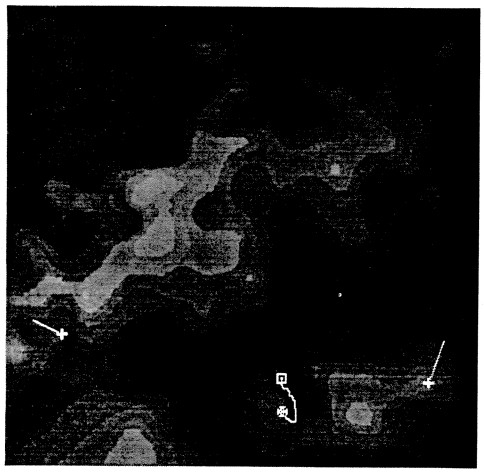
(e)



(f)



(g)



(h)

Fig. 7. Cont'd.

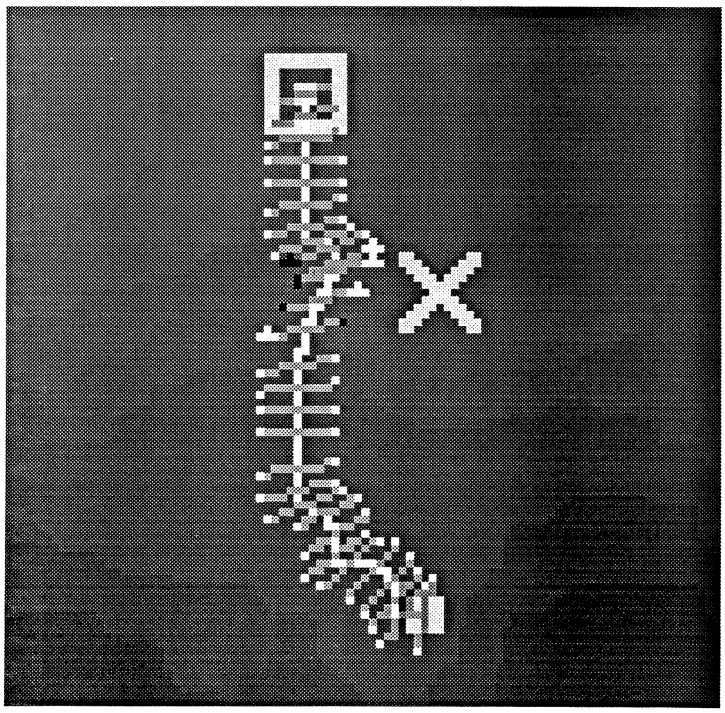
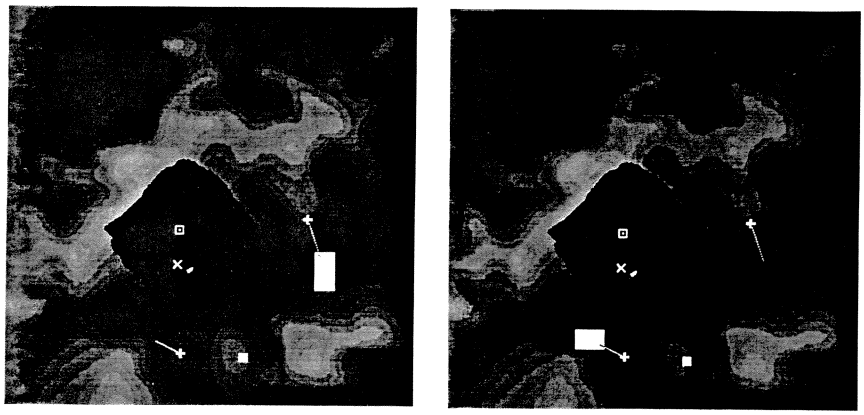
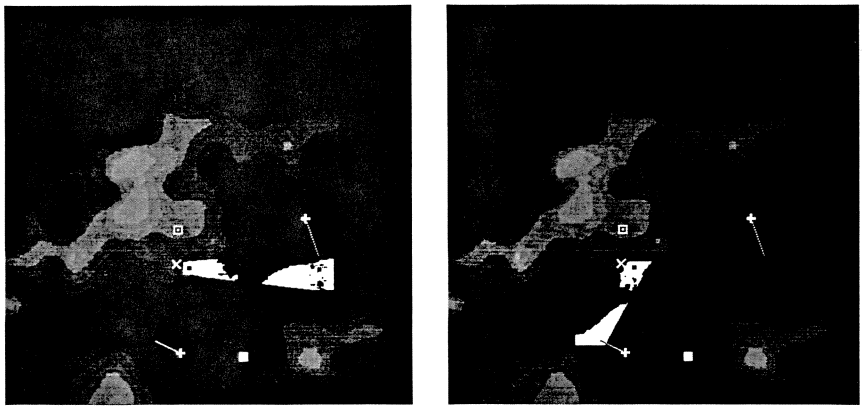


Fig. 8. The configuration of the group and the actual location of each agent along the path of the second stage.



(a) (b)

Fig. 9. The source and destination regions in the region-to-region visibility analysis of the third stage.



(a) (b)

Fig. 10. The visibility from the vicinity of the subgoal to the destination regions in the third stage.

focused on an instance of path planning problems in which two groups of vehicles move in a bounding overwatch manner. Since this kind of problem is known to be hard, our algorithms use approximations based on both temporal and spatial sampling. Our method is an extension of the stealth terrain navigation approach used in Ref. 1. However, the problems considered in Ref. 1 require only point-to-region visibility analysis, while the bounding overwatch movement requires region-to-region visibility analysis. A fast parallel algorithm has been developed to conduct this analysis with reduced running time. The effectiveness of our planning system has been shown by experiments.

ACKNOWLEDGEMENTS

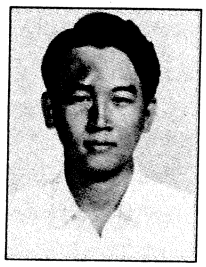
The support of the Defense Advanced Research Projects Agency (DARPA Order No. 6350) and the U.S. Army Engineer Topographic Laboratories under Contract DACA76-88-C-0008 is gratefully acknowledged.

REFERENCES

1. Y. A. Teng, D. DeMenthon and L. S. Davis, "Stealth terrain navigation", Technical Report CAR-TR-532, Center for Automation Research, University of Maryland, 1991; also to appear in *IEEE Trans. Syst. Man Cybern.* **22**, 6 (1992).
2. J. Canny, *The Complexity of Robot Motion Planning*, MIT Press, 1988.
3. J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles", in *Proc. 26th Symp. on Foundations of Computer Science*, 1985, pp. 144–154.
4. J. S. B. Mitchell, "An algorithmic approach to some problems in terrain navigation", *Artif. Intell.* **37** (1988) 171–201.

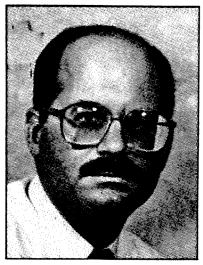
5. M. B. Metea and J. J.-P. Tsai, "Route planning for intelligent autonomous land vehicles using hierarchical terrain representation", in *Proc. Int. Conf. on Robotics and Automation*, 1987, pp. 1946–1952.
6. W. Daniel Hillis, *The Connection Machine*, MIT Press, 1985.
7. Y. A. Teng, D. DeMenthon and L. S. Davis, "Region-to-region visibility analysis using massively parallel hypercube machines", Technical Report CAR-TR-578, Center for Automation Research, University of Maryland, 1991. Also appears in *Proc. 1991 Workshop on Computer Architecture for Machine Perception*, eds. B. Zavidovique and P. L. Wendel.
8. G. E. Blelloch and J. J. Little, "Parallel solution to geometric problems on the scan model of computation", in *Proc. 1988 Int. Conf. on Parallel Processing*, 1988, pp. 218–222.
9. R. Cole and M. Sharir, "Visibility problems for polyhedral terrains", *Symbolic Comput.* **7** (1989) 11–30.
10. L. De Floriani, B. Falcidieno, C. Pienovi, D. Allen and G. Nagy, "A visibility-based model for terrain features", in *Proc. Int. Symp. on Spatial Data Handling*, 1986, pp. 235–250.
11. D. M. Jung, "Comparisons of algorithms for terrain visbility", Master's thesis, Rensselaer Polytechnic Institute, Troy, New York, 1989.
12. J. Reif and S. Sen, "An efficient output-sensitive hidden-surface removal algorithm and its parallelization", in *Proc. 4th ACM Symp. on Computational Geometry*, 1988, pp. 193–200.
13. K. Mills, "A pilot study of intervisibility analysis on the connection machine", *Parallel Comput. News* **4**, 3 (1991) 4–7, Northeast Parallel Architecture Center at Syracuse University.
14. D. Hearn and M. P. Baker, *Computer Graphics*, Prentice-Hall, 1986.

Received 15 June 1991; revised 20 December 1991.



Y. Ansel Teng received the B.S. degree in electrical engineering from National Taiwan University, Taipei, in 1986. He is currently a doctoral candidate in the Department of Computer Science at the University of Maryland and was awarded the Departmental Graduate Fellow in 1992. Since 1989, he has been a Research Assistant in the Center for Automation Research at UMD. His research interests include terrain navigation, parallel processing, geographical data processing, computational geometry, and visualization. Mr. Teng is a student member of the Association for Computing Machinery.

Daniel DeMenthon received a graduate engineering degree from Ecole Centrale de Lyon, France, the M.S. degree in applied mathematics from Université Claude Bernard, France, and the M.S. degree in engineering from the University of California at Berkeley. He is completing a Ph.D. course in computer science with Université Joseph Fourier, Grenoble, France. For the past seven years he has been a senior research engineer at the Computer Vision Laboratory of the Center for Automation Research at the University of Maryland, College Park. His research interests are in the areas of object recognition, planning, and 3-D user interfaces.



Larry S. Davis is a tenured Professor in the Department of Computer Science and the Director of the University of Maryland Institute for Advanced Computer Studies. He received his Ph.D. in 1976 from the University of Maryland, College Park in computer science. Prof. Davis has published over 50 articles on topics in image processing and computer vision. He is a member of the IEEE and is on the editorial board of both *Computer Vision, Graphics, and Image Processing* and the *International Journal of Computer Vision*. He is Program Co-chair of the upcoming workshop "Computer Architectures for Machine Perception (CAMP) '93".