Two Algorithms for Determining Partial Visibility and Reducing Data Structure Induced Error in Viewshed Analysis

Abstract

In geographic information systems, viewshed analysis is used to calculate the areas on a topographic map that can be seen from a specified set of locations. Grid cell ("raster") based viewshed routines typically create binary map results, in which each cell is listed as being entirely visible or entirely not visible. Prior research has shown that sampling errors in elevation data can reduce the accuracy of the resulting visibility maps. This study demonstrates that further errors may be introduced in binary viewshed maps by the geometry of the raster data structure itself. Two techniques are illustrated for reducing data-structure induced error in viewshed analysis.

Introduction

Within a geographic information system (GIS), viewshed analysis makes use of elevation data to determine which areas on a map can be seen from one or more locations. Such analysis has many useful geographical applications. For example, viewshed analysis can be used to determine the visibility of proposed architectural projects and highway construction (Fisher, 1990), to find the best locations for fire lookout towers in a rugged landscape (Travis *et al.*, 1975), to plan the least visible routes for military troop movements, and to locate radar systems for maximum visibility of the surrounding terrain (Goodchild and Lee, 1989).

Computer algorithms for viewshed determination require a digital map of elevation data, generally referred to as a digital terrain model. Typically, elevation data are presented in one of three formats: contours, raster grids, or triangulated irregular networks (Burrough, 1986; Peucker and Chrisman, 1975; Goodchild and Lee, 1989). This study focuses on raster-based viewshed algorithms. Elevation data stored in raster format is commonly referred to as a digital elevation model (DEM). Within a raster DEM (referred to simply as a DEM from here on), each cell is assigned an elevation value that corresponds to the average elevation of land within the cell (USGS, 1990). The two-dimensional matrix structure of a DEM can be easily implemented and manipulated in computer programs, and it is associated with a well-understood implicit cell ordering scheme (Peucker and Chrisman, 1975). As such, DEMs are well-suited to numerical analyses necessary for calculating contours, slope angles and aspects, hill shading, basin delineation, and viewshed information.

An algorithm for generating a viewshed from raster elevation data can be based on the calculation of slopes between different cells in the DEM. The determination as to whether one cell (the destination) can be seen from another (the source) can be accomplished by examining each of the intermediate cells that lie between the two (the "line of sight"). If the slope between the source and any intermediate cell in the line of sight is greater than the slope between the source and the destination cell, then the visibility between source and destination is at least partially blocked (Figure 1).

Although specific algorithms used by various raster-based viewshed routines are often more complicated (Felleman and Griffin, 1990), they rely on the same principles. All result in binary viewshed maps, in which each cell is marked as either visible or not visible. Figure 2 shows the results of a standard binary viewshed calculation based on sample elevation data. Considering the error present in most digital elevation data, however, the accuracy implied by a binary viewshed analysis may not be justified (Felleman and Griffin, 1990; Fisher, 1991). To account for this problem, Felleman and Griffin (1990) and Fisher (1992) propose a method of calculating probabilistic visibility weights by generating and combining multiple viewsheds based on the addition of simulated error to the original elevation data.

Data-Structure Induced Errors

Error in the elevation data, however, is not the only potential source of inaccuracies for binary viewshed maps calculated from raster elevation data. Inaccuracies can also result from limitations in the algorithms that analyze visibility between cells in a raster grid. Such inaccuracies are referred to as data-structure induced errors–they are caused by a failure to properly account for all cells in the data structure that can effect visibility between the source and destination.

The basic problem is as follows. The visibility between the source and destination can be determined by checking all of the intermediate cells lying between the two. These intermediate cells are tested to see if they block the view from source to destination. In the case where the source and destination are located in the same row or column of the grid, a single intermediate cell can block the view entirely (Figure 3a). In the case where the source and destination lie diagonally to one another, however, a single cell may only block part of the view between the two, not entirely obscuring the line of sight (Figure 3b). In a standard binary viewshed, however, a destination cell with partially blocked visibility is de-

Photogrammetric Engineering & Remote Sensing, Vol. 59, No. 7, July 1993, pp. 1149–1160.

0099-1112/93/5901–1149$03.00/0

Paul A. Sorensen
David P. Lanter
Department of Geography/NCGIA, University of California, Santa Barbara, CA 93117

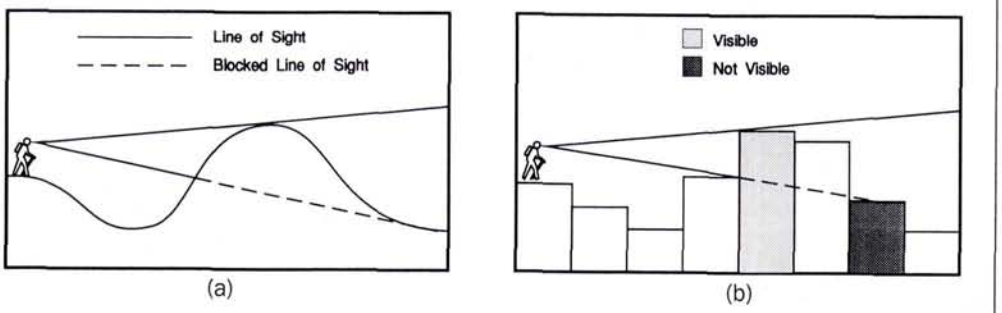

Figure 1. (a) Cross cut view of line of sight blocked by a hill. (b) Cross cut view of line of sight principles generalized within a non-interpolated raster structure.

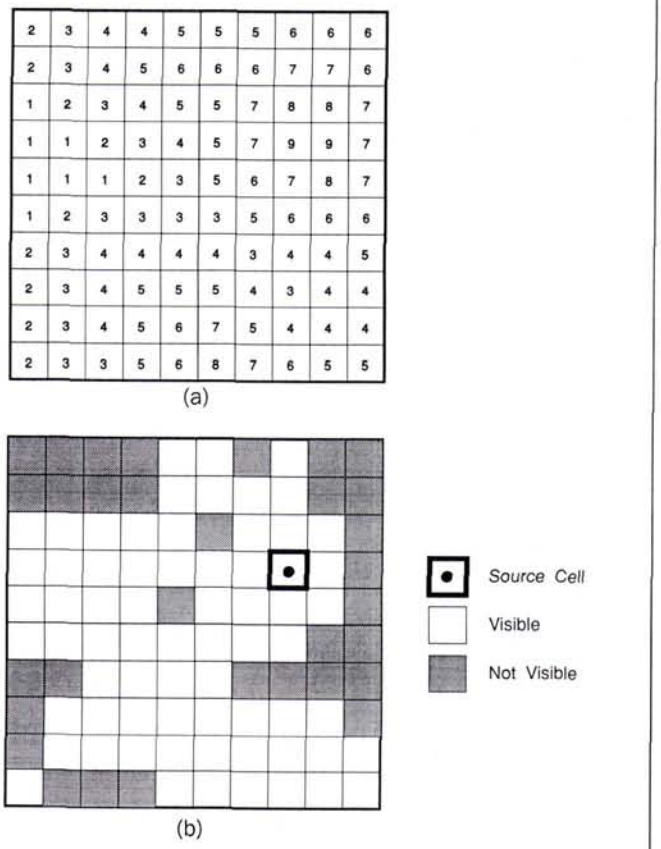

Figure 2. (a) Sample elevation data used for viewshed calculations. (b) A standard binary viewshed.

clared either entirely visible or entirely blocked. The degree to which the cell is not entirely visible or blocked constitutes data-structure induced error.

To reduce or eliminate data-structure induced viewshed error, two methods for handling partial visibility are explored. The first method, "vector analysis," approximates partial visibility measures for each cell. The second method, "sub-cell binary analysis," subdivides each cell into visible and non-visible areas.

The Vector Analysis Method

The vector analysis method for determining visibility between a source and a destination cell begins by identifying all intermediate cells that could potentially block the line of sight between the two. This group includes all cells that fall at least partially within the visible corridor between the source and destination. These cells are then organized into one, two, or three vectors: one vector if the source and the destination are located on the same row or column in the grid (Figure 4a); two or three vectors if the source and destination lie diagonally from one another (Figures 4b and 4c). Each vector is then checked for cells that block the view between the source and the destination. Overall visibility for the destination cell is approximated as

$$\text{Visibility} = \#\ \text{Unblocked Vectors} / \#\ \text{Vectors}.$$

For example, if there are two vectors and one is blocked, the destination is marked as 1/2 visible (Figure 5a). If there are three vectors and two are blocked, the destination cell is marked as 1/3 visible (Figure 5b). The algorithm is summarized by the following steps:

(1) Identify the set of cells between the destination and source that need to be checked for visibility blockage.
(2) Divide the set of cells into one, two, or three vectors as appropriate.
(3) Analyze visibility along each vector. Any vector with one or more cells that block the view is considered blocked. All other vectors are considered visible.
(4) Calculate the visibility proportion as the number of visible vectors divided by the total number of vectors–assign this visibility estimate to the destination cell.

Determining the Intermediate Cells that Must Be Checked

Two steps are required to identify the set of cells between the source and destination that must be checked.

(1) Determine the two parallel bounding lines that delimit the visible corridor between the source and destination.
(2) Find all cells that intersect the boundary lines or lie entirely within the boundary lines. This is the set of cells (the "intermediates") that must be checked for visibility blockage.

In order to illustrate the implementation of these two steps, the following notation must be introduced. Let the coordinates of the center point and the corner points of a generic cell C be defined as shown in Figure 6. Let S be the source cell at which the viewer is located. Let D be the desti-

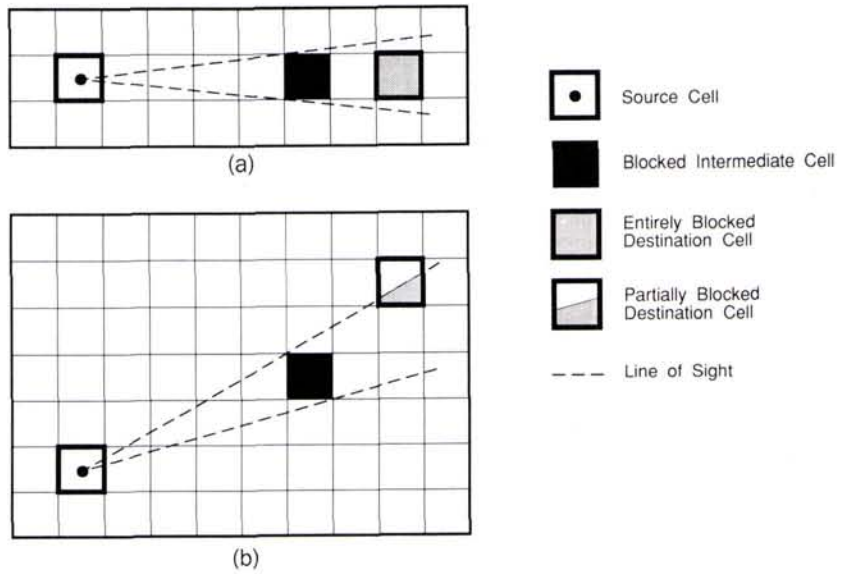

Figure 3. Examples of an intermediate cell blocking the view between the source and destination (a) entirely and (b) partially.

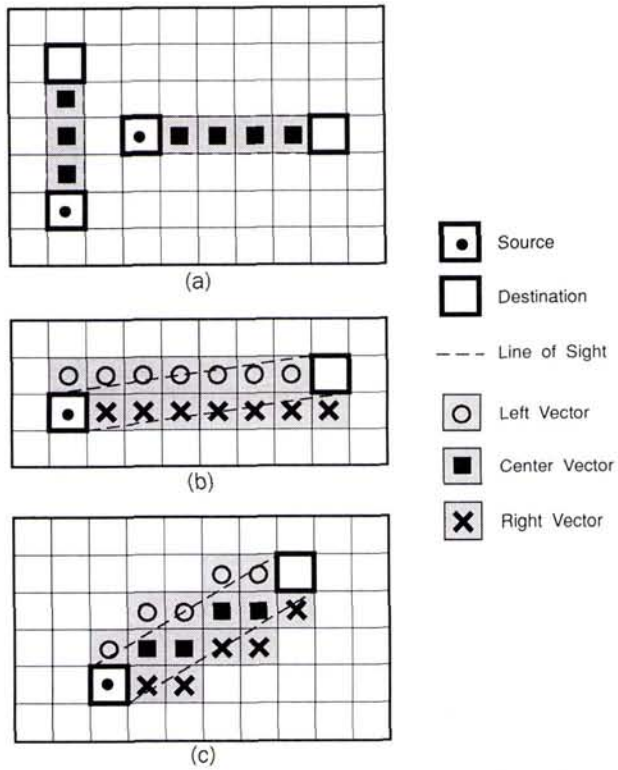

Figure 4. When the source and destination are located horizontally or vertically from each other, there will be (a) only one vector of cells between them. When the source and destination are located diagonally, the area of cells between them may be divided into (b) two vectors or (c) three vectors.

nation cell for which visibility is being determined. Let L1 and L2 be lines that bound the visible line of sight between S and D. L1 is defined by the points $l_{1,1}$ and $l_{1,2}$ and L2 is defined by the points $l_{2,1}$ and $l_{2,2}$.

To determine the endpoints of L1 and L2, we must check the position of D relative to S. There are four possible orientations: vertical, horizontal, positive slope, and negative slope. Figure 7 illustrates examples of each of these orientations.

As Figure 7 shows, if sx (the x coordinate of the center of S) = dx (the x coordinate of the center of D), S and D are oriented vertically (they lie in the same column). L1 is assigned the endpoints s2 and d1, and L2 is assigned the endpoints s3 and d4. If sy = dy, S and D are oriented horizontally (they lie in the same row). L1 is then defined by s3 and d2, and L2 is defined by s4 and d1. If (sx − dx)(sy − dy) > 0, S and D are oriented with a positive slope. L1 is defined by s2 and d2, and L2 is defined by s4 and d4. If (sx − dx)(sy − dy) < 0, S and D are oriented with a negative slope. L1 is assigned the endpoints s1 and d1, and L2 is assigned the endpoints s3 and d3. Note that in each case shown in Figure 7, D and S could be interchanged and the same tests and line endpoint assignments would hold.

Once the endpoints of L1 and L2 have been determined, the intermediate cells can be identified as follows. In the case of vertical or horizontal orientation, the task is simple: all cells in the same column or row as S and D that fall between the two must be included. In the case of positive or negative slope orientation, the algorithm is more complex. To determine the intermediates, the area of cells between S and D are examined on a row by row basis (all rows between and including y = sy and y = dy). For each row, two cell addresses are determined: the left-most cell that is intersected by L1 and the right-most cell that is intersected by L2 (Figure 8). The following pseudo-code demonstrates how this may be accomplished when S and D are oriented with a positive slope. This code relies on a simple line intersection routine (for example, see Goodchild and Kemp, 1990)): that is,

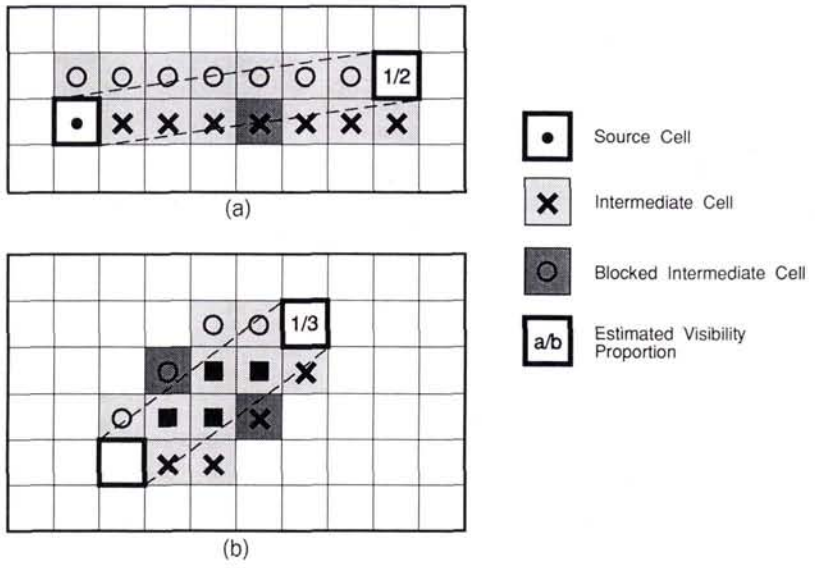

Figure 5. Determining the visibility proportion estimate using the vector method.

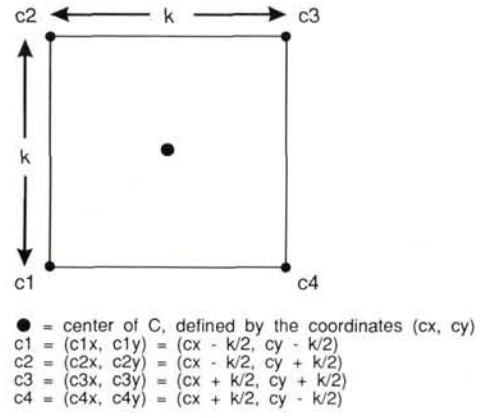

Figure 6.

```
if ((sx < dx) and (sy < dy))
   for (row = sy; row ≤ dy; row++)
      /* identify the left most x on this row */
      if (row = sy)
         left_x[row] = sx;
      else
         (xi, yi) = intersection (L1, y = row − k/2);
         left_x[row] = xi;
      /* identify the right most x on this row */
      if (row = dy)
         right_x[row] = dx;
      else
         (xi, yi) = intersection (L2, y = row + k/2);
         right_x[row] = xi;
```

Once the left- and right-most cells on each row have been identified, identifying the remaining cells is straightforward: on each row between sy and dy, all cells between the left- and right-most are included. Some of these cells may intersect L1 or L2 as well (but simply not be the left-most or right-most to do so); others may lie entirely between L1 and L2. The following pseudo-code demonstrates this step:

```
for (i = sy; i ≤ dy; i++)
    for (j = left_x[i]; j ≤ right_x[i]; j++)
        add (i, j) to the "intermediates";
```

Dividing the Group of Intermediate Cells into Vectors

Before discussing the method by which the set of intermediate cells is partitioned into vectors, the relationship between S and D must be further specified. First, assume that S is located at cell (i, j), where i represents the horizontal and j the vertical position within the raster grid. Assume that m and n are integers greater than 1. The terms horizontal, vertical, near horizontal, near vertical, and diagonal are defined as follows:

	Source Location	Destination Location
Vertical	(i, j)	$(i, j \pm n)$
Horizontal	(i, j)	$(i \pm m, j)$
Near Vertical	(i, j)	$(i \pm 1, j \pm n)$
Near Horizontal	(i, j)	$(i \pm m, j \pm 1)$
Diagonal	(i, j)	$(i \pm m, j \pm n)$

The steps required to divide the intermediate cells into vectors can be summarized as follows:

(1) If **S** and **D** are located vertically or horizontally from one another, construct a single vector of cells between the two (refer to Figure 4a).
(2) If **S** and **D** are located near vertically or near horizontally from one another, divide the intermediates into two side-by-side vectors $\mathbf{I}_{left}$ and $\mathbf{I}_{right}$ (refer to Figure 4b).
(3) If **S** and **D** are located diagonally from one another, divide the intermediates into three vectors (refer to Figure 4c) as follows:
   (a) Create the $\mathbf{I}_{left}$ and $\mathbf{I}_{right}$ vectors as explained below.
   (b) Assign the remaining intermediates to $\mathbf{I}_{center}$.

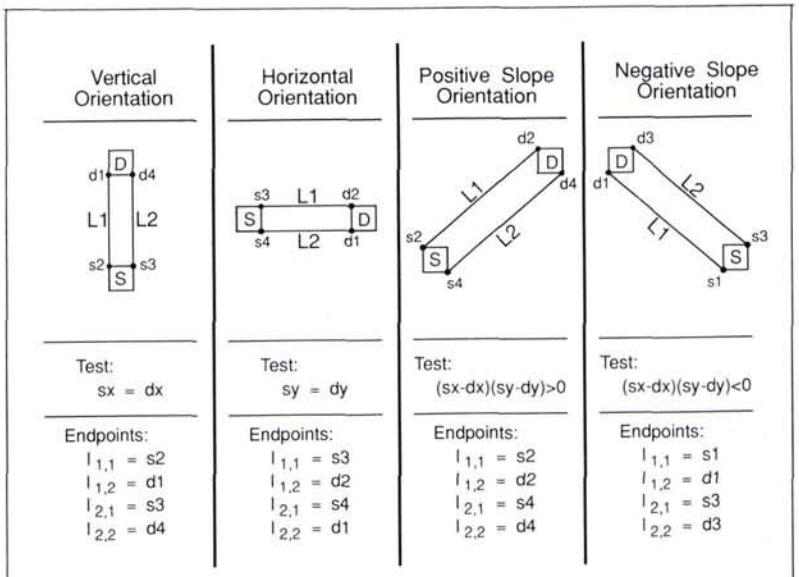

Figure 7. Construction of the boundaries of the visible corridor between S and D for different orientations of the two cells.

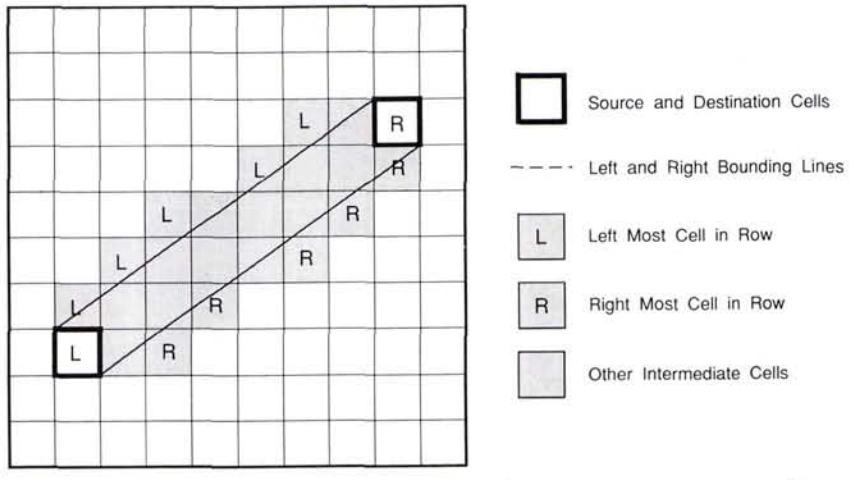

Figure 8. Identification of the left and right most cells for each row between S and D.

The process of assigning a single vector in the horizontal or vertical cases or two straight side-by-side vectors in the near horizontal or near vertical cases is straightforward. If the source and destination are in the same row or column, all cells between the two form a single vector. If the source and destination are separated in a near horizontal or near vertical orientation, the area of cells between them will form a rectangle two cells in width. This rectangle can be divided in half lengthwise to form two straight vectors.

Dividing the intermediates into three vectors for the diagonal case is more complicated. The first step is to create the $\mathbf{I}_{left}$ and $\mathbf{I}_{right}$ vectors. These vectors will always contain the left-most and right-most cell, respectively, in each row in the intermediates. They will also contain additional cells along the outside edges of the intermediates that serve to connect the left-most cells in different rows and right-most cells in different rows, into the $\mathbf{I}_{left}$ and $\mathbf{I}_{right}$ vectors, respectively.

The algorithm for determining the $\mathbf{I}_{left}$ and $\mathbf{I}_{right}$ vectors traverses the left side and the right side of the intermediates, selecting cells in a manner that results in two "continuous" and "direct" vectors. The term "continuous" signifies that any two cells that are adjacent within the vector must be adjacent (either horizontally, diagonally, or vertically) within the raster. For example, if one cell in the vector is located at (i, j), the next cell may be located at $(i, j+1)$, $(i+1, j+1)$, or $(i+1, j)$. Locating the next cell at $(i+1, j+2)$, on the other hand, would constitute a jump that would break the continuity of the vector. The term "direct" signifies that, whenever possible, the vector will move diagonally. For example,

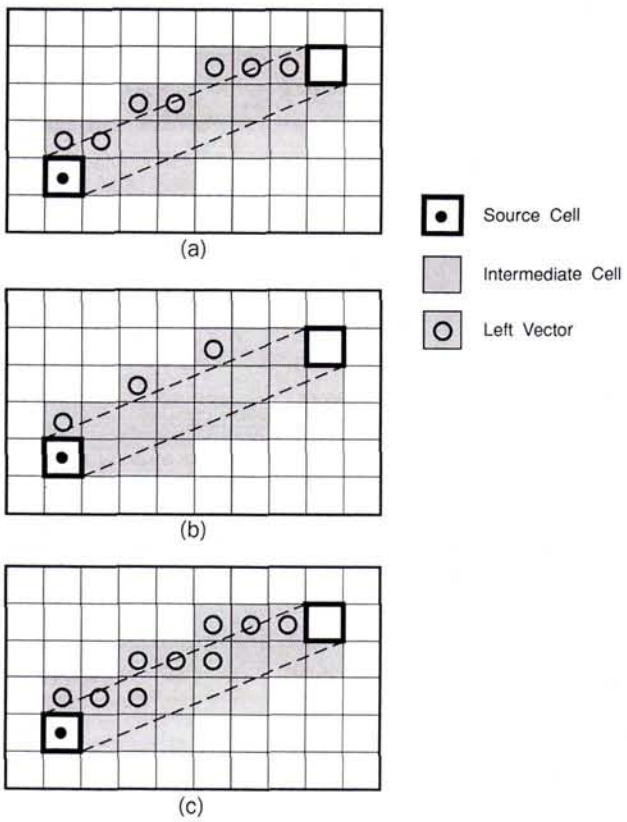

Figure 9. Examples of (a) a continuous and direct vector, (b) a non-continuous vector, and (c) a non-direct vector.

rather than the series of vector cells (i, j), $(i+1, j)$, $(i+1, j+1)$, the vector would proceed directly from (i, j) to $(i+1, j+1)$. Figure 9a shows an example of a continuous and direct vector; Figure 9b shows a discontinuous vector; Figure 9c shows a non-direct vector.

After the two outer vectors have been determined, the center vector is created using a "fill-in-the-blanks" method that includes all the intermediate cells not already in $\mathbf{I}_{left}$ or $\mathbf{I}_{right}$. In other words, $\mathbf{I}_{center}$ = intermediates − ($\mathbf{I}_{left}$ + $\mathbf{I}_{right}$). Based on this selection criteria, the center vector, unlike the outer two vectors, is not necessarily direct (Figure 10a) and not necessarily continuous (Figure 10b). Note that the fill-in-the-blanks criteria is used because it would not serve to assign a cell which is already a member of the outer vectors to the center vector. If such a cell were doubly assigned, it would have the potential to block two vectors.

Using the Vectors to Determine a Visibility Estimate

After the intermediates have been divided into vectors, the cells along each vector are checked for visibility blockage. Any vector that contains at least one blocking cell is considered blocked; vectors with no blocking cells are clear. (To test whether a cell blocks the view, the slope between the source and the intermediate cell is calculated. If this slope is greater than the slope between the source and the destination, the intermediate cell blocks the view.) The visibility proportion of the destination cell is then estimated as the number of unblocked vectors divided by the total number of vectors. With this scheme, if there is only one vector, the visibility estimate can be 0 or 1; with two vectors it can be 0, 1/2, or 1; with three vectors it can be 0, 1/3, 2/3, or 1. Figure 11 shows the results of a viewshed calculated using the vector method.

In the case of three vectors, this approach is admittedly simplistic in that it assumes that the right and left vectors should have the same weight, or same visibility blockage potential, as the center vector. In fact, cells in the center vector actually tend to block much more than 1/3 of the visible corridor between the source and the destination. From empirical observations, cells in the center always block at least 1/2 the view and sometimes may block the entire view. For this rea-

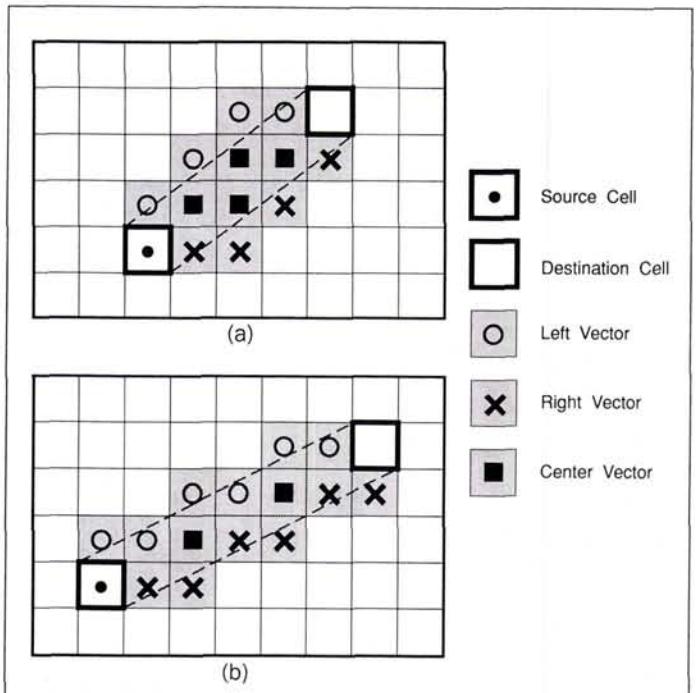

Figure 10. Examples of a center vector that is (a) not direct and (b) not continuous.

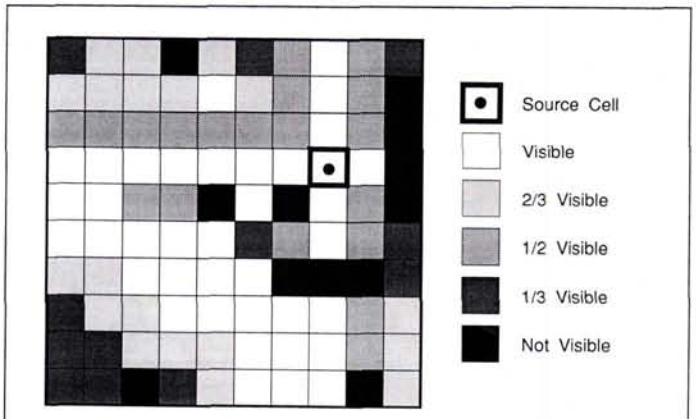

Figure 11. Viewshed calculated with the vector analysis method using elevation data from Figure 2a.

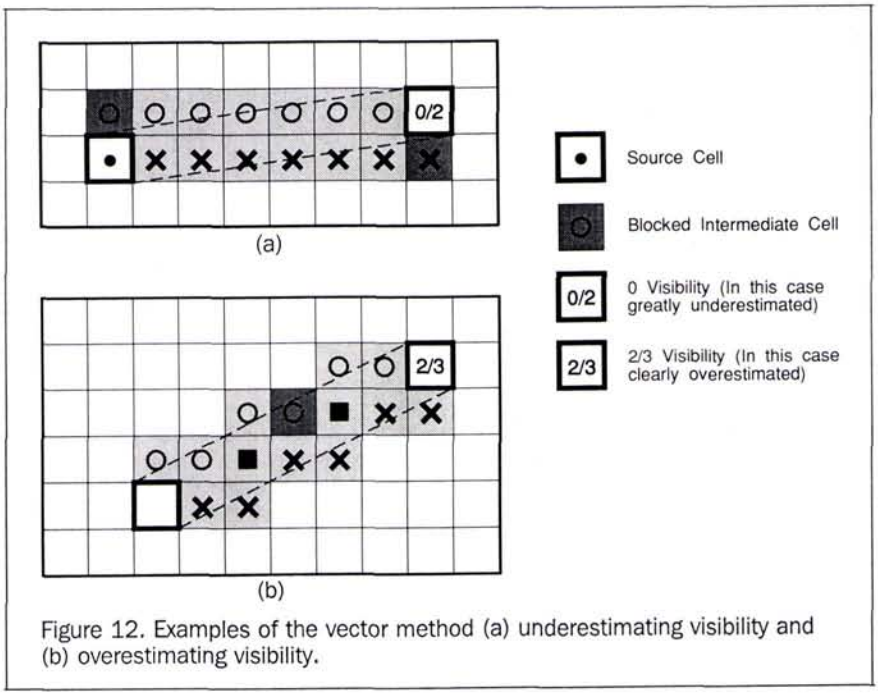

Figure 12. Examples of the vector method (a) underestimating visibility and (b) overestimating visibility.

son, a more reasonable weighting could be designed which gives the center vector greater blocking potential than the outer two vectors.

Drawbacks of the Vector Method

Although the vector method provides a more accurate portrayal of viewshed information than a simple binary analysis, it is still not very precise–in certain cases it may seriously overestimate or underestimate the actual visible proportion of a destination cell. The basic problem is that one cell in a vector may block the line of sight between the source and the destination cell more than another cell in the same vector. The vector analysis technique makes no allowance for such differences. Each cell in a vector is treated equally, regardless of the degree to which it actually blocks the view. Figures 12a and 12b show two problematic cases: in Figure 12a the visibility percent is clearly underestimated; in Figure 12b it is significantly overestimated. This problem may persist even when an adjusted vector weighting scheme is used.

The Sub-Cell Binary Analysis

While the vector method offers an approximate visibility proportion estimate, the sub-cell binary analysis calculates a more geometrically precise measure of viewshed blockage. To do this, the sub-cell binary technique examines each intermediate cell separately to determine its individual impact on the visibility between the source cell **S** and the destination cell **D**. The results of the individual analyses are integrated to find the overall visibility measure for **D**. In the results, each partially blocked destination cell is sub-divided into visible and non-visible zones. Hence, the term sub-cell binary. The steps in this algorithm can be summarized as follows:

(1) Identify all potentially blocking intermediate cells using the same strategy as in the vector analysis.

(2) Check the slope between the source cell **S** and each intermediate cell to determine which cells contribute to blocking the view.

(3) For each blocking cell, determine and define the area within **D** that will be obscured from view. Store this area in the form of a polygon.

(4) Perform a logical union (i.e., Boolean "OR") on the areas of the polygons within D obscured by each blocking intermediate cell. This union represents the total area of the cell that cannot be seen by the viewer. The complement of this area (cell area - union(blocked polygons)) equals the area of **D** that can be seen from the observation point.

Finding the Exact Area that Is Blocked from View by a Single Intermediate Cell

In order to find the exact area within **D** that is obscured by a blocking intermediate cell, it is necessary to draw two lines of sight from the viewer's position within the source (for simplicity, the midpoint of the source cell is assumed) past the edges of the blocking cell towards the destination cell. The area of the destination cell that falls between these two lines will be hidden from the viewer's vantage point. The area outside of the lines will remain visible.

Note that unlike the lines delimiting the visible corridor within the vector analysis method, the lines of sight discussed here are not parallel. Rather they are radial lines emanating from a specific viewer location. This modification enables the determination of the exact area within a destination cell that is blocked from view by an intermediate cell from a viewer's perspective.

For each potentially blocking intermediate cell, there are four possible scenarios. First, **D** may lie entirely within the two lines of sight, in which case it is totally blocked from view (Figure 13a). Second, **D** may lie entirely outside the two lines, in which case it remains totally visible (Figure 13b). Third, the viewer's left line of sight (the line that passes to the left of the blocking cell) may intersect **D**, in which case

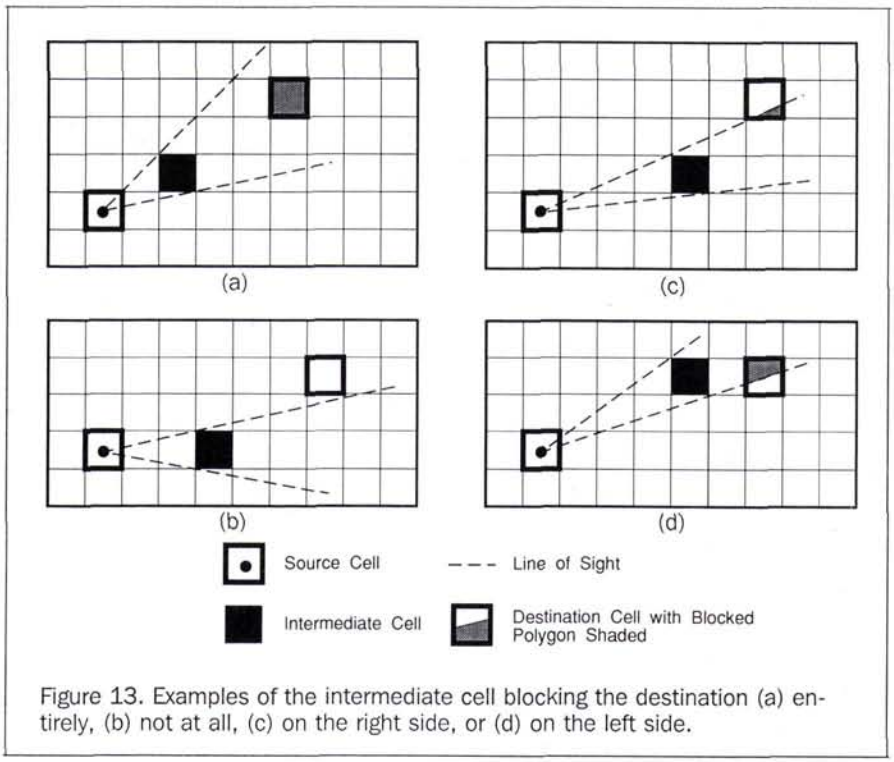

Figure 13. Examples of the intermediate cell blocking the destination (a) entirely, (b) not at all, (c) on the right side, or (d) on the left side.

the portion of **D** to the right of the intersection line (from the viewer's perspective) will be blocked from sight (Figure 13c). Fourth, the viewer's right line of sight may intersect **D**, in which case the portion of **D** to the left of the intersection line will be blocked from sight (Figure 13d).

The basic algorithm to determine the degree to which a single intermediate cell effects the view of the destination cell **D** can be summarized as follows:

(1) Construct the two sightlines
(2) Check whether the left sightline intersects **D**
  (a) If it does intersect **D**, determine the polygon within **D** that is blocked from view–done.
  (b) If it does not intersect **D**, check whether **D** lies to the inside (to the right) or to the outside of the left sightline.
    (i) If **D** lies to the outside, it is unobscured by the intermediate cell – done.
    (ii) If **D** lies to the inside, a potential intersection of **D** with the right sightline must be analyzed – continue.
(3) Check whether the right sightline intersects **D**.
  (a) If it does intersect **D**, determine the polygon within **D** cell that is blocked from view – done.
  (b) If it does not intersect **D**, check whether **D** lies to the inside (to the left) or to the outside (to the right) of the right sightline.
    (i) If **D** lies to the outside, it is unobscured by the intermediate cell – done.
    (ii) If **D** lies to the inside, it is totally obscured by the intermediate cell – done.

As these steps are broken down individually, the following notation is used. As before, let **S** represent the source cell and let **D** represent the destination cell. Additionally, let **B** represent a blocking intermediate cell. Let the coordinates of the center and corners of each of these cells be defined as in Figure 6.

First, the two sightlines are constructed. Both emanate from the viewer's position at (**sx, sy**). The first, **SL**, passes to the left of **B** and is defined by the points **sl1** = (**slx1, sly1**) and **sl2** = (**slx2, sly2**). The second, **SR**, passes to the right of **B** and is defined by the points **sr1** = (**srx1, sry1**) and **sr2** = (**srx2, sry2**). Because both sightlines emanate from the viewer's position at the center of **S** (**sx, sy**), the first points of **SL** and **SR** are defined: **sl1** = **sr1** = (**sx, sy**).

To determine the second points of **SL** and **SR**, it is necessary to identify the outer corners of **B** through which the sightlines will pass. This depends on the relative location of **B** to **S**: to the right, to the upper right, directly above, to the upper left, to the left, to the lower left, directly below, or to the lower right. Figure 14 illustrates the assignments of **sl2** and **sr2** for each of these cases. In each example, the area between **SL** and **SR** behind **B** is shaded. Any portion of **D** falling within this shaded area will be hidden from the viewer. The specific logic for determining the locations of **sl2** and **sr2** is as follows:

```
if (bx > sx) and (by = sy)
    sl2 = b2; sr2 = b1;
else if (bx > sx) and (by > sy)
    sl2 = b2; sr2 = b4;
else if (bx = sx) and (by > sy)
    sl2 = b1; sr2 = b4;
else if (bx < sx) and (by > sy)
    sl2 = b1; sr2 = b3;
else if (bx < sx) and (by = sy)
    sl2 = b4; sr2 = b3;
else if (bx < sx) and (by < sy)
    sl2 = b4; sr2 = b2;
else if (bx = sx) and (by < sy)
    sl2 = b3; sr2 = b2;
else if (bx > sx) and (by < sy)
    sl2 = b3; sr2 = b1;
```

Once **SL** and **SR** have been defined, attention is focused on the position of **D** relative to the area between the two sightlines. To determine what portion of **D** (if any) falls be-

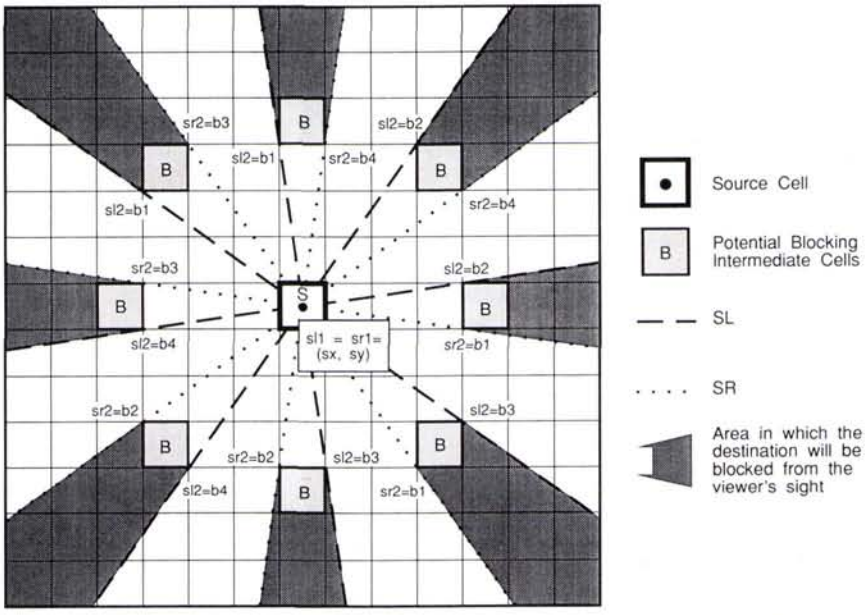

Figure 14. Determining the endpoints of the lines of sight SL and SR.

tween **SL** and **SR**, potential intersections of the two sightlines with **D** are calculated. This is accomplished by testing the intersection of each sightline with each of the four sides of **D**. For example, if **D** lies to the upper right of **S**, it is possible for the sightline **SL** to enter through the left or bottom and to exit through the top or right. The algorithm first determines each of these potential intersections:

```
if ((dx > sx) and (dy > sy))
    left_entry_pt = (left_x, left_y)
      = intersection of SL and the left side of D (defined
        by d1 and d2)
    bottom_entry_pt = (bottom_x, bottom_y)
      = intersection of SL and the bottom of D (defined
        by d1 and d4)
    right_exit_pt = (right_x, right_y)
      = intersection of SL and the right of D (defined by
        d3 and d4)
    top_exit_pt = (top_x, top_y)
      = intersection of SL and the top of D (defined by
        d2 and d3)
```

Next, the algorithm checks whether **SL** enters through the left side of **D**. If it does, the algorithm determines whether **SL** exits through the right or the top side:

```
if (d1y < left_y < d2y)
    entry_pt = left_entry_pt;
    if (d2x < top_x < d3x)
        exit_pt = top_exit_pt;
    else
        exit_pt = right_exit_pt;
```

If **SL** does not enter through the left side of **D**, it may enter through the right:

```
else if (d1x < bottom_x < d4x)
    entry_pt = bottom_entry_pt;
    if (d2x < top_x < d3x)
        exit_pt = top_exit_pt;
    else
        exit_pt = right_exit_pt;
```

If **SL** is found to intersect **D** (entering through the left or bottom and exiting through the top or right), it is necessary to calculate the exact area that will be blocked from view. The following logic demonstrates how the entry point, exit point, and destination cell corner points are used to define the blocked and visible polygons within **D** and the areas of each in the case in which **SL** enters through the left and exits through the top of **D**.

```
if ((entry_x = left_x) and (entry_y = left_y))
    if ((exit_x = top_x) and (exit_y = top_y))
        visible polygon is defined by entry_pt, d2, exit_pt,
          entry_pt;
        blocked polygon is defined by entry_pt, d1, d4,
          d3, exit_pt, entry_pt;
        area of visible polygon = (exit_y − entry_y) *
          (exit_x − entry_x)/2;
        area of blocked polygon = k² − area of visible
          polygon;
```

Figure 15 shows an example of the construction of these polygons.

If **D** is to the upper right of **S** (**sx** < **dx** and **sy** < **dy**) and **SL** does not intersect **D** through the left or the bottom, then it must not intersect **D** at all. If this is the case, the algorithm must determine whether **D** lies inside or outside of **SL**. The position of **D** relative to **SL** can be checked using one of the intersection points determined earlier:

```
if SL does not enter D through the left or through the
bottom
  if (bottom_x < d1x)
     D is inside of SL;
  else
     D is outside of SL;
```

If **D** lies outside of **SL**, it is not obscured by this intermediate cell. If it lies inside, however, the potential intersection with **SR** must be checked. If **SR** intersects **D**, the blocked and visible polygons must be calculated in a manner similar to the algorithm illustrated above. If **SR** does not intersect **D**, a test is performed to determine whether **D** lies to the inside (to the left) or to the outside (to the right) of **SR**. Recall that,

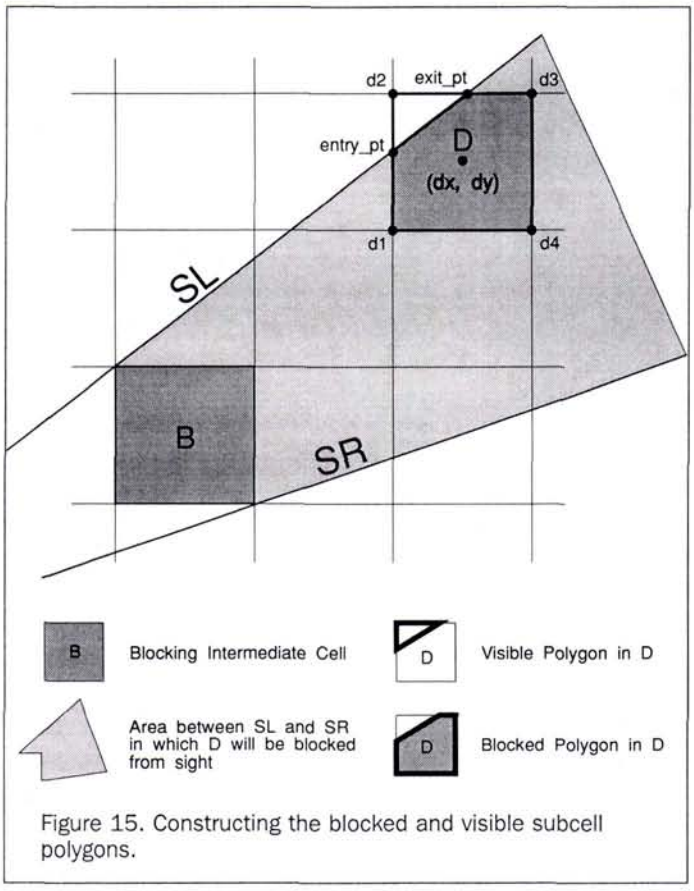

Figure 15. Constructing the blocked and visible subcell polygons.

in order for **SR** to be tested at all, **D** must lie to the inside of **SL**. Therefore, if **D** lies to the inside of **SR** as well, then **D** must lie completely within the two sightlines **SL** and **SR**. Hence, it is completely blocked from view. If **D** lies to the outside of **SR**, on the other hand, it must be totally unobscured.

Finding the Total Blockage Area for the Destination Cell

After each intermediate cell has been checked, the total destination blockage can be determined by overlaying all of the blocked areas within the destination cell using a polygon overlay operation based on the logical union operation. To do this efficiently, the algorithm must keep track of two things: the largest blocked polygon on the left side of **D** (caused by an intersection of **SR** and **D**) and the largest blocked polygon on the right side of **D** (caused by an intersection of **SL** and **D**). These two polygons are overlaid to determine the total blockage of **D**. This process is outlined as follows:

(1) Check each intermediate cell for visibility blockage.
  (a) If a cell is found that blocks the entire view of the destination, terminate the algorithm and mark the destination as completely blocked (as in Figure 13a).
  (b) If a cell is found that partially blocks the destination, calculate the blocked polygon and blocked area and note whether the blockage occurred on the left side or the right side of the destination relative to the viewer (i.e., whether it was caused by an intersection with **SR** or with **SL**). Check whether the blocked polygon is larger than any other blocked polygon already identified for that side of the cell. If it is, save the information.
  (c) Check whether the largest currently identified blocked polygons on the left side and the right side of the destination overlap one another. This will be the case if their areas sum to more than k_2, the total area of the destination cell. If they do, terminate the algorithm and mark the destination as completely blocked (Figure 16a).
(2) Once all cells have been checked, identify the largest blocking polygon on both the left and right side of the destination cell.
  (a) If left and right side blocking polygons both exist, then the destination cell will be partially blocked on both sides (Figure 16b).
  (b) If only a right side blocking polygon exists, the corresponding portion of the right side of **D** is blocked (refer to Figure 13c).
  (c) If only a left side blocking polygon exists, the corresponding portion of the left side of **D** is blocked (refer to Figure 13d).
  (d) If there are no blocked polygons on either the left or the right side of the destination cell, the destination is entirely visible.

It is worth noting that because of steps 1a and 1c, this algorithm does not necessarily check every intermediate cell. If at any point the destination is determined to be completely blocked by those cells already checked, the algorithm terminates.

Figure 17 shows the results of a viewshed calculated using the sub-cell binary analysis algorithm. In the viewshed, a distinct pattern of visibility lines can be discerned emanating from the source cell. This method of visualizing a viewshed is intuitively pleasing, for it more closely mimics the manner in which vistas in real life may open or close in wedge-like shapes that increase in width with greater distance from the viewer.

Implementation

The algorithms discussed in this paper were implemented as a set of programs designed to help users visualize the geometric problems that arise during viewshed analysis in a gridded data structure. The programs were implemented in Allegro Common Lisp running under AIX and X Windows on an IBM RS6000 workstation. The interface to the programs allows users to select different menu options and point and click on cells in a display grid to perform the following actions:

(1) Select a source cell and a destination cell and calculate the group of cells between them that must be checked during the visibility analysis.
(2) Divide a group of intermediate cells into vectors.
(3) Select a source cell, a destination cell, and an intermediate pixel and visualize the degree to which the intermediate cell blocks the visibility between the source and destination:
  (a) Assuming that the viewer is standing in the middle of the cell, or
  (b) Designating an arbitrary viewer position within the source cell.
(4) Load sample elevation data into the grid.
(5) Calculate a viewshed from any point in grid using:
  (a) A standard binary viewshed analysis.
  (b) The vector analysis method.
  (c) The sub-cell binary analysis method.

Conclusions

In this paper we have demonstrated that there are geometric limitations to gridded visibility analysis that will lead to inaccuracies in any viewshed algorithm that designates each cell as totally visible or totally blocked. Such inaccuracies are considered data-structure induced errors.

We have presented two methods for assigning "partial

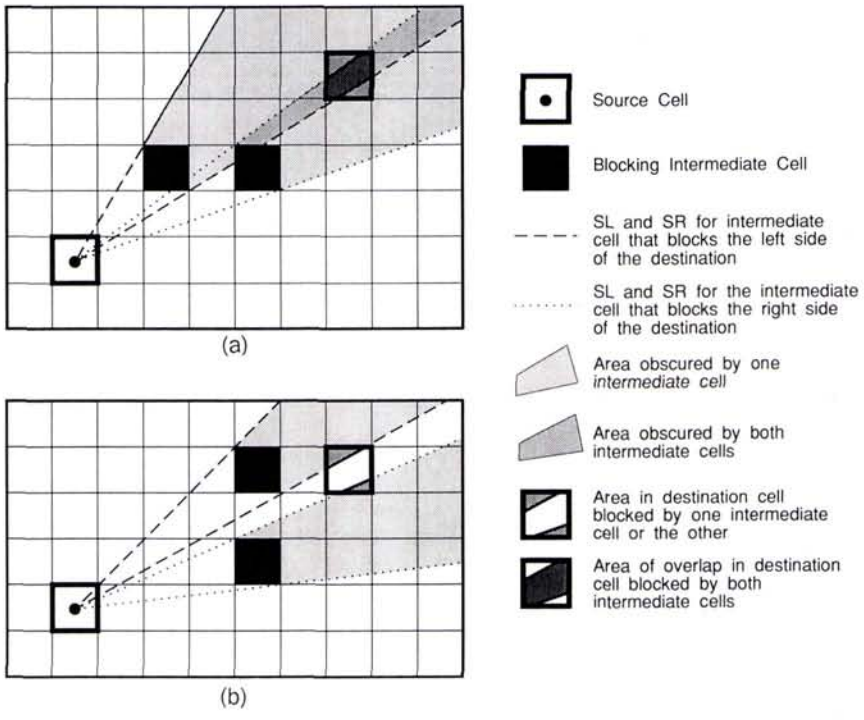

Figure 16. Boolean overlay of blocked polygons within the destination cell. In (a), the polygons overlap and obsc. the destination entirely. In (b), the overlay blocks the left side and right side of the destination, but leaves the middle clear.

visibility" weights to take into account that visibility between any two cells in a gridded data structure may be partly, but not totally, blocked. In the vector analysis method, intermediate cells between the source and destination are divided into one, two, or three vectors and visibility is calculated along each vector. The partial visibility weight is derived as the ratio of the vectors along which vision is not blocked to the total number of vectors.

The vector analysis technique serves as a quick method for approximating partial visibility, but cases exist in which it will significantly overestimate or underestimate the degree to which visibility is actually impeded. In the sub-cell binary analysis method, the geometric effect of each intermediate cell on the visibility between the source and destination is calculated separately. The results are then added together to determine the cumulative visibility blockage, a measure between 0 and 1. The blocked polygon approach returns a more accurate measure of visibility, but requires a significant amount of computational processing.

It should be noted that both techniques for deriving partial visibility measures discussed in this paper treat each cell as a plane of constant elevation – an assumption that is not entirely realistic. Hence, the accuracy of the techniques is sensitive to the resolution of the DEM. One direction for future research is to extend the calculation of partial visibility to interpolation-based viewshed calculations.

Another possible direction for future research is to combine the subcell binary analysis with Monte Carlo error simulation (Fisher, 1992). This would allow the calculation of different areas within each destination cell that are likely to be visible at given probability levels. Such information would present interesting challenges both in terms of analysis and visualization of error. A final topic for further research involves a quantitative comparison between standard binary viewshed maps, vector analysis viewshed maps, and sub-cell binary viewshed maps. The goal of such an effort would be to quantify the degree of error induced by raster-based algorithms for generating binary viewsheds.

Acknowledgments

We would like to acknowledge the comments of three anonymous reviewers which were helpful in refining the content of this paper.

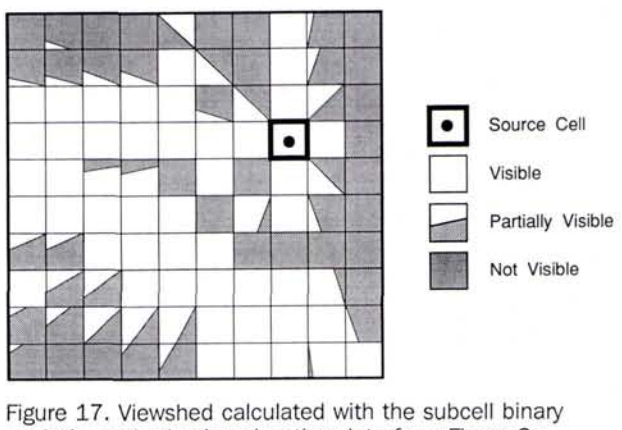

Figure 17. Viewshed calculated with the subcell binary analysis method using elevation data from Figure 2a.

References

Burrough, P.A., 1986. *Principles of Geographical Information Systems for Land Resources Assessment.* Oxford University Press, New York.

Felleman, J., and C. Griffin, 1990. *The Role of Error in GIS-Based Viewshed Determination: A Problem Analysis.* IEPP Report No. IEPP-90-2, Syracuse, New York.

Fisher, P.F., 1991. First Experiments in Viewshed Uncertainty: The Accuracy of the Viewshed Area. *Photogrammetric Engineering & Remote Sensing,* 57(10):1321–1327.

———, 1992. First Experiments in Viewshed Uncertainty: Error Simulation and the Fuzzy Viewshed. Photogrammetric Engineering & Remote Sensing, 58(3):345–352.

Goodchild, M.F., and J. Lee, 1989. Coverage Problems and Visibility Regions on Topographic Surfaces, *Annals of Operations Research,* 18:175–186.

Goodchild, M.F., and Karen Kemp, 1990. *Technical Issues in GIS: Volume II of the NCGIA Core Curriculum.* National Center for Geographic Information and Analysis, University of California, Santa Barbara.

Peucker, T.K., and N. Chrisman, 1975. Cartographic Data Structures, *The American Cartographer,* 2(1):55–69.

Travis, M.R., G.H. Elsner, W.D. Iverson, and C.G. Johnson, 1975. *VIEWIT: Computation of Seen Areas, Slope, and Aspect for Land-Use Planning.* General Technical Report PSW-11/1975, Forest Service, U.S. Department of Agriculture.

USGS, 1990. *Digital Elevation Models: Data User's Guide.* U.S. Geological Survey, Reston, Virginia.

(Received 23 June 1992; accepted 29 September 1992; revised 2 December 1992)

Call for Nominations
ASPRS Vice President

ASPRS is seeking nominations for 1994 Vice President from industry. Nominations must be made by a nominating letter signed by not less than 155 voting members and must contain a biographical sketch of the nominee.

Submit nominations to the Executive Director, 5410 Grosvenor Lane, Suite 210, Bethesda, Maryland, 20814-2160 by Wednesday, December 8, 1993 (20 weeks prior to the day of the 1994 Annual Meeting).

If you have any questions or need additional information, please contact Stanley A. Morain, chair, ASPRS Nominating Committee (505-277-4000).

Call for Nominations
ASPRS Fellow and Honorary Member Awards

ASPRS would like to encourage members to submit nominations for its Fellow and Honorary Member awards.

The Fellow Award is relatively new. It is conferred on active Society members who have performed exceptional service in advancing the science and use of the mapping sciences and for service to the Society. Nominees must have been active members of the Society for at least ten years at the time of their nomination.

The Honorary Member Award is ASPRS's highest honor. It recognizes an individual who has rendered distinguished service to the Society and/or who has attained distinction in advancing the science and use of the mapping sciences. It is awarded for professional excellence and service to the Society. Nominees must have been active members of the Society for at least fifteen years at the time of their nomination.

For more information, or to obtain a nomination form for these very special awards, please call Mindy Saslaw at 301-493-0290.