Lionel E. Deimel, Jr.
Robert J. Fornaro
David F. McAllister
Department of Computer Science
North Carolina State University
Raleigh, NC 27650
Calvin Lee Doss, II
IBM Corporation
Systems Communication Division
Research Triangle Park, NC 27709

Techniques for Computerized Lake and River Fills in Digital Terrain Models

The automatic collection of DTM, using correlation techniques, imposes special problems when bodies of water are encountered.

Abstract: *Difficulty in obtaining correlation over bodies of water demands that lakes and rivers be handled with special procedures for incorporation into digital terrain models. Such procedures designed for Defense Mapping Agency Aerospace Center's Integrated Photogrammetric Instrument Network are described. Boundaries for bodies of water are collected, and computerized processing provides "fill," that is, consistent elevation values for grid points within the bodies of water.*

Introduction

The use of digital terrain models (DTM's) can be expected to become increasingly more commonplace. A DTM is a collection of numbers representing the properties of a terrain. Profile lines, triangulated irregular networks, contours, and points on a regular grid are examples of DTM representations. If a DTM represents only elevation data, it sometimes referred to as a DEM or digital elevation model (see Collins and Moon, 1981). In this paper, by a DTM we mean a collection of regularly-spaced elevation values on a rectangular grid (Doyle, 1978), although the techniques discussed below may be applicable to other representations. A DTM so defined is easily stored and processed by digital computers. Before a DTM can be used for applications, however, it must be constructed. DTM data acquisition and quality assurance present a number of interesting technical problems. This paper is concerned with problems associated with bodies of water—lakes, rivers, and the like. The authors first encountered the issues addressed while working on the design of the Edit Processor component of the Defense Mapping Agency Aerospace Center's Integrated Photogrammetric Instrument Network, or IPIN (Elphingstone and Woodford, 1978).

IPIN is a collection of instruments and computers whose main purpose is the production of DTM's. The primary data sources are aerial photographs, from which data are extracted by means of analytical stereoplotters. The operation of some of these instruments has been highly automated to collect dense elevation information very rapidly. Nonetheless, the stereoplotters, even when combined with sophisticated computer hardware and software, cannot directly produce high quality DTM's without human intervention. Operators are involved both in the data collection (directing the compilation of certain features such as lake and river boundaries, ridge lines, and roads) and data editing. Data editing is carried out at "edit sta-

0099-1112/82/4809-1431$02.25/0

tions," computer graphics workstations where operators can display and modify raw or processed data and can generate and display profiles, contours, etc. Data acquisition errors can be corrected at this stage, and additional sources of data can be incorporated into the model. It is our opinion that any system producing DTM's, whatever the data acquisition process, will require manual review and modification if the final models are to be of high quality. Thus, the IPIN editing procedures may be a reasonable prototype for other systems, at least insofar as they serve demanding applications such as those of the Department of Defense.

Bodies of water present several problems for IPIN. Ideally, lake boundaries are represented by simple closed curves of constant elevation. Translated into the domain of the DTM, this means that grid points inside such a curve should all be at the same elevation. Unfortunately, correlation is difficult over water, making data collection problematical (Allam, 1978). Lakes are, therefore, best handled by collecting a boundary and shore elevation and having a computer automatically set all grid points within that boundary to the desired shore height. We describe this insertion of elevation values into the DTM as a "fill" and refer to the points involved as "fill points." In IPIN, most elevation data are either collected along arbitrary lines or along specific features, and an interpolation algorithm is then used to establish elevations at grid points. Our knowledge of lakes, however, should allow us to calculate exact grid point elevations within the lake without resorting to interpolation. Furthermore, near the shoreline, the lake elevation should neither influence the computed elevation at nearby grid points on the shore, nor should the height of the terrain be allowed to influence that of the lake. Thus, fill points are inserted late in the DTM production process and are not subject to further modification.

The processing of double-line drains (that is, rivers large enough to be represented as having two distinct banks) entails all the problems associated with processing lakes plus some additional ones. In the final model, we expect opposite banks to have equal elevations, and we expect points within the drain to be consistent with shore elevations and of nonincreasing height in the downstream direction. Our solution here as well is to collect boundaries and to generate fills, though the consistency requirements make this task particularly involved.

We discuss the computer techniques required for processing lakes and double-line drains. We will not present detailed algorithms but will try to elucidate trouble spots and suggest solutions to these problems. At North Carolina State University, the authors have implemented all the algorithms in FORTRAN programs which are independent of any idiosyncrasies of the IPIN environment.

Generating Lake Fill

The lake fill process begins with the manual collection of the lake boundary using an AS-11 analytical stereoplotter. This boundary is in the form of a sequence of points representing a polygon (the first and last points are considered adjacent). Elevation values of individual points are ignored; a determination of the lake elevation is made, and all boundary and fill points are given this elevation. Actually, the boundary may also include embedded, non-intersecting polygons representing islands.

Since the fill-generation process is merely one of setting points found to be within the lake to some fixed elevation, the essential problem is that of determining which grid points lie within the lake. This problem is quivalent to a standard computer graphics task for raster-scan displays known as *solid-area scan conversion* (Newman and Sproull, 1979). Solid-area scan conversion algorithms rely on the simple fact that a line crossing a region boundary goes from outside to inside or vice versa (Figure 1). Interior points may, therefore, be identified by examining one by one all possible candidates (within the rectangle bounded by the minimum and maximum X- and Y-coordinates among the boundary points, say) and connecting them with a point known to be outside. If the line between the candidate point and outside point crosses the boundary an odd number of times, the point is inside; otherwise it is outside.

For the process described to be carried out efficiently, advantage must be taken of the form of the boundary data. Although it is difficult to access directly boundary data near any particular grid point, it is easy to traverse the boundary and determine through simple interpolation where boundary segments cross grid lines. In particular, we may determine where the boundary intersects horizontal (y = constant) grid lines. Because we are only interested in setting elevations at grid points, we need only identify the vertical lines (x = constant and x = constant + Δ, where Δ is the grid spacing) between which such intersections occur. As we find these intersections, we sort them by X-Y-coordinates (see below). After the boundary has been processed, we should have lists of intersections along each horizontal line, with these intersections arranged in order by increasing X-coordinates. Between intersections, all grid points along the horizontal line are either inside or outside the lake, depending on whether the

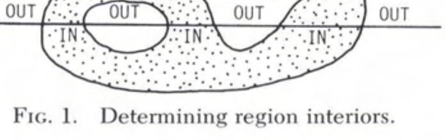

FIG. 1. Determining region interiors.

number of intersections to the left is odd or even. (see Figure 2). This method generates information about many points at once, a great advantage because the number of points with which we must deal is proportional to the square of the number of points on the boundary.

Storing and sorting the intersection information can be done simultaneously and with reasonable storage efficiency by using linked lists (Knuth, 1973), a scheme in which data are stored in *nodes* along with a pointer to the next such node in the sequence. Information about a horizontal grid line is kept in a linked list. These lists are arranged in order of increasing Y-coordinate by creating a linked list of pointers to them. The scheme is illustrated in Figure 3. Once the entire boundary has been processed and the structure in Figure 3 has been created, it is a simple matter to work our way from left to right along the grid lines from top to bottom, in order to find all points as illustrated in Figure 2. These points may be temporarily stored in a file in some convenient format or may be inserted directly into the DTM.

The algorithm described requires a slight modification to avoid failure in the rare circumstances that a boundary point falls on a grid line. The problem is illustrated in Figure 4. Whereas point *A* should be considered a single intersection of grid line and boundary, point *B* needs to be considered either as none or two. The problem may be overcome by taking into account slope changes of the boundary or by adjusting Y-coordinates so that the problem does not occur. For IPIN, coordinates are represented to a precision greater than their accuracy, so the latter solution was chosen. If a boundary point falls on a grid line, its Y-coordinate is increased by the smallest increment allowed by the data representation.

The lake fill algorithm is not limited to use with lakes. It is also used to provide fill for bodies of water such as an ocean whose boundaries are not completely within a model. In this case, the model boundary is taken as part of the "lake" boundary. As we shall see in the next section, the algorithm is useful for the general problem of finding points within a boundary and is an important part of the processing scheme devised for rivers.

Generating Double-Line Drain Fill

Generating double-line drain fill is more difficult than generating lake fill for several reasons. To begin with, it is much less clear what a "correct" solution to the problem looks like. The apparently simple requirement that rivers flow downhill does not necessarily tell us the relative elevations of two particular points. Neither does the requirement that shore points opposite one another be at the same elevation provide unambiguous directions for assigning "correct" elevations. These problems are illustrated in Figure 5. Is point *B* downstream from point *A* or not? Is point *D*, *E*, or even some other point opposite point *C*? These sorts of questions would be more easily answered if boundary data for double-line drains were accurate. Unfortunately, the manual collection of these data using the analytical stereoplotter tends to be noisy and inconsistent. This results from the operator's tendency to make discrete rather than continuous adjustments to the floating mark as boundaries are collected and to make systematic errors (consistently undershooting elevations on one boundary, for example). Also, the shore may be obscured by buildings, trees, bluffs, or other features which make it difficult to assess the shore elevation at a given point.

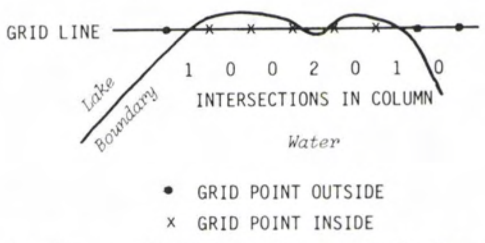

FIG. 2. Use of grid line crossings to find interior points.

As in the lake-fill problem, we wish to generate fill data from boundary data, so our first step must be to transform the raw boundary data into rational, usable boundaries. This process is facilitated by requiring a small quantity of additional data in the form of *control points* to be collected by the stereoplotter operator. These points should be nonincreasing in elevation in the downstream direction. The elevations are actually obtained from the shorelines, but the control points are placed in midstream. Control points are established at approximately fixed intervals or wherever the curvature of the drain changes substantially. A computer program ties these control points to the nearest boundary points on the shorelines in order to establish unequivocal elevations for those points. A least-squares fit of nearby points on each

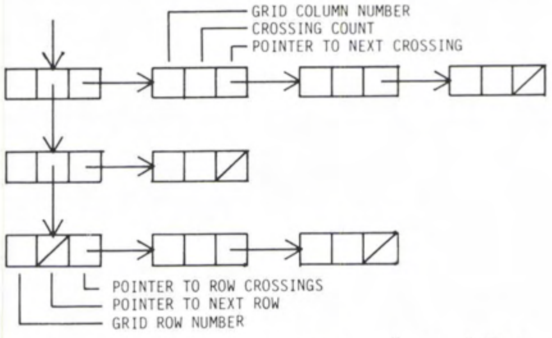

FIG. 3. Linked list representation for grid line-boundary intersection information.

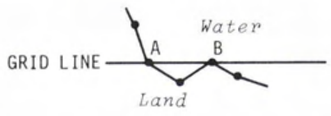

FIG. 4. Special cases for boundary analysis.

boundary is used to establish slopes at these boundary points. A quadratic spline interpolation technique uses this elevation and slope information to construct a profile of the drain along an imaginary center line. The Z-values of the profile are then projected onto the boundary points to give consistent and reasonable shores—boundaries which are nonincreasing in elevation in the downstream direction and for which corresponding points on opposite shores have the same elevation. (These modified boundaries are adjusted only in the Z-direction. X- and Y-coordinates are left unchanged.) The technique is described elsewhere by the authors (Deimel, 1978).

FILL PRODUCTION FROM MODIFIED BOUNDARIES

We now describe in detail the fill-production algorithm itself. The boundary of the river is given by two lineal strings of boundary points, P and Q, where

$$P = P_1, P_2, \ldots, P_m,$$
$$Q = Q_1, Q_2, \ldots, Q_n,$$

and each element is an ordered triple (x, y, x). If $P_i = (x, y, z)$, we write $P_{iX} = x$, $P_{iY} = y$, and $P_{iZ} = z$. It is assumed that boundaries are supplied in the downstream direction, so that if $i \leq j$, then $P_{iZ} \geq P_{jZ}$ and $Q_{iZ} \geq Q_{jZ}$. Any fill-generation technique should use only boundary points from each shore "near" the fill point. Ideally, we wish to construct a surface containing the boundary points P and Q which represents the river surface and can be used to estimate elevation levels. To accomplish this, we construct an "optimal" drain surface using triangles, where optimality is defined in terms of an acceptable surface of minimum area. (One does

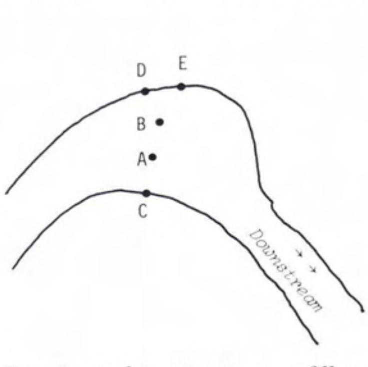

FIG. 5. Ambiguities in river fill requirements.

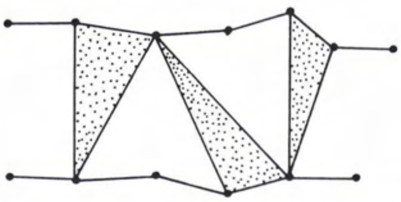

FIG. 6. Some acceptable triangles to be used in surface construction (projection onto X-Y plane).

not expect rivers to have substantial rises or depressions in midstream.) An acceptable surface completely covers the drain and has no holes. Clearly, the triangles should not overlap. Each triangle is formed from two adjacent points on one boundary and a point from the opposite boundary. Some acceptable triangles are illustrated in Figure 6. We now show how the optimal surface can be found. The technique is adapted from Fuchs *et al.* (1977).

We assume that control points allow us to partition the entire double-drain boundary into successive segments. Because these segments are chosen so that the four ends of the boundary segments are acceptably matched, we may construct our optimized surface for the entire drain segment by segment. This is important. The processing time for construction of this surface increases rapidly with the number of boundary points processed, so that having to process a long river in one piece might require unacceptably long processing times. Because the end points on segment boundaries are matched, we require that the top and bottom ends of drain segments be connected to form edges of triangles used in the surface being constructed (see Figure 7). Because the surface constructed must cover the drain without holes or overlaps, either triangle $P_1Q_1Q_2$ or triangle $Q_1P_1P_2$ must be included in the surface. Likewise, either triangle $P_mQ_{n-1}O_n$ or triangle $Q_nP_{m-1}P_m$ must be included. The minimal surface area is defined recursively as follows: Let $S(P_1,P_i,Q_1,Q_j)$ denote the areas of the minimal surface using points P_1 through P_i and points Q_1 through Q_j, and let $A(TUV)$ denote the area of triangle TUV. (Note that this area is calculated in three dimensions.) Then

$$S(P_1,P_i,Q_1,Q_j) = \min\{S(P_1,P_i,Q_1,Q_{j-1}) + A(P_iQ_{j-1}Q_j),\ S(P_1,P_{i-1},Q_1,Q_j) + A(Q_jP_{i-1}P_i)\}. \quad (1)$$

The optimal surface, $S(P_1,P_m,Q_1,Q_n)$ for the segment in question, can be found in time proportional to $m \times n$ using the following scheme. Let D be a directed graph such that the vertices of D form a rectangular $m \times n$ grid (Figure 8). Let V_{ij} denote the vertex in the ith row and jth column. With V_{ij}, associate $W(V_{ij}) = S(P_1,P_i,Q_1,Q_j)$. Clearly, $W(V_{11}) =$

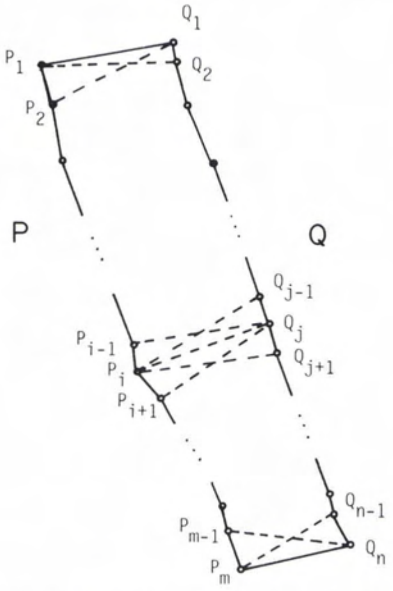

FIG. 7. Selection of triangles for optimal surface.

0. The value we seek, the area of minimal surface over the whole segment, is $W(V_{mn})$. From equation 1, it is clear that we may calculate the W's using the relation

$$W(V_{ij}) = \min \{W(V_{i,j-1}) + A(P_iQ_{j-1}Q_j), W(V_{i-1,j}) + A(Q_jP_{i-1}P_i)\}. \quad (2)$$

We do so by finding the W's from left to right on each row and processing the rows from top to bottom. As a value of W is assigned to each vertex, we note whether the value is derived from the vertex above or to the left. When V_{mn} is reached, we may follow these pointers back to V_{11} to generate a list of triangles actually used in the optimal surface so found. (See Aho *et al.* (1974) for discussion of implementation of such an algorithm.)

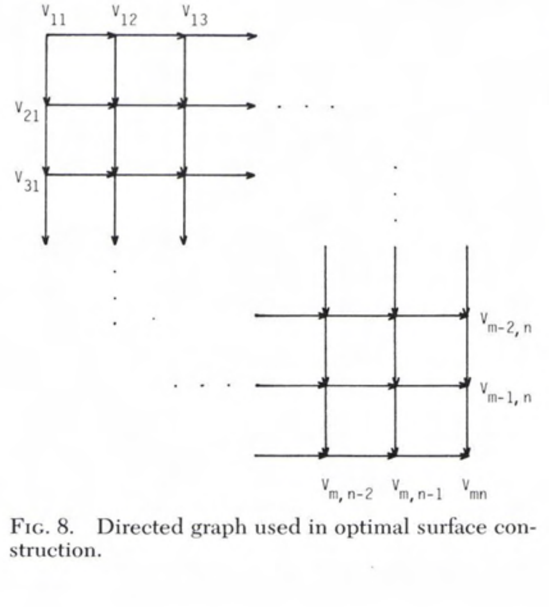

FIG. 8. Directed graph used in optimal surface construction.

Once we have found a surface of triangles to form the interior of our double-line drain, it would seem an easy task to find grid points within the drain and to establish fill points by interpolating elevation values for such points by projection onto the surface triangles. This task is less straightforward than it seems. The major problem is that the surface found does not always cover the same area as the surface on the drain itself. This problem is illustrated in Figure 9. Notice that the twisting nature of the river bank causes the surface generation algorithm to place erroneously the area of triangle BCD within the river. Further, the area of triangle ABC is covered by two surface triangles. The result is that certain grid points not in the drain could be identified as being within it and other grid points actually within the drain could be assigned any of several elevation values depending on the surface triangle used for interpolation. Of these two problems, the former is the more serious. The elevation ambiguity of points in triangle ABC is unlikely to be great. If the river slope is steep, its shores will tend to be smooth, so that the situation shown in the figure seldom occurs. If the river slope is slight, little effect is seen, because nearby boundary points are at about the same elevation anyway. To be sure that only points actually inside the drain appear in the final fill, we first generate fill point locations in a *fill file* by applying the lake fill algorithm to the boundary $P_1, P_2, \ldots, P_m, Q_n, Q_{n-1}, \ldots, Q_1$. We then take the resulting points from the fill file and determine in which triangles they fall, interpolating elevation values as we do so. In particular, if a triangle has vertices (x_0, y_0, z_0), (x_1, y_1, z_1), and (x_2, y_2, z_2), it may be used to assign an elevation value z to fill point (x, y, z) using the formulae

$$z = ax + by + c$$

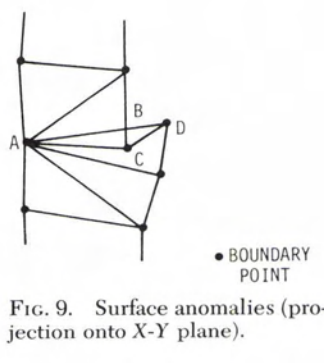

FIG. 9. Surface anomalies (projection onto X-Y plane).

where

$$
\begin{aligned}
a &= ((y_1 - y_2)z_0 - (y_0 - y_2)z_1 + (y_0 - y_1)z_2)/d, \\
b &= ((z_1 - z_2)x_0 - (z_0 - z_2)x_1 + (z_0 - z_2)x_2)/d, \\
c &= ((y_1z_2 - y_2z_1)x_0 - (y_0z_2 - y_2z_0)x_1 \\
&\quad + (y_0z_1 - y_1z_0)x_2)/d, \text{ and} \\
d &= (y_1 - y_2)x_0 - (y_0 - y_2)x_1 + (y_0 - y_1)x_2.
\end{aligned}
$$

The surface triangle formed by points (x_0, y_0, z_0), (x_1, y_1, z_1), and (x_2, y_2, z_2) should be used to interpolate an elevation value for the grid point at $D = (x, y)$ if D falls within triangle T formed by points $A = (x_0, y_0)$, $B = (x_1, y_1)$, and $C = (x_2, y_2)$. Figure 10 shows how we can determine if this is the case. If $D = C$, then D is inside T by definition. Assume this is not the case. Since A, B, and C are not collinear, either lines AB and CD intersect in a point E or they are parallel. In the latter case, D is clearly not in the triangle T. Assume point E exists. We assert that D is inside if and only if D is between C and E on line CD and E is between A and B on line AB. If D is inside, line CD must intersect the triangle boundary at at least one point Q other than C. If the point Q is on segments AC or BC, then E is either A or B, which implies that E lies between A and B. If the point Q is on segment AB, then $Q = E$, which again is between A and B. On the other hand, if E is between A and B, E is inside. If D is between C and E, both of which are inside T, D must be inside T, because a triangle is a convex polygon.

FURTHER CONSIDERATIONS

Practical application of the double-line drain technique we have proposed requires care in certain cases. Where rivers flow into larger bodies of water, the lake and drain boundaries should not overlap. Islands within rivers require that channels on either side of the island be treated as separate rivers. (Alternately, only the outermost banks are considered in finding a surface and the lake-fill algorithm is relied upon for removing points falling on the island. This technique would work poorly in pathological cases, for Niagara Falls, say.)

Some experience has been gained using this technique, and it is believed to give generally satisfactory results. However, results of this or any other algorithm should always be verified for reasonableness by finding contours for the drain surface or in some other way.

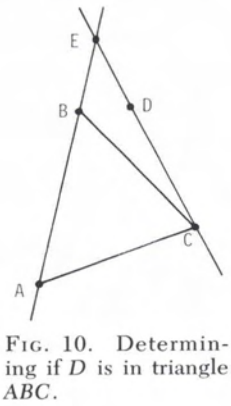

FIG. 10. Determining if D is in triangle ABC.

ACKNOWLEDGMENTS

This work was supported in part by the Rome Air Development Center under contract F30602-75-C-0118 and by NSF grant MCS76-0403.

REFERENCES

Aho, A. V., J. E. Hopcroft, and J. D. Ullman, 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, Mass.

Allam, M. M., 1978. DTM application in topographic mapping. *Photogrammetric Engineering and Remote Sensing.* 44(12), pp. 1513-1520.

Collins, S. H., and G. C. Moon. Algorithms for dense digital terrain models. *Photogrammetric Engineering and Remote Sensing.* 47 (1), pp. 71-76.

Deimel, L. E., C. L. Doss, R. J. Fornaro, D. F. McAllister, and J. A. Roulier, 1978. Application of shape-preserving spline interpolation to interactive editing of photogrammetric data. *Proceedings, SIGGRAPH '78.* Atlanta, Ga., pp. 93-99.

Doyle, F. J., 1978. Digital terrain models: an overview. *Photogrammetric Engineering and Remote Sensing.* 44 (12), pp. 1481-1485.

Elphingstone, G. M., and G. F. Woodford, 1978. Interactive graphics editing for IPIN. *Proceedings, Digital Terrain Models (DTM) Symposium,* supplement. St. Louis, Mo., pp. 597-609.

Fuchs, H., Z. M. Kedem, and S. P. Uselton, 1977. Optimal surface reconstuction from planar contours. *Communications of the ACM.* 20 (10), pp. 693-702.

Knuth, D. E., 1973. *The Art of Computer Programming,* volume 1, 2nd edition. Addision-Wesley Publishing Co., Reading, Mass.

Newman, W. M., and R. F. Sproull, 1979. *Principles of Interactive Computer Graphics,* 2nd edition. McGraw-Hill Book Co., New York, N.Y.

(Received 2 July 1981; accepted 6 January 1982; revised 5 March 1982)