A COMPACT TERRAIN MODEL BASED ON CRITICAL TOPOGRAPHIC FEATURES

Lori L. Scarlatos
Grumman Data Systems
1000 Woodbury Road
Woodbury, NY 11797

ABSTRACT

A broad range of applications, from military programs to survey and land use systems, rely on Digital Terrain Models (DTMs) for timely and accurate information. As more and more applications make use of this data, demands for both greater land coverage and finer, more accurate details are on the increase. Meeting these requirements can result in vast volumes of data which strain the memory limits of a computer system. Large digital elevation models can also create a bottleneck in the input/output processes and 3-D perspective rendering algorithms. Therefore, algorithms that generate compact and accurate elevation models are an important topic for research. We present one such algorithm here.

This paper describes a triangulation method which builds a DTM from a series of critical line features such as elevation contours, ridge and valley lines, and other breaklines. The method described is an improvement over current techniques because it triangulates any set of critical lines without human intervention, retains the original lines in the triangulation, adds no more than four points to the data, and runs relatively fast. Implementation results are given at the conclusion of the paper.

INTRODUCTION

Digital Terrain Models (DTMs) contain important topological information for applications such as 3-D terrain modeling, simulation, navigation, hydrology studies, visibility calculations, and route planning. For all of these applications, increasing demands for both greater land coverage and finer, more accurate details result in greater data volumes. For example, a typical DTM covers a one degree cell, which is about 3600 square nautical miles, an area smaller than Connecticut. With the elevations sampled every 3 arc seconds, or slightly less than 100 meters, this DTM occupies about 3 megabytes of memory. Elevations sampled at 10 meter intervals, a more desirable resolution for applications that rely on the DTM's accuracy, will occupy 300 megabytes for the same coverage. The United States covers over 3 million square nautical miles, so a data base for that area will grow correspondingly. This increase in data volume can strain the memory limits of a computer system. Large volumes of data also create a bottleneck in the input/output processes and 3-D perspective rendering algorithms. Therefore, an ideal DTM will provide highly accurate data in the smallest possible storage space.

DTMs developed from maps and imagery are generally stored either in a grid format or as a triangulated irregular network (TIN). Figure 1 shows how a simple contour map might be converted to these two formats.

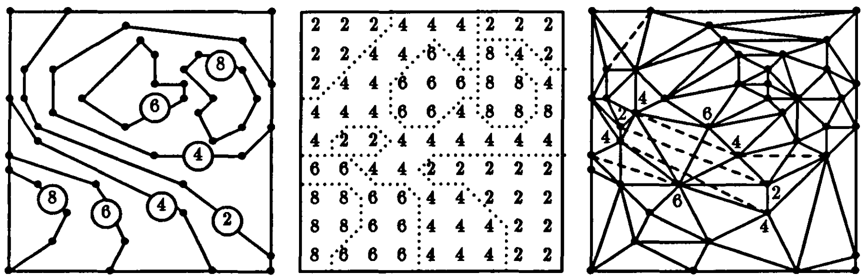

Figure 1. DTMs created with Grid Sampling and Delaunay Triangulation

Grid Sampling

Terrain in a grid format is represented by evenly spaced elevation samples or posts. DTMs produced by the U.S. Geological Survey and Defense Mapping Agency are of this form. Although a grid structure is easy to manipulate, it is limited by its dependence on the sample rate. For example, widely spaced elevation posts may miss important terrain features, producing an inaccurate model. This is clearly demonstrated by comparing DTMs of the same area created with different sample rates as shown in Figure 8 and Figure 9. It is for this reason that recent studies (such as U.S.A.E.T.L. 1987) call for the generation of data with 10 meter resolution or better.

Contrarily, narrow spacing between posts greatly increases the data volume, sometimes unnecessarily. For example, samples taken over a relatively flat area are translated into many polygonal facets when only a few are necessary. This can greatly increase the cost of computer image generation. Figure 1 shows how a grid structure can both contain many more samples than necessary in one area, yet miss important terrain features in another area.

Triangulated Irregular Networks

TINs, on the other hand, contain only those points which significantly affect the topology of the terrain. Included are points along contour lines or ridge and valley breaklines, as well as peaks and pits. These points are linked by edges that define a network of triangular facets conforming to the terrain. TIN structures are widely used in commercial geographic information systems because high resolution may be achieved with relatively little data. For example, flat or smoothly sloped areas may be represented by only a few points, whereas the fine detail in rocky or irregular areas may contain a great number of points. TIN structures are also convenient for generating perspective views, as many rendering algorithms work best with triangles.

Numerous methods have been developed to generate TINs (for example Christiansen 1978, Dennehy 1982, DeFloriani 1984, Watson 1984, Preparata 1985, Christensen 1987, Correc 1987, Dwyer 1987, McKenna 1987). Most of these algorithms are based on Delaunay triangulation, which forms connections between nearest neighbors within a scattered set of isolated elevation posts (Preparata 1985). Extensions to this algorithm typically increase time efficiency, simplify the steps in the algorithm, or add further structure to the TIN. However, Delaunay triangulation has one major drawback. It ignores the natural connections between points along contours and breaklines. These line segments are commonly used in maps to describe critical terrain features, and are frequently as important as the points themselves. As noted by Christensen (1987) and shown in Figure 1 , lines produced with Delaunay triangulation can cross these important breaklines, creating triangular plateaus over or under important features. This results in a misleading and inaccurate DTM.

The cartographic community has recognized this problem and proposed solutions which utilize connections between points along contour lines. However, these algorithms suffer because they require extensive human interaction (Christiansen 1978), do not extend well to handle complex terrain (Dennehy 1982), or double the number of data points as a side effect (Christensen 1987). In addition, none of these algorithms can triangulate intersecting breaklines such as ridge and valley lines and contour lines merged with other breaklines.

THE NEW TRIANGULATION

Our problem was to devise a way of building accurate, yet compact DTMs from common input sources. These sources include contour maps and ridge and valley lines traced from stereo imagery. This data would be input as a series of connected points forming closed polygons, open curves, connected graph structures, and a few isolated points. Our algorithm had to conform to these lines without adding data points.

The resulting algorithm, presented here, is an extension of Fournier and Montuno's 1984 triangulation algorithm developed for simple polygons with non-intersecting edges. This algorithm was an attractive basis for a solution to the more general terrain problem because

1. the polygon edges are inherently part of the triangulation, contributing at least one edge to each triangular facet
2. only the original points defining the polygon are used, resulting in a small data set
3. the algorithm is shown to run in O(n log n) time with a method that is relatively simple to understand and implement.

Although theirs is an excellent academic treatise, the Fournier and Montuno triangulation algorithm required a great deal of alteration before it could triangulate diverse terrain data. We created a valuable tool for terrain modelling applications by extending the algorithm to handle the following:

1. Any number of lines and points may be input. This makes the algorithm general enough to triangulate the data from any topological map.
2. Input points may be linked to form closed polygons, open chains of line segments, or even complex graph structures with several edges emanating from each point. This covers all combinations of contour lines, ridge and valley lines, and other breaklines which may occur in a topographic representation. Because maps often contain isolated peak elevations, isolated points may also be included.
3. Lines may connect points that have either a constant elevation (as do contours) or variable elevations (common with breaklines).

The algorithm takes the following steps. First, the data is sorted on point positions. Second, the data set is decomposed into a series of trapezoids. Third, these trapezoids are split by new edges linking points on the trapezoids. Finally, all of the resulting edges, new and old, are used to define the triangular mesh. The remainder of this section describes these steps in detail.

Setup

Digitized contour lines and/or breaklines are input as lists of connected 3-D coordinates. An edge list is maintained so that a point with multiple references may be condensed to a single point reference with multiple connections. Naturally, if two edges intersect, a point must be placed at the intersection. The only restriction on the data is that elevations for repeated points must agree.

Two steps must be taken before triangulation. First, the points are sorted on their Y value, from bottom to top. Points with the same Y value are sorted on their X value, left to right.

Next, the points must be bounded by a single closed polygon. Although the bounding polygon can be any shape, we selected a rectangle for convenience. We determine the bounding rectangle for the data set, add points at the corners of that rectangle (if necessary), and add edges connecting points along the periphery of that rectangle. New points are assigned elevations which are the weighted averages of their neighbors' elevations, where neighbors are adjacent points connected by an edge.

Defining the Trapezoids

Decomposition of the data into trapezoids parallels the definitions and steps outlined by Fournier and Montuno (1984). As shown in Figure 2 , a trapezoid is defined as a four-sided figure with its top and bottom edges parallel to the X axis. These imaginary top and bottom edges each pass through one of the input data points. The side edges, left and right, come from the collection of connecting lines, which are defined by two endpoints each. Therefore, the program stores each trapezoid as a list of six point indices, even though some of the indices may be repeated. For example, Figure 2 also shows that some trapezoids may look very much like triangles. In this case, the top point is also listed as a point on the left edge and a point on the right edge.

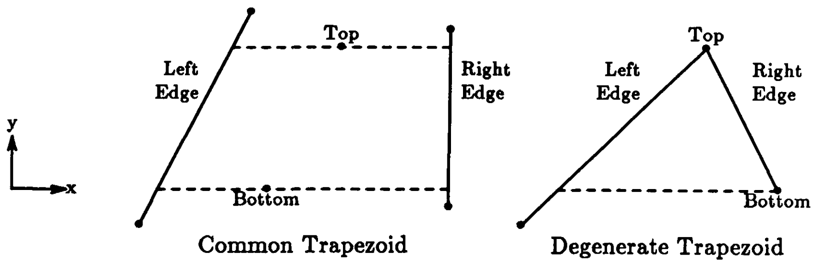

Figure 2. Trapezoids

Trapezoids are developed in the following manner. First, an active list of incomplete trapezoids is initialized. The trapezoids in the active list each have a bottom and two side edges, but no top edge. Initially, every active trapezoid has a bottom edge with the minimal Y value and side edges which are connections to points with the minimal Y value.

Then each data point is examined once, in its sorted order. This point is used to complete active trapezoids and contribute to new trapezoids. First, the active list is searched for trapezoids which have side edges that either surround the current point or belong to that point's edge list. These trapezoids are said to be completed by the current point. Typically, a point with m edges extending downward will complete m+1 trapezoids, as shown in Figure 3 Thus, an unconnected point will complete one trapezoid. Once a trapezoid is completed, it is removed from the active list and placed on a separate trapezoid list, with the current point given as its top point. The data set is completely decomposed into trapezoids when all of the vertices have been examined.

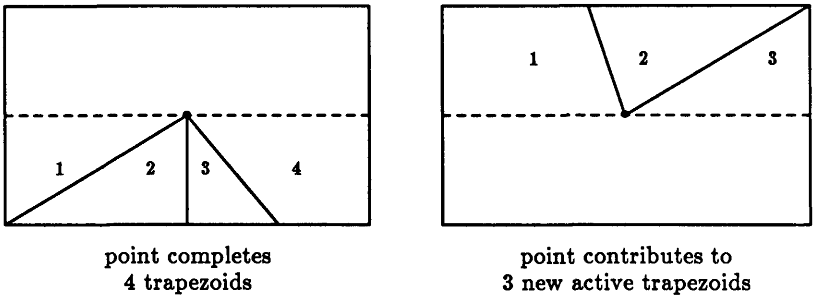

Figure 3. How points contribute to trapezoids

Second, new active trapezoids are created and inserted to the list where the old ones resided. As shown in Figure 3 , a point with n edges extending upward contributes to n+1 new active trapezoids. The current point is the bottom point of each new active trapezoid.

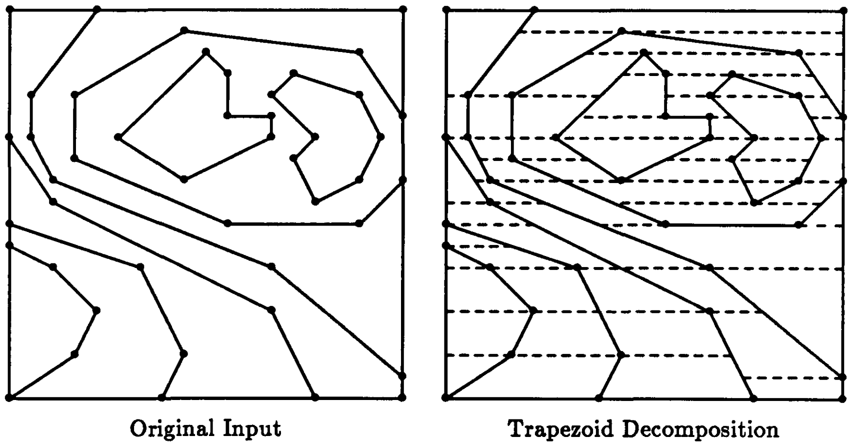

Figure 4. Contours decomposed into trapezoids

Figure 4 shows how the elevation contours from Figure 1 would be decomposed into trapezoids.

Finding the Remaining Triangle Edges

Once all the trapezoids have been found, they are split into triangles by connecting points that lie on opposing sides of the trapezoids. These splitting edges are added to the edge lists of the newly connected points. Each trapezoid that is completely split into triangles is removed from the trapezoid list. When the trapezoid list is empty, all necessary edges have been found. These new edges, combined with the original contour edges, will form the triangular network covering the terrain.

Trapezoids are split in two passes. The first pass recursively splits individual trapezoids. If two or more points lie on the top or bottom edge of the trapezoid, then these points are linked by new edges, added to the points' edge lists. Any point appearing on the top or bottom edge may then serve as the top or bottom point. If a trapezoid has a top point and bottom point which do not lie on the same side edge, then these points are also connected with a new edge. This splits the original trapezoid into two new trapezoids, each of which are examined for further splits. Once a trapezoid becomes triangular in shape, and no more splits are possible, it is removed from the list. Figure 5 shows how the trapezoids from Figure 4 would be split in the first pass.

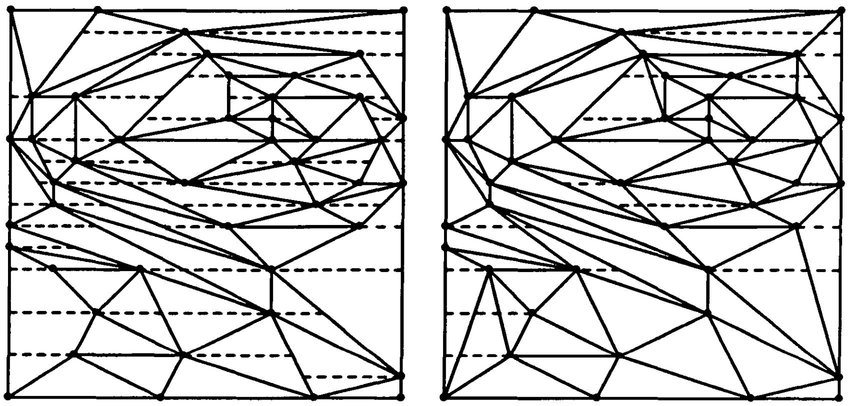

Figure 5. Splitting trapezoids

After the first pass, all remaining trapezoids on the list may be collected to form one or more unimonotone polygons (Fournier 1984). A unimonotone polygon is characterized by a single major edge and a two or more minor edges. The major edge is defined by two endpoints which have the maximum and minimum Y values for the polygon. All points on the minor edges fall within this vertical range. Therefore, all remaining trapezoids with the same major edge are part of the same unimonotone polygon.

The second pass splits the unimonotone polygons into triangles. Two approaches may be taken to decompose each unimonotone polygon into triangles.

1. If a point on a minor edge is linked to the major edge, a triangle may be formed by connecting its minor neighbor to its major neighbor. The new edge must fall within the unimonotone polygon, and therefore the angle formed by the two pre-existing edges must be checked.
2. The two minor neighbors of a minor point may be linked by an edge if that edge falls within the unimonotone polygon.

Figure 5 also shows how the remaining trapezoids would be split in the second pass.

Collecting Edges to Build Triangles

The edges splitting the trapezoids, along with the original edges, form a triangular network covering the area within the original bounding polygon. In the final step, these edges are organized to form a triangle list. This is done by repeating the following steps for each vertex.

1. The current point's connecting edges are sorted in counter-clockwise order.
2. Each pair of adjacent edges on the sorted list form a triangle, because the points at the ends of those edges are linked to one another. Add this triangle to the list only if the indices of the two end points are greater than the index of the current point. Otherwise, this triangle is already on the list.

IMPLEMENTATION

To test our algorithm, we traced contour lines from the field map shown in Figure 6 and fed them to our triangulation routine. This small test area covers 304x482 meters, and is represented by 2014 points. Our routine triangulated this data in 4.85 CPU seconds on a VAX/8530, with 0.35 seconds of that time spent on the initial sorting. The resulting DTM contains 2018 points on 4020 triangles. Figure 7 shows a perspective view of this data.

For comparison, we also generated a grid format DTM from the same contour map. This data was sampled at a regular post spacing of 2 meters and placed in a grid structure containing 152x241 points, or 72480 triangles. Figure 8 shows the resulting grid rendered in perspective. Although this produces an equally accurate representation of the scene, it also contains 18 times the number of points and triangles contained in the TIN. We also sampled the data with 8.25 meters between posts to produce a grid data set with approximately the same number of points and triangles as the TIN. Figure 9 shows the lower resolution data set rendered in perspective. Notice how much detail is lost in the lower resolution.

TABLE 1. Triangular vs. Grid DTMs

DTM	No. of Points	No. of Polygons	Render Time (min.)
Triangulated Contours	2018	4020	2:46.04
2m posts	36,632	72,480	3:52.14
8m posts	2088	3990	2:36.66

Table 1 summarizes our results. Rendering time is measured as minutes of CPU time on a VAX/8530. It is important to note that this sample area is small relative to the coverage required by most applications. Thus, the savings incurred by the triangulation will be greater as the area of interest grows.

CONCLUSION

At a time when demands for highly detailed data over large areas is placing a strain computer systems, we have demonstrated that TINs can represent details far more compactly than a grid format. Furthermore, we have presented a proven algorithm that produces a DTM from critical line features in a reasonable amount of time. This algorithm is superior to previous algorithms because it 1) maintains the integrity of the critical lines, 2) triangulates any series of input lines, including intersecting lines, without requiring any human intervention, and 3) adds no more than 4 points to the original data set.

ACKNOWLEDGEMENTS

I would like to thank J. Mendelson, R. Kelly, H. Tesser, and G. Gardner, without whom this research would not have been possible.

REFERENCES

Christensen, A.H.J., 1987. Fitting a triangulation to contour lines, *Proceedings of AUTO-CARTO 8*, 57-67.

Christiansen, H.N. and Sederberg, T.W., 1978. Conversion of complex contour line definition into polygonal element mosaics, *Proceedings of SIGGRAPH '78*, 187-192.

Correc, Y. and Chapuis, E., 1987. Fast computation of Delaunay triangulations, *Advances in Engineering Software*, 9(2), 77-83.

DeFloriani, L., Falcidieno, B., Nagy, G., and Pienovi, C., 1984. A hierarchical structure for surface approximation, *Computers and Graphics*, 8(2), 183 - 193.

Dennehy, T.G., and Ganapathy, S., 1982. A new general triangulation method for planar contours, *Proceedings of SIGGRAPH '82*, 69-74.

Dwyer, R.A., 1987. Faster divide-and-conquer algorithm for constructing Delaunay triangulations, *Algorithmica*, 2(2), 137-151.

Fournier, A., and Montuno, D., 1984. Triangulating simple polygons and equivalent problems, *ACM Transactions on Graphics*, 3(2), 153 - 174.

McKenna, D.G., 1987. The inward spiral method: an improved TIN generation technique and data structure for land planning applications, *Proceedings of AUTO-CARTO 8*, 670-679.

Preparata, F.P. and Shamos, M.I., 1985. *Computational Geometry*, Springer-Verlag, New York.

U.S. Army Engineer Topographic Laboratories, 1987. Digital terrain data requirements, *Army Environmental Sciences*, U.S. Army Corps of Engineers, 5(3), 10-11.

Watson, D.F. and Philip, G.M., 1984. Survey: systematic triangulations, *Computer Vision, Graphics, and Image Processing*, 26, 217-223.

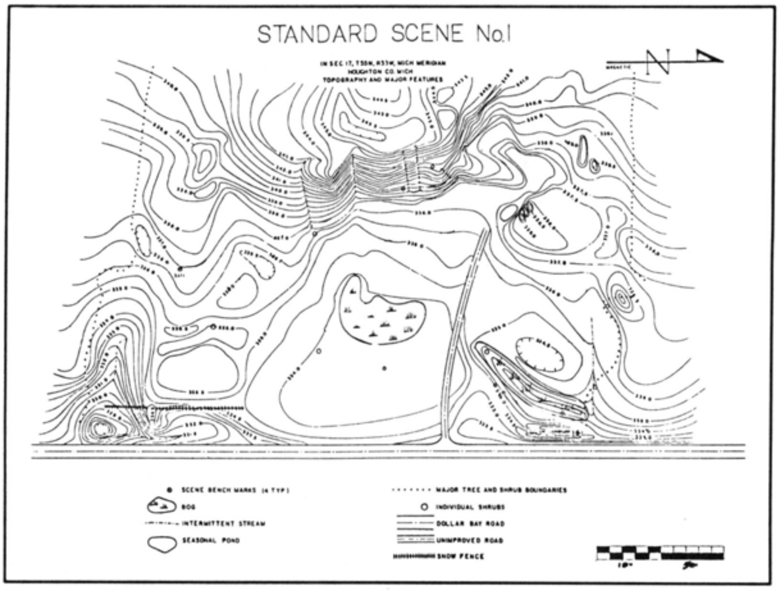

Figure 6. Field map used as input for triangulation

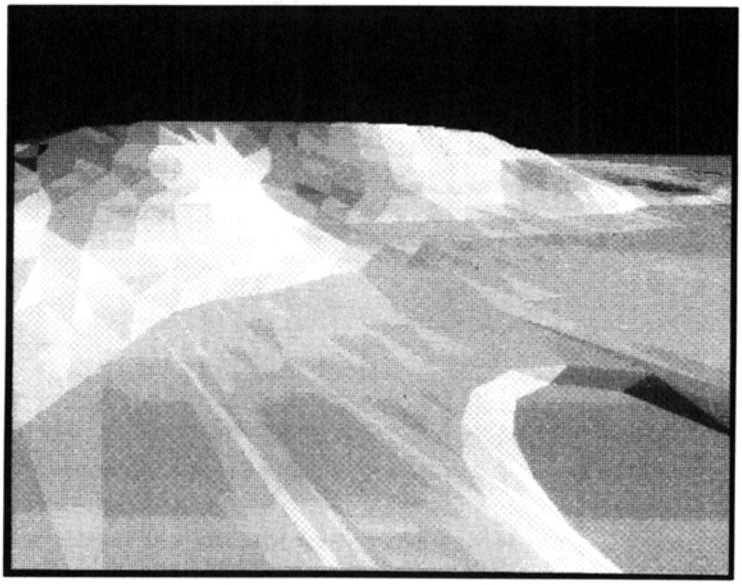

Figure 7. Triangulated field map

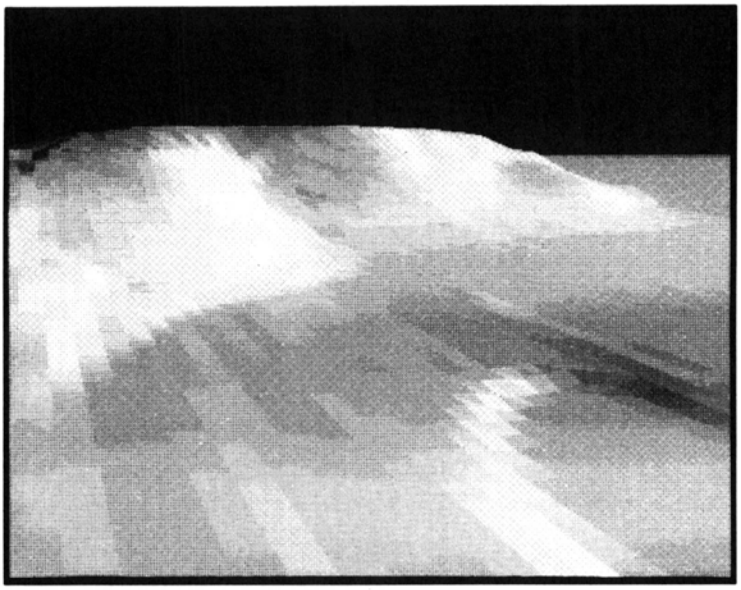

Figure 8. Grid data, sampled every 2 meters

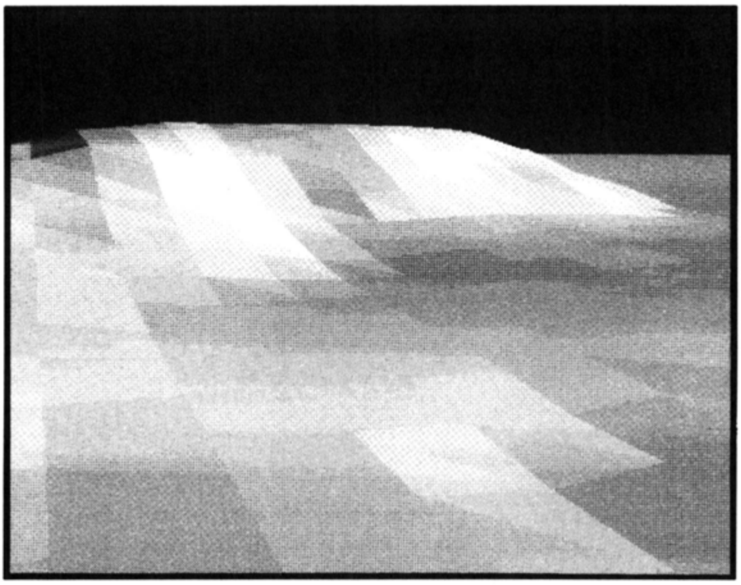

Figure 9. Grid data, sampled every 8.25 meters