HIERARCHICAL TRIANGULATION USING TERRAIN FEATURES

Lori Scarlatos
Grumman Data Systems
1000 Woodbury Rd.
Woodbury, NY 11797

Theo Pavlidis
State University of New York
Dept. of Computer Science
Stony Brook, NY 11794

ABSTRACT

Triangulated Irregular Networks (TINs) have been used for surface approximation in many applications. A hierarchical representation is often desirable if the surface is to be rendered at different resolutions. Past work has emphasized techniques where a coarse triangulation is refined by focusing on plane geometry using very little the surface data. For example, a new point is introduced where the deviation of the surface is maximum and the triangle is subdivided into four others. Variants of Delauney triangulations have also been used. We propose a technique where we look more carefully into the features of the surface to be approximated. For example, if there is a ridge, the original triangle is divided by a line along the ridge and one of its vertices is used to subdivided the resulting quadrilateral. In this way the number of very thin triangles (slivers) is significantly reduced. Such triangles produced undesirable effects in animation. In addition the number of levels of the TIN tree is reduced which speeds up searching within the data structure. Tests on data with digital elevation input have confirmed the above theoretical expectations. On eight such sets the average "sliveriness" with the new method was between 1/5 and 1/10 of old triangulations and number of levels was about one third. There was an increase in the number of descendants at each level but the total number of triangles was also lower.

Note: Because of space limitations many details and examples have been omitted from this version of the paper. Interested readers should request from the authors a technical report with the same title providing full details of the method, as well as additional examples of implementation than poresented here.

1. Introduction

Triangulation is a popular technique for approximating surfaces. Many cartographic applications rely on triangulated terrain models for simulation, visualization, and analysis. Applications such as the finite element method, medical imaging, and graphics in general (including animation) also utilize triangulated surfaces. Triangulations are popular because triangles are simple geometric objects that are simple to manipulate and render. Triangulated Irregular Networks (TINs) offer additional advantages because they are not bound by regularity constraints. Hence TINs can approximate any surface at any desired level of accuracy with a minimal number of polygons.

In general treatises on triangulation fall into two groups: those that search for efficient algorithms to triangulate a given polygon and those that emphasize the application of the triangulation to approximate surfaces. In the former category, [1] outlines a greedy triangulation for polygons using dynamic programming, and [2,3,4] outline even faster algorithms. Reference [5] describes an algorithm for triangulating a polygon with the constraint that no obtuse angles are allowed.

However, we are more interested in approximating surfaces with triangulation. Since triangulation of a surface is done once and the results are used repeatedly, computational efficiency of the triangulation is not as critical. Instead, we strive to produce the triangulation that best approximates the surface. A good surface triangulation meets given accuracy requirements while minimizing the number of triangles used to model the surface. A good triangulation also minimizes the number of very thin, slivery triangles which produce artifacts in renderings of surface models. Surface triangulation algorithms fall into two sub-categories: those that triangulate a set of isolated points, and those that triangulate a set of connected points (critical lines) by adding non-intersecting connections.

Triangulation algorithms for isolated points are numerous, although many are a variation on the Delaunay triangulation scheme. References [6,7] describe only a few of these. Triangulations of both points in a plane and simple polygons is covered in [8] while a survey of greedy, Delaunay, and optimal triangulations of isolated points (the last of which is still an open problem) is found in [9]. Other algorithms for triangulating points on the plane can be found in [10,11]. Yet many surfaces -- especially natural ones -- form continuous patterns which aren't adequately represented by isolated points. Because these algorithms ignore the third dimension, they can produce triangle edges that contradict the topology of the actual surface.

Triangulations of planar graphs -- i.e. points with initial connections -- are more likely to produce accurate models because the lines describing surface topology are given. Some papers such as [12,13] deal with triangulating cross-sections from tomographic scans, although the methods of both of these papers require human intervention when the contours get complex. Other triangulations of cartographic critical lines have been recently published [14,15,7].

This paper describes a hierarchical triangulation built from a digital elevation model (DEM) in grid form. Each level in the hierarchy corresponds to a different level of detail that approximates the surface within a given tolerance (i.e. maximum error). The top level is the coarsest, containing the smaller number of triangles which approximate the surface within some tolerance t_0. The $i+1^{th}$ level is related to the i^{th} level as follows. Tolerance t_{i+1} is smaller than t_i. For each triangle T_j^i of the i^{th} level there exists a set of triangles $T_{j_1}^{i+1}, \cdots, T_{j_n}^{i+1}$ at the $i+1^{th}$ level, such that

$$\bigcup_{k=1}^{n} T_{j_k}^{i+1} = T_j^i.$$

The number of descendent triangles n can be any integer, possibly 1. Such a hierarchical triangulation allows easy implementation of such operations as zooming when viewing the surface. It also facilitates searching and other geometrical operations such as finding the intersection of two surfaces. Furthermore, it makes real-time simulation and visualization possible for applications that can represent less important areas with less detail in mixed-resolution models.

2. Past Work

Previous work on triangulations by one of the authors has researched techniques to find critical points and lines [16] , triangulate them [15] , and then refine those triangulations to produce a hierarchy of detail levels for fast spatial search with maximum accuracy [17] . These algorithms represent significant improvements over other algorithms, producing good triangulations. However, the above algorithms do not allow for refinement down to a specified level of accuracy.

Although several refinement techniques have been suggested in the literature [18, 19, 20] , these algorithms can introduce artifacts to a terrain model because they consider only the locality of points in a 2D plane instead of actual terrain topology. Consider, for example, DeFloriani's first algorithm for triangle refinement [19] which splits a triangle by connecting its corners to a selected interior point (usually, the point of maximum distance between the surface and the plane defined by the vertices of the triangle). The algorithm ignores the *coherence* of cartographic features: valleys or ridges have a linear structure. Figure 1 shows the results of ignoring such coherence. We assume that a ridge (its points marked by small circles in (a)) crosses the triangle. (b) shows that the maximum point triangulation will produce an unreasonably large number of triangles. Even worse, the triangles will have very sharp angles, which is an undesirable property [5] . Such triangulations may cause numerical stability problems in finite element methods and also produce undesirable display artifacts. In contrast, if we realize that we deal with a ridge and introduce a dividing line along it as shown in (c) we will end up with fewer triangles, none of them very sharp. We should point out that triangles with very sharp angles may be inevitable for some types of data. For example, if we have a steep cliff we will see large differences in the value between adjacent elevation points. Then triangles with very sharp angles cannot be avoided.

3. New Method

A generalization of the critical line method could produce better accuracy with fewer triangles. We have implemented such a strategy as follows. We start with a coarse triangulation produced by techniques outlined in Scarlatos' three papers. Our refinement technique pays particular attention to terrain characteristics, approximating critical lines at each step. To accomplish this, we find for each triangle four error values: one inside the triangle, and one on each of the three edges.

Figure 2 shows the five ways that a triangle may be refined. In all cases our goal is to approximate the surface form by splitting edges if necessary, thereby reducing the number of splits or refinements required to achieve a desired level of detail. If an isolated peak or pit resides within the triangle, it is split at that central peak or pit point as shown. If a single ridge or channel line travels up to that peak or pit, the triangle is split where that line crosses the edge of the triangle and at the central peak or pit point. If, however, a single ridge or channel line enters the triangle and ends at a saddle point or flat, then the center point is insignificant and the triangle is split by one edge as shown. If a ridge or channel line passes through the triangle, significant errors will be found on two edges of the triangle. A line connecting these points approximates the topographical line, and an additional edge splits the remaining quadrilateral. Finally, if a triangular patch corresponds to a rapidly fluctuating surface, many points are likely to have significant errors. Splitting this type of triangle on all edges segments the high-frequency regions which may then be further refined.

During triangulation, special care must be taken to avoid quantization artifacts caused by vertices falling between grid points. This topic is discussed in our technical report of the same title.

We repeatedly split the triangles until they all meet the given accuracy requirements for the current level of detail. This accuracy is checked by projecting all original grid points to the surface of the triangulated model and measure the difference. Intermediate triangles, used to produce but not included in the final triangulation for the current level of detail, are discarded. This reduces the number of levels in the hierarchy and the number of triangles within each level, making faster search, display, and processing possible. If polygon constraints are more important than the level of error, we can easily check the polygon count and terminate a level when the limit is approached.

4. Implementation of the Algorithm

Here we outline the implementation of the algorithms associated with our method. Each point in the original elevation matrix has an associated ground position (also used as an array index) and elevation. Two global functions, GET_POINT_INDEX and GET_ROW_COL, respectively return a point index given row and column numbers, and row and column numbers given a point index. Each point also is associated with zero, one or two triangles in a Membership list. Triangle vertices are associated with no triangles. If a point is associated with 2 triangles, then it belongs to an edge and a distance to that edge is stored. Otherwise, the point is within a single triangle. Three point indices define a triangle. Each triangle is associated with a level of detail and contains pointers to its parent, its children, its siblings, and its three neighbors (ie the triangles that share its edges). In addition, triangles have temporary structures keeping track of their splitting points, the maximum error found within them, and the number of edges to be split. A flag indicates whether the triangle meets the accuracy standards of the current level.

The main program retrieves the input data, calls the appropriate triangulation routines, and writes out the results to a data base.

The level of error within a triangle is found by taking all grid points within the boundaries of that triangle, projecting them to the surface of the triangle, and comparing the results to the original elevation values. Errors are found in four regions on a triangle: on each of the three edges, and within the triangle. The errors found by this routine determine how the triangle will be split (if at all) later on.

The procedure FIND_SPLITS relies on several local routines to do its work. Routine PROJECT calculates the elevation of a point projected to a given triangle. The procedure SORT uses the quicksort method to reorder LIST and DISTLIST so that the values in DISTLIST appear in descending order; and the point indices in LIST are the ones corresponding to the appropraite DISTLIST values.

Routine SIGNIFICANCE returns TRUE if the given error value is significant enough to merit splitting at the related point. This may be calculated in two ways. Using the first way, if the given value is greater than the threshholded error for the current level of detail, then that point is significant. Using the other way, if the given value is more than some percentage of the maximum error found within a triangle, then that point is significant. In either case, a point is insignificant if its error falls at or below the threshhold for the current level of detail.

Although each of the five regular triangulation algorithms is different, they all follow the same pattern of steps.

There are two important local procedures for adding edges to the triangulation. The first is ADD_EDGE which connects two previously unconnected vertices. MEMBER values are updated for points on and alongside the new line. Endpoints are given null MEMBER values. The second routine, SPLIT_EDGE, splits an existing edge into two new edges, possibly adding a bend in the middle. If the edge has not already been split, it calls ADD_EDGE twice. Otherwise, it finds the new neighbors added by the split and returns those.

GEN_TRIANG generates a new triangle number. To save space, the current triangle number is recycled, becoming the first child of T. ADD_TRIANG actually generates the new triangle.

When a splitting point on one edge is colinear with another edge, special triangulation must take place.

5. Data Base Structure

This algorithm produces all of the information required to both render the 3D surface and search for spatial relationships. A header record includes information such as a ground position for the lower-left corner of the triangulation; spacing between posts in the original grid; elevation ranges in the triangulation; number of levels. This is followed by the level records. Each level has a threshhold of allowable error, used to produce the triangulation. It also has a number of points, number of triangles, and a list of triangles. Each triangle is defined by 3 point indices, and has a parent pointer, child pointers, and neighbor pointers. All this is followed by a single point list. Only points that appear in the triangulation are written to the data base; all others are unnecessary. Points are ordered such that if level L uses N points, then it uses points $1, \cdots ,N$. This reduces retrieval time for a level of detail.

6. Results

To test our algorithm, we implemented and ran it on several test cases. We selected eight (8) areas of interest (AOI) as test data, each covering 75x75 elevation points. These areas, representing various terrain types, come from the Defense Mapping Agency's Digital Terrain Elevation Data (DTED Level 1).† A triangulation

† DTED Level 1 elevation points are three seconds of an arc

employing all 5625 points in an AOI would contain 10,952 triangles.

We implemented our algorithm with varying parameters to see which behaved best. The first parameter is how we determine the significance of point p's error e_p. Error e_p may be considered significant compared to 1) tolerance value t_i for level i, so that $e_p \geq t_i$, or 2) a percentage N of the maximum error $e_{t_{max}}$ found for current triangle t, so that $e_p \geq e_{t_{max}}$. The second parameter determines when we split a triangle at one edge and a significant center (as shown in Figure 2). Center point c may be considered significant compared to 1) the error e_v of splitting point v on the edge of the triangle, so that $e_c \geq e_v$, or 2) the significance value used to determine the significance of all other points, as determined by the first parameter. Hence we ran four optional programs. Option 1 uses tolerance to determine significance, and requires a center point to be at least as significant as an edge point in order to be used. Option 2 uses 75% of the maximum error within a triangle to determine significance, and also requires a center point to be at least as significant as an edge point. Options 3 and 4 are like options 1 and 2 respectively, except that a center point's significance is determined by the usual measures. As a basis of comparison, we implemented DeFloriani's first algorithm [19] and ran it with the same test data.

All four options and the basis algorithm were executed on the eight AOIs, producing triangulations with a minimum error of 10 meters. All tests demonstrated that our algorithm works well. Table 1 shows some of our results.

A better triangulation will produce fewer slivery triangles. The table shows how slivery the resulting triangles were. We measured sliveriness with the following ratio, calculated for each triangle in the triangulation:

$$\frac{Perimeter^2}{Area}$$

(For equilateral triangles this ratio equals 20.78.) We then calculated the average sliveriness for the entire structure, and divided the result by the sliveriness ratio for an equilateral triangle. A lower sliveriness value is better. Note that on the average most of the triangles have much sharper angles than sixty degrees. Using the basis case, some angles are as small as 0.25 degrees. Notice how much better our algorithm performed, using all four options. Options 1 and 2 seem to work about equally well, indicating that the best measure of point significance is determined by data characteristics. Options 3 and 4 consistently performed a little worse. This leads us to conclude that a center point should only be included in the division of a triangle if it is more significant than the edge point.

A better triangulation will permit fast spatial search. The time required for a search is determined by the number of levels that must be searched, and the number of child nodes that must be examined at each level. DeFloriani's algorithm, which always splits a parent triangle into 2 or 3 children, has an average of about 2.5 children per parent node. The number of levels in a hierarchy depend on the number of iteration levels required to build the triangulation. Our algorithm, on the other hand, guarantees a fixed number of levels in the hierarchy, but can split a parent triangle into any number of children. Although one may presume that a very large number of children will be produced, table 7 shows that this is not the case. In this table, the "old" algorithm is the basis case, and the "new" algorithm is our option 1. Table 2 shows that search times using our hierarchy will be as fast as, or faster than, the other. Additional results can be found in our technical report.

Figure 3 demonstrate the significances of the improvements our algorithm makes. Figure 3a shows a view of AOI 1 using the original grid data, represented by 10,952 triangles each. Figure 3b shows the same views of the data triangulated with DeFloriani's algorithm (the basis case), containing 1714 and 2318 triangles respectively for a maximum error of 10 meters. Figure 3c shows the same views of our algorithm (using option 1), containing 1836 and 1979 triangles respectively for a maximum error of 10 meters.

While Delaunay triangulations have been proposed as means for reducing the number of very sharp triangles within hierarchical structures, [20] Delaunay triangulations have serious drawbacks as discussed in [14] . In some cases, using Delaunay triangulation to add points can actually increase error levels in the model, even though the model contains more triangles. The algorithm of [5] while it avoids generating obtuse triangles, it generates far too many points and triangles for our purposes.

7. Conclusions

We have presented an algorithm that produces a hierarchy of triangulations in which each level of the hierarchy corresponds to a guaranteed level of accuracy. Because our algorithm focuses on the topology of a surface, it minimizes the number of triangles, and produces fewer long and slivery triangles, within each level of detail. These features add up to a triangulation that provides great accuracy in a model that can be rapidly searched, rendered, and otherwise manipulated.

We plan to extend this work by using triangulation for representing other data besides terrain. Finite element methods, for example, use triangulations extensively, yet

apart, about 100 meters.

suffer from occurrences of slivers. We will consider what the triangulation criteria are for related applications. Other data such as images may also be triangulated. We will address numerous issues in this area, such as how to fuse disparate triangulations such as a terrain model with a Landsat overlay.

References

1. Aho, A.V., Hopcroft, J.E., and Ullman, J.D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.
2. Garey, M. R., Johnson, D. S., Preparata, F. P., and Tarjan, R. E., ''Triangulating a simple polygon,'' *Information Processing Letters*, vol. 7, pp. 175-180, 1978.
3. K. L. Clarkson, R. E. Tarjan, and C. J. Van Wyk, ''A Fast Las Vegas Algorithm for Triangulating a Simple Polygon,'' *preprint*.
4. Fournier, A. and Montuno, D., ''Triangulating simple polygons and equivalent problems,'' *ACM Transactions on Graphics*, vol. 3, pp. 153-174, April 1984.
5. B. S. Baker, E. Grosse, and C. S. Raferty, ''Nonobtuse Triangulation of Polygons,'' *Disrete Comput. Geom.*, vol. 3, pp. 147-168, 1988.
6. Watson, D.F., ''to Voronoi polytopes,'' *The Computer Journal*, vol. 24, pp. 167-172, 1981.
7. Dwyer, R.A., ''Delaunay triangulations,'' *Algorithmica*, vol. 2, pp. 137-151, 1987.
8. Preparata, F.P and Shamos, M.I., in *Computational Geometry*, Springer-Verlag, New York, NY., 1985.
9. Watson, D.F. and Philip, G.M., ''Survey: systematic triangulations,'' *Computer Vision, Graphics, and Image Processing*, vol. 26, pp. 217-223, 1984.
10. Mirante, A and Weingarten, N., ''triangulated irregular networks,'' *IEEE Comuters Graphics & Applications*, vol. 2, pp. 11-21, 1982.
11. Manacher, G.K. and Zobrist, A.L., ''point set approximates the Optimal triangulation,'' *Information Processing Letters*, vol. 9, pp. 31-34, 1979.
12. Christainsen, H.N. and Sederberg, T.W., ''Conversion of complex contour line definition into polygonal element mosaics,'' *Proceedings of SIGGRAPH*, pp. 187-192, 1978.
13. Dennehy, T.G. and Ganapathy, S., ''A new general triangulation method for planar contours,'' *Proceedings of SIGGRAPH*, pp. 69-74, 1982.
14. Christensen, A., ''Fitting a Triangulation to Contour Lines,'' *Proceedings of Auto-Carto 8*, pp. 57-67, Baltimore, Maryland, April 1987.
15. Scarlatos, L.L., ''A Compact Terrain Model Based On Critical Topographic Features,'' *Proceedings of Auto-Carto 9, Baltimore, MD*, pp. 146-155, April 1989.
16. Scarlatos, L.L., ''An Automated Critical Line Detector for Digital Elevation Matrices,'' *Proceedings of the 1990 ASPRS/ACSM Annual Convention*, Denver, Colorado, March 1990. to be published
17. Scarlatos, L.L., ''A Refined Triangulation Hierarchy for Multiple Levels of Terrain Detail,'' *Proceedings of the IMAGE V Conference*, Phoenix, AZ, June 1990. to be published
18. Fowler, R.J. and Little, J.J.,, ''Automatic Extraction of Irregular Network Digital Terrain Models,'' *Proceedings of SIGGRAPH '79, Chicago, Illinois,*, pp. 199-207, 1979.
19. DeFloriani, L., Falcidieno, B., Nagy, C. and Pienovi, C., ''A Hierarchical Structure for Surface Approximation,'' *Computers and Graphics, 8(2)*, pp. 183-193, 1984.
20. DeFloriani, L., ''A Pyramidal Data Structure for Triangle-based Surface Description,'' *IEEE Computer Graphics & Applications*, vol. 9, no. 2, pp. 67-78, March 1989.

Table 1

Measures of Sliveriness*					
AOI	Basis	Option 1	Option 2	Option 3	Option 4
1	32.294	5.037	5.071	6.301	6.578
2	52.487	9.864	12.107	11.074	11.107
3	35.889	6.047	5.739	6.398	5.998
4	50.682	11.619	11.164	12.581	12.854
5	56.835	15.054	8.521	14.329	8.676
6	40.932	5.461	5.578	7.376	7.437
7	51.261	4.336	5.029	6.089	7.367
8	39.925	5.873	6.112	6.805	7.153

* normalized to 1 for an equilateral triangle

Table 2

Comparison of Hierarchies				
AOI	Number of Levels*		Average Number of Children**	
	Old	New	Old	New
1	15	5	2.5	2.8
2	17	5	2.5	2.4
3	17	5	2.5	3.6
4	17	5	2.5	3.6
5	19	5	2.5	2.4
6	18	5	2.5	2.5
7	17	5	2.5	2.4
8	18	5	2.5	2.3

* number of levels specified for new algorithm
** number of children assumed to be 2.5 for old algorithm

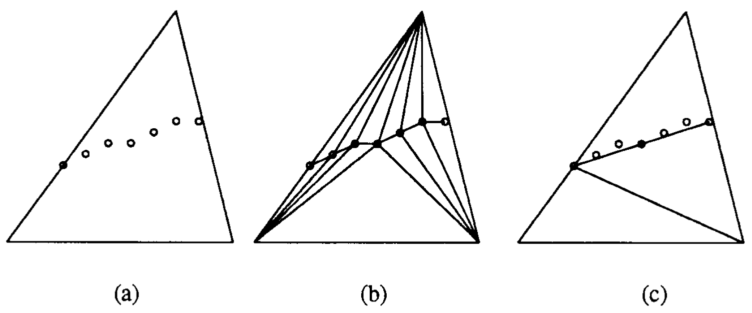
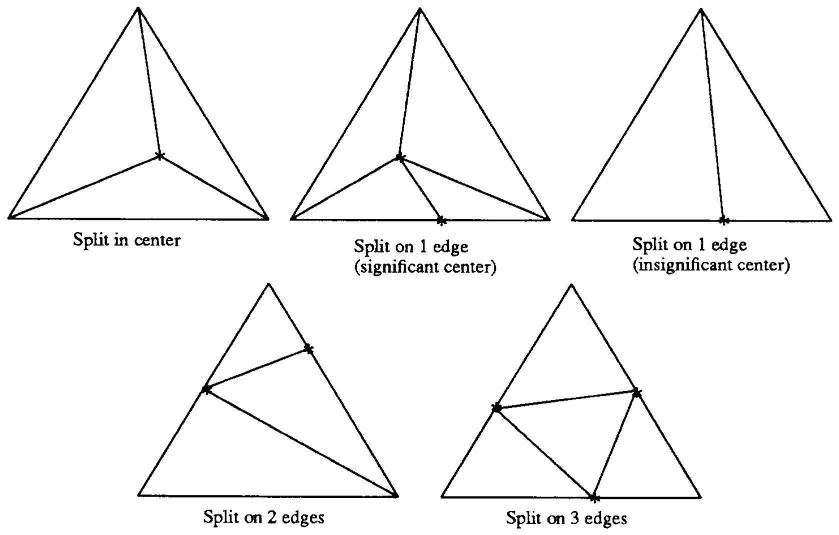


Figure 1: (a) Elevation points along a ridge intersecting the triangle, (b) results of triangulation with respect to maximum error point, (c) results of triangulation using cartographic coherence.



Split in center
Split on 1 edge (significant center)
Split on 1 edge (insignificant center)
Split on 2 edges
Split on 3 edges

Figure 2: Split strategies for preserving cartographic coherence.

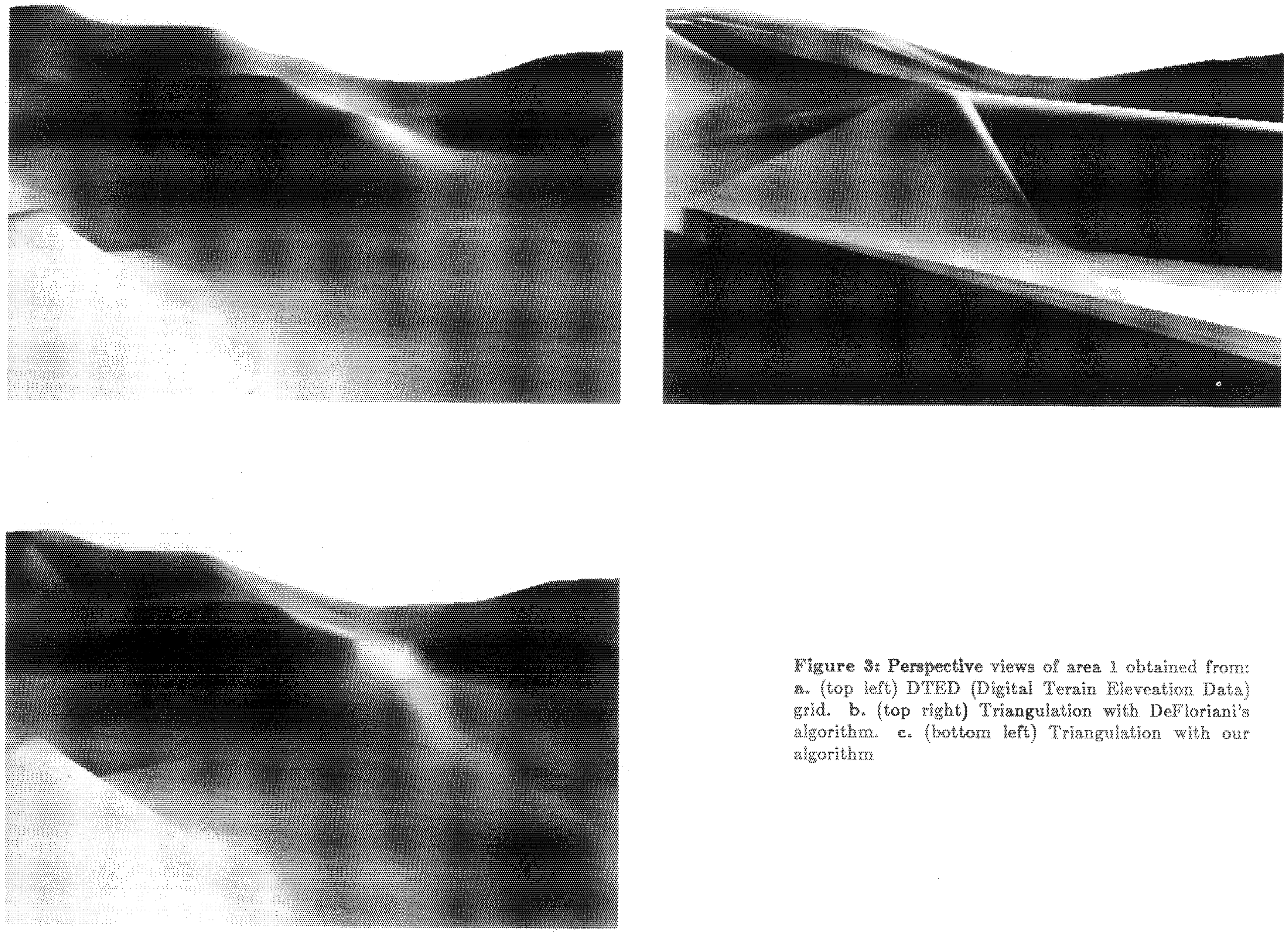


Figure 3: Perspective views of area 1 obtained from: **a.** (top left) DTED (Digital Terain Eleveation Data) grid. **b.** (top right) Triangulation with DeFloriani's algorithm. **c.** (bottom left) Triangulation with our algorithm