

Title: Superlative TINs
NTR number: 40608

Funding Disclosure

JPL Contributors

Badge #	First	M.	Last	Lab Ext	Section Number	Specific Inventive Contribution	Main Contact
105005	Robert	G	Chamberlain	48978	366	Created the concept and most of the initial designs. Important: See the comments in the "This software accomplishes..." field.	√
104940	Eric	A	Taylor	49130	366	Developed and tested code for line simplification, Faugeras bending, and Delaunay triangulation	
104960	Gary	M	Gutt	45844	366	Designed, developed and tested the code for retrieving the NIMA DEM and for richline deduction and display; made improvements in the algorithm for line simplification; developed and tested the code for 2-3 tree priority queuing, and TIN annealing; suggested the inclusion of vertex pulsing during annealing.	

Non-JPL Contributors

Software Description

This software accomplishes the following:

This software produces triangulated irregular networks (TINs) that incorporate ridgelines, channels, and shorelines whose locations have been deduced from large-scale digital elevation maps (DEMs) that consist of rectangular arrays of elevation values.

IMPORTANT COMMENTS: (Here because this form has no explicit place for comments.) Implementation was terminated before the software was released because we discovered a better way to address the problem for which the TINs were going to be used by the sponsoring project. TINs have considerable value for other applications, and the algorithm described here is definitely a new approach to their construction. It is being documented in a JPL TR and will be presented at a professional conference. It is "New Technology".

Because implementation was terminated, there are several lines of development that are incomplete. In particular, the line simplification code needs modification and retesting to incorporate the new procedures suggested by Dr. Gutt for computing the horizontal and vertical distances from raster points to the line segment. Furthermore, the designs for 2-3

trees, TIN annealing, and TIN refinement are complete, but neither the code nor the testing have been completed.

What are the unique features of the software:

Lines that are rich in information about the structure of the terrain are automatically extracted from a regular array of elevation values. Similar results have been obtained manually (rather than automatically) in the past using topographic maps (rather than digital elevation maps) as input.

A standard algorithm for simplifying rasterized lines is used, but the information contained in the rasterized representation is retained (for later use), rather than discarded. Furthermore, an improved algorithm is used for determining the horizontal and vertical distances of raster points to the simplified line segments.

An algorithm similar to that described by Faugeras to chop a set of lines into pieces from which a Delaunay triangulation of their endpoints will contain those lines as edges in the triangulation was modified to use the raster-level information about the line segments to restrict the points that are used when chopping the line segments.

A Delaunay triangulation is used to create a starting triangulation for TIN construction. Other current processes either (1) construct a Delaunay triangulation from significant points, and are not constrained to use the significant line segments as just described and consider the process complete, or (2) they start TIN construction from a pair of triangles across the entire region to be triangulated, then iteratively refine the TIN until defined stopping criteria have been satisfied. (3) Exception: Our previous algorithm uses a particular fractal pattern we have called a "Persian Rug" as the starting point for TIN refinement, rather than a pair of triangles.

Before refining the TIN, this algorithm uses a 2-3 tree priority queue with a composite quality metric based on a variety of specific quality metrics to consider all edges for "flipping" (i.e., replacement by the other diagonal of the quadrilateral that it defines) and all vertices for "pulsing" (i.e., changes in elevation). None of the "richlines" are allowed to be flipped.

The specific quality metrics that make up the composite quality metric provide a means to make concrete, quantitative statements about how well a TIN represents the information contained in a DEM. The composite metric provides an easy and unambiguous means to determine which edges are most in need of annealing and which triangles are most in need of refinement.

The optimization protocol allows the user of the algorithm to tune the construction process to make TINs that are most appropriate for his particular purpose.

After annealing, the TIN is iteratively refined by adding additional vertices and their associated edges and triangles. This is accomplished by maintaining a list of edges, prioritized by the composite quality metric. The composite metric is significantly more sophisticated than is used by other TIN construction processes.

After refinement, the TIN is annealed again.

The documentation also describes additional steps to construct seamless mixed-resolution TINs for very large regions.

The algorithms take into account the convergence of the meridians and the curvature of the Earth.

What improvements have been made over existing similar software application:

This was covered in the previous section.

Software Release Version:

Not applicable; development was terminated before the software was released.

What problems are you trying to solve in the software:

The TIN construction algorithm currently used by CBS is similar to the data-driven refinement process designed to minimize the maximum error described in [Polis et al 1996], though it uses a fractal “Persian Rug” starting pattern to facilitate tile-by-tile construction of mixed-resolution TINs for large playboxes, and splits adjacent-triangle quadrilaterals into fours of triangles instead of individual triangles into threes.

Usage revealed a variety of desired changes:

- o Low-resolution portions of the playbox were being represented at *very* low resolution, due to a RAM budget on the computer then in use of about 400 MB. In a $1^\circ \times 1^\circ$ tile, only 11 data-dependent vertices were being added to the 433 pre-located vertices of the Persian Rug. High-resolution tiles contained about 7,000 vertices. Although it is unknown how good is *good enough*, high resolution throughout the playbox would clearly be a major improvement.
- o With TINs generated to iteratively minimize the maximum error, lake and ocean shorelines tend to be “tented”, rather than sharp. Credibility requires well-defined lakes and coastlines.
- o Differences between the simulated terrain and the maps used by the training audience provide “negative training” unless those differences are comparable to the discrepancies between their maps and the real terrain.
- o When slivers (very long, very thin triangles) are artifacts of the process, they can cause anomalous results. (Sometimes, however, slivery triangles do represent the terrain best.)
- o The increasing use of small units in CBS requires better LOS assessment. Due to computing resource limitations — speed and memory, better LOS demands better TINs, not a higher density of triangles.

Meanwhile, CBS had been ported from once-grand VAX computer hardware to modern PCs with better performance and more memory, and the Shuttle Radar Topography Mission (SRTM) had produced high-resolution data with much higher quality for most of the Earth’s surface. Furthermore, NIMA, the National Imagery and Mapping Agency, has been assigned the mission of providing map products to the United States’ armed services, including the Army.

Please cut and paste the abstract describing the program (this abstract should focus on the programmatic and/or utilization aspects of the software):

The automated process described here was designed to emulate the manual process of selecting vertices and edges that characterize the topography and drawing “nice” triangles, then use digital elevation data to improve the topographic model, using accuracy in performing line-of-sight checks at the ranges and in the kinds of terrain that actually occur during exercises to guide design trades.

This process uses the new very-high-quality high-resolution SRTM data from NIMA throughout. The first step is to extract rasterized “richlines” [Douglas 1986] to describe ridges, channels, shorelines, and elevation profiles along the boundaries of the playbox. A three-dimensional version of the standard [Douglas & Peucker 1973] line simplification algorithm converts these richlines to vectors. A constrained Delaunay triangulation based on the [Faugeras 1993] algorithm makes nicely shaped triangles that use the richlines. Annealing creates non-Delaunay triangles that improve the triangulation, and result from considering prioritized lists of edges for flipping and of vertices for pulsing. More triangles are added during refinement until the budget is reached, then the edges and vertices are annealed again.

Line of sight fidelity is used by an externally applied optimization to choose process parameters, such as line simplification tolerances and priorities given to a variety of quality metrics.

Does your work relate to current or future NASA (include reimbursable) work that has value to the conduct of aeronautical and space activities? If so, please explain:

Yes. TINs provide an efficient piecewise planar approximation of a terrain surface. The TINs produced by this process should be extraordinarily efficient. In fact, the TIN construction parameters can be tuned to produce TINs particularly well suited to any well-defined purpose.

Outside Interest...

What advantages does this software have over existing software?

Already answered above.

Are there any known commercial applications? What are they? What else is currently on the market that is similar?

Yes. Hydrology. Topographic scene generation. Line of sight determination. There are many products now on the market that construct TINs.

Is anyone interested in the software? Who? Please list organization names and contact information.

No.

What are the current hardware and operating system requirements to run the software? (Platform, RAM requirement, special equipment, etc.)

The programs are written in the C language and consequently run under most operating systems. RAM must be sufficient to deal with the input data, which depends on the region for which a TIN is desired.

Software Status

What is the status of the software?

(Choices are: BETA, MATURE, PROTOTYPE)

Prototype

How has the software performed in tests? Describe further testing if planned.

Testing was not completed.

Awards

Do you want this information published in Tech Brief magazine

www.nasatech.com YES

Attachments: The TR is in process of being written

Software Available for Public Release Award

Do you believe your software is eligible? Yes

If so, please identify the customer(s) and sponsor(s) outside of your section that requested and are using your software.

U.S. Army National Simulation Center

U.S. Army Program Executive Office Simulation, Training, and Instrumentation

The algorithm is sufficiently complete to qualify for public release. Software that implements that algorithm is not that complete.

Software Dissemination

Third Party Contributions

Was this software built upon previously existing code/software? NO

If yes then, was the previously existing code/software developed at JPL and/or Campus?

If the previously existing code was not developed at JPL or Campus, please answer the following:

Name and description of the previously existing code/software:

Not applicable.

Contact information for the code/software (web site, name/email, name/phone number):

Did you accept any license terms for the previously existing code/software? NO

Has this software been disclosed (e.g. presented as an enabling flowchart) or distributed to others external to JPL or Campus? YES

If so, please identify to whom the disclosure was made and provide date, place, and manner of distribution:

It will be submitted for publication to the INFORMS journal Interfaces and, if not accepted there, published as a JPL Technical Report. It will be presented at the Annual Meeting of the Institute for Operations Research and the Management Sciences in Atlanta, GA, on October 21, 2003.

Do you need to disseminate your software? NO

Editors Comments: Roger Carlsson
 Final version attached,
 Questions and Comments

Chk	Location	Question or Comment
	Overall	Author instructions specify a maximum length of 10—15 pages. Has the 32+ length of this paper been checked with the conference editor?
	Overall	It would be good to have text references to the figures.
	Overall	Minor grammar comment: The U.S. usage (not some other countries) is to have punctuation inside the quotations.
	Title	"...Are Superfluous" in title gives the casual reader the impression that the paper is superfluous. It would be better to make the title something like "New Line-of-Sight Algorithm Renders Superlative TINs Superflous"
	page i	A couple words changed to get under the 150-word limit
	p. 5, Terrain Skins	A <i>TIN</i> , for <i>Triangulated Irregular Network</i> (Peucker et al 1976) does not match the reference at the back Peucker, Thomas K., Robert J. Fowler, James J. Little, David M. Mark. 1976. Digital representation of three-dimensional surfaces by triangulated irregular networks. <i>Technical Report No. 10</i> , Office of Naval Research (ONR) Geography Programs 1-63.
	p. 7	"...splits adjacent-triangle quadrilaterals into fourths of triangles instead of individual triangles into thirds" [Is this fourths of triangles instead of thirds?]
	p. 6, SURF	The SURF Fellow asked the right questions: [Do we need this line?]
	p. 7	The observed
	p. 7	"...paper, by Gutt & Chamberlain (2003) is in preparation" JPL won't allow use of an in-preparation paper as a reference—it can be a footnote.
	p. 8	Peucker 1976 referenced here may be different from the earlier one.
	p. 8	Gutt and Chamberlain 2003 cannot be a reference until it is at least "in press." We could quickly slap together an internal document (D-series). That would not be easily accessible to readers, but it would be legit.
	p. 10, Fig. 4	NIMA?
	p. 10, Fig. 4	Ridgelines and richlines are used in near each other. Just to be sure, these are separate terms are they not?
	p. 14, Richlines	Gutt & Chamberlain (2003) again—not yet a document.
	p. 16	"...associated endball are "chopped" by the endball ."

New Line of Sight Algorithm Renders Superlative TINs Superfluous

Robert G. Chamberlain

email: <rgc@jpl.nasa.gov>

Corps Battle Simulation Project
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Joseph E. Gonzalez

email: <josephg@caltech.edu>

Summer Undergraduate Research Fellow
California Institute of Technology
Pasadena, California

Gary M. Gutt

email: <Gary.M.Gutt@jpl.nasa.gov>

Senior Member of the Technical Staff
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Eric A. Tailor

email: <Eric.A.Tailor@jpl.nasa.gov>

Corps Battle Simulation Project
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

Abstract

Assessment of geometric line of sight (LOS) is the most important purpose of the topography model in the Corps Battle Simulation (CBS), a constructive combat simulation program that supports training exercises for senior commanders and staff officers of the U.S. Army. CBS currently uses a triangulated irregular network (TIN) to model topography. LOS is obstructed if the sensor-target line passes below any TIN edge.

First, we describe a sophisticated new process to use the very-high-quality, high-resolution Shuttle Radar Topography Mission raster elevation data to construct “superlative” TINs.

The need to optimize the free parameters in this process led to a Summer Undergraduate Research Fellowship project to produce an optimization protocol and algorithm. Surprisingly, the search for an appropriate optimization metric revealed a new, elegant algorithm for evaluating LOS.

In part 2, we describe a ray-tracing-inspired quadtree-based approach to LOS assessment that uses the raster elevation data directly and very efficiently.

Key words

CBS (Corps Battle Simulation project), constructive combat simulation, constrained Delaunay triangulation, DEM (digital elevation map *or* digital elevation model), LOS (line of sight), lossless data compression, optimization, OR (operations research) practice, quadtrees, ray tracing, richlines, SRTM (Shuttle Radar Topography Mission), SURF (Summer Undergraduate Research Fellowship), TIN (triangulated irregular network), terrain model, 2-3 trees.

A Breakthrough

Terrain Skins in the Corps Battle Simulation (CBS)

CBS is the U.S. Army's premier computer-based constructive combat simulation program. It has been used as a part of the "final exam" for *all* graduates of the U.S. Army's Command and General Staff College since 1987. Even more, it is used for continuing training in large-scale, worldwide, joint, corps, and division combat exercises. Training audiences consist of two-star and three-star generals and their staffs.

A *TIN*, for Triangulated Irregular Network, is a piecewise planar model of the configuration of terrain. The seminal reference is Peucker et al. (1976); Scarlatos (1993) gives an excellent discussion of the theoretical and practical issues involved in their construction. TINs are also used in other contexts to model three-dimensional shapes.

Since 1997, the CBS terrain model has used TINs to model the topography of the large playboxes used by CBS. TINs were selected because they could be constructed at a level of approximation dictated by available computer memory. Other approaches, such as digitized contour lines, were considered, but rejected as insufficiently efficient.

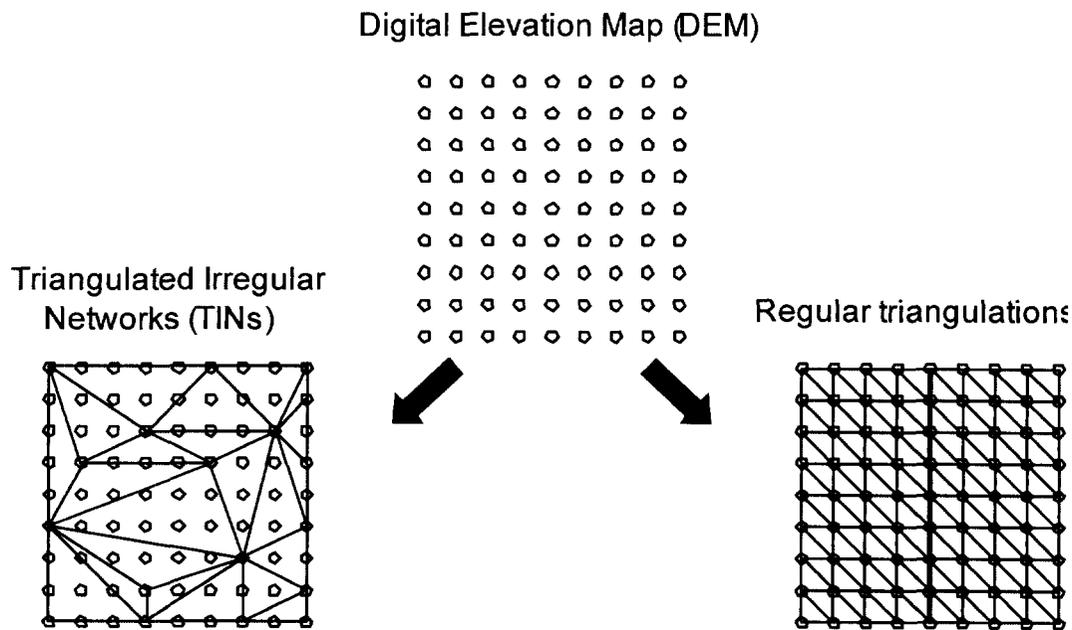


Figure 1. Regular arrays of elevation data are treated as ground truth, then used to define a terrain skin that is a piecewise planar approximation to the surface. Higher-order interpolations are possible, but there is no reason to assume they describe the terrain any better. Individual TIN triangles usually represent many more DEM posts than this figure suggests.

The diagonals in regular triangulations can go in either direction. To avoid the asymmetry that results from either choice, an alternative is to use as a terrain skin the four triangles in each DEM cell whose interior edges extend from the corners to the center, with the elevation of that center point chosen to be the average of the elevations of the corners. This terrain skin model is a planar version of a bi-linear fit to the four corners.

State of the Art of LOS Assessment in Military Simulations

Assessment of geometric line of sight (LOS) queries is the most important purpose of the topography model in CBS. The only other purpose is to estimate slopes along a route for use in the mobility model. This is accomplished by differencing elevations obtained from the TIN. This procedure has the benefit that it converts the TIN's sharp creases, which are preferred for LOS assessment, to rounded edges for use in the mobility model.

There are two major LOS algorithms in current use in military simulations. There are other algorithms, some of which are discussed by Richbourg et al. (2001). Periodic sampling, sometimes known as the "telephone pole algorithm", compares the height of the LOS to the height of the surface model at uniformly spaced steps; LOS is obstructed if the LOS is below the surface at any step. This algorithm has the advantage that computation is easily reduced (at the expense of accuracy) by increasing the length of the step. It can incorrectly indicate that LOS is not obstructed in terrain that is well suited to ambushes; to mitigate that problem, the steps must be close together. Thus, it is often either too inaccurate or too slow. Furthermore, a naïve implementation can give asymmetric results.

The other major LOS algorithm traverses the edges of a piecewise planar model of the surface, and is generally considered to be as accurate as is justified by the model of the topography. In this algorithm, LOS is assessed by comparing a line between the sensor and the target to each edge it crosses, stopping the first time the line is below the edge. On an efficient TIN, this algorithm is fast and accurate. This algorithm can be used with a regular triangulation of a DEM, but the large number of edge crossings to be considered usually makes it prohibitively slow.

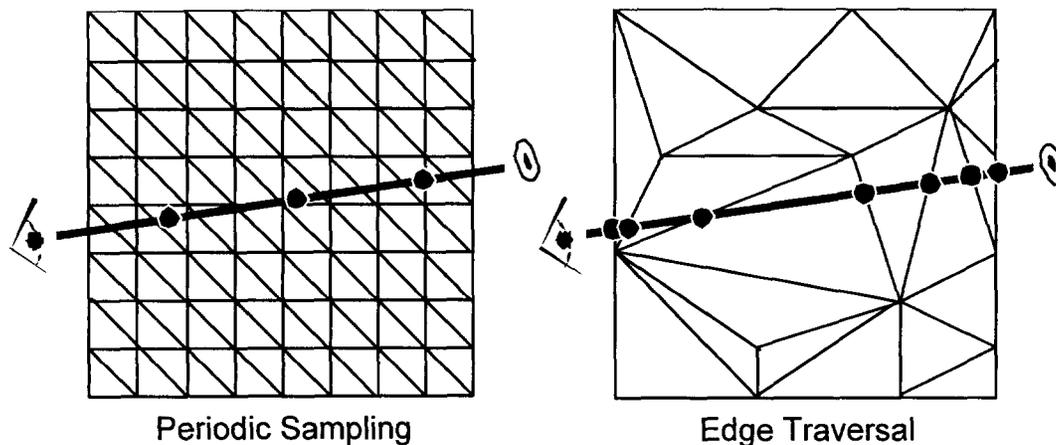


Figure 2: Periodic sampling of the relative elevations along the sensor-target line of sight provides computational flexibility, but it can miss just those features that are important for ambushes. Edge traversal takes advantage of efficient TINs but is computationally expensive on inefficient triangulations. (Either algorithm can be used with either kind of terrain skin.)

TIN Construction in CBS

The TIN construction algorithm currently used by CBS is similar to the data-driven refinement process designed to minimize the maximum error described by Polis et al. (1996), though it uses a fractal “Persian Rug” starting pattern to facilitate tile-by-tile construction of mixed-resolution TINs for large playboxes, and it splits an adjacent-triangle quadrilateral into four triangles instead of an individual triangle into three.

Usage revealed a variety of desired changes:

- Low-resolution portions of the playbox were being represented at *very* low resolution, due to a RAM budget on the computer then in use of about 400 MB. In a $1^\circ \times 1^\circ$ tile, only 11 data-dependent vertices were being added to the 433 pre-located vertices of the Persian Rug. In contrast, high-resolution tiles contained about 7,000 vertices. Although it is unknown how good is *good enough*, high resolution throughout the playbox would clearly be a major improvement.
- With TINs generated to iteratively minimize the maximum error, lake and ocean shorelines tend to be “tented”, rather than sharp. Credibility requires well-defined lakes and coastlines.

- Differences between the simulated terrain and the maps used by the training audience provide “negative training” unless those differences are comparable to the discrepancies between their maps and the real terrain.
- When slivers (very long, very thin triangles) are artifacts of the process, they can cause anomalous results. (Sometimes, however, slivery triangles *do* represent the terrain best.)
- The increasing use of small units in CBS requires better LOS assessment. Due to computing resource limitations — speed and memory, better LOS demands better TINs, not a higher density of triangles.

Meanwhile, CBS had been ported from once-grand VAX computer hardware to modern PCs with better performance and more memory, and the Shuttle Radar Topography Mission (SRTM) had produced high-resolution data with much higher quality for most of the Earth’s surface. Furthermore, NIMA, the National Imagery and Mapping Agency, has been assigned the mission of providing consistent map products to the United States’ armed services, including the Army.

The SURF Project

The time was right to implement the improved TIN construction process that had been under consideration for seven years. It was designed to address all of the issues noted above and would, it was hoped, produce “superlative” TINs.

The need to optimize the process’s free parameters led to a ten-week SURF (Summer Undergraduate Research Fellowship) project for which the objective was to develop, implement, and, if time allowed, apply an optimization protocol.

The first step in the SURF project was to construct a way to compare the quality of TINs produced by competing sets of parameters. The primary use of the TIN in CBS is to determine whether there is an unblocked line of sight between a sensor and a target. (Accurate determination of elevation for slope determination, the only other use of the TIN, is much less critical to operations.) The best metric, then, would compare how accurately a TIN answers line-of-sight queries. Using actual queries extracted from a CBS exercise would bias the statistical sample space to the ranges and types of terrain actually experienced during the exercise — a desirable bias. Ideally, sets of queries from several playboxes would be used to tune the optimal values of the parameters, which would be checked for robustness against additional sets not used in the tuning.

The obvious question is, “If a regular triangulation of the SRTM-based DEM is the touchstone, why not just use one?” The answer, based on the current state of the art, is that the data would occupy too much memory and dealing with all those edge traversals would require too much computation.

The SURF Fellow observed that the combat-simulation line-of-sight problem has much in common with the ray-tracing problem in computer graphics and suggested using a very fast octree-based “trivial rejection” approach to the line-of-sight problem.

Because the bottoms of the bounding boxes are all at the center of the Earth, we realized that the appropriate data structure was quadtrees, rather than octrees. Had we not had this insight, the

memory requirements implied by 10,000 or so vertical levels in the octrees would have been truly overwhelming. Even with a quadtree data structure, memory is a major issue.

The triangulation can be implicit, so pointers are not needed to find adjacent triangles, as they are in TINs. The quadtree structure can also be implicit, so it requires very little additional data storage. Furthermore, the quadtree structure facilitates data compression. Consequently, it appeared that all the DEM data for DTED1 resolution would fit in the RAM allocated to TINs. That assessment turned out to be a bit optimistic, but a much better alternative was soon identified: Run the quadtree LOS algorithm as a second process. A TCP/IP approach to interprocess communication demonstrated feasibility, producing speeds similar to those of edge traversal on existing TINs. In an operating system with 32-bit address space, implementation as a second process increased the address space available by an order of magnitude (to about 3 GB) while allowing the previous allocation (about 350 MB) to be used for other purposes by the simulation. We anticipate being able to use existing hard disk space with an operating-system-provided RAM cache (to provide sufficient performance) to store the quadtrees for even the largest playbox.

A New Algorithm for LOS

The result is a ray-tracing-inspired quadtree-based approach to line-of-sight assessment that makes direct use of the raster elevation data.

CBS is an engineering project, not an academic one. Since TINs are used in CBS only to assess LOS and to determine spot elevations, the new TIN construction process was tabled when the breakthrough occurred, even though implementation was about 80% complete. It is documented here to provide a beachhead for further development. .

Part 1: How to Construct Superlative TINs

State of the Art of TIN Construction for Military Simulations

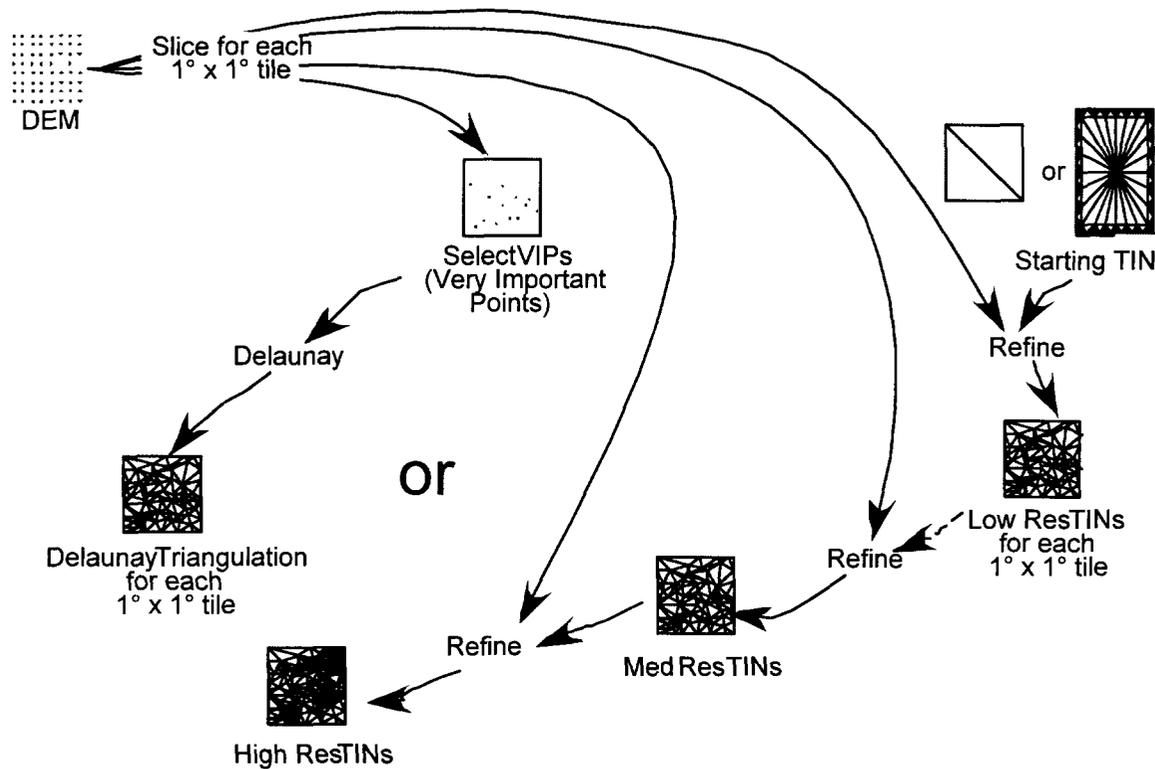


Figure 3: State-of-the-art TINs for military simulations are either Delaunay triangulations of carefully selected DEM posts or data-driven refinements of a simple initial starting pattern. Delaunay triangulations have nicely shaped triangles but may contain egregious errors. Highly refined triangulations may contain large numbers of poorly shaped triangles (slivers).

As suggested by the above figure, almost all automated TIN construction processes start with a regular array of elevation data, usually referred to as a DEM, which stands for *digital elevation map* (or, sometimes, *digital elevation model*). Most of these processes would also work with an irregular array of elevation data, though refinement processes would not be able to assume that error integrals can be approximated by simple summations over data points.

Early TIN builders (see, for example, the seminal work by Peucker et al. in 1976 and 1978), manually inspected paper maps to find peaks, passes, pits, ridges, and channels. Triangulation was then a matter of adding lines that, to the extent possible, neither jumped across valleys nor pierced ridgelines. This procedure leads to very efficient TINs, but is unacceptably labor-intensive for large regions.

Probably the most prevalent approach in current use is typified by that used in ArcInfo, a widely available product of the Environmental Systems Research Institute (ESRI 1989). In this approach, an algorithm of some sort inspects the DEM for “very important points”, which are then connected by a Delaunay triangulation. Zeiler (1999) describes some additional steps performed in the ArcInfo algorithm to insert user-designated breaklines (some carrying elevation data and some not). The ESRI implementation allows the user to define excluded areas (such as inside lakes and outside project boundaries) that are not triangulated. Use of that feature would preclude line of sight checks across excluded areas. In the CBS context, having excluded areas would also preclude eventual interplay with naval and air models.

In the other major technique, an initial TIN consisting of a pair of very large triangles defined by the corners of the playbox is refined by repeatedly splitting its “worst” triangle. In most procedures, such as those described by Fowler & Little (1979) and Polis et al. (1996), the “worst” triangle is identified as the one containing the most poorly fit point. In Fowler & Little, the new point is used to construct a new constrained Delaunay triangulation. In Polis et al., the triangle is split, and local annealing (edge flipping) is used to reduce slivers. Refinement continues until some stopping criterion — usually reaching the triangle budget, but sometimes reducing the worst deviation below a specified goal — is reached. The technique currently used for CBS is very similar to that of Polis et al., but the initial TIN for each tile has a “high-resolution edge” so mixed-resolution TINs for large playboxes can be constructed by simply concatenating TIN files for adjacent geotiles.

Overview of the Improved TIN Construction Process

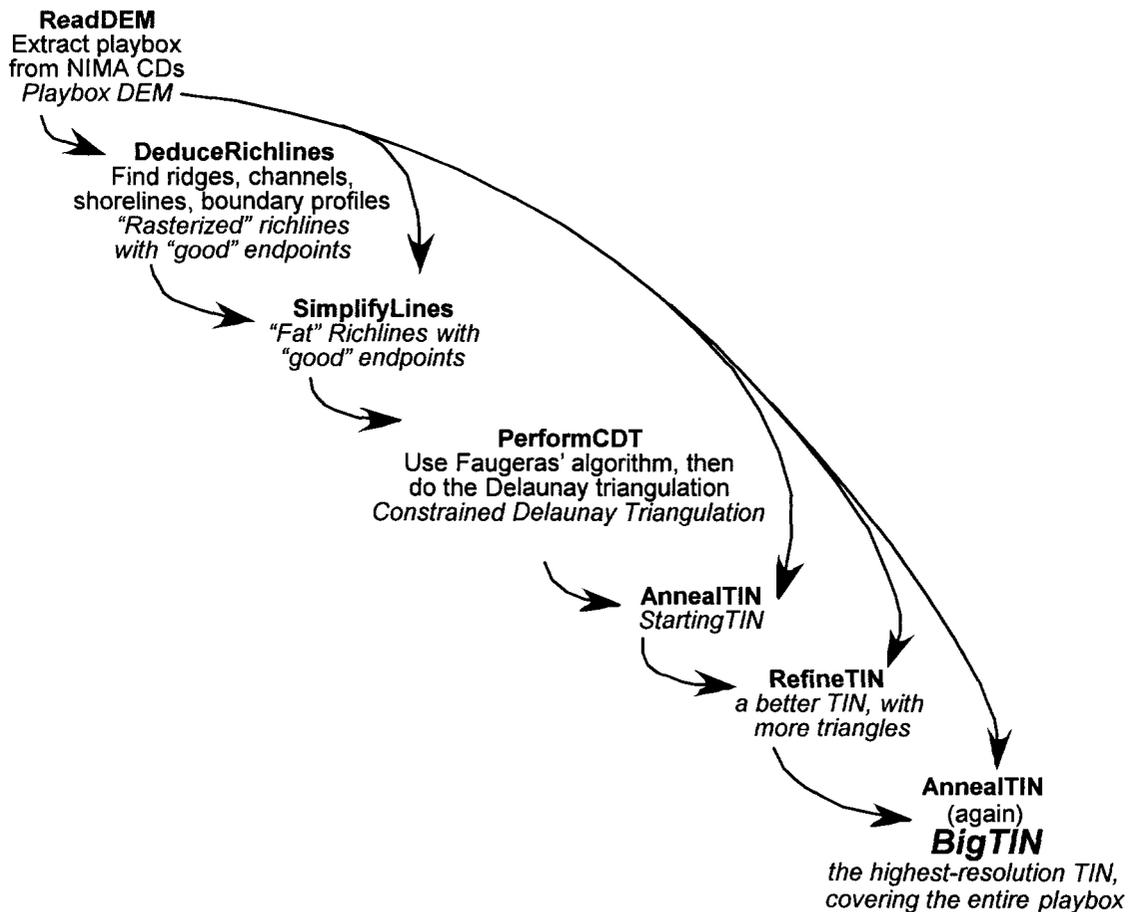


Figure 4: The improved TIN construction process, which we hoped would produce “superlative” TINs, mimics the manual process by identifying important linear features (“richlines”), uses a constrained Delaunay triangulation to start with nicely shaped triangles, removes the egregious anomalies, then uses the extensive digital data and an optimized collection of quality metrics to refine the triangulation. The final step removes anomalies that may have been introduced during the refinement.

The new automated process described here was designed to emulate the manual process of selecting vertices and edges that characterize the topography and drawing “nice” triangles, then use digital elevation data to improve the topographic model, using accuracy in performing line-of-sight checks at the ranges and in the kinds of terrain that actually occur during exercises to guide design trades.

This process is illustrated in the figure above. It uses the new very-high-quality high-resolution SRTM data from NIMA throughout. The first step is to extract rasterized “richlines” (Douglas 1986) to describe ridges, channels, shorelines, and elevation profiles along the

boundaries of the playbox. A three-dimensional version of Douglas & Peucker's (1973) standard line simplification algorithm converts these richlines to vectors. A constrained Delaunay triangulation based on Faugeras' (1993) algorithm makes nicely shaped triangles that use the richlines. Annealing creates non-Delaunay triangles that improve the triangulation. These triangles result from considering prioritized lists of edges for flipping and of vertices for pulsing. More triangles are added during refinement until the budget is reached. Then the edges and vertices are annealed again.

Line of sight fidelity is used by an externally applied optimization to choose process parameters, such as line simplification tolerances and the priorities given to a variety of quality metrics.

If the high-resolution *BigTIN* is too big for the application to handle, a mixed-resolution TIN could be constructed from some combination of *Low-*, *Medium-*, and *High-Resolution TINs* for smaller tiles, as is currently done in CBS. Additional modules, not described in this document, would be needed: *ExtractRichLines*, *SliceBigTIN*, and *ExtractBoundaryTriangles*. The modules *PerformCDT*, *AnnealTIN*, and *RefineTIN* would be applied to each tile; *AnnealTIN* will be applied both before and after *RefineTIN*.

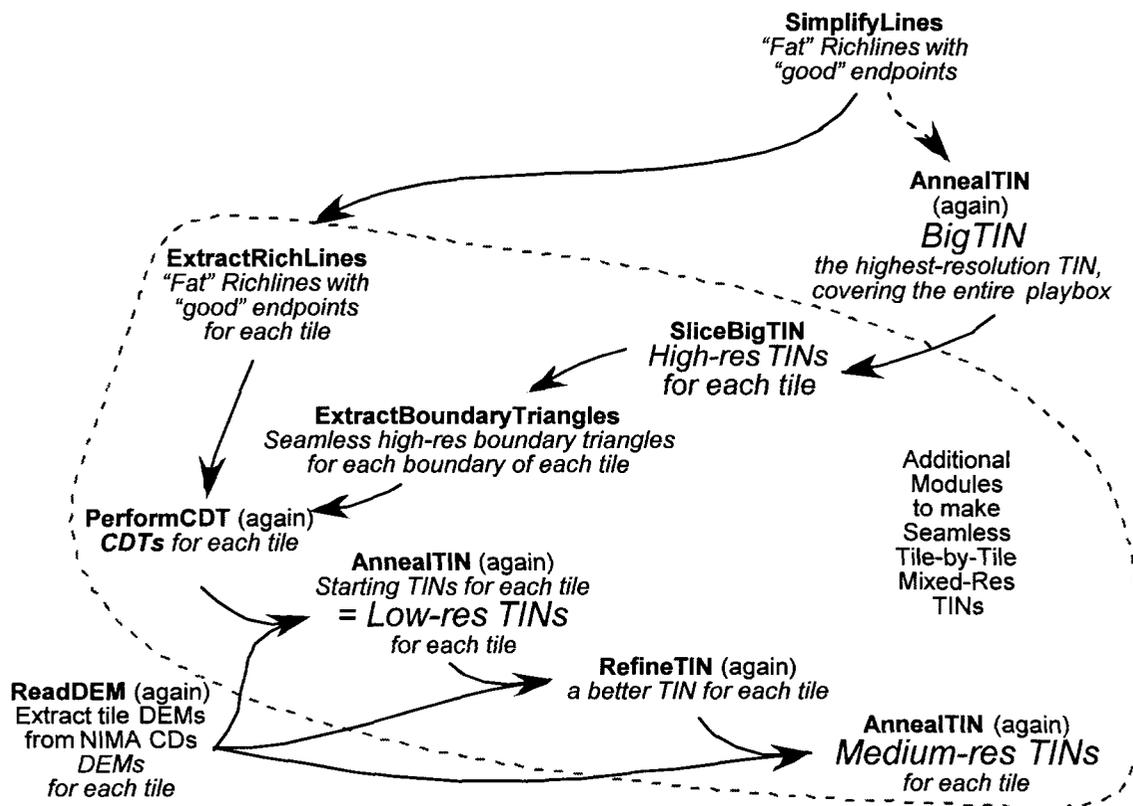


Figure 5. If the application is not able to use the high-resolution TIN of the entire playbox, additional steps may be needed to create TINs at different resolutions for smaller pieces of the playbox. These pieces should fit together seamlessly so players are not able to gain unrealistic advantages by using artifacts of the model.

To obtain seamless edges across tile boundaries, the *BigTIN* is sliced along tile boundaries so triangles that cross the boundaries are replaced by a larger number of co-planar triangles that do not cross the boundaries. The result is a set of tile-by-tile *High-Resolution TINs*. Next, sets of lower-resolution TINs that can be seamlessly abutted by concatenating their data files are obtained by using the triangles along each tile's *High-Resolution TIN* boundaries and the tile's richlines to perform CDTs (constrained Delaunay triangulations) for each tile. Extensive annealing makes these the tile-by-tile *Low-Resolution TINs*. These TINs are then refined and annealed by the same code as is used to produce the *BigTIN* to produce tile-by-tile *Medium-Resolution TINs*.

Richlines

The DEM implicitly contains channels, ridges, and shorelines. This algorithm examines the DEM data to deduce lines that define the locations of these features—and playbox boundaries, as well. These lines consist of adjacent DEM locations. The line simplification process that will convert these “rasterized” lines to vectors always keeps the endpoints of line segments, so it is important to find “good” endpoints for the line segments. The point at which a shoreline intersects a boundary line is an example of a “good” endpoint, both for the shoreline segment and for two elevation profile segments. This will ensure that “smart boundary” endpoints will connect with the endpoints of other richlines.

The algorithm described here was inspired by Toma et al. (2002), a website that gives a very brief description (and a link from which the source code can be downloaded) of Duke University’s TerraFlow, “a software package for computing flow routing and flow accumulation on massive grid-based terrain.” Their algorithm, formally documented in Arge et al. (2003), is based on the work by Jenson (1985) and Jenson & Domingue (1988).

The algorithm we finally implemented was also influenced by Peucker & Douglas (1975) and Band (1986). It uses an inverted channel-finding approach to find ridges. It does not attempt to find the watersheds drained by channels, nor does it attempt to find the divides that are the boundaries of the watersheds. Since hydrography is not the issue, spurious sinks and gaps in channels or ridges (which will likely be filled by the triangulation) are not a problem.

Another algorithm, described in Little & Shi (2001), from the University of British Columbia, fits a higher-order surface to the DEM, then uses derivatives to identify creases. They call the creases that are convex upward *p-lines*, those that are convex downward *n-lines*. The *p* lines and *n* lines could augment the other richlines.

Douglas-Peucker Line Simplification

Douglas-Peucker line simplification (Douglas & Peucker 1973) is a heuristic method of finding a line with a small number of segments that fits a sequence of points within specified maximum errors. Figure 4 in Whyatt & Wade (1988), p 21, gives a clear recursive illustration of how the algorithm works. Elevation profiles have vertical deviations but no horizontal deviation, while shorelines have horizontal deviations but no vertical deviation. Ridges and channels have both vertical and horizontal deviations, so a three-dimensional version is used. Since horizontal and vertical errors have different significance, they can have different error bounds.

In most applications, the data from which the lines were simplified is discarded, but this situation is unusual. During preparatory processing for constrained Delaunay triangulation and at each step of the TIN refinement process, edges (that is, segments of a simplified richline) are often replaced by two shorter line segments derived from the line’s unsimplified version. This extra information is stored with the simplified line segments (but only during TIN construction). Due to their conceptual nature, they may be thought of as “fat” line segments.

Line simplification is performed on all of the deduced richlines to convert rasterized lines to vectors. It is also used during the deduction of richlines to find channel mouths and ridgeline endpoints.

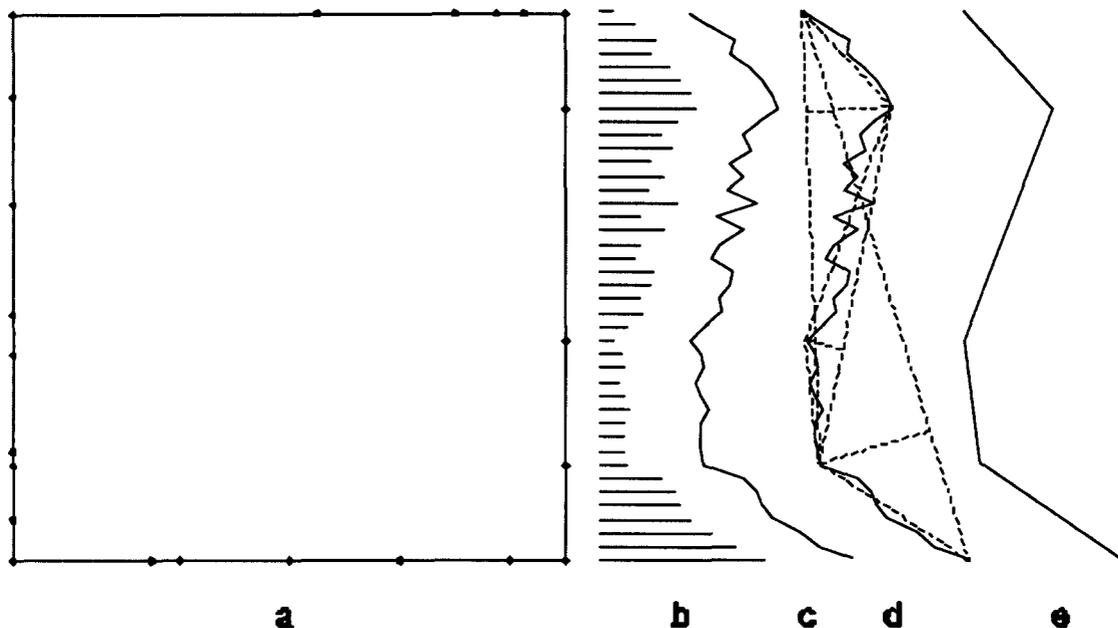


Figure 6. Douglas-Peucker line simplification is applied to the richlines, including the elevation profiles of the boundaries (a) of the playbox. The DEM posts in (b) are connected by short line segments in (c), simplified as in (d), with the result shown in (e). The process is iterative: The first approximation is the line connecting the endpoints. If all of the intermediate points are within tolerance, the process terminates. If not, the segment is bent into two pieces at the location of the largest error, and the procedure is applied to each piece. When it terminates, none of the original points is farther from the simplified line than the specified tolerance. Our algorithm remembers the unsimplified version of the line because the TIN might be improved by shortening some of the longer segments.

Constrained Delaunay Triangulation

To summarize: Richlines consist of sequences of adjacent DEM posts, some of which are marked as endpoints of line segments. These “rasterized” lines are converted to vectors by application of the line simplification algorithm, then the segments are extracted and made into edges for use in the triangulation.

Lawson (1977) showed that the dual of a Delaunay triangulation is a Voronoi polygon. In particular, the vertices of a Delaunay triangulation define the regions of an associated Voronoi polygon. The boundaries of regions in a Voronoi polygon enclose all points that are closer to their Delaunay vertices than to any other. Thus, in this sense, the endpoints of edges in a Delaunay triangulation are nearest neighbors. Put another way, if two vertices are so far apart that their Voronoi polygons do not share an edge, the line between them cannot be an edge in the Delaunay triangulation.

Thus, some of the richline segments are too long to be edges in a Delaunay triangulation of their endpoints. These edges can be forced to be in the triangulation, as was done by Chew (1989) and by Heller (1990), but the result is no longer a Delaunay triangulation, and it is likely to have extremely slivery triangles adjacent to the too-long edges.

A better algorithm, adapted from Faugeras (1993), is used to chop those too-long edges into pieces that will be Delaunay edges. Lee & Schachter (1980) give a somewhat similar approach, but theirs assumes that the set of line segments is sparse. Since the line segment vectors represent sequences of DEM posts in our implementation, it is those sequences, rather than the geometrical line segments, that are chopped.

Faugeras chopping. Another property of every triangle in a Delaunay triangulation (Lawson 1977) is that there are no vertices in the interior of the circumscribing circle through its vertices. Faugeras' algorithm uses this property by trimming segments so that there are no vertices inside the circles that have the edges as their diameters.

Our adaptation of Faugeras' algorithm cycles through a list of richline segments — and the vertices that are their endpoints — and selects intermediate points along those segments to be new vertices. Each segment in that list is inspected. If the segment has already been marked as being a Delaunay edge, the loop proceeds to the next segment. Otherwise, three temporary lists are made. *Adjacent* edges are those that share an endpoint with the segment. There are two lists, one for each end. The *neighbors* of the segment are those other edges for which any point is inside the circle for which the segment is the diameter.

If the segment has neither neighbors nor adjacent edges at either end, it is marked as a “Delaunay edge”, and processing proceeds to the next segment in the list.

If there are neighbors and/or adjacent edges, *end balls* are created for each end of the segment. An *end ball* is an open circle (Faugeras' context was three-dimensional) whose center is on the segment, whose circumference passes through the associated segment end, and whose radius is such that there are neither neighbors nor endpoints of adjacent edges inside the circle.

All adjacent edges that intersect the associated end ball are “chopped” by the end ball. In our adaptation, that means we select a point from the unsimplified version of the edge that is as close to the circumference as possible but still inside the end ball. The adjacent edge is made into two segments. The part that is inside the end ball is marked as “Delaunay”. The other part is put at the end of the list of segments for subsequent processing.

The segment itself (the one for which the end balls were constructed) is also chopped. If the end balls intersect each other, an intermediate point from the portion of the segment that is inside both end balls is selected as the division point, and both pieces are marked as “Delaunay”. If they do not intersect each other, the segment is chopped by the end balls into three pieces, instead of two. The two end pieces are marked as “Delaunay”. The center piece is put at the end of the list for subsequent processing.

Processing continues until the last segment in the list (which has been growing as the algorithm proceeds) has been considered. At that point, a Delaunay triangulation of the endpoints of the segments, all of which have been marked “Delaunay”, will contain all of the segments as edges in the triangulation. (It will probably contain many other edges as well.)

Delaunay triangulation. There are algorithms for constructing Delaunay triangulations with computational complexity ranging from $O(n \log n)$ to $O(n^2)$ to $O(n^3)$. A textbook reference is Preparata & Shamos (1985); their reference is to Lee & Schachter (1980), which contains clear descriptions of two algorithms with pseudocode.

We implemented the $O(n^2)$ algorithm described in Faugeras (1993), but tabled development before determining whether the size of our problem makes a more complicated $O(n \log n)$ algorithm essential — which it probably does.

Some of the many other references that may be useful are Chew (1989), Guibas & Stolfi (1985), Heller (1990), and Shewchuk (1996).

Annealing

Richline locations are based on the topographic information contained in the DEM and are used to construct the constrained Delaunay triangulation (CDT). Since the vertices used in the CDT are drawn from the richlines, most of the new edges can be expected to connect vertices on channels to vertices on ridges. The construction process, however, is strictly geometric, adding edges to create nicely shaped triangles without direct consideration of the elevation data. Inevitably, using *some* non-Delaunay triangles would describe the terrain better.

Iteratively consider each edge as the diagonal of the quadrilateral composed of the two triangles that share that edge. Replace the edge by the other diagonal of the quadrilateral if doing so improves the TIN. (See Appendix A, “Quality Metrics”.) Of course, neither richlines nor the diagonals of non-convex quadrilaterals are eligible for this “flipping” process.

The term *annealing* comes from recognizing the analogy to the process of relieving stresses in a forged piece of metal.

A second annealing process attempts to improve the TIN by considering variations in the elevations of the TIN vertices, each of which is located at a DEM post and (initially) has the DEM elevation. In addition to allowing vertex elevations to represent the elevations of nearby points, *pulsing* the vertices should mitigate the effects of any spurious local pits and peaks due to noise in the DEM elevation data.

The number of triangles in the TIN is not changed by either annealing process, nor is the number of vertices or the number of edges.

After the CDT has been annealed, it is *refined* by adding vertices and edges. TIN refinement is an iteratively applied process. After new triangles are created, their edges are considered for flipping. One of the major benefits of edge flipping during refinement is a reduction in the frequency of poorly shaped triangles — but only the edges of the new triangles are considered at this time; possible propagation of stress relief is ignored. Hence, global annealing is also applied after refinement.

The annealing algorithm. The first stage of the annealing algorithm consists of maintaining a list of TIN edges sorted by the value of the composite quality metric, then considering each edge in that list as a candidate for flipping. Sorting is accomplished by use of a 2-3 tree as described

by Aho et al. (1976), which allows search, insertion, and deletion to be accomplished with $O(\log n)$ operations.

If the metric for the flipped edge would be smaller than the metric for the edge being considered, the flip is performed and the sorted list is updated. The edges that originated as richlines are *unflippable* and do not appear on the list. After an edge is flipped (or not), it is put on a tabu list. It is removed from the tabu list when either of its triangles is changed as a result of flipping one of its other edges.

The second stage is similar, except that vertex elevations are considered instead. After the vertex metric is improved by changing the elevation of the “worst” vertex, the sorted vertex list is updated. A tabu list is also used for vertices.

In principle, annealing continues until all edges and vertices are on the tabu lists. Practicality, however, may dictate that computation continue only until a computing budget is reached.

Refinement

The annealed constrained Delaunay triangulation, which is input to this module, contains information from the richlines and has nicely shaped triangles everywhere except where non-Delaunay triangles fit the topography better. There may be significant topographic features between the ridges, channels, and shorelines, so this module uses DEM data to split these triangles into smaller ones, using a procedure inspired by Polis et al. (1995 and 1996). Starting with two triangles for the entire playbox, their approach is to find the triangle with the largest deviation, split it into three triangles with a new vertex at the location of the largest deviation, then consider each of the edges of the original triangle for flipping. They iterate until they reach any of several goals or exceed a triangle budget.

In our 1996 implementation by Mark A. Kordon, we modified their approach for CBS by introducing a starting pattern that resembles a Persian rug (see the upper right of Figure 3) so that TINs for adjacent geotiles, possibly at different resolutions, could be abutted by concatenating their data files. Kordon found that many fewer slivery triangles would be produced by splitting the quadrilateral formed by the worst triangle and the adjacent triangle nearest the worst point into four triangles, instead of splitting the worst triangle itself into three. After splitting, the untouched edges of the original quadrilateral are locally annealed — that is, flipped if doing so reduces the maximum deviation. Kordon’s innovation is equivalent to using Polis’s approach, then always flipping the edge nearest the most deviant point and proceeding to check the two farthest edges of that edge’s quadrangle.

The design described here uses a much better starting TIN and measures several characteristics of the fit to the topography. The refinement algorithm focuses on edges, rather than triangles. All non-boundary edges have a triangle on each side, thereby implying a quadrilateral. Non-convex quadrilaterals and boundary edges are treated as special cases in which triangles, rather than quadrangles, are split. An external evaluation process based on line-of-sight queries at ranges and in the kinds of terrain that are typical of line-of-sight queries in CBS exercises is used to find optimal weighting factors (and other parameter values) for combining the various quality metrics into a composite metric. Due to the discovery of the

quadtree algorithm for line-of-sight assessment, however, this optimization process was never actually performed.

When a quadrilateral is split, four new triangles replace two old ones. When the edge is *not* a constrained richline, the edge is replaced by two new edges from the edge's original endpoint vertices to the quadrilateral's most deviant point, which becomes a new vertex. The other two new edges are from the two triangles' "other" vertices to the same new vertex. When the edge *is* a constrained richline, the new vertex is the most deviant point *taken from the unsimplified "interior" posts associated with that richline edge*. Thus, the richline-derived edges continue to be constrained to be in the TIN, and they are still simplified, but they are shorter and less simplified.

The refinement algorithm. The logic for refinement is very similar to that for annealing. The quality metrics for triangles and edges are essentially the same. Vertex metrics are not used.

Stopping criteria are similar, except that an edge (or triangle) budget must be introduced. The relation between TIN quality and the number of triangles is the fundamental determinant of the efficiency of the TIN construction process.

Optimal values of the weights of the metrics and the value of the composite goal are likely to be similar for both of the annealing steps and for the refinement step. Whether it is important to keep them as separate variables is another part of the optimization study that was not pursued.

During annealing, the list of edges to be considered for flipping contains only the added edges and none of the richlines. In refinement, all edges must be considered, as the action that is taken is not replacement of the edge by the other diagonal, but splitting of the associated quadrilateral.

When the quadrilateral associated with a richline edge is selected for splitting into four triangles, the new edges are constructed by bending the richline, rather than by discarding the existing edge and constructing new edges that connect the most deviant point to the four corners. This is accomplished by using the most deviant point *along that edge's unsimplified version* instead. Afterward, the two shorter pieces of the richline are still identified as the same kind of richline.

Non-convex quadrangles and boundary triangles are treated as special cases.

Transition: The Parameter Optimization Project

Parameters

Under normal circumstances, input information required from the user who wants to construct a TIN will be minimal. A digital elevation map of a playbox is defined by the coordinates of a corner, the uniform angular spacing between DEM posts in longitude and in latitude (each of which is constant throughout the playbox, but the two values are not necessarily the same), and the number of posts in each direction. Detailed software specifications include file naming conventions and file locations.

In addition, several free parameters control the operation of the process, such as the horizontal and vertical tolerances for line simplifications, the number of annealing iterations allowed, and the triangle budget. Details are given in Appendix B. The initial goal of the SURF project was to develop a method to find optimal values of these parameters.

The Fitness Metric

The first step in choosing an optimization procedure is to consider the character of the response surface. If there are only a few free parameters and the goal of the optimization is “well-behaved” in response to variations in those parameters, just about any kind of hill-climbing algorithm will produce the desired values of the parameters. If these conditions do not hold, a non-deterministic approach, such as “genetic optimization” might be more effective.

As Appendix B shows, many parameters may be important in the construction of “superlative” TINs. The goal is to find the parameters that will produce TINs that do the best job of determining whether geometric lines of sight are blocked when the kinds of terrain and the distribution of ranges correspond to those that occur when the Corps Battle Simulation is used in training exercises.

Thus, the first step was to get an idea of how the response surface depends on the controllable parameters. The first part of that first step is to create an explicit definition of the optimization criterion — known in the parlance of genetic optimization as “the fitness metric”.

While constructing a metric for line of sight accuracy, we realized that we were forced to assume that an arbitrary terrain skin draped over the DEM represented ground truth.

Thus, an algorithm that used the terrain skin draped over the DEM directly would have perfect accuracy — if we had one that was fast enough and would fit into memory.

Part 2: Quadtree Assessment of Line of Sight on Large DEMs

Ray Tracing

Scene-rendering algorithms in computer graphics trace the paths of large numbers of simulated rays of light through complex three-dimensional scenes that may contain millions of objects. To avoid consideration of ray-object intersection with every object in the scene, objects are grouped within bounding boxes. A ray that does not intersect a bounding box does not strike any of the objects contained in that box. LOS evaluation is a special case of the ray-tracing problem. In LOS evaluation, the scene to be evaluated is the regular triangulation of the DEM, the objects are the triangular facets of the terrain skin, and the ray is the LOS.

To reduce the number of intersections that have to be evaluated even further, the bounding boxes can be recursively grouped within other bounding boxes. Commonly, as in Whitted & Rubin (1980) or Agate et al. (1991), the scene space is recursively divided into halves in each of the three dimensions and reassembled into an octree data structure.

In principle, an octree approach could be applied to the LOS evaluation problem, but the number of leaves required to describe the volume of space occupied by the terrain skin for a large playbox would be prohibitive (up to 70 trillion). Fortunately, the elevation of the terrain skin is a single-valued function of the location. Therefore, a quadtree, which recursively partitions the data in halves only in latitude and longitude (but not in elevation), as indicated in figure 7, can be used instead. Only 7 billion leaves are required for a quadtree that describes the largest CBS playbox.

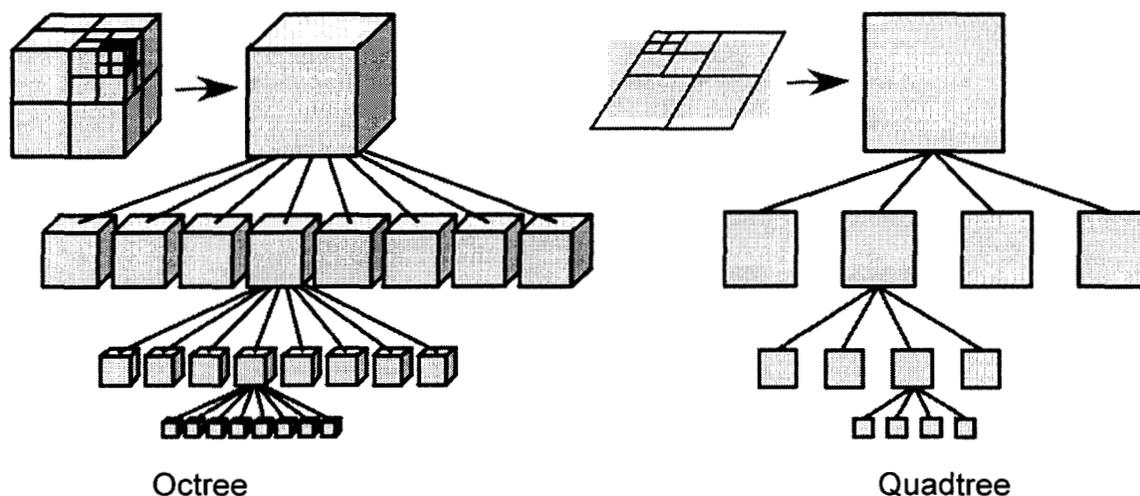


Figure 7. An octree data structure, as shown on the left, describes the result of recursively dividing a volume in half in each of three dimensions. Unlike the three-dimensional scenes encountered in computer graphics, the DEM describes a simple surface with only a single elevation for each latitude-longitude pair. Thus, only two dimensions must be halved at each step, and a quadtree can be used instead.

The Quadtree Data Structure

Quadtree spatial partitioning is applied by quartering the DEM along the posts into progressively smaller quadrants until each quadrant contains only 4 posts. Any quadrant that contains more than 4 posts may be further divided and is represented in the quadtree data structure by a node containing the highest elevation in the quadrant. Quadrants that contain only 4 DEM posts are not divided further and are represented by leaves containing the elevations of the four DEM posts.

To ensure that the DEM is always evenly divided, each dimension (rows and columns) of the DEM is enlarged to the nearest $2^n + 1$, but no data is stored for leaves that are outside of the DEM or for nodes that do not intersect the DEM. Consequently, no additional memory is required to enlarge the dimensions.

The only information stored in the nodes is the elevation. The dimensions and location of the bounding boxes are implied by the level of the tree and the path of traversal, respectively. At a given level in the tree, all the boxes share the same dimensions. By knowing the location of the root node, the location of any other node can be computed.

Pointers to link nodes with their children are not needed. Instead, each level of the tree is stored in a one-dimensional array. The indices and level of child nodes are computed from the index and level of the parent node. The level representing the leaves contains only the original DEM data.

By starting quadtree construction at the leaves, the number of comparisons necessary to compute the maximum elevation within a quadrant is always exactly three, rather than the number of pairs of DEM posts within the quadrant.

The lowest level of the quadtree, the leaves, contains the DEM elevation data. At higher levels, every node contains a single elevation value and implicitly points either at four leaves or at four nodes; hence, every level is one-fourth as large as the one below it. Thus, the overall size of the quadtree is a little less than $4/3$ (which is the sum of the infinite series $1 + (1/4) + (1/4)^2 + \dots$) times the size of the DEM.

In summary: The elevations stored in the nodes of the quadtree data structure correspond to the tops of three-dimensional bounding boxes that completely contain the terrain within the domain of the quadrant represented by the node. The bottoms of the bounding boxes are implicitly at the center of the Earth. The root node contains the highest elevation anywhere in the DEM. The box defined by the root node spans the length and width of the DEM and touches the top of the highest point. No post in the DEM protrudes from the root's bounding box. Together with a model specification (such as the choice of a direction for the diagonal of a pair of triangles), the elevations stored in the leaves of the quadtree data structure are sufficient to define the terrain skin.

Quadtree Line Of Sight Algorithm

The new algorithm for evaluating LOS is a recursive, depth-first search of the quadtree data structure for the intersection of the LOS line segment with a regular triangulation of the DEM. By searching the largest bounding boxes first, it is possible to reject LOS-terrain intersection over large regions of the DEM in relatively few steps. An unobstructed LOS is recognized as high in the tree as possible, thereby minimizing the number of intersection evaluations.

The initial question is whether the LOS — that is, the line segment from the sensor to the target — intersects the bounding box implied by the root node. If either end of the LOS is inside that box, then it certainly intersects the box. If both ends are outside the box, then the LOS intersects the box only if it intersects one of the sides of the box. To intersect a side, the LOS line must intersect the plane associated with the side between its two limits and below the top at a point on the LOS that is between the sensor and the target. If the LOS does not intersect the root node, it is unobstructed. If it does intersect the root node, its intersection with each of the four children of the root must be examined.

If both ends of the LOS are inside the box associated with one of the child nodes, the other child nodes may be ignored. The intersection tests for children are the same as for the parents. Failure to intersect implies that LOS is not blocked by this node. If the LOS intersects any node's box, its four children must be examined. If none of the children of a node block the LOS, it is unobstructed by the node.

Eventually, the node is a leaf, and the intersection test is slightly different. The four associated DEM posts define a terrain skin. We are currently using a regular triangulation consisting of the two triangles obtained by connecting the four corners and a constant-direction diagonal. This choice of a terrain skin model can lead to asymmetrical terrain resolution, depending upon how closely the LOS is aligned with the diagonals. We are planning to change to the four triangles defined by the four corner DEM posts and a center point whose elevation is the average of the four corners. An alternative we considered is to choose the direction of the diagonal by determining which of the two directions would give a better fit to a pair of triangles fit to the sixteen DEM posts of which the current four are the center. Yet another alternative, if higher resolution data is available, would be to use that data to choose a preferred direction for the diagonal.

The leaf intersection test is similar to the box intersection test, except that the top of each side slopes instead of being horizontal. Also, there is a diagonal (or two) to be considered. The diagonal is treated like a side. To keep the intersection computations simple and fast, both the bounding boxes and the LOS are expressed in the same rectangular coordinate system. While floating-point numbers are used, the smallest possible denominator is the spacing between posts. Leaf intersection evaluation adds only one additional step to bounding box intersection. Using a higher order interpolation between DEM posts would require changing only the leaf intersection calculation.

Because the DEM is always divided into quadrants, the complexity of this new approach is $O(\log(n))$ where n is the number of DEM edges along the LOS. The complexity of the edge traversal technique is $O(n)$. The periodic sampling technique has constant time complexity, but suffers from stochastic inaccuracy as a function of the time complexity constant.

While a rectangular coordinate system facilitates efficient intersection mathematics, it hides the curvature of the Earth and the effects of atmospheric refraction. Three adjustments are made during the intersection evaluation.

First, the elevation of the end of the LOS is reduced (in rectangular coordinates) by an amount proportional to the square of the distance from the sensor to the target. The elevation of any edge being considered is reduced in proportion to the square of its distance from the sensor.

Second, the constant of proportionality accounts for atmospheric refraction in addition to the radius of curvature as suggested by Bowditch (2002).

Third, the convexity of the boxes is addressed by adding a pre-computed value based on the maximum horizontal dimension of the box. This adjustment ensures the top of the box is indeed at least as high (in rectangular coordinates) as any point on the terrain. Leaves are so small that this correction is negligible; the triangles in the regular triangulation are flat.

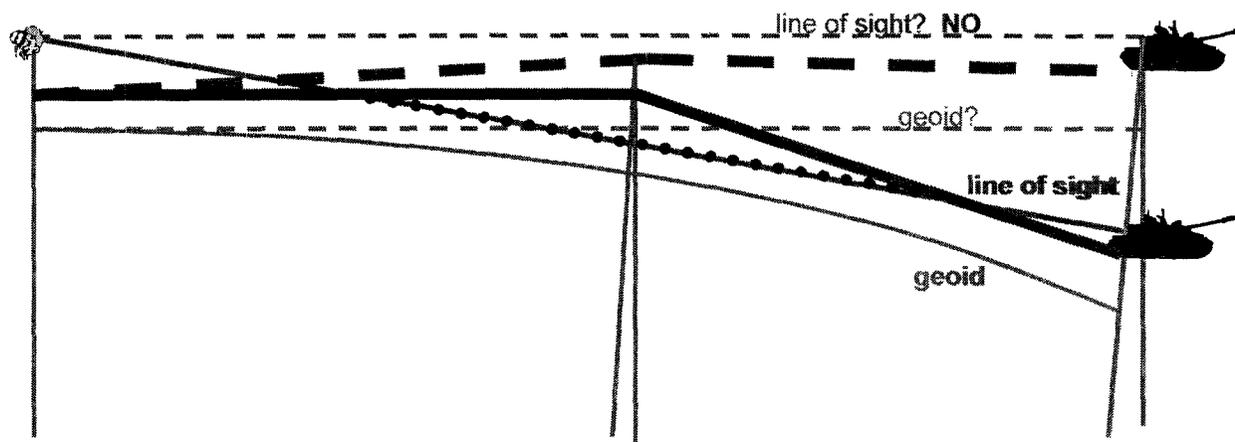


Figure 8. Curvature of the Earth must be considered, as elevation data is presented relative to average sea level (technically, the geoid). The geoid is lower than the horizon (a plane tangent to the Earth at the location of an observer) by an amount proportional to the square of the distance from the observer.

Memory Management and Compression

For large data sets, the DEM alone can require several gigabytes of memory. A data structure that is 4/3 the size of the DEM may be too large to be loaded into low-latency memory (RAM). These demands necessitate the development of memory management and compression techniques.

Because each level of the tree is a linear array of binary elevation data describing a relatively smooth surface, compression techniques may be feasible. One possible technique is to store changes in elevation relative to the parent bounding box rather than absolute elevations. By storing delta values, far fewer bits would be required per elevation value. The only absolute elevation would be the elevation of the root node. To determine the absolute elevation of a given node or leaf would require summing the changes in elevation of all the nodes traversed to reach that node. Only the data that is needed would be decompressed, and it could be decompressed as the algorithm requires the information, without any loss in data accuracy.

Even effective data compression, however, can only reduce the data storage requirement by an order of magnitude. A more promising approach is to map only the necessary information to RAM while leaving large portions of the unread data on high-latency memory (hard disk), to be brought into RAM only as needed. It will often be possible to determine when LOS is not obstructed by reading from the first few levels of the quadtree data structure. These levels take insubstantially small amounts of memory relative to the DEM and can easily be stored in RAM. Data is required from the deepest levels in the quadtree only when LOS is obstructed. When the application is a military simulation, LOS queries are not uniformly distributed throughout the DEM. Information describing only the combat regions can easily be paged to RAM by the operating system.

TO
@PJL SET RESOLUTION=600
@PJL ENTER LANGUAGE=PCL

Appendix A. Quality Metrics

Assessment of line-of-sight queries, while not a global issue, is not local, either. Hence, we have not thought of a good direct way to determine whether flipping an edge or changing the elevation of a vertex would improve the quality of the TIN from that point of view. We have, however devised several quality metrics that describe how well the TIN triangles match the topography. A weighted sum of these quality metrics is then used as a composite optimization criterion.

As discussed elsewhere in this paper, the original product of the 2003 Summer Undergraduate Research Fellowship (SURF) project was to be a protocol and an algorithm that uses line-of-sight queries to find values of the weights (and other parameters) that produce the best TINs. Then the composite quality metrics (one for edges and one for vertices) could be used as proxies for the true line-of-sight objective.

Triangles

We have identified six quantifiable aspects of quality for TIN triangles: *NumPosts*, *MaxDev*, *SumDevs*, *SumSquares*, *RMS*, and *Sliveriness*; each of which is defined and discussed below. We do not have quantitative measures for the correspondence of TIN edges with ridges, cliffs, watercourses, and other terrain breaklines. However, to the extent that the deduction of richlines was successful, those features are “frozen” into the TIN.

NumPosts, N_T . The first measure is simply the number of DEM posts covered by the triangle. Posts that fall exactly on an edge are counted as one-half for each of the two triangles affected. (Posts on playbox boundaries count fully since they affect only one triangle.) Posts at vertices count as the reciprocal of the number of triangles meeting at the vertex.

The next four measures are based on the difference between the surface of the faceted TIN model and the surface of the earth as represented by the elevations of the DEM posts, which are treated as if they represented error-free ground truth.

MaxDev, D_T . The maximum deviation between the model and the data is an intuitively important measure of fidelity, and can be considered with respect to individual triangles and to the TIN as a whole. A *deviation* is the perpendicular distance from the tip of a DEM post to the surface of the TIN triangle that contains the DEM post. Deviations of posts that fall exactly on edges or at vertices are weighted in sums in the same way they are weighted in the computation of N_T . In a triangle, the location at which the maximum deviation occurs is used for a new vertex if the triangle is split.

D_T Absolute value of the largest of the deviations in triangle T.

The maximum deviation of the TIN for an entire playbox (or tile) is the largest of these, and can be reduced by splitting the triangle that contains it.

SumDevs, A_T . The algebraic sum of the deviations in a triangle is expected to be most useful in adjusting the elevations of vertices.

A_T Algebraic sum of the deviations in triangle T.

Appendix A. Quality Metrics

Assessment of line-of-sight queries, while not a global issue, is not local, either. Hence, we have not thought of a good direct way to determine whether flipping an edge or changing the elevation of a vertex would improve the quality of the TIN from that point of view. We have, however devised several quality metrics that describe how well the TIN triangles match the topography. A weighted sum of these quality metrics is then used as a composite optimization criterion.

As discussed elsewhere in this paper, the original product of the 2003 Summer Undergraduate Research Fellowship (SURF) project was to be a protocol and an algorithm that uses line-of-sight queries to find values of the weights (and other parameters) that produce the best TINs. Then the composite quality metrics (one for edges and one for vertices) could be used as proxies for the true line-of-sight objective.

Triangles

We have identified six quantifiable aspects of quality for TIN triangles: *NumPosts*, *MaxDev*, *SumDevs*, *SumSquares*, *RMS*, and *Sliveriness*; each of which is defined and discussed below. We do not have quantitative measures for the correspondence of TIN edges with ridges, cliffs, watercourses, and other terrain breaklines. However, to the extent that the deduction of richlines was successful, those features are “frozen” into the TIN.

NumPosts, N_T . The first measure is simply the number of DEM posts covered by the triangle. Posts that fall exactly on an edge are counted as one-half for each of the two triangles affected. (Posts on playbox boundaries count fully since they affect only one triangle.) Posts at vertices count as the reciprocal of the number of triangles meeting at the vertex.

The next four measures are based on the difference between the surface of the faceted TIN model and the surface of the earth as represented by the elevations of the DEM posts, which are treated as if they represented error-free ground truth.

MaxDev, D_T . The maximum deviation between the model and the data is an intuitively important measure of fidelity, and can be considered with respect to individual triangles and to the TIN as a whole. A *deviation* is the perpendicular distance from the tip of a DEM post to the surface of the TIN triangle that contains the DEM post. Deviations of posts that fall exactly on edges or at vertices are weighted in sums in the same way they are weighted in the computation of N_T . In a triangle, the location at which the maximum deviation occurs is used for a new vertex if the triangle is split.

D_T Absolute value of the largest of the deviations in triangle T.

The maximum deviation of the TIN for an entire playbox (or tile) is the largest of these, and can be reduced by splitting the triangle that contains it.

SumDevs, A_T . The algebraic sum of the deviations in a triangle is expected to be most useful in adjusting the elevations of vertices.

A_T Algebraic sum of the deviations in triangle T.

SumSquares, U_T . A sign-independent sum of residuals would be a measure of the gross overall fidelity of the fit of a model. The sum of the squares of residuals is not only sign-independent, but emphasizes larger errors and is related to statistical assessment of the quality of the fit. The sum of the squares of residuals in a triangle is a measure of the gross overall fidelity of the fit to the region covered by the facet.

U_T Sum of the squares of the deviations in triangle T.

The sum of squares of deviations for the TIN for an entire playbox (or tile) is the grand sum of U_T over all of its triangles. Larger triangles tend to have larger values of U_T .

RMS, R_T . A sign-independent measure of the average fidelity of the fit that can be easily obtained from the SumSquares is the root-mean-square deviation.

R_T Root mean square of the deviations in triangle T. This metric carries the same information as the standard deviation, but is easier to compute. Because annealing does not change the shape of the two-triangle quadrilateral associated with an edge (and therefore does not change the number of interior posts), this metric is identical with U_T during annealing — do not use it. That is, *use this metric when refining, but not when annealing*.

$$R_T = \sqrt{U_T/N_T}$$

The RMS of the fit of the TIN for an entire playbox (or tile) is given by the square root of the quotient of the grand sum of U_T divided by the number of posts in the entire playbox (or tile).

The statistical *sample variance*, s_T , (an estimate of the *standard deviation*, σ_T), could be computed from

$$s_T = \sqrt{(U_T - [A_T/N_T]^2)/(N_T - 1)}$$

By assuming that deviations from the model occur randomly with a known distribution, one could then make statements about probable deviations. The RMS is easier to compute and is sufficient to compare the average fit of alternative models.

The sum of the squares, U_T , describes the overall fit of the model, while the RMS, R_T , describes the average fit. When applied to a fixed region, as in edge flipping, they are equally effective in identifying the better fit, but U_T requires less computation.

Sliveriness, S_T . “Slivers” are triangles with at least one very small angle. They are the bane of all triangulation schemes that make smaller triangles out of bigger ones by adding vertices. Edge flipping is the most (perhaps the only) effective way of eliminating slivers, but our algorithm (quite properly) only allows flips when the fit is improved. (We also try to avoid slivers by starting with a Delaunay triangulation.)

The preferred edges to consider for flipping are the long sides of slivery triangles. There are several ways that sliveriness could be identified. The most intuitive would be the size, in degrees, of the smallest angle in a triangle. This would require many evaluations of inverse trigonometric functions and be computationally expensive. A more convenient measure of sliveriness is a dimensionless ratio based on the square of the perimeter of the triangle to its area.

From an elementary table of trigonometry formulas, such as §63 of Burington (1956), we have

$$2s = p = a + b + c$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where a , b , c are the lengths of the sides of a triangle, p is the perimeter, and A is the area. The semi-perimeter $s = (a + b + c) / 2$. Using these equations, we find that the ratio of the square of the perimeter to the area contains a $3/2$ power and a square root:

$$\frac{p^2}{A} = \frac{4s^{3/2}}{\sqrt{(s-a)(s-b)(s-c)}}$$

Since the only purpose for this measure is to identify the sliveriest triangle, we can use the simpler expression obtained by dividing out the constant and squaring:

$$S_T = \frac{s^3}{(s-a)(s-b)(s-c)}$$

where a , b , c = lengths of the three sides of triangle T

By this definition, the Sliveriness of an equilateral triangle—the smallest value possible—is exactly 27. The Sliveriness of a half-square is 0.03 less than 34. The Sliveriness of an isosceles triangle with a 1° peak angle is 53,450. This measure depends only on the shape of the triangle, not its size. (Note that the denominator goes to zero only if s , half of the perimeter, equals one of the sides, which can only happen if the triangle is completely flat. This should never happen, but bulletproof programming can set Sliveriness to a very large value if floating-point computation causes this condition to occur.)

Edges

State-of-the-art TIN refinement algorithms such as Polis et al. (1996) focus on lists of triangles, sorted by a quality metric — usually the maximum deviation within the triangle, D_T . The worst triangle is then split into three triangles and its original three (untouched) edges are considered for flipping. One of Mark Kordon's innovations in CBS's 1996 TINning algorithm, designed to produce fewer slivers, was to find the adjacent triangle that was closest to the *MaxDev* point, then split the quadrilateral consisting of those two triangles into four triangles — then consider the quadrilateral's original four (untouched) edges for flipping.

Kordon's idea can be applied more easily if it is *quadrilaterals*, rather than *triangles*, that are assessed and sorted. The easiest way to characterize all of the possible quadrilaterals, which overlap, is by the TIN *edges* that are their diagonals. (Hence, the following *edge* metrics could be thought of as *quadrilateral* metrics.) We have identified nine of them: *NumPosts*, *MaxDev*, *Curvature*, *SumDevs*, *SumSquares*, *RMS*, *Sliveriness*, *SliverBullet*, and *ZeroInteriorPosts*; each of which is defined and discussed below.

All of the edge metrics are normalized so that the order of magnitude of optimal values is expected to be *roughly* unity in most cases.

NumPosts, M_{NE} . The number of DEM posts associated with an edge is the sum of the numbers in the two associated triangles. Edges on the boundaries of the playbox are associated with only one triangle.

$$M_{NE} = N_{T_1} + N_{T_2}$$

MaxDev, M_{DE} . The maximum deviation associated with an edge is the maximum deviation in the two associated triangles.

M_{DE} Normalized maximum deviation metric for edge E, computed from

$$M_{DE} = \frac{\max(D_{T_1}, D_{T_2})}{N_D}$$

where N_D = Average relative vertical error for the DEM files used to construct the playbox. A suitable value can usually be extracted from the DEM files.

Curvature, M_{CE} . The facets of the TIN are treated as if they were planar. The TIN, however, is built on the elevation data provided in the DEM, which corresponds to the geoid, which is round (and very nearly spherical, with a radius of curvature given by RE). Thus, the TIN facets should be thought of as (nearly) spherical, with a radius of curvature of RE, and a line of sight could be blocked by a point interior to a TIN triangle.

A line of sight could also be blocked by variations in the surface at a level of resolution below that which is represented by the TIN. Thus, for consistency, we need only ensure that the bowing due to the curvature of the Earth is no more than the typical resolution. The resolution, however, depends upon how good the TIN is. This circular definition can be avoided by simply treating the maximum error due to curvature as one of the quality metrics to be considered along with the others. The maximum error due to curvature is at the middle of the longest edge of the triangle.

M_{CE} Normalized metric for Earth curvature of edge E, computed from

$$M_{CE} = \frac{L^2 \cdot (RF / 2 \cdot RE)}{N_D}$$

where L = Length of the edge

RF = Atmospheric refraction factor

RE = Radius of curvature of the Earth

SumDevs, M_{AE} . The algebraic sum of the deviations is expected to be most useful in adjusting the elevations of vertices.

M_{AE} Normalized sum of deviations metric for edge E, computed from

$$M_{AE} = \frac{A_{T_1} + A_{T_2}}{N_{T_1} + N_{T_2}} / N_D$$

SumSquares, M_{UE} . The sum of the squares of the deviations in the two triangles.

M_{UE} Normalized sum of squares metric for edge E, computed from

$$M_{UE} = (U_{T_1} + U_{T_2}) / N_D^2$$

RMS, M_{RE} . The RMS of the deviations in the two triangles.

M_{RE} Normalized RMS metric for edge E, computed from

$$M_{RE} = \sqrt{\frac{U_{T_1} + U_{T_2}}{N_{T_1} + N_{T_2}}} / N_D = \sqrt{\frac{M_{UE}}{N_{T_1} + N_{T_2}}}$$

Again: use this metric while refining, but not while annealing.

Sliveriness, M_{SE} . It would not make sense to define an analogous sliveriness metric for the quadrilateral, as it is the slivery triangles themselves that are undesirable.

M_{SE} Normalized sliveriness metric for edge E, computed from

$$M_{SE} = (S_{T_1} + S_{T_2}) / N_S$$

where $N_S =$ Twice the sliveriness of an equilateral triangle $= 2 \times 27 = 54$

SliverBullet, M_{BE} . A characteristic of offensive slivers is that they extend unreasonable distances across the playbox—small slivers are less distressing than large ones. The *SliverBullet* heuristic, invented by Mark Kordon, seeks triangles whose splitting will reduce both the most offensive slivers and the maximum deviation by weighting the *MaxDev* by the square of L, the length of the side of the triangle nearest the *MaxDev* point. Here, every edge is considered, but the longer ones will have worse (larger) values of the metric.

M_{BE} Normalized “sliver bullet” for edge E, computed from

$$M_{BE} = (D_{T_1} + D_{T_2}) \cdot L_E^2 / N_B$$

where T_1 and $T_2 =$ the triangles associated with edge E

$L_E =$ length of edge E

$N_B = 2 \cdot N_D \cdot dlat^2$

$dlat =$ DEM post spacing in latitude (in meters)

ZeroInteriorPosts, M_{ZE} . Triangles that contain no interior DEM posts are likely to be extreme slivers. They will *never* be detected by metrics that look at deviations between the TIN and the DEM.

M_{ZE} Zero interior posts pathology metric for edge E, computed from

$$M_{ZE} = \begin{cases} 1 & \text{if } T_1 \text{ or } T_2 - \text{ but not both - has no interior DEM posts} \\ 0 & \text{if both } T_1 \text{ and } T_2 - \text{ or neither - have any interior DEM post} \end{cases}$$

For this determination, DEM posts at the vertices do not count as “interior”, though they are used in computing the other metrics.

Composite Quality Metric for Edges, QE_E . None of the metrics defined above measure how well the TIN assesses line-of-sight queries, which is the primary reason for having a TIN (in the CBS context). However, they do all measure aspects of the TIN that are likely to be related to its performance on LOS tests.

QE_E Composite quality metric for edge E, computed from

$$QE_E = W_N \cdot M_{NE} + W_D \cdot M_{DE} + W_C \cdot M_{CE} + W_A \cdot M_{AE} + W_U \cdot M_{UE} + W_R \cdot M_{RE} \\ + W_S \cdot M_{SE} + W_B \cdot M_{BE} + W_Z \cdot M_{ZE}$$

where W_N = Relative weight of the number of posts associated with the edge
 W_D = Normalized relative weight of the maximum deviation metric for edges
 W_C = Normalized relative weight of the Earth curvature metric for edges
 W_A = Normalized relative weight of the algebraic sum of deviations metric for edges
 W_U = Normalized relative weight of the sum of squares metric for edges
 W_R = Normalized relative weight of the RMS metric for edges
(not used when annealing)
 W_S = Normalized relative weight of the sliveriness metric for edges
 W_B = Normalized relative weight of the “sliver bullet” metric for edges
 W_Z = Normalized relative weight of the zero interior posts pathology metric for edges

Since we suspect that large triangles are desirable, it may be that W_N , the optimum weight associated with M_{NE} , the number of posts in the triangles associated with the edge, is negative; all of the others are expected to be positive. Some of these metrics may turn out to be useless in improving the quality of the TIN; if that can be determined, their weights will be zero.

Vertices

Normalized Quality Metric for Vertices, MH_V . Changing the height of a TIN vertex affects the fit of all of the TIN triangles that share that vertex.

MH_V Height quality metric for vertex V, computed from

$$MH_V = \frac{\sum_{T=\text{all triangles around vertex V}} A_T}{N_V}$$

where $N_V = 6 \times N_D$

Appendix B. Free Parameters (subject to optimization)

Tolerances used during deduction of richlines:

`BoundaryMouthTolerance` = Maximum allowable vertical deviation during line simplification of the boundary elevation profile for finding where channels drain off the playbox (and the corresponding high points) along the boundary.

BranchingSlopeTolerance = The amount by which the slope from the end of a propagating ridge end to neighboring DEM posts to which it drains can differ and still be considered at the same slope. This is used to identify branching ridges.

GeneralSlopeTolerance = The amount by which the slope between neighboring DEM posts can differ from each other and still be considered the same.

MinNegSlopesForPit = Minimum number of neighbor posts with negative slope for this post to be a “pit”.

MinShorelineElevationRangeForMouth = Minimum elevation difference along a line one DEM post inland of a shoreline required to recognize a channel mouth. (Otherwise, the entire shoreline defines a single watershed.) Nominal value = 10 meters.

MinShorelinePerimeterForMouth = Minimum number of rasterized posts along a shoreline required before there will be an attempt to recognize a channel mouth. (Otherwise, the entire shoreline defines a single watershed.)

MinPosSlopesForPeak = Minimum number of neighbor posts with positive slope for this post to be a “peak”.

ShorelineMouthTolerance = Maximum allowable vertical deviation during line simplification of the elevation profile of the temporary string of posts just outside a shoreline for the purpose of finding where channels drain into the shoreline (and the corresponding watershed limits along the shoreline).

TributarySlopeTolerance = The amount by which the slope from the end of a propagating channel end to neighboring DEM posts that drains to it can differ and still be considered at the same slope. This is used to identify tributaries.

ZeroSlopeTolerance = The amount by which the slope between neighboring DEM posts can differ from zero and still be considered “flat”.

Minimum richline lengths:

MinPostsForBoundaryLine = Boundary lines with fewer than this number of posts will be discarded before line simplification. Nominal value = 2 posts.

MinPostsForChannel = Channels with fewer than this number of posts will be discarded before line simplification. Nominal value = 2 posts.

MinPostsForRidgeLine = Ridgelines with fewer than this number of posts will be discarded before line simplification. Nominal value = 2 posts.

MinPostsForShoreline = Shorelines with fewer than this number of posts will be discarded before line simplification. Nominal value = 2 posts.

Tolerances used while simplifying rasterized lines

H = Maximum allowable horizontal tolerance during line simplification

V = Maximum allowable vertical tolerance during line simplification

It may be desirable to have separate pairs of these values for each of the types of richlines.

Budgets

`FlippingBudget` = Maximum number of edge flips during annealing

`PulsingBudget` = Maximum number of vertex elevation changes during annealing

`TriangleBudget` = Maximum number of triangles during refinement. (In the interior of a triangulation, the ratios of vertices:triangles:edges = 1:2:3. Consequently, the vertex budget \approx `TriangleBudget` / 2 and the edge budget \approx `TriangleBudget` \times 3/2.) Lawson (1977) gives the exact relationships that account for boundary conditions:

Annealing and refinement parameters

There are three sets of the following values: One for annealing the CDT to produce the `StartingTIN`, one for use in `RefineTIN`, and a third for annealing the output of `RefineTIN` to produce the `BigTIN`. When used, the weights should sum to 1.0, but it would be user-friendly to have the computer normalize the input values by dividing by their sum, then using the normalized values. Note that `MHVGOAL`, which is associated with “pulsing” vertices, is not used in refinement. Also note that the RMS metric (see Appendix A), and therefore `WR` (see Appendix A and the list below), is not used during annealing.

`WB` = Normalized relative weight of the “sliver bullet” metric for edges

`WC` = Normalized relative weight of the Earth curvature metric for edges

`WD` = Normalized relative weight of the maximum deviation metric for edges

`WR` = Normalized relative weight of the RMS metric for edges

`WS` = Normalized relative weight of the sliveriness metric for edges

`WU` = Normalized relative weight of the sum of squares metric for edges

`WZ` = Normalized relative weight of the zero interior posts pathology metric for edges

`MHVGOAL` = “Good enough” value of M_{HV} the quality metric for vertices. Annealing terminates before the number of pulses reaches the `PulsingBudget` in the unlikely event that Q_{HV} , for the worst vertex gets below this value. Conversely, setting this value to zero forces annealing to continue until the `PulsingBudget` is reached. This parameter is not used during refinement.

`QEGOAL` = “Good enough” value of Q_E , the composite quality metric for edges. Refinement (or annealing) terminates before the number of triangles reaches the `TriangleBudget` (or the number of flips reaches the `FlippingBudget`) in the unlikely event that Q_E , for the worst edge gets below this value. Conversely, setting this value to zero forces continuation of refinement until the `TriangleBudget` is reached and forces annealing to continue until the `FlippingBudget` is reached.

`VertexDeviationFactor` = Amount by which the average sum of the deviations associated with a vertex is multiplied to obtain an adjustment when “pulsing the vertices”. If the deviations were continuously distributed, the value that would zero the metric is 3. (However, using a greedy algorithm that sets this metric to zero on a vertex-by-vertex basis may not be the best strategy for optimizing line-of-sight tests.)

Acknowledgements

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the U.S. Army Program Executive Office Simulation, Training, and Instrumentation under an agreement with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

- Agate, M., R. L. Grimsdale, P. F. Lister. 1991. The HERO Algorithm for Ray Tracing Octrees. *Advances in Computer Graphics Hardware IV*, R. L. Grimsdale, W. Strasser (eds), Springer-Verlag, New York.
- Aho, Alfred V., John E. Hopcroft, Jeffrey D. Ullman. 1976. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Reading, MA, ISBN 0-201-00029-6, Third Printing. 145–152.
- Arge, Lars, Jeffrey S. Chase, Patrick Halpin, Laura Toma, Dean Urban, Jeffrey S. Vitter, Rajiv Wickremasinghe. 2003. Flow computation on massive grid terrains, *GeoInformatica*, (to appear). An earlier version appeared in *Proc. 10th ACM International Symposium on Advances in Geographic Information Systems (ACM-GIS '01)*. The contact author is Laura Toma, laura@cs.duke.edu, Box 90129, Durham, NC 27708.
- Band, Larry E. 1986. Topographic partition of watersheds with digital elevation models, *Water Resources Research* 22/1: 15–24.
- Bowditch, Nathaniel. 2002. *The American Practical Navigator*, Table 12, p 559, Superintendent of Documents, U.S. Government Printing Office, Mail Stop SSOP, Washington, D.C. 20402-0001; downloadable from (pollux.nss.nima.mil/pubs/).
- Burington, Richard Stevens. 1956. *Handbook of Mathematical Tables and Formulas*, Handbook Publishers, Inc, Sandusky, Ohio, Third Edition.
- Chew, L. Paul. 1989. Constrained Delaunay triangulation. *Algorithmica*, 4 97–108.
- Douglas, David H., Thomas K. Peucker (now Poiker). 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2) 112-122.

- Douglas, David H. 1986. Experiments to locate ridges and channels to create a new type of digital elevation model, *Cartographica*, 23(4) 29-61.
- Environmental Systems Research Institute. 1989. *TIN Users Guide—ARC/INFO Surface Modeling and Display*. Environmental Systems Research Institute Redlands, CA.
- Faugeras, Olivier. 1993. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, MA. 438–445.
- Fowler, R. J., J. J. Little. 1979. An automatic method for the construction of irregular network digital terrain models. *Proceedings of SIGGRAPH '79*, 199–207.
- Guibas, Leonidas, Jorge Stolfi. 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2) 74–123.
- Heller, Martin. 1990. Triangulation algorithms for adaptive terrain modeling. *Proc. 4th Intl. Symp. on Spatial Data Handling*, Zurich, 1, 163-174.
- Jenson, Susan K. 1985. Automated derivation of hydrologic basic characteristics from digital elevation model data. *Proceedings of Auto-Carto 7*, 301–310.
- Jenson, Susan K, Julia O. Domingue. 1988. Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric Engineering and Remote Sensing* 54(11) 1593–1600.
- Lawson, Charles L. 1977. Software for C^1 surface interpolation. *Mathematical Software III*, Academic Press Inc., New York.
- Lee, D. T., B. J. Schachter. 1980. Two algorithms for constructing a Delaunay triangulation. *Intl. J. Computer and Information Sciences*, 9(3) 219–242.
- Little, J. J., P. Shi. 2001. Structural lines, TINs, and DEMs. *Algorithmica* 30 243–263.
- Peucker, Thomas K., David H. Douglas. 1975. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4 375–387.
- Peucker, Thomas K., Robert J. Fowler, James J. Little, David M. Mark. 1976. Digital representation of three-dimensional surfaces by triangulated irregular networks. *Technical Report No. 10*, Office of Naval Research (ONR) Geography Programs 1-63.
- Peucker, Thomas K., Robert J. Fowler, James J. Little, David M. Mark. 1978. The triangulated irregular network”, *Proceedings of the Digital Terrain Models (DTM) Symposium*, American Society of Photogrammetry, Falls Church, VA. 516—532.
- Polis, Michael F., Stephen J. Gifford, and David M. McKeown Jr. 1995. Automating the construction of large-scale virtual worlds, *Computer*, 28(7) 57–65.

- Polis, Michael F., Stephen J. Gifford, David M. McKeown Jr. 1996. *Integrated TIN Generation User's Manual*, Rev 1.0, April 15, 1996. Digital Mapping Laboratory, Carnegie-Mellon University, Schenley Park, Pittsburgh, PA 15213-3891.
- Preparata, Franco P., Michael Ian Shamos. 1985. *Computational Geometry: An Introduction*, Springer-Verlag, New York.
- Richbourg, Robert F., Robert J. Graebener, Tim Stone, Keith Green. 2001. Verification and validation (V & V) of federation synthetic natural environments. Institute for Defense Analyses, Simulation Center, Alexandria, Virginia. *Proceedings of the 2001 Interservice/Industry Training, Simulation, and Education Conference*; CDs may be ordered from NTSA, Suite 400, 2111 Wilson Boulevard, Arlington, VA, 22201
- Scarlato, Lori Lynne Lemanczyk. 1993. *Spatial Data Representations for Rapid Visualization and Analysis*, [Ph.D. thesis] Department of Computer Science, State University of New York at Stony Brook.
- Sedgewick, Robert. 1990. *Algorithms in C*. Addison-Wesley Publishing Company, Inc.
- Shewchuk, Johnathan Richard. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. *First Workshop on Applied Computational Geometry*, ACM. School of Computer Science Carnegie Mellon University, Pittsburgh, PA 15213. The paper was downloaded from links found at (www.cs.cmu.edu/~quake/triangle.html).
- Sinnott, R. W. 1984. Virtues of the haversine. *Sky and Telescope*, 68(2) 159.
- Toma, Laura, Lars Arge, Jeff Chase, Pat Halpin, Dean Urban, Jeff Vitter, Rajiv Wickremesinghe. 2002. Computations on massive grids; the TerraFlow Project. Retrieved on January 8, 2003 from (www.cs.duke.edu/geo*/terraflow/).
- Whitted, Turner, Steven M. Rubin. 1980. A 3-dimensional representation for fast rendering of complex scenes. *Proceedings of the 7th annual conference on computer graphics and interactive techniques*. ACM Press. 110–116.
- Whyatt, J. D., P. R. Wade. 1988. The Douglas-Peucker line simplification algorithm. *Bulletin of the Society of University Cartographers*, 22(1) 17–27.
- Williams, Ed. 2002. Aviation formulary V1.35 retrieved on October 10, 2002 from (williams.best.vwh.net/avform.htm).
- Zeiler, Michael. 1999. *Modeling our world, the ESRI guide to geodatabase design*, ESRI Press, Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100.