The Constrained Delauney Pyramid - A Model for Reconstructing Surfaces at Multiple Resolutions

Andreas Voigtmann, Ludger Becker and Klaus Hinrichs
Westfälische Wilhelms-Universität, Institut für numerische Mathematik - Informatik
Einsteinstr. 62, D-48149 Münster, Germany
email: {avoigt,beckelu,khh}@math.uni-muenster.de

(Extended Abstract)

1. Introduction

The problem of surface representation occurs in a variety of disciplines, including computer graphics, computer-aided design (CAD) and geographic information systems (GIS). In this paper we restrict ourselves to surface representations in geographic applications. In this area the reconstruction of the topography is needed for three-dimensional map design, terrain modeling (e.g. in conjunction with CAD) or feature analysis requiring three-dimensional data (e.g. for solving hydrographic problems).
Given a set of topographical data, we want to reconstruct the shape of the surface at a variety of predefined resolutions. In mathematical terms, the reconstruction process at a fixed level of detail can be considered as the interpolation of a function of two variables. Our topographical data is given as a finite set of three-dimensional points $V = \{ (x, y, z) \in \mathbf{R}^3 | z = H(x, y)\}$ representing the height $z = H(x, y)$ of the surface at each coordinate point (x,y). This set S of two-dimensional points (x, y) may be distributed in the plane non-uniformly. In addition, we assume the existence of a set L of non-intersecting line segments describing surface-specific features like coast-lines, river banks, ridges (i.e. lines connecting peaks with passes) or valleys. We assume that the endpoints of the line segments are elements of S, and no point of S lies in the relative interior of a line segment, i.e. the line segments may only intersect in their endpoints. The set L provides constraints for the surface-reconstruction process. Since we use triangulations to reconstruct the surface the constraints help us to preserve the surface-specific features. Without these constraints triangles are created whose edges cross characteristic lines of the surface. These triangles produce undesirable errors in the reconstructed surface (e.g. rivers flowing over little hills, etc.).
We consider the surface at different levels of detail to provide a significant data reduction for low resolutions which is useful for browsing very large data sets. The use of a hierarchical structure allows a tree-like traversal of the generated levels and therefore supports zooming in a given part of the surface without browsing large lists of triangles.
Many hierarchical surface models and related data structures have been proposed (see [Flo89] and [Flo87] for a survey) including ternary hierarchical triangulations [PF87], [Flo84], quaternary hierarchical triangulations [BV84],[GG79], quadtree-based models [CT86], [SG85] and other triangulation methods like [SP92]. Most of these models use triangulations of the given point-data to reconstruct the surface but none of them considers the existence of constraints for the reconstruction process. Furthermore, some models suffer from the requirement of regularly sampled data points (quadtree-based models, quaternary hierarchical triangulation and [SP92]) or from triangulations which contain elongated triangles and therefore cause numerical instability during interpolation (ternary hierarchical triangulation).
A "good" model should allow non-uniform data distributions and should be based on an "optimal" triangulation, i.e. a triangulation where the resulting triangles have a regular shape. A well known triangulation of this kind is the Delauney triangulation [PS85]. A hierarchical data structure for multiresolution surface reconstruction based on the Delauney triangulation is the Delauney pyramid [Flo89]. Since the Delauney pyramid cannot deal with the constraints defined by a set of line segments, we introduce the constrained Delauney pyramid.

2. The constrained Delauney pyramid

2.1 Basic definitions

The set L of line segments can be described by a *constraint graph* G_c on the given set S of vertices, i.e. two points s and t are connected by an edge in G_c if and only if $\overline{st} \in L$. We present the *constrained Delauney pyramid*, a multiresolution surface model based on the Delauney pyramid, and its underlying data structure. The constrained Delauney pyramid represents surfaces at multiple resolutions and considers the constraints during the reconstruction process.

Two vertices s and t of the constraint graph G_c are called *mutually visible* if and only if either $\overline{st} \cap l = \emptyset$ for all line segments $l \in L$ or if $\overline{st} \in L$.

Similar to the (standard) Delauney triangulation we may now define the constrained Delauney triangulation by the *constrained circumcircle property*: A constrained triangulation CT, i.e. a maximal planar straight line graph which contains G_c as a subgraph, is a *constrained Delauney triangulation CDT* if and only if no vertex which is mutually visible to each vertex of a triangle $t \in CT$ lies inside the circumcircle of triangle t. Algorithms to compute constrained Delauney triangulations are presented in [Che89], [FP92] and [MV93].

A *constrained Delauney sequence* $CDS = [CDT_0, ..., CDT_m]$ is a sequence of constrained Delauney triangulations with the following properties:

1. CDT_i is a constrained Delauney triangulation of subsets $S_i \subseteq S$ and $L_i = \{\overline{st} \in L | s, t \in S_i\}$, $i = 0, ..., m$
2. $S_{i-1} \subset S_i, i = 1, ..., m$
3. $E(CDT_i) \leq \varepsilon_i, i = 1, ..., m$, where ε_i denotes the predefined maximum error at level i
4. $E(CDT_i) \leq E(CDT_{i-1}), i = 1, ..., m$

Starting with an initial triangulation CDT_0, we obtain each constrained Delauney triangulation CDT_{i+1} from its predecessor by inserting points into S_i until property 3 is satisfied. The constraints must be inserted if the corresponding endpoints have been included in the triangulation. To maintain the constrained Delauney triangulations during the insertion of points and line segments we use the algorithm of [FP92]. This algorithm computes the constrained Delauney triangulation for sets $S' = S \cup \{p\}$ and $L' = L \cup \{l\}$ based on a constrained Delauney triangulation for sets S and L. Only local modifications are required to obtain the new triangulation.

The relationship between two consecutive triangulations DT_{i-1} and DT_i is given by the following two difference sets. The *difference set* D_{i-1} between DT_{i-1} and DT_i is the set of all triangles of DT_{i-1} which do not belong to DT_i, i.e. the union of all triangles violating the circumcircle property due to the point set $S_i \setminus S_{i-1}$. The difference set D_i' between DT_i and DT_{i-1} is the set of all triangles of DT_i which do not belong to the predecessor DT_{i-1}.

A *Delauney pyramid CDP* is the representation of a Delauney sequence $CDS = [CDT_0, ..., CDT_m]$ having the following properties:

1. CDT_0 is the top level of CDP
2. Level i in CDP corresponds to CDT_i
3. Each triangle t of CDT_{i-1} which does not belong to D_{i-1} is contained in CDT_i. We connect these identical triangles of CDT_{i-1} and CDT_i by a conceptual link.
   Each triangle t of D_{i-1} is replaced in CDT_i by the subset of triangles of D_i' intersecting t. Triangle t is connected with each element of this subset by a conceptual link.

To store the constrained Delauney pyramid in main memory, we use a modified winged-edge representation of the triangulations ([Flo87], [Vo93], [Wei85] or [Woo85] give a survey of data structures for representing triangular grids).

2.2 Generalization of constraints

Since the line segments are defined for the finest resolution of the data points, only few constraints will be inserted into the triangulations at coarse resolutions. This results in a loss of information of the topography for many levels. To prevent these errors, we have to generalize the constraints in order to obtain a *generalized constraint graph* for each resolution. These generalized constraints are used to construct the final pyramid.

During the generalization process we search for paths in the constraint graph G_c connecting points a and b which have already been inserted into the triangulation. If such a path is a *proper candidate path* (see definition in properties 1 and 2 below) and is close to edge $\overline{ab}$ (property 3), $\overline{ab}$ is said to be a *generalized constraint* for this path. A candidate path may be formed by several pieces which are separated by small gaps representing for example small valleys (property 1). These gaps do not affect the shape of the reconstructed surface at coarser resolutions. The candidate path needs not directly connect a and b. It is sufficient if the path intersects a circle around points a and b (property 2). Furthermore, the path must be close to the generalized edge (property 3). We now define these properties formally. Let $U(\bullet)$ denote a circle around a vertex of the triangulation at level i whose size depends on the corresponding maximum error ε_i.

1. Property: *candidate paths*
   Let $P = c_1, ..., c_n$ be a path of G_c and let $c_j \notin S_i, 1 \leq j < n$. P is said to be a *candidate path*.
   Let $Q = d_1, ..., d_m$ be another candidate path and let $c_n, d_1 \notin S_i$.
   - $PQ = c_1, ..., c_n, d_1, ..., d_m$ is a candidate path if $c_n \in U(d_1)$, i.e. $d_1 \in U(c_n)$.
   - $PQ_1 = c_1, ..., c_n, d_1, ..., d_m$ and $PQ_2 = c_1, ..., c_n, d_2, ..., d_m$ are candidate paths, if $\overline{d_1 d_2} \cap U(c_n) \neq \emptyset$ and either $d_1 \notin U(c_n), d_2 \notin U(c_n)$ or $d_1 \in U(c_n), d_2 \in U(c_n)$.

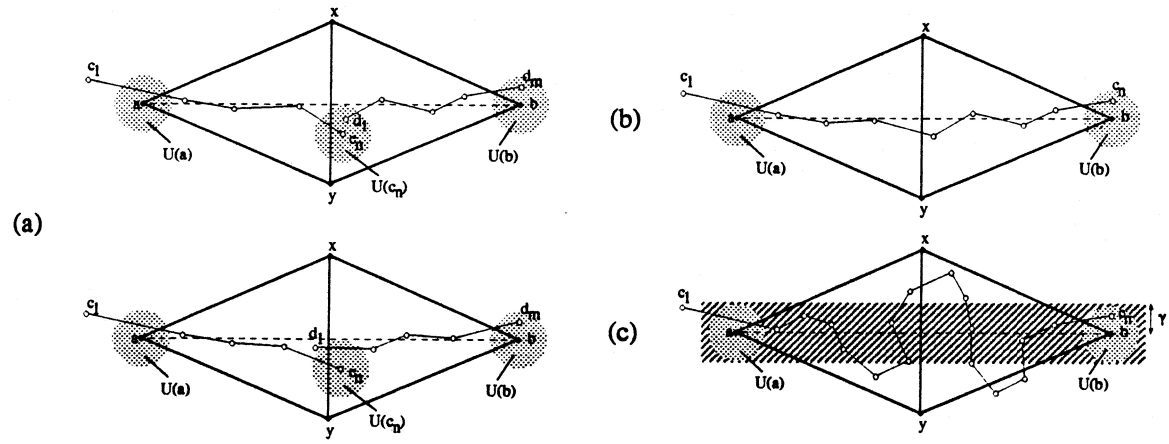

Figure 1: *Properties for generalized constraints.*

2. Property: *proper candidate paths*

   Let $P = c_1, \ldots, c_n$ be a candidate path and let $t_1, \ldots, t_k$ be Delauney triangles forming a convex polygon R defined by the set of vertices $\{a_1, \ldots, a_l\} \in S_i$. Let none of the diagonals of R be a constraint. If there exist two vertices a_i, a_j $(i \neq j)$ such that $\overline{c_1 c_2} \cap U(a_i) \neq \emptyset$ and $\overline{c_{n-1} c_n} \cap U(a_j) \neq \emptyset$ we call path P a *proper candidate path* for vertices $a_i, a_j \in S_i$ (see figure 1(a) and figure 1(b)).

3. Property: *generalized constraint*

   Let $P = c_1, \ldots, c_n$ be a proper candidate path for vertices $a, b \in S_i$. Let $\mathrm{dist}(c, \overline{ab})$ denote the distance of point c to edge $\overline{ab}$ and let $\gamma = \gamma(\varepsilon_i)$ denote an error value depending on the error value ε_i of the current level.

   If P satisfies $dist(c_j, \overline{ab}) \leq \gamma = \gamma(\varepsilon_i)$, $1 < j < n$, the proper candidate path P may be generalized by edge $\overline{ab}$ in the current level (see figure 1(c)).

We split the computation of the generalized constraints into a preprocessing step and a search process.
Let G_c' be the subgraph of the constraint graph G_c consisting of all constraints, i.e. line segments and their corresponding endpoints, which have not yet been inserted into the triangulation. Starting with G_c', a *search graph SG* is built by performing the following transformation steps (i is the current level of the pyramid):

1. (corresponds to property 1 - "candidate paths")
   Let $l = \overline{ab}$ and let $l' = \overline{cd}$ be two disjoint edges of G_c' with $x \cap U(s) = \emptyset$ for all $x \in \{a, b, c, d\}$ and for all $s \in S_i$. Let $l' \cap U(a) \neq \emptyset$.
   (a) If $c \cap U(a) \neq \emptyset$ but $d \cap U(a) = \emptyset$, we add the edge $\overline{ac}$ in SG.
   (b) If $c \cap U(a) \neq \emptyset$ and $d \cap U(a) \neq \emptyset$, we add the edges $\overline{ac}$ and $\overline{ad}$ in SG.
   (c) If $c \cap U(a) = \emptyset$ and $d \cap U(a) = \emptyset$, we add the edges $\overline{ac}$ and $\overline{ad}$ in SG.
2. (corresponds to property 2 - "proper candidate paths")
   (a) Each vertex a of G_c' with $a \in U(s)$, $s \in S_i$ is replaced by s in SG, i.e. s is inserted into SG if it is not yet contained in SG, and all vertices of SG which were connected to a are now linked to s.
   (a) Each edge $l = \overline{ab}$ of G_c' with $a \cap U(s) = b \cap U(s) = \emptyset$ for all $s \in S$ and $l \cap U(s) \neq \emptyset$ for at least one $s \in S_i$ is replaced by edges $\overline{as}$ and $\overline{bs}$ in SG.

All edges of G_c' which are not modified by these transformation steps remain unchanged in SG. All vertices of SG contained in S_i are marked, all other vertices of SG are not marked.
Since the size of the circles $U(\bullet)$ depends on the level i of the pyramid, we have to recompute the search graph each time we construct a new level of the pyramid. When creating a new level of the pyramid step 1 in the computation of the search graph is performed by testing the edges of G_c' against each other and step 2 is performed by testing each edge/vertex of G_c' against each vertex in S_i. While constructing level $i+1$ of the constrained Delauney pyramid, the search graph can be updated after each point insertion by computing the difference to its predecessor.
The search for the generalized constraints is performed by a *depth first search* on the search graph SG. In SG a generalized constraint between vertex p and vertex q is represented by a path connecting p and q which satisfies property 3 (generalized constraint). The construction of the search graph SG implies that properties 1 and 2 are always satisfied.

Let p be the last point which has been inserted into the triangulation during the reconstruction process. By applying a depth first search-type algorithm we search for paths in SG starting at point p and ending at any marked vertex q of SG. These paths have to satisfy property 3 (generalized constraint), i.e. we only have to consider edges e such that $dist(a, \overline{pq}) \leq \gamma$ for each endpoint a of e. Paths found by this algorithm may be generalized by edge pq. In addition we have to check whether the generalized constraints are disjoint with the "real" constraints, i.e. the edges of G_c which have already been inserted into the triangulation. Since generalized edges are less important and the constraints may intersect in their endpoints only, we have to delete all generalized constraints intersecting "real" constraints. A detailed algorithm can be found in [Vo93].
We already mentioned above that the constraints must be inserted into the triangulation if the corresponding endpoints have been included. The generalized constraints are inserted after the "real" constraints. Therefore we have to check for intersection of "real" and generalized constraints after computing the generalized constraints. During the triangulation process the generalized constraints are treated like "real" constraints. But if we insert a constraint ("real" or generalized) intersecting a generalized one which has been inserted earlier this generalized constraint is removed from the set of generalized constraints appearing in the triangulation.
The size of the surroundings defined by $U(\bullet)$ and γ depends on the maximum error ε_i but not on the data set which is processed and may be chosen unsuitably for a specific data set. Since a data dependant definition is not possible and the generalization is a heuristic approach we may produce local errors.
Finally, we determine the worst case time complexity for the computation of a constrained Delauney pyramid. The number l of constraints depends on the number n of vertices. The number m of levels in the pyramid is a small and constant value independent of n. It can be shown that the computation of a constrained Delauney pyramid requires $O(n^3)$ time.

3. The constrained Delauney pyramid in database systems

Topographic data sets usually consist of a large number of data items (e.g. topographic data derived from remote sensing data). The geographic information system which is used to access the topographic data stores these data sets in an underlying database system. If we use the constrained Delauney pyramid to browse and zoom in such sets of geometric data, the size of these data sets may exceed the main memory size. Hence we have to provide secondary storage structures and algorithms working on such structures to integrate our multiresolution surface model into a database system.
Geographic information systems offer browse- and zoom-operations on geographic data sets. Hence, we assume that the given sets of point elevation data and line segments are stored using appropriate spatial data structures for geometric objects, e.g. quadtree [Sam90b], R-Tree [Gu84], R*-Tree [Be90] or grid file [NHS84]. We propose to use a modified R*-Tree to store the triangles and edges of the generated triangulations. Since we do not want to store the point data twice, we further assume that we can access the point elevation data via tuple identifiers.
As presented in [Vo93] there is an upper bound for the size of a constrained Delauney pyramid in main memory. If the given data set and the resulting main memory structure of the constrained Delauney pyramid fit into main memory, we compute the pyramid as presented in section 2. In a second step we transform the resulting pyramid into the format needed for the secondary storage data structure and write the data to disk. If the given data set and the resulting constrained Delauney pyramid do not fit into memory, we compute several parts which fit into memory and combine these parts. We use a *divide-&-conquer* algorithm for the stepwise construction of the constrained Delauney pyramid.
In the following we assume that the topographic data (i.e. the set S of points and the set L of line segments) are stored in a database using spatial data structures. For our purposes we further assume that the spatial data structure used to store the point data performs a quadtree-like subdivision of the data space (a comprehensive overview on spatial data structures is given in [Sam90a] and [Sam90b]). Since the divide-&-conquer paradigm requires to divide the given data set efficiently, we assume that there is a function which enables splitting the given data set on the internal boundaries of the data structure.
The divide-&-conquer algorithm can be described by the following steps:

- *Divide phase (divide the topographic data)*
  Let $\mathfrak{I}$ be an index set. We consider a rectangular subdivision of the data space which induces a partitioning of S into pairwise disjoint sets $S^k, k \in \mathfrak{I}$. The set L of line segments is divided into sets $L^k, k \in \mathfrak{I}$ where L^k contains all constraints $\overline{ab}$ with $a \in S^k$ or $b \in S^k$. Obviously, these subsets fulfill the property $S = \bigcup_{k \in \mathfrak{I}} S^k, L = \bigcup_{k \in \mathfrak{I}} L^k$. Since the upper bound of memory consumption of a constrained Delauney pyramid can be estimated based on the size of the sets S^k, L^k (see [Vo93] for further details), we can choose the subsets S^k, L^k small enough such that their (constrained) Delauney pyramid can be computed in main memory.

- *Construction phase*
  For each $k \in \mathfrak{I}$: Compute the (constrained) Delauney pyramid of S^k, L^k in main memory and store the resulting data structure on secondary storage.
- *Merge phase*
  Perform a merge of the (constrained) Delauney pyramids on secondary storage.

In the merge phase the algorithm merges the triangulations of the pyramids level by level starting at level 0. To merge two triangulations, we use the algorithm presented in [MV93] which is an extension of the merge algorithm for constrained Delauney triangulations presented in [Che89]. This extension fixes several problems of the original algorithm. Details can be found in [MV93], [Che89] and [LS80].
Let S^l and S^r denote the point data of the triangulations to be merged. We call $a \in S^r$ visible from S^l if there is a $b \in S^l$ with the property $\overline{ab} \cap t = \emptyset$ for all triangles t in the triangulation of S^r.
To merge two triangulations, it is sufficient to load the set $T = T_l \cup T_r$ of required triangles into main memory. T_r consists of the following triangles:

- All triangles at the current level of the pyramid having at least one vertex which lies on the boundary of the convex hull of S^r and is visible from S^l.
- All triangles t in the triangulation of S^r such that there is a point of S^l lying inside the circumcircle of t.
- All triangles in the triangulation of S^r intersected by a constrained edge $l = \overline{ab}$ with $a \in S^r$ and $b \in S^l$.
- All triangles in the triangulation of S^r sharing at least one edge with a triangle chosen by the criteria above. This is required due to the merge algorithm of [LS80] which is part of the merge algorithm presented in [MV93].

Since we use a merge algorithm for constrained Delauney triangulations we have to insert all constraints ("real" and generalized) connecting the two partial triangulations into both triangulations and then we have to apply the algorithm of [MV93] to merge the triangulations. Let G_c' be the subgraph of the constraint graph G_c consisting of all line segments $l \in L^l \cup L^r$ which intersect the set T of triangles defined above and have not yet been inserted into the triangulation. The subgraph G_c' is used to search generalized constraints between the triangulations of S^l and S^r as follows: For each level k of the pyramid processed by the merge step we use G_c' to compute a search graph SG according to the rules presented in section 2. We search for generalized constraints $\overline{ab}$ by applying the search method presented in section 2 to the search graph SG. The endpoints a and b of the generalized segment must be chosen from the point sets S^l and S^r, i.e. $a \in S^l$ and $b \in S^r$ or vice versa.
The whole merge process of two constrained Delauney pyramids may be described by the following algorithm:
For each level i of the pyramid do:

- Let T_l, T_r be the sets of triangles as defined above, and let $S_l(T_l) \subseteq S^l$, $S_r(T_r) \subseteq S^r$ be the corresponding sets of vertices. Load these sets into main memory.
- Let L^1 be the set of "real" constraints $l = \overline{ab}$ with $a \in S_l(T_l)$ and $b \in S_r(T_r)$. Remove all triangles from T_l, T_r intersected by any $l \in L^1$.
  Insert each $l \in L^1$ into both triangulations.
- Let L^2 be the set of generalized constraints computed by the method presented in the previous paragraph. For each $l \in L^2$: If l does not intersect any "real" constraint then remove from both triangulations all previously inserted generalized constraints which intersect l. Further, remove all triangles from T_l, T_r intersected by l, and insert l into both triangulations. If l intersects any "real" constraint, discard l.
- Use the algorithm presented in [MV93] to merge both triangulations.

4. Conclusions

We have presented a multiresolution surface model called constrained Delauney pyramid which reconstructs a given topographic surface at a variety of predefined resolutions. Since our model is based on processing point elevation data and linear surface features a "good" reconstruction of the original surface is achieved for each level of detail. Delauney triangulations allow us to process non-uniformly distributed point sets and result in triangulations with only few non-regularly shaped triangles.
Since geographic information systems store very large sets of topographic data, we presented a method for constructing a constrained Delauney pyramid on secondary storage by a divide-&-conquer approach. The integration of our model and its underlying data structure into an existing database system allows different users to browse and zoom on the data structure concurrently without having to recompute the triangulations.
The main benefit of the constrained Delauney pyramid for the surface reconstruction process is the generalization of

the constraints which are given for the finest resolution. Using the constraint generalization characteristic topographic features are also visible at coarse resolutions. We defined three properties how to derive generalized constraints from the original set of line segments. Due to the fact that the process of generalization is not mathematically exact, our generalized constraints may be inaccurate in specific situations and may therefore produce local errors in the triangulations. We are currently implementing the constrained Delauney pyramid. This implementation will include the implementation of the pyramid as a search structure for an extensible database system. Using the implementation we plan to apply our model to real-world data and to assess the constraint generalization. These experiments will help to determine the exact size of neighborhoods $U(\bullet)$ and error values γ depending on the predefined maximum error for each level of the pyramid.

Literature

[Be90] N. Beckmann et. al.; "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles"; *Proc. 1990 ACM SIGMOD Int. Conf. on Management of Data*, Atlantic City, NY, 1990, pp. 322-332

[BV84] R. Barrera, A.M. Vazques; "A Hierarchical Method for Representing Relief"; *Proc. Pecora IX Symposium in Spatial Information Technologies for Remote Sensing Today and Tomorrow*, Sioux Falls, South Dakota, Oct. 1984, pp. 87-92

[Che89] L.P. Chew; "Constrained Delauney Triangulations"; *Algorithmica*; Vol. 4, 1989; pp. 97-108

[CT86] Z.T. Chen, W.R. Tobler; "Quadtree Representation of Digital Terrain"; *Proc. Autocarto*, London, 1986; pp. 475-484

[Flo84] L. De Floriani et. al.; "A Hierarchical Structure for Surface Approximation"; *Computer and Graphics*; Vol. 8, No. 2, 1984; pp. 183-193

[Flo87] L. De Floriani; "Surface representations based on triangular grids"; *The Visual Computer*; Vol. 3, No. 1, Feb. 1987; pp. 27-50

[Flo89] L. De Floriani; "A Pyramidal Data Structure for Triangle-Based Surface Description"; *IEEE Computer Graphics & Applications*; Vol. 9, No. 2, Mar. 1989; pp. 67-78

[FP92] L. De Floriani, E. Puppo; "An On-Line Algorithm for Constrained Delauney Triangulation"; *CVGIP: Graphical Models and Image Processing*; Vol. 54, No. 3, Jul. 1992; pp. 290-300

[GG79] D. Gomez, A. Guzman; "Digital Model for Three-Dimensional Surface Representation"; *Geo-Processing*; Vol. 1, 1979; pp. 53-70

[Gu84] A. Guttman; "R-Trees: a dynamic index structure for spatial searching"; *Proc. 1984 ACM SIGMOD Int. Conf. on Management of Data*, 1984, pp. 47-57

[LS80] D.T. Lee, B.J. Schachter; "Two Algorithms for Constructing a Delauney Triangulation"; *Int. Journal of Computer and Information Sciences*; Vol. 9, No. 3, 1980; pp. 219-242

[MV93] J.-M. Moreau, P. Volino; "Constrained Delauney Triangulation Revisited"; *Proceedings 5th Canadian Conference on Computational Geometry*; Waterloo, 1993, pp. 340-345

[NHS84] J. Nievergelt, H. Hinterberger, K.C. Sevcik; "The Grid File: An Adaptable, Symmetric Multikey File Structure"; ACM TODS; Vol. 9, No. 1, 1984; pp. 38-71

[PF87] J. Ponce, O. Faugeras; "An Object Centered Hierarchical Representation for 3D Objects: The Prism Tree"; *Computer Vision, Graphics and Image Processing*; Vol. 38, No. 1, Apr. 1987

[PS85] F.P. Preparata, M.I. Shamos; *Computational Geometry - An Introduction*; Springer, Berlin, 1985

[Sam90a] H. Samet; *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*; Addison-Wesley, Reading, MA, 1990

[Sam90b] H. Samet; *The Design and Analysis of Spatial Data Structures*; Addison-Wesley, Reading, MA, 1990

[SG85] F. Schmitt, B. Gholizadeh; "Adaptive Polyhedral Approximation of Digitized Surfaces"; *Proc. SPIE Conf. on Computer Vision in Robots, SPIE*, Bellingham, Wash.; Vol. 595, 1985; pp. 167-171

[SP92] L. Scarlatos, T. Pavlidis; "Hierarchical Triangulation Using Cartographic Coherence"; *CVGIP: Graphical Models and Image Processing*; Vol. 54, No. 2, Mar. 1992; pp. 147-161

[Vo93] A. Voigtmann; "Hierarchische Repräsentation diskreter geometrischer Daten"; Diplomarbeit; Fachbereich Mathematik, Universität-GH Siegen, 1993

[Wei85] K. Weiler; "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments"; *IEEE Computer Graphics and Applications*; Vol. 5, No. 1, Jan. 1985; pp. 21-40

[Woo85] T.C. Woo; "A Combinatorial Analysis of Boundary Data Structure Schemata"; *IEEE Computer Graphics and Applications*; Vol. 5, No. 3, Mar. 1985; pp. 19-27