Title
Representation and Understanding of Scientific Data

Author
Konkle (Schussman), Shirley Ellen

Publication Date
2000

Peer reviewed

Representation and Understanding of Scientific Data

BY

SHIRLEY ELLEN KONKLE
B.S. (University of California at Davis) 1996

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTERS OF SCIENCE

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

_____

_____

_____

Committee in Charge

2000

Abstract

Representation and Understanding of Scientific Data

by

SHIRLEY ELLEN KONKLE

MASTERS OF SCIENCE in Computer Science

University of California at DAVIS

Professor Kenneth I. Joy, Chair

Outstanding problems in scientific visualization include creating a generalized multiresolution data representation scheme that can handle high-dimensional, scattered data, and using feature detection for data understanding and exploration. To address the first problem, we present a data hierarchy of Voronoi diagrams as a versatile solution. Given an arbitrary set of points in the plane, the goal is the construction of an approximation hierarchy using the Voronoi diagram [39] as the essential building block. We have implemented two Voronoi diagram-based algorithms to demonstrate their usefulness for hierarchical scattered data approximation. The first algorithm uses a constant function to approximate the data within each Voronoi cell, and the second algorithm uses Sibson's interpolant [35]. The second topic of this thesis addresses using feature detection for data understanding and exploration. Betti numbers, well know in classical topology, provide a method for feature detection and can aid in the exploration of complex large-scale time-varying data sets. We

present a fast algorithm that calculates Betti numbers for isosurfaces along with a set of examples of their interpretation and use. Once an isosurface is obtained from a data set, calculating Betti numbers requires time and space proportional to the size of the isosurfaces. The presented algorithm can be used for large data sets, since the overhead of obtaining Betti numbers is relatively small.

Professor Kenneth I. Joy
Dissertation Committee Chair

To Tim

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Patrick Moran for his continued help with the Betti number project, and Chris Henze for the use of his molecular data sets. I would also like to thank TJ Jankun-Kelly for his mathematical explanations of topology fundamentals.

In general, I would like to thank all the members of CIPIC's visualization and computer graphics group for the fun and collaborative work environment, and specifically to Martin Bertram for his input and collaborative work on Voronoi hierarchies. In addition, I would like to give my specials thanks to my advisor Ken for his help, focus, and inspiration.

Chapter 1

Introduction

This thesis consists of two parts. The first part introduces a novel means for multiresolution data hierarchies based on Voronoi diagrams. The second part describes a new, fast method, using Betti numbers, to understand and explore isosurfaces extracted from scientific data sets.

We present a new solution for constructing multiresolution data representations: data hierarchies based on Voronoi diagrams. This approach is motivated by the need to interactively explore very large data sets that consist of scattered or arbitrarily gridded data. A hierarchy of Voronoi diagrams is a natural solution for a number of reasons. First, Voronoi diagrams define a "natural mesh" for scattered data, i.e., data without explicit point connectivity. Second, point insertion and deletion operations for Voronoi diagrams are expected constant-time operations [27, 13]. In addition, Voronoi cells, or tiles, can be sorted in depth in linear time, which is important for volume visualization. Furthermore Voronoi diagrams can be extended to n dimensions. Although the Voronoi diagram's dual–

the Delaunay triangulation (or complex)–has very similar properties, the Voronoi diagram provides a more intuitive tessellation and it is unique.

A hierarchy is generated by inserting points into the Voronoi diagram where error is maximal. We use two interpolation methods, one based on constant functions and the other one based on Sibson's interpolant [35, 14] and discuss their advantages and disadvantages. Our algorithm is a top-down approach that produces a hierarchy of Voronoi diagrams, where each diagram has an associated approximation that is within a certain threshold of the original data. The error calculations are local, which makes the algorithm fairly efficient. Our algorithm utilizes only the original data points for refinement, which allows for a very compact representation.

We also present a new method for data understanding and exploration that uses *Betti numbers* to provide information about the topology of isosurfaces extracted from a trivariate scalar field. The first three Betti numbers have intuitive physical interpretations that provide information about the number of *connected components*, the number of *tunnels*, and the number of *closed regions* of a simplicial complex. We can use these values to obtain additional information about a scalar field, and to detect features within the field. For example, using the isosurface of the Laplacian of the electron density of two molecules, we can use Betti numbers to determine when the molecules have bonded. We can also use Betti numbers to prove the correctness of mesh reduction schemes. (Given a three-dimensional mesh M and a reduced version of the mesh, called M', we can examine the Betti numbers of the two meshes to determine if the topology of the reduced mesh is the same as the topology of the original one.) In addition, we can use Betti numbers to highlight isovalues

where complex topology is implied by an isosurface. This allows a user to quickly focus on interesting parts of a data set.

An algorithm for efficient calculation of Betti numbers for triangulated isosurfaces is given. This contribution includes a new algorithm for computing the second Betti number of the triangulated mesh that does not rely on a tetrahedral mesh. We also provide a set of examples for the use of Betti numbers for data exploration.

Chapter 2

Previous Work

A number of approaches have been developed during the past two decades to visualize scientific data that is scattered [18] or defined on very large and often highly complicated grids. The most common methods for hierarchical data representations are based on mesh reduction. These techniques associate a mesh with the data sites, apply various reduction techniques to the mesh, and use reduced meshes as basis for progressive visualization at multiple levels of detail.

Several data decimation and hierarchical schemes have been developed over the past years by the computer graphics and visualization communities. For example, Schroeder et al. [34] and Renze and Oliver [32] have developed algorithms that simplify a mesh by removing vertices. Removing a vertex creates a hole in a mesh that must be re-triangulated, and several strategies may be used.

Hoppe [22, 23] and Hoppe and Popović [31] describe a progressive-mesh representation for triangle meshes. They describe a continuous-resolution representation based on

edge-collapse operations. The data reduction problem is formulated in terms of a global mesh optimization problem ordering the edges according to an energy function to be minimized. As edges are collapsed, and the priorities of the edges in the neighborhood of the transformation are recomputed. The result is an initial coarse representation of a mesh, and a linear list of edge-collapse operations. Garland and Heckbert [19] utilize a different strategy, based on quadratic error metrics for efficient calculation of a hierarchy. Hoppe [24] has extended this method to multidimensional meshes with appearance attributes.

Trotts et al. [38, 37] and Staadt and Gross [36] have extended the edge collapse paradigm to tetrahedral meshes. Cignoni et al. [4] also treat the tetrahedral mesh problem. They use a top-down Delaunay-based procedure to define a tetrahedral mesh that represents a three-dimensional set of points. The mesh is refined by selecting a data point whose associated function value is poorly approximated by an existing mesh and inserting this point into the mesh. The mesh is modified locally to preserve the Delaunay property.

As part of their work on the topology of *alpha shapes*, Delfinado and Edelsbrunner[12] developed an incremental algorithm that computes the Betti numbers β_0, β_1, and β_{d-1} for a simplicial complex $\mathcal{K}$ embedded in the d-sphere $\mathcal{S}^d$. They illustrate the use of this method when calculating β_0, β_1, and β_2 for tetrahedral meshes. Their procedure requires $O\left(n\alpha(n)\right)$ time and $O(n)$ storage, where n is the size of the simplicial complex and α is the slow-growing inverse of the Ackermann function defined by Cormen et al. [7], p452 as

$$\alpha(m,n) = \min\{i \geq 1 : A(i, \lfloor m/n \rfloor) > \lg n\},$$

where m is the number of operations and n is the number of items in the set. To give an idea of how slow-growing the function is, $\alpha(m,n) \leq 4$ for $m \leq 10^{80}$, where 10^{80} far exceeds

the estimated number of atoms in the universe. We review Delfinado and Edelsbrunner's incremental algorithm for the case of β_0 in Section 4.2.

The first three Betti numbers characterize the topology of isosurfaces. We use the results of Delfinado and Edelsbrunner's work to calculate β_0, and develop a new algorithm for β_2, which only uses the isosurface to determine closed isosurface components, but does not consider the underlying tetrahedralization. The time and space bounds are $O(n\alpha(n))$ and $O(n)$ where n is the number of triangles of the isosurface.

Chapter 3

Hierarchical Voronoi Diagrams

A Voronoi hierarchy consists of a set of Voronoi diagrams and interpolating functions defined on the Voronoi diagrams that approximate a given data set at different resolutions and qualities of approximation. Any implementation requires methods for selecting points from a given finite data set, choosing an interpolant for each Voronoi cell, and choosing an error metric to determine the overall approximation quality of each level in the Voronoi hierarchy.

3.1 Refinement of a Voronoi Diagram

Given a point set $D = \{p_1, \ldots, p_n\}$ and n associated functions values $f_1, \ldots, f_n$, we define a sequence of Voronoi diagrams $V^{n_0}, V^{n_1}, \ldots, V^{n_h}$, where each V^i is defined by i points selected from D. The initial Voronoi diagram V^{n_0} is defined by the n_0 points of D that lie on and completely define the boundary of the convex hull of the given point set.

An intermediate Voronoi diagram V^{n_k} is refined by inserting additional data points

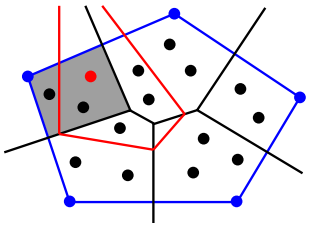


Figure 3.1: Example of vertex insertion. Black lines: current Voronoi diagram; black polygon: convex hull of point set; grey region: Voronoi cell with maximal error; grey point: vertex p_{c_j} with maximal error ε_{max}; grey lines: Voronoi cell to be inserted.

into the Voronoi diagram in cells of high error as is shown in Figure 3.1. More specifically, a set of Voronoi cells with high errors is identified, and a point is inserted into each cell c_j in that set. The error ε_{c_j} is calculated using the L^1 norm, an average of the error over all n_{c_j} data points of D lying in cell c_j. We define the error of cell c_j as

$$\varepsilon_{c_j} = \frac{1}{n_{c_j}} \sum_{p_i \in c_j} \|f(c_j, p_i) - f_i\|. \tag{3.1}$$

where $f(c, p)$ is an interpolant over cell c_j containing p_i and where f_i denotes the associated function value at p_i. Once ε_{c_j} is calculated, it is compared with some threshold value to determine whether or not it belongs to the set of cells to be refined. It is convenient to determine the threshold as a percentage of the average global error ε_{avg}, which we define as

$$\varepsilon_{avg} = \frac{1}{n} \sum_{j=1}^{n_k} \sum_{p_i \in c_j} \|f(c_j, p_i) - f_i\|. \tag{3.2}$$

Once the set of cells to be refined is determined, a point $p_{c_j} \in D$ is inserted into each cell c_j. Ideally, p_{c_j} defines a cell that eliminates, or at least minimizes, the error in the resulting local cell configuration. Rather than searching exhaustively for the optimal point, we simply choose the point p_{c_j} in c_j with the largest error ε_{max}, where

$$\varepsilon_{max} = \max_{p_i \in c_j} \|f(c_j, p_i) - f_i\|. \tag{3.3}$$

3.2 Constructing the Hierarchy

The following pseudocode describes the basic algorithm for generating a hierarchy of Voronoi diagrams.

Algorithm: Voronoi Hierarchy Construction

Input:

- Set of n points $p_i = (x_i, y_i)$ and associated scalar or vector function values f_i, $i = 1, \ldots, n$
- Number of levels, h, to be calculated and error tolerances for all levels, called ε_k, $k = 1, \ldots, h$

Output:

- Set of h Voronoi diagrams, where the global error associated with each Voronoi diagram V^{n_k} is smaller than the level-specific global error tolerance ε_k

Steps of the Algorithm:

- Determine minimal point set defining boundary polygon of convex hull of given points (2D case).
- Create initial Voronoi diagram for this minimal point set.

- Compute global approximation error for initial Voronoi diagram, called V^{n_0}.
- Assuming that the global approximation error of V^{n_0} is larger than ε_1, determine a set of cells in V^{n_0} with large errors.
- For each cell in this set, choose an appropriate data point in D that lies in it and update the diagram accordingly.
- Check whether the global approximation error of refined Voronoi diagram still exceeds ε_1.
- Continue the process of point selection and insertion until diagram's global error approximation is smaller than ε_1; call this Voronoi diagram V^{n_1}.
- Construct Voronoi diagrams $V^{n_2}, \ldots, V^{n_h}$ in the same manner.

3.3 Point Insertion and Selection

One reason Voronoi hierarchies are a general and suitable format for multiresolution representations is that there are multiple ways to choose points for refinement. Rather than exploring all possibilities and determining an optimal solution, a research topic in its own right, we developed the method described in Section 3.1, which is designed to be fast, to refine Voronoi diagrams adaptively, and to capture patterns quickly in the underlying data. The method can also be extended to higher-dimensional domains.

The method of point insertion described in Section 3.1 refines a Voronoi diagram in a way such that it captures high-gradient regions and discontinuities early in the refinement process. The point insertion strategy detects "extreme values" first, since ε_{max} is always

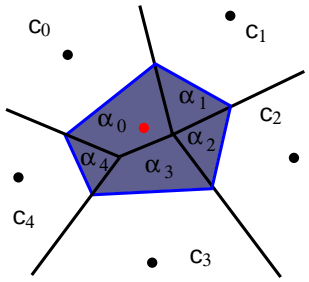


Figure 3.2: Example of Sibson interpolant. Black lines: current Voronoi diagram; c_i: cells in current Voronoi diagram; grey point: simulated insertion point; grey region: simulated cell resulting from point insertion; α_i: regions where simulated cell overlaps c_i

the maximum or minimum function value in a cell c_j. If the point p_j defining c_j is defined by a maximum value in a previous iteration, then p_{c_j} is a minimum value and vice versa.

A problem arises when there are multiple points in cell c_j with the same maximal error value ε_{max}. In this case, a random point is selected from the set of candidates for insertion. Random selection is crucial in this context, as the order of indices of the points in D should not bias point insertion.

Selecting an appropriate threshold value to determine which cells should be refined is another issue. As stated in Section 3.1, when a cell error ε_{c_j} is greater than a threshold value, the cell should be refined. In the context of data approximation, high threshold values are appropriate since only areas with high errors will be refined. Unfortunately, it is difficult to determine a "good" threshold value for an arbitrary data set. Another approach is to only insert a point in a cell with maximal error.

3.4 Interpolation Functions

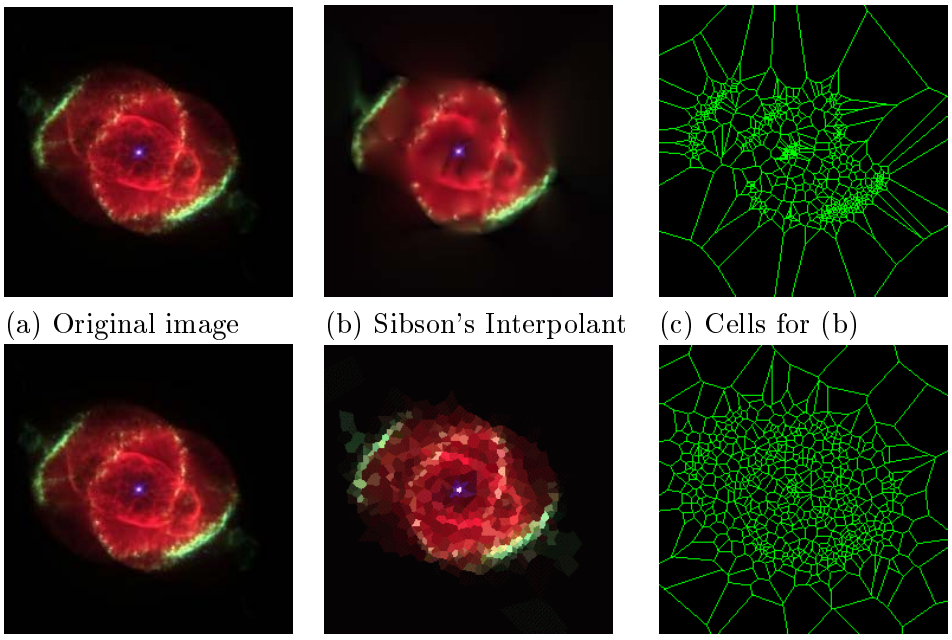
In principle, any function that interpolates values at the cell centers can be used. We implemented a piecewise-constant approach and in a collaborative project with Martin Bertram, also used Sibson's interpolant, see Figures 3.3b and Figure 3.3c. The example provides a basis for comparing interpolants.

Using a constant value per cell, where the value is that of the defining point (the "center") of the cell, is a simple and efficient interpolant. The constant function can be rendered quickly, which means that more data points can be rendered in the same amount of time. A piecewise-constant representation permits the representation of discontinuities, but cannot represent smoothly varying data at the same time.

Sibson's interpolant is a smoothly varying function that smoothly represents the underlying data. Sibson's interpolant is based on blending the function values f_j associated with the points defining a Voronoi diagram. The resulting interpolation defines a smooth function that is C^1-continuous everywhere, except at the points themselves. The interpolating function $f(p)$ is evaluated at a point p by "simulating its insertion" into the Voronoi diagram, without actually changing the Voronoi diagram, and by estimating the areas a_j cut away from Voronoi cells c_j in a local neighborhood. The value of Sibson's interpolant at p is defined as

$$f(p) \;=\; \frac{\sum_j \; a_j f_j}{\sum_j \; a_j},$$

see Figure 3.2.



(a) Original image (b) Sibson's Interpolant (c) Cells for (b)

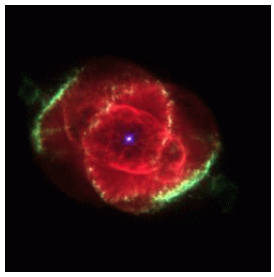
(d) Original image (e) Piecewise-constant function (f) Cells for (d)

Figure 3.3: Approximations of the Cat's Eye Nebula with 580 points.

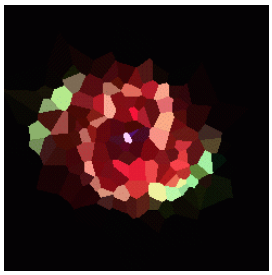
3.5 Results

We present results that compare the constant-function and Sibson's interpolant approach as well as two hierarchies generated with the piecewise-constant function approach. All input data sets are defined on a 250×250 uniformly spaced rectilinear grid, representing color images produced by the Hubble Space Telescope, courtesy of NASA.

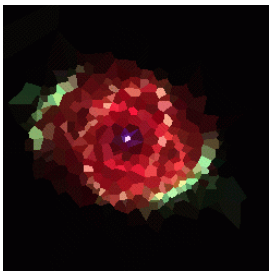
Figure 3.3 shows the characteristics of the constant-function approach and Sibson's interpolant for generating Voronoi diagrams. Sibson's interpolant tends to cluster cells around discontinuous regions of high contrast. Since there are fewer cells in the smoother regions, they are not represented well. The constant-function approach captures the discontinuities with fewer points and can capture more detail in the more smoothly varying



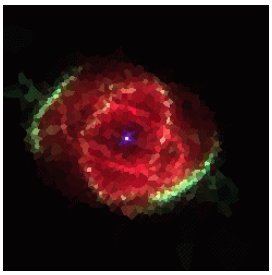
(a) Original image of Cats Eye Nebula



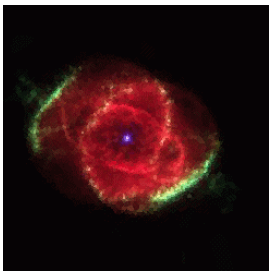
(b) 161 points



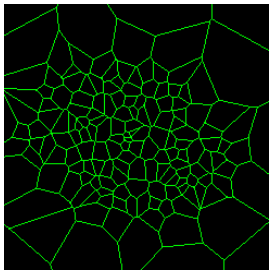
(c) 362 points



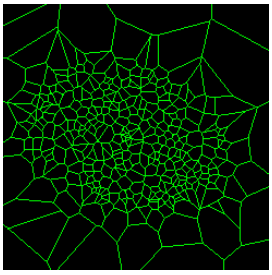
(d) 959 points



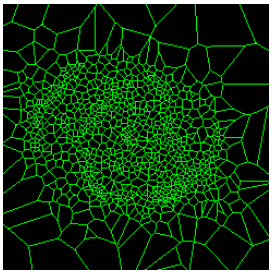
(e) 2734 points



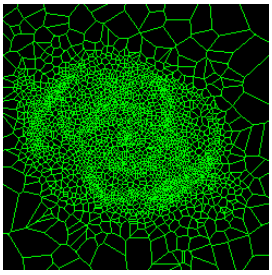
(f) Cells for (b)



(g) Cells for (c)



(h) Cells for (d)



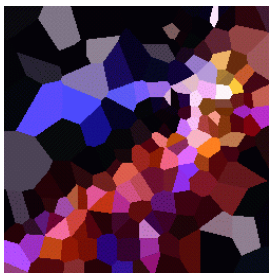
(i) Cells for (e)

Figure 3.4: Approximations of Cats Eye Nebula image with 161, 362, 959, and 2734 points.

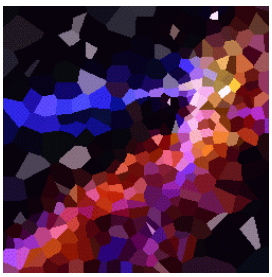
regions.

A hierarchy of Voronoi diagrams is shown in Figure 3.4. Figure 3.4b captures the rough shape of the nebula with only 161 points, or 0.26 % of the total number of points in the image. With 362 points, see Figure 3.4c, the areas of high contrast are represented well. Figure 3.4e represents low-contrast as well as high-contrast regions well with 2734 points, or 4.4% of the total number of points.

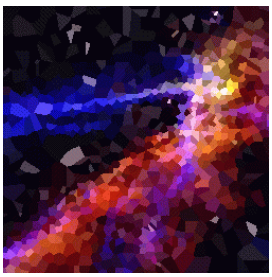
(a) Original image of Cygnus Loop Nebula



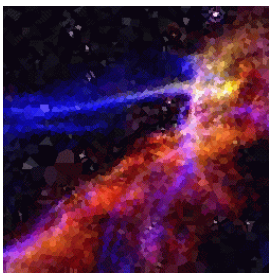
(b) 178 points



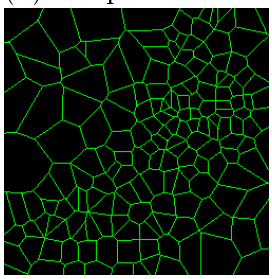
(c) 384 points



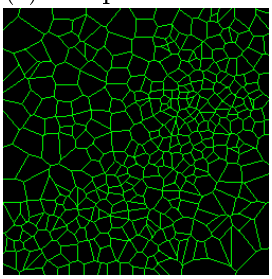
(d) 1034 points



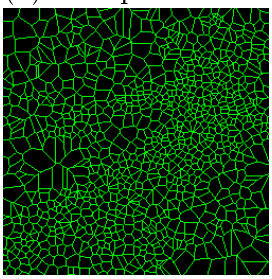
(e) 2915 points



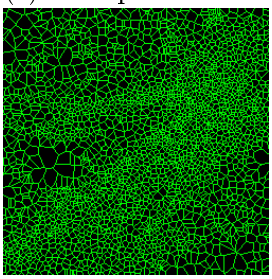
(f) Cells for (b)



(g) Cells for (c)



(h) Cells for (d)



(i) Cells for (e)

Figure 3.5: Approximations of Cygnus Loop Nebula image with 178, 384, 1034, and 2915 points.

A second Voronoi hierarchy represents the Cygnus Loop Nebula, shown in Figure 3.5. Shapes start to evolve when at least 384 points are used, see Figure 3.5c. At this level of resolution, stars are coarsely represented by large cells, creating high errors. The regions near the stars are refined in Figure 3.5e. The three bright stars that form an arc in the upper-left corner of the image are represented with 2915 points, or 4.1% of the total number of points. Several dimmer stars in that region are represented as well.

Cat's Eye Nebula			Cygnus Loop Nebula		
no. cells	L^1 Error [%]	L^2 Error [%]	no. cells	L^1 Error [%]	L^2 Error [%]
161	4.4	0.17	178	15	0.34
362	3.0	0.11	384	9.9	0.24
959	1.8	0.06	1034	6.5	0.15
2734	1.1	0.03	2915	4.1	0.08

Table 3.1: Numerical errors for two hierarchies of Voronoi diagrams.

The corresponding numerical results for Figures 3.4 and 3.5 are listed in Table 3.1. Although the numbers of cells are not exactly the same in the two hierarchies, they are close enough to allow a comparison. The L^1 and L^2 errors for the images of the Cygnus Loop Nebula are three to four times higher than those for the corresponding images of the Cat's Eye Nebula. This is due in part to the large background in the Cats Eye Nebula image, which contains no detail and can be represented with few cells. Although the Cygnus Loop image has a somewhat black background, the background is smaller and contains a large number of "dim" stars.

Chapter 4

Betti Numbers and the Exploration of Isosurfaces

4.1 Terminology and Concepts

Prior to discussing the use of Betti numbers for isosurface exploration, we summarize and review some needed definitions from topology.

A *k-simplex* σ is the convex hull of a set T of $k+1$ affinely independent points. A *vertex* is a 0-simplex, an *edge* is a 1-simplex, a *triangle* is a 2-simplex, and a *tetrahedron* is a 3-simplex. A proper *face* σ' of σ is any simplex σ' consisting of a non-empty set U of points, where $U \subset T$. A simplex $\sigma = [u_0, u_1, \ldots, u_k]$ can be oriented by defining an order for its vertices.

A simplicial complex $\mathcal{K}$ is a collection of simplices such that these two conditions hold: (1) if $\sigma \in \mathcal{K}$, then any face of σ is also contained in $\mathcal{K}$ and (2) two simplices contained

in $\mathcal{K}$ are either disjoint or intersect in a face of both. The k^{th} Betti number β_k of a simplicial complex $\mathcal{K}$ is the rank of the k^{th} homology group of $\mathcal{K}$, see [33].

The *boundary* of a simplex σ is defined as

$$\partial_k(\sigma) = \sum_{i=0}^{k}(-1)^i[u_0, u_1, \ldots, \hat{u_i}, \ldots, u_k],$$

where $u_0, u_1, \ldots, u_k$ are the points defining the simplex. For example, the boundary of a triangle is the set of its edges, and the boundary of a tetrahedron is the set of its bounding triangles. The boundary of a complex $\mathcal{K}$ uses the *boundary homomorphism*

$$\partial_k(\sum_j a_j\sigma_j) = \sum_j a_j\partial_k\sigma_j,$$

where a_j are integers and σ_j are the simplices in $\mathcal{K}$. For an oriented planar mesh of triangles, the boundary is the set of outer edges, since the inner edges cancel. In other words, each "double" inner edge has a positive weight for one direction and a negative weight for the other one. The boundary of a closed manifold triangulated mesh has a boundary of zero, since all edges cancel.

Betti numbers have mathematical meaning and properties. The 0^{th} Betti number β_0 is the number of connected components of a simplicial complex $\mathcal{K}$. The 1^{st} Betti number β_1 is the number of independent tunnels in $\mathcal{K}$. For example, a torus has two tunnels: one that goes through the center and one that goes around the center. A nice property of all closed, orientable surfaces is that they are all homotopic to a sphere with h handles, and β_1 is always $2 * h$, as is illustrated in Figure 4.1.

The 2nd Betti number β_2 is the number of closed regions in a simplicial complex $\mathcal{K}$. A regular triangulation of a sphere has an associated second Betti number $\beta_2 = 1$.

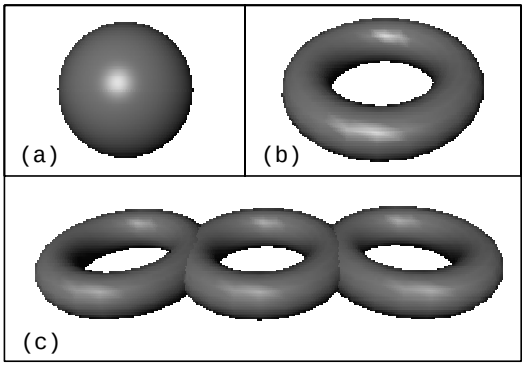


Figure 4.1: Orientable, closed surfaces: Betti number β_1 is two times the number of handles; (a) $\beta_1 = 0$; (b) $\beta_1 = 2$; (c) $\beta_1 = 6$.

Triangulations of two disjoint spheres have Betti numbers $\beta_2 = 2$. If two triangulated spheres intersect at a point, then there is one connected component, i.e., $\beta_0 = 1$, but two closed regions exist, which implies that $\beta_2 = 2$.

4.2 The Algorithm for Computing Betti 0, 1, and 2

A triangulated isosurface is a simplicial complex $\mathcal{K}$ of dimension 2. In the following, we provide algorithms to effectively calculate the Betti numbers β_0, β_1, and β_2. We use the *Euler number* (or *Euler-Poincaré Characteristic*) χ, which is defined as

$$\chi = \beta_0 - \beta_1 + \beta_2$$

or, equivalently, as

$$\chi = v - e + t,$$

where v, e, and t are the number of vertices, edges, and triangles in $\mathcal{K}$.

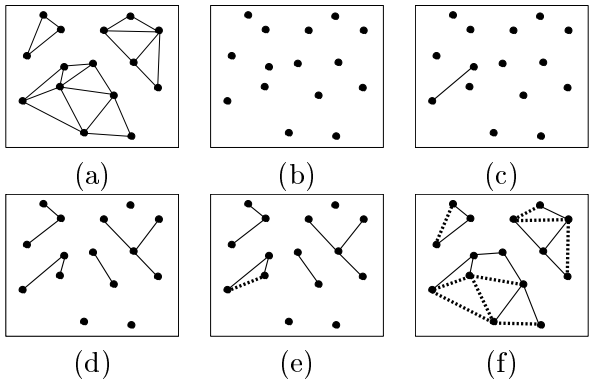


Figure 4.2: Detecting 1-cycles with incremental method: (a) original triangulation; (b) $\beta_0 = v$; (c) $\beta_0 = v - 1$; (d) $\beta_0 = v - 8$; (e) $\beta_0 = v - 8$; (f) $\beta_0 = v - e' = 3$.

4.2.1 Betti 0

The value of β_0 is calculated using Delfinato and Edelsbrunner's incremental approach[12]. Given an initial triangulation, see Figure 4.2a, consider the vertices only and ignore all edges and triangles, see Figure 4.2b. Betti number β_0 is initially set to be identical to the number of vertices. Adding the first edge connects two components, see Figure 4.2c, and β_0 is decremented by one. More edges that connect components are shown in Figure 4.2d. Other edges, shown as dotted lines in Figures 4.2d and 4.2e, do not connect two components since the components are already connected; instead, they form a cycle. The final number of components, shown in Figure 4.2f, is the number of vertices v minus the number of edges e', that do not form cycles. In our example, the number of components is 15-12=3.

The implementation is based on the union-find data structure, see Cormen et al. [7]. The initial set of vertices is used to initialize the data structure. Adding edges corresponds to *Union* and *Find* operations. First, a Find operation is done to see whether

an edge's two endpoints are already in the same set. If they are not in the same set, the Union operation is applied to the endpoints, and e' is incremented by one. If the endpoints were already in the same set, no further action is needed. The total run times for the Union and Find operations are $O(n\alpha(n))$, where α is the inverse of the Ackermann function.

The most difficult part of the algorithm is the construction of exactly one copy of each vertex and edge in the isosurface triangulation to define the input for the Union-Find data structure.

Many isosurface rendering algorithms generate a set of triangles where certain vertices and edges are represented multiple times. One method of avoiding this problem might be to compute and represent an isosurface directly with a topological data structure such as the specific quad-edge data structure, using, for example, the quad-edge data structure described by Guibas and Stolfi [20]. However, we chose to hash the vertices and edges, which is straightforward with the *NASA FEL Library* [28].

Hashing vertices of an isosurface is simple. Since each isosurface vertex is obtained from linearly interpolating function values along mesh edge, an isosurface vertex can be hashed on the associated mesh vertex, which has a unique tag in the FEL Library. Topologically, it does not matter where an isosurface point is located on an edge; it only matters that a point exists.

Assuming that an isosurface is generated using a marching-tetrahedra algorithm[42], edges in the isosurface triangulation are only slightly more difficult. Most edges lie on the faces of tetrahedra, see Figure 4.3a and 4.3c. These edges use the FEL tag for a triangle as a hash value. Fortunately, edges lying in the interior of tetrahedra, see Figure 4.3b, do

not need to be stored in the hash table. The reason is this: when a Union operation is first applied to the four edges of the isosurface that lie on faces of the tetrahedra, the interior edge can never connect two components. Therefore, the interior edge does not need to be stored.

4.2.2 Betti 2

Betti number β_2 is the number of closed regions in space, and it can be calculated by checking whether all components are closed. The easiest way to determine whether a component is closed is to perform a depth-first search from each triangle to each of its neighboring triangles. If one of the triangle's edges does not have a neighbor, then the isosurface touches the edge of the data set, and the surface is not closed. The total run time is $O(e)$.

There is a mathematical basis for the correctness of the algorithm. For a complex of triangles, β_2 is the number of $2-cycles$, or components, whose boundary is zero. Because all isosurface triangulations are orientable–the normal always facing "outward"– a boundary is easy to compute as a depth-first search is performed. The first time an edge is encountered in the search it will have either a positive or negative weight and is incrementally added to the value of the boundary. When the search reaches that triangle's neighbor, that triangle will use the opposite weight. As a result, the weights will always cancel, unless the surface hits the edge of the data set. Any surface that stays within the data set has a boundary of zero and is a 2-cycle.

For the algorithm to lead to correct results, two surfaces must not meet at an edge and surfaces cannot have "wings." In other words, an edge cannot be part of more than

two triangles. To understand the algorithm's correctness, it is necessary to see how edges are generated and degenerate cases are handled. First, all data points are classified as "+" or "-". If the data point is exactly equal, it is assigned a "-". All cases of a surface passing through a tetrahedron are listed in Figure 4.3. A point on the isosurface corresponds to an edge in the dataset that is bounded by a "+" and "-". Topologically, it does not matter where a point is located or if one of the endpoints are equal. Figure 4.3 shows that there are only two symmetric cases for a surface passing through a tetrahedra and two types of edges in those surfaces. The first type of edge lies completely inside the tetrahedron and is obviously part of only one surface. The second type of isosurface edge lies on a triangle face of the tetrahedron. Only one isosurface triangle touches it in the tetrahedron. By symmetry, the neighboring tetrahedron will also have one isosurface triangle at the edge. As a result, there will never be more than two triangles meeting at any edge in an isosurface generated by a tetrahedral mesh.

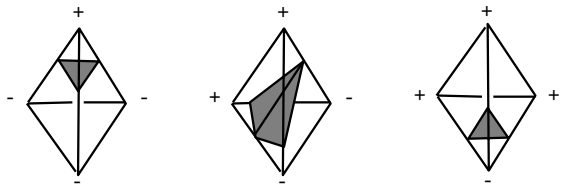


Figure 4.3: Two cases to be considered for an isosurface passing through a tetrahedron: (a) and (c) are symmetrical cases of a one-triangle configuration, and (b) shows a two-triangle configuration.

The algorithm requires time $O(n)$ time and space, where n is the number of triangles in the isosurface triangulation.

4.2.3 Betti 1

The value of β_1 is calculated from the Euler number χ, β_0, and β_2. Knowing the values of β_0, β_2, and χ, β_1 is given by

$$\beta_1 = \beta_0 + \beta_2 - \chi.$$

4.2.4 Extension to Hexahedral Meshes

Isosurfaces extracted from hexahedral meshes are more complex than isosurfaces extracted from tetrahedral meshes. The difficulties arise from ambiguous faces and multiple isosurfaces in a single hexahedral cell. Ambiguous faces, shown in Figure 4.4, have to be resolved consistently to preserve topology: if on one cell face the "+"'s are separated and the neighboring cell sharing this face separates the "-"'s, then there would be a crack in the isosurface triangulation. Our implementation uses the ambiguity decider from Nielson and Hamann to produce crack-free triangulations[29]. Even though the connectivity on faces should be uniquely decided with this method, floating point errors can potentially cause different results when the same shared face is calculated from the two hexahedral cells that share it. A simple solution to prevent cracking is to hash each ambiguous face first, apply the ambiguity decider once, and reference the result from the hash table.

One also has to deal with multiple isosurfaces in a single hexahedral cell. The depth-first search for calculating β_2 is more complicated, because there are two isosurfaces that pass through an ambiguous face. When calculating β_2, extra bookkeeping must be done to ensure that the depth-first search chooses the correct edge.

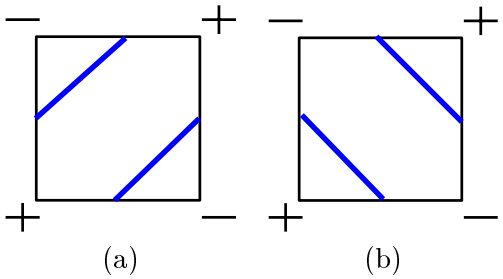


Figure 4.4: Example of an ambiguous face: (a) Positive points are separated; (b) Negative points are separated

4.3 Results

Figure 4.5 shows isosurfaces of the Laplacian of the electron density of adenine. For this example, $\beta_1 = 20$, indicating the existence of more than the ten components visible. Clipping these isosurfaces reveals the additional parts inside the visible triangulations. Clearly, the Betti number reveals additional information about the isosurface.

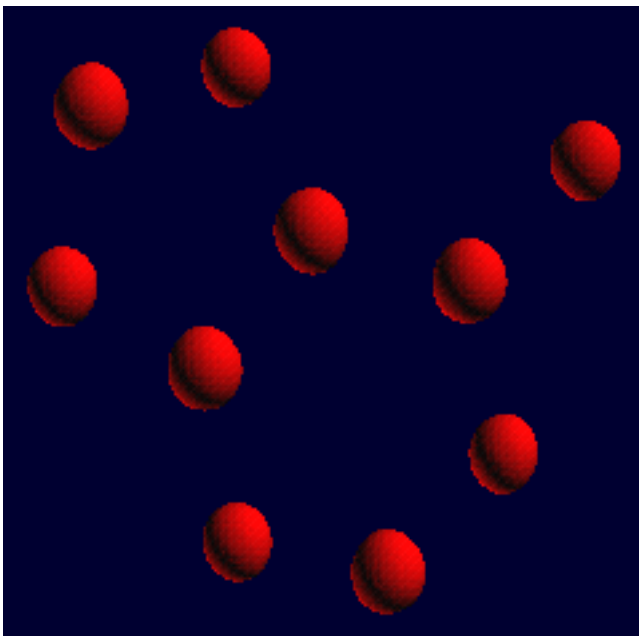


Figure 4.5: Isosurfaces of the Laplacian of the electron density of adenine. Here, $\beta_0 = 20$, and this number is larger than the number of clearly visible objects.

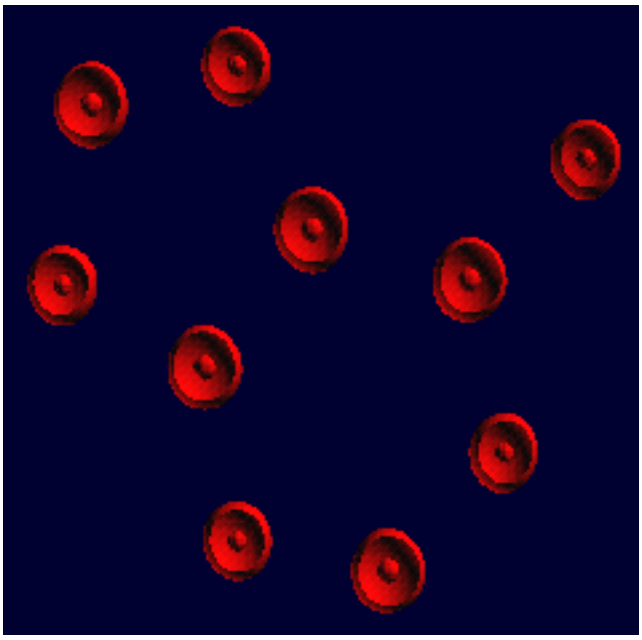


Figure 4.6: Isosurfaces of the Laplacian of the electron density of adenine. Clipping the isosurfaces of Figure 4.5 reveals the additional, previously invisible components.

Figures 4.7–4.9 show isosurfaces of the Laplacian of water. Figure 4.7 shows the isosurface at isovalue -0.931, and the associated Betti numbers are $\beta_0 = 3$, $\beta_1 = 1$, and $\beta_2 = 2$. This implies that the isosurface has three connected components, one tunnel, and two closed regions. Figure 4.8 shows the isosurface for isovalue -0.935, and the Betti numbers are $\beta_0 = 5$, $\beta_1 = 0$, and $\beta_2 = 3$. They indicate that this isosurface has two additional components and one additional closed region. In this case, one original component has "split" into three components. Figure 4.9 shows the components in the interior of the isosurfaces. (This image was generated by clipping the isosurface shown in Figure 4.8.)

One of the features of the Blunt Fin data set is the presence of vortices near the base of the fin. These vortices are detected by sweeping a range of isovalues, see Table 4.1, and observing that β_2, the number of tunnels, is equal to two for a wide range of isovalues.

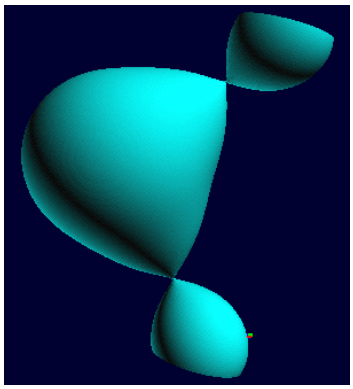


Figure 4.7: Isosurfaces of the Laplacian of water. For the isovalue of -0.931, three components exist.

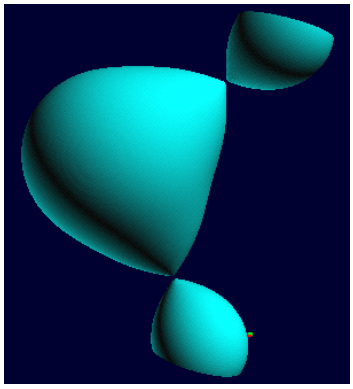


Figure 4.8: Isosurfaces of the Laplacian of water. For the isovalue of -0.935, five components exist.

Since tunnels in the isosurface of the velocity magnitude are a possible indicator for vortices in the data set, a range of interesting isovalues is identified. By visually inspecting some of the resulting isosurfaces for tunnels, the region in space for the potential vortices is identified, and is shown in Figure 4.10. Figure 4.10 shows where one tunnel split off into a separate component.

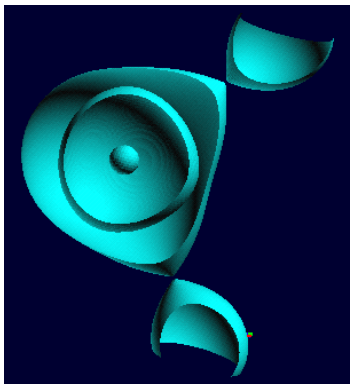


Figure 4.9: Isosurfaces of the Laplacian of water. Clipping the isosurface shown in Figure 4.8 reveals full structure of the isosurface.

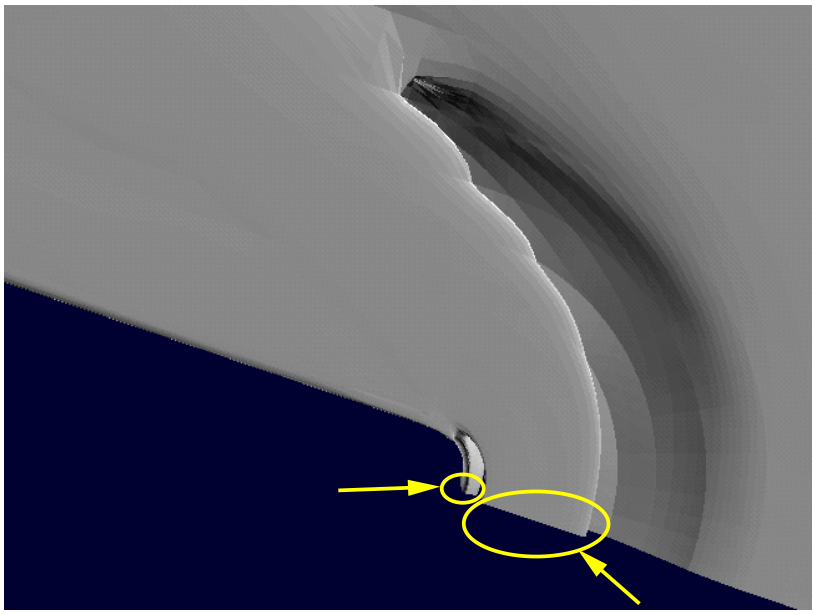


Figure 4.10: Isosurfaces of velocity magnitude of the the Blunt Fin data set, courtesy of NASA/Ames Research Center. Shown is the isosurface for the isovalue 1.3. Two tunnels are seen near the base of the fin. The yellow circles show the base of the tunnels.

Isovalue	β_0	β_1	β_2	Meaning
0	1	1	0	
0.75	3	0	1	
0.8	1	1	0	
0.85	1	2	0	two stable tunnels appear indicating vortices near the base of the fin
0.9	1	2	0	
0.95	1	2	0	
1	1	2	0	
1.05	1	2	0	
1.1	1	2	0	
1.15	1	2	0	
1.2	1	2	0	
1.25	1	2	0	
1.3	1	2	0	
1.35	1	2	0	
1.4	1	2	0	
1.45	1	2	0	
1.5	1	2	0	
1.55	1	2	0	
1.6	1	2	0	
1.65	2	2	1	another component appears briefly
1.7	2	2	1	
1.75	3	1	1	one of the stable "tunnels" splits off into another component
1.8	2	1	0	
1.85	4	1	2	topology becomes less stable and meaningful
1.9	2	2	0	
1.95	3	1	1	
2	3	1	1	

Table 4.1: Isovalues and Betti numbers for the Blunt Fin data set.

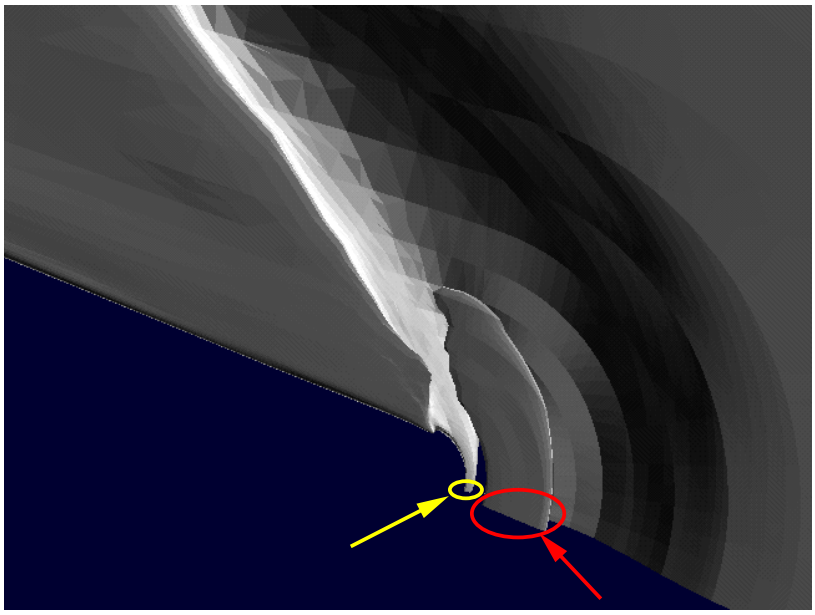


Figure 4.11: Isosurfaces of velocity magnitude of the Blunt Fin data set. Shown is the isosurface for the isovalue 1.9. The yellow circle indicates the base of the remaining tunnel. The red circle shows the region that had been the base of a tunnel for a lower isovalue, but the region split into a separate component in the current isovalue.

Chapter 5

Conclusion

We have introduced a method for the hierarchical and gridless representation of planar scattered data. The method is based on Voronoi diagrams and can thus be generalized to higher dimensions. We believe that Voronoi diagram-based approaches provide an appropriate framework for constructing hierarchical approximations for gridless, scattered data. Voronoi diagrams provide flexibility and enable adaptive and local refinement. Unfortunately, they require rather involved data structures for their efficient manipulation, but we are convinced that this is acceptable due to the gain in flexibility. We plan to extend our methods to volumetric data, and to time-varying data as well. We intend to develop efficient ray-casting and isosurface extraction methods for Voronoi diagram hierarchies of volumetric data sets.

There are a number of applications in data exploration and analysis where Betti numbers provide meaningful information about complex data. We have presented a fast algorithm for the calculation of Betti numbers for isosurface triangulations and have dis-

cussed techniques that use Betti numbers to provide information about isosurface topology. Calculating Betti numbers for isosurface triangulations requires time and space proportional to the size of the triangulations. This is desirable for isosurface applications, since obtaining additional topology information with Betti numbers would require only a minimal overhead. Concerning future work, we plan to extend this algorithm to more general non-manifold surfaces.

Bibliography

[1] P. Alexandroff. *Elementary Concepts of Topology.* Dover Publications, Inc., New York, 1961. Translated by Alan E. Farley.

[2] Franz Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys, Sept 1991*, 23(3), 1991.

[3] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, Heidelberg, New York, 1997.

[4] P. Cignoni, L. De Floriani, C. Montoni, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In Arie Kaufman and Wolfgang Krueger, editors, *1994 Symposium on Volume Visualization*, pages 19–26. ACM SIGGRAPH, October 1994.

[5] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution representation and visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):352–369, 1997.

[6] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber,

Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[8] L. De Floriani, P. Marzano, and E. Puppo. Hierarchical terrain models: Survey and formalization. In *Proc. IEEE Sympos. Applied Comput.*, pages 323–327, 1994.

[9] L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description geometric design. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

[10] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. A Delaunay-based method for surface approximation. In *Eurographics '83*, pages 333–350. Elsevier Science, 1983.

[11] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. A Delaunay-based method for surface approximation. In *Eurographics '83*, pages 333–350. Elsevier Science, 1983.

[12] Cecil Jose A. Delfinado and Herbert Edelsbrunner. An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995. ISSN 0167-8396.

[13] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[14] Gerald Farin. Surfaces over dirichlet tessellations. *Computer Aided Geometric Design*, 7(1-4):281–292, June 1990.

[15] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 4th edition, 1997.

[16] L. De Floriani and E. Puppo. Constrained Delaunay triangulation for multiresolution surface description. In *Proceedings Ninth IEEE International Conference on Pattern Recognition*, pages 566–569, Los Alamitos, California, 1988. CS Press.

[17] L. De Floriani and E. Puppo. Constrained delaunay triangulation for multiresolution surface description. In *Ninth International Conference on Pattern Recognition (Rome, Italy, November 14–17, 1988)*, pages 566–569, Washington, DC, 1988. Computer Society Press.

[18] R. Franke and G. M. Nielson. Scattered data interpolation and applications: A tutorial and survey. In H. Hagen and D. Roller, editors, *Geometric Modeling*. Springer-Verlag, 1991.

[19] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997.

[20] Leonidas J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graphics*, 4(2):74–123, April 1985.

[21] Bernd Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, 1994. ISSN 0167-8396.

[22] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.

[23] Hugues Hoppe. View-dependent refinement of progressive meshes. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 189–198. ACM SIGGRAPH, Addison Wesley, August 1997.

[24] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization 99*, pages 59–67. IEEE, November 1999.

[25] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. volume 25, pages 285–288, July 1991.

[26] D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242, June 1980.

[27] Arne Maus. Delaunay triangulation and the convex hull of n points in expected linear time. *BIT*, 24(2):151–163, 1984.

[28] P. Moran, C. Henze, and D. Ellsworth. The FEL 2.2 user guide. Technical report, National Aeronautics and Space Administration, 2000. NAS-00-002.

[29] Gregory M. Nielson and Bernd Hamann. The asymptotic decider: Removing the ambiguity in marching cubes. In *Visualization '91*, pages 83–91, 1991.

[30] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tesselations — Concepts and Applications of Voronoi Diagrams.* Wiley, Chichester, 1992.

[31] Jovan Popović and Hugues Hoppe. Progressive simplicial complexes. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 217–224. ACM SIGGRAPH, Addison Wesley, August 1997.

[32] Kevin J. Renze and James H. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics & Applications*, 16(6):24–32, November 1996.

[33] J. J. Rotman. *An Introduction to Algebraic Topology.* Springer-Verlag, New York, 1988.

[34] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, July 1992.

[35] R. Sibson. Locally equiangular triangulation. *The Computer Journal*, 21:243–245, 1978.

[36] Oliver G. Staadt and Markus H. Gross. Progressive tetrahedralizations. In David S. Ebert, Hans Hagen, and Holly Rushmeier, editors, *Proceedings of IEEE Visualization 98*, pages 397–402. IEEE Computer Society Press, Los Alamitos, California, October 1998.

[37] Issac J. Trotts, Bernd Hamann, and Kenneth I. Joy. Simplification of tetrahedral

meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, 1999.

[38] Issac J. Trotts, Bernd Hamann, Kenneth I. Joy, and David F. Wiley. Simplification of tetrahedral meshes. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *Proceedings of Visualization 98*, pages 287–296. IEEE Computer Society Press, Los Alamitos, California, October 1998.

[39] G. Voronoï. Nouvelles applications des paramètres continus à la théorie des formes quadratiques — premier Mémoire: Sur quelques propriétés des formes quadtratiques positives parfaites. *J. Reine Angew. Math.*, 133:97–178, 1907.

[40] G. F. Voronoï. Deuxième mémoire: recherches sur les paralléloedres primitifs. *J. Reine Angew. Math.*, 136:67–181, 1909.

[41] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[42] Yong Zhou, Baoquan Chen, and Arie Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization ý7*, pages 135–142. IEEE, November 1997.

[43] Yong Zhou, Baoquan Chen, and Arie Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *IEEE Visualization '97*, October 1997.