Hierarchical Data Representations Based on Planar Voronoi Diagrams

Shirley Schussman, Martin Bertram, Bernd Hamann, and Kenneth I. Joy

Center for Image Processing and Integrated Computing (CIPIC),
Department of Computer Science, University of California at Davis
Davis, CA 95616-8562, USA
{schussms, bertram, joy, hamann}@cs.ucdavis.edu

Abstract. Multiresolution representation of high-dimensional scattered data is a fundamental problem in scientific visualization. This paper introduces a data hierarchy of Voronoi diagrams as a versatile solution. Given an arbitrary set of points in the plane, our goal is the construction of an approximation hierarchy using the Voronoi diagram 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 the Sibson interpolant [14].

1 Introduction

This paper presents 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, data without explicit point connectivity. Second, point insertion and deletion operations for Voronoi diagrams are expected constant-time operations [10, 3]. In addition, Voronoi cells can be sorted in depth in linear time, which is important for volume visualization, and Voronoi diagrams can be extended to n dimensions. Although the Voronoi diagram's dual–the Delaunay triangulation–has the same properties, the Voronoi diagram provides a more intuitive tessellation.

2 Related Work

A number of approaches have been developed during the past two decades to visualize scientific data that is scattered [5] or defined on very large and often highly irregular 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 visualization.

Several data decimation and hierarchical schemes have been developed over the past few years by the computer graphics and visualization communities. Schroeder et al. [13] and Renze and Oliver [12] have developed algorithms that simplify a mesh by removing vertices. Removing a vertex creates a hole in the mesh that must be re-triangulated, and several strategies may be used.

Hoppe [7, 8] and Hoppe and Popović [11] describe a progressive-mesh representation of a triangle mesh. This is a continuous-resolution representation based on an edge-collapse operation. 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 [6] utilize a different strategy, based on quadratic error metrics for efficient calculation of a hierarchy. Hoppe [9] has extended this method to multidimensional meshes with appearance attributes.

Trotts et al. [17, 16] and Staadt and Gross [15] have extended the edge collapse paradigm to tetrahedral meshes. Cignoni et al.[1] 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.

This paper presents a new technique that produces a hierarchy of Voronoi diagrams. These diagrams can be used to approximate massive data sets by utilizing functional approximations over the Voronoi cells. We generate a hierarchy by inserting points into the Voronoi diagram that represent the largest error in individual cells. We construct two interpolation methods, one based on constant functions and the other one based on the Sibson interpolant [**?**,4]. We 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, which allows for a very compact representation.

3 Voronoi Hierarchies

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 accuracy of each level in the Voronoi hierarchy.

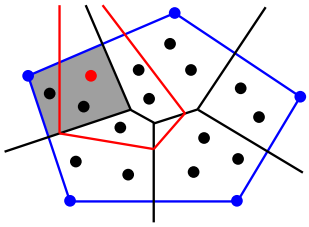


Fig. 1. Example of vertex insertion. Black lines: current Voronoi diagram; black polygon: convex hull of point set; grey region: Voronoi cell with maximal error; grey point: vertex p_{c_j} with maximal error ε_{max}; grey lines: Voronoi cell to be inserted.

3.1 Refinement of a Voronoi Diagram

Given a data 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 the boundary of the minimal point set defining the closed boundary polygon of the convex hull of D.

A Voronoi diagram, V^{n_k}, is refined by inserting additional data points of D into the Voronoi diagram in cells of high error as is shown in Figure 1. More specifically, a set of Voronoi cells with high error 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

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

where $f(c, p)$ is used to represent an interpolant over cell c_j containing p_i and where f_i denotes the associated function value at p_i. Once ε_{t_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

$$\varepsilon_{avg} = \frac{1}{n} \sum_{j=1}^{n_k} \sum_{p_i \in c_j} \|f(c_j, p_i) - f_i\|. \tag{2}$$

Once the set of cells to be refined is identified, a point $p_{c_j} \in D$ is inserted into each cell c_j. Ideally, p_{c_j} would define a cell that would eliminate or at least minimize the error in the resulting local cell configuration. Rather than searching exhaustively for the ideal point, we simply choose the point p_{c_j} in c_j with the highest error ε_{max}, where

$$\varepsilon_{max} = \max_{p_i \in c_j} \|f(c_j, p_i) - f_i\|. \tag{3}$$

3.2 Constructing the Hierarchy

The following pseudocode describes the basic algorithm for generating the 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.
- 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 high error.
- For each cell in this set, choose an appropriate data point in D that lies in it and update the diagram accordingly.
- Check whether global approximation error of refined Voronoi diagram still exceeds ε_1.
- Continue 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 form for multiresolution representations is that there are multiple ways to choose points for refinement. Rather than exploring all possibilities, which is a research topic in its own right, we developed the method described in Section 3.1. This method is designed to be fast, to refine Voronoi diagrams adaptively, and to quickly capture patterns in the underlying data. It can also be extended to higher dimensional domains, as can be seen by all equations.

The method of point insertion described in Section 3.1 refines a Voronoi diagram in a way that captures high-gradient regions and discontinuities very early in the refinement process. The point insertion strategy detects "extreme

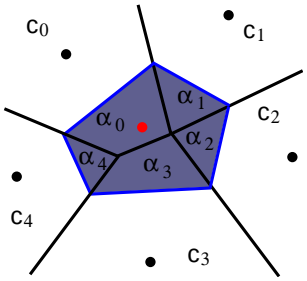


Fig. 2. Example of Sibson interpolant. Black lines: current Voronoi diagram; c_i: cells in current Voronoi diagram; grey point: simulated insertion point; grey region: simulated cell resulting from point insertion; α_i: regions where simulated cell overlaps c_i

values" first, since ε_{max} is always the maximum or minimum function value in a cell c_j. If the point p_j defining c_j was 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 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, if 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 good since only areas of high error 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 two interpolants for comparison, a piecewise constant and the Sibson interpolant The strengths and weaknesses of both interpolants are compared.

Using a constant value per cell, where the value is that of the defining point for 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 interpolant representation permits the representation of discontinuities, but cannot represent smoothly varying data well.

The Sibson interpolant is a smoothly varying function that smoothly represents the underlying data. The Sibson 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

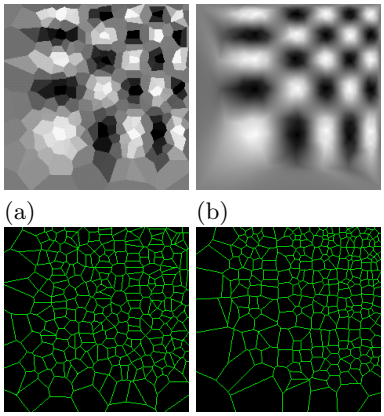


Fig. 3. (a) Piecewise constant function used to evaluate analytical function with 270 points; (b) Sibson's interpolant used to estimate same function with 270 points; (c) and (d) show corresponding Voronoi meshes for (a) and (b), respectively

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 the Sibson interpolant at p is defined as

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

which is illustrated in the Figure 2.

4 Results

We present the results of the constant function and Sibson interpolant based algorithms for three data sets shown in Figures 3, 5 and 6, along with their numerical performance data in Table 1. We also show a hierarchy using the Sibson interpolant in Figure 4. All of the input data sets are defined on a 250x250 uniformly spaced rectilinear grid representing color and grey scale images.

The first greyscale data set, see Figure 3, was generated by evaluating the function

$$\omega(x, y) \;=\; sin(x^2)\; sin(y^2), \quad x,\; y \,\epsilon\, [0,\; 4]\,.$$

Although both algorithms pick up the pattern quickly, the Sibson interpolant obtains better results because it can represent smooth functions well.

The remaining data sets are color images produced by the Hubble Space Telescope, courtesy of NASA. Figure 4 shows a Voronoi hierarchy with four levels. The basic pattern is represented well with only 100 points, as is shown is Figure 4(b). Successive levels refine the center and represent additional stars.

Dataset	No. Tiles	Piecewise Constant		Sibson Interpolant	
		L^1 Error [%]	L^2 Error [%]	L^1 Error [%]	L^2 Error [%]
ω	270	7.1	21	2.9	3.4
"Cat's Eye"	580	2.3	8.3	2.3	2.9
"Cygnus Loop"	620	8.2	19	5.3	6.4
	1740	5.2	11	4.4	5.3

Table 1. Numerical approximation results.

Figure 4(e) shows a good approximation of the original data set with 2000 points, which is only three percent of the original data points.

Figure 5 shows the effectiveness of both algorithms on the Cat's Eye Nebula data set using 580 points. Although they obtain the same numerical performance for the L^1 norm, the constant function algorithm detects more features. Namely, it detects the second elliptical path in the center, the one whose primary axis has a negative slope. As is seen from the Voronoi cells in Figure 5(e), the Sibson interpolant algorithm has fewer cells in that region. The Sibson interpolant, a smooth interpolant, places its points around discontinuous regions, which is the only way the Sibson interpolant can represent discontinuities.

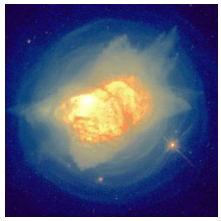
Another data set is an image of the Cygnus Loop Nebula, see Figure 6. Although the Sibson interpolant is better visually and numerically, it depicts fewer stars than the constant function algorithm. The lack of stars results from the point insertion technique. Instead of using a threshold value for the entire data set, like the constant function algorithm, it only inserts a point into the cell with the worst error before it updates the Voronoi diagram. Since the cells with missing stars never counted at the worst cell, they were never refined.

5 Conclusions and Future Work

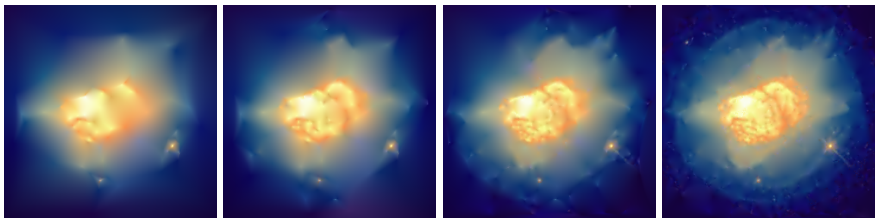
We have introduced a method for the hierarchical, gridless representation of planar scattered data. The method is straightforward and can be generalized to higher dimensions. We believe that Voronoi diagram-based approaches provide an appropriate framework for constructing hierarchical approximations for gridless, scattered data. Voronoi diagrams provide flexibility and enable adaptive and localized refinement. Voronoi diagram hierarchies require rather involved underlying 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 implementations to volumetric data, and eventually to time-varying data. We will develop efficient ray-casting and isosurface extraction methods for Voronoi diagram hierarchies of volumetric data sets.

6 Acknowledgments

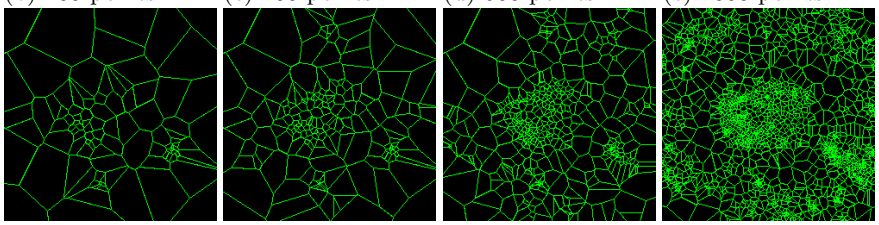
This work was supported by the National Science Foundation under contracts ACI 9624034 and ACI 9983641 (CAREER Awards), through the Large Scien-



(a) Original image of a dying sun



(b) 100 points (c) 200 points (d) 500 points (e) 2000 points



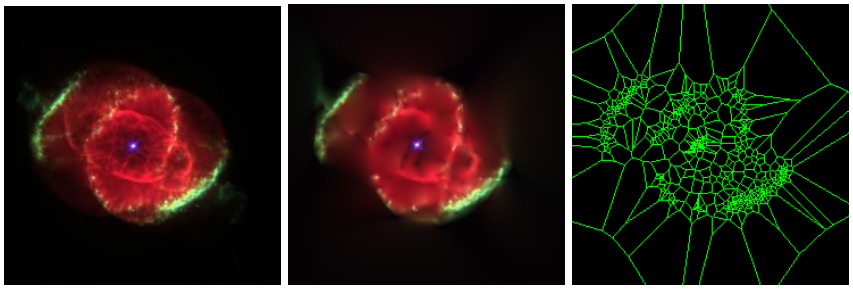
(f) Cells for (b) (g) Cells for (c) (h) Cells for (d) (i) Cells for (e)

Fig. 4. Approximations of the dying sun image with 100, 200, 500 and 2000 points

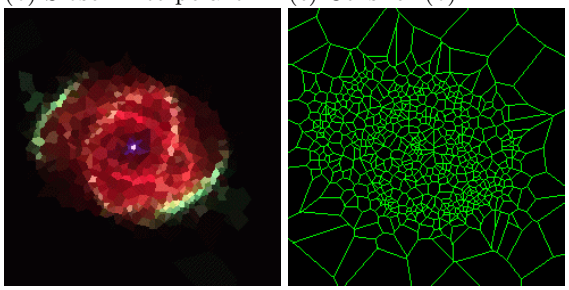
tific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the Office of Naval Research under contract N00014-97-1-0222; the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We also acknowledge the support of ALSTOM Schilling Robotics, Chevron, Silicon Graphics, Inc. and ST Microelectronics, Inc. We thank the members of the Visualization Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

1. 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.
2. L. De Floriani, P. Marzano, and E. Puppo. Hierarchical terrain models: Survey and formalization. In *Proc. IEEE Sympos. Applied Comput.*, pages 323–327, 1994.
3. O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.
4. Gerald Farin. Surfaces over dirichlet tessellations. *Computer Aided Geometric Design*, 7(1-4):281–292, June 1990.
5. 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.
6. 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.
7. Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.
8. 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.
9. 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.
10. Arne Maus. Delaunay triangulation and the convex hull of n points in expected linear time. *BIT*, 24(2):151–163, 1984.
11. 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.
12. Kevin J. Renze and James H. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics & Applications*, 16(6):24–32, November 1996.
13. William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, July 1992.
14. R. Sibson. Locally equiangular triangulation. *The Computer Journal*, 21:243–245, 1978.
15. Oliver G. Staadt and Markus H. Gross. Progressive tetrahedralizations. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *Proceedings of Visualization 98*, pages 397–402. IEEE Computer Society Press, Los Alamitos, California, October 1998.
16. 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.
17. 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.

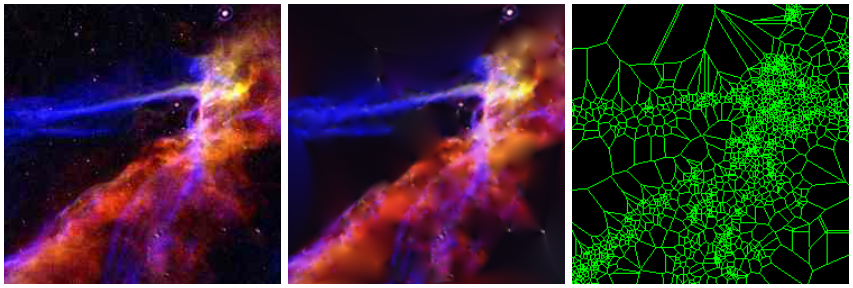


(a) Original image (b) Sibson Interpolant (e) Cells for (b)

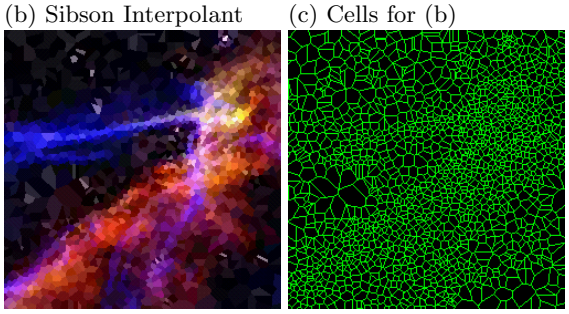


(d) Constant function (e) Cells for (d)

Fig. 5. Approximations of the Cat's Eye Nebula made with 580 points



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



(d) Constant function (e) Cells for (d)

Fig. 6. Approximations of the Cygnus Loop Nebula made with 1740 points