Data-Dependent Triangulation in the Plane with Adaptive Knot Placement

René Schätzl
University of Kaiserslautern *

James C. (Fritz) Barnes
Vanderbilt University †

Bernd Hamann
University of California, Davis ‡

Kenneth I. Joy §
University of California, Davis

Hans Hagen¶
University of Kaiserslautern

Center for Image Processing and Integrated Computing (CIPIC)
Department of Computer Science
University of California, Davis
Davis, CA 95616–8562, USA

University of Kaiserslautern
FB Informatik / AG Computergrafik
Postfach 3049
67663 Kaiserslautern, Germany

Abstract

In many applications one is concerned with the approximation of functions from a finite set of scattered data sites with associated function values. We describe a scheme for constructing a hierarchy of triangulations that approximates a given data set at varying levels of resolution. Intermediate triangulations can be associated with a particular level of a hierarchy by considering their approximation errors. We present a data-dependent triangulation scheme using a Sobolev norm to measure error instead of the more commonly used root-mean-square (RMS) error. Triangles are split by selecting points in a triangle, or its neighbors, that are in areas of potential discontinuites or areas of hight gradients. We call such points "significant points."

1 Introduction

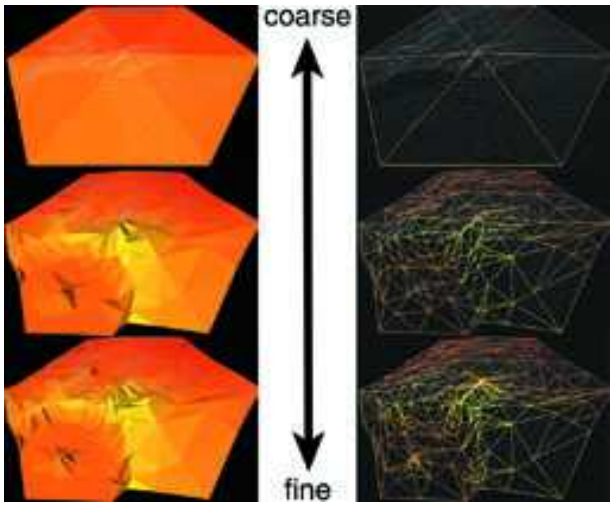
We describe a method to create piecewise linear approximations for scattered bivariate data of the form $\{(x_i, y_i, f_i) \mid i = 1, \ldots, N\}$. Our algorithm creates an initial triangulation of the region defined by the boundary polygon of the convex hull of the given data. Using this triangulation, a refinement process produces a sequence of piecewise linear functions that improve the approximation of the given scattered data at each step. The method can be applied to general multi-valued scattered data, defined as a set

$$\{(x_i, y_i, f_{i,1}, f_{i,2}, \ldots, f_{i,k}) \mid i = 1, \ldots, N\}, \qquad (1)$$

where multiple function values $f_{i,j}$ are associated with each site.

The input to our method is a set of error tolerances, denoted as $\varepsilon_1, \varepsilon_2, ..., \varepsilon_n$, each of which specifies the allowable error per triangulation level. We iteratively refine intermediate triangulations by triangle subdivision until the next error tolerance is met. Each triangulation implies a piecewise linear approximation of the given scattered data. Refinement is performed until we have n triangulations that meet the n prescribed error tolerances. These n triangluation levels define a "hierarchy," which is illustrated in Figure 1.

Our method does not require connectivity information for the given sites. First, we create a coarse triangulation. This is done by calculating the boundary polygon of the convex hull of the set of all given sites in the plane and triangulating the region defined by the point subset defining the boundary polygon.



captionHierarchy of triangulations (left: flat-shaded triangulated surface, right: triangulation).

We perform triangle subdivision to improve an intermediate linear spline approximation. The triangle with the greatest local error is split into at least two and at most four subtriangles by using at most one split point per edge. This process is then repeated iteratively.

We have used different types of error metrics to determine estimates of the local error of a triangulation. The Sobolev norm, which also considers the gradient of the original data, leads to very good results. By considering the gradients, triangles containing "significant" data sites, like discontinuities or high-gradient data, have larger associated errors than triangles in relatively low-gradient areas. We do not need the gradient to be part of the given data set, as it can be approximated in a preprocessing step.

To get an approximation of the gradient, we compute the surface at each original data site, which is computed using the original data site and its ten closest neighbors in a discrete Gaussian least-square fit.

*Fachbereich Informatik, Universität Kaiserslautern, D-67653 Kaiserslautern, Germany; email: schaetzl@informatik.uni-kl.de

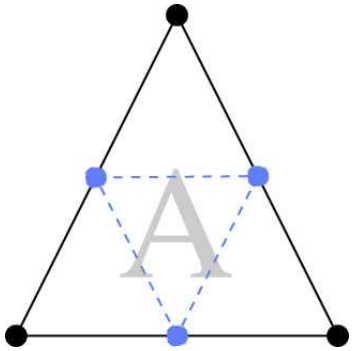
†Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37235, USA; email: barnes@cs.ucdavis.edu

‡Co-Director of CIPIC; Department of Computer Science, University of California, Davis, CA 95616-8562, USA; email: hamann@cs.ucdavis.edu

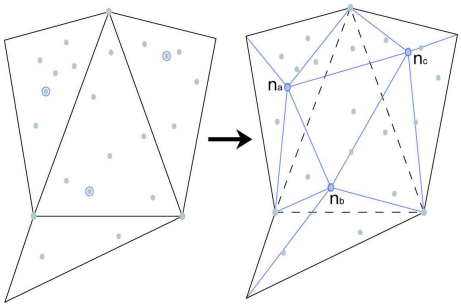
§Department of Computer Science, University of California, Davis, CA 95616-8562, USA; email: joy@cs.ucdavis.edu

¶Fachbereich Informatik, Universität Kaiserslautern, D-67653 Kaiserslautern, Germany; email: hagen@informatik.uni-kl.de

When we perform triangle subdivision to improve an approximation, we consider two different refinement schemes, which we refer to as "Type-A" and "Type-B" refinement. Type-A refinement splits triangles by generating split points along one or all three edges of a triangle. An example of this technique is shown in Figure 1(a) for three points on the edges of a triangle. When a triangle is split, so-called "implied splits" must be performed in neighboring triangles ("edge neighbors").



(a) Type-A refinement: The original black triangle is subdivided into four subtriangles.



(b) Type-B refinement: The original triangle is subdivided into four triangles using existing data sites.

Figure 1: Two types of triangle subdivision.

There are some problems with Type-A refinement. These are due to the fact that Type-A refinement introduces split points lying exactly on triangle edges. As a result of this restriction, long edges in a coarse initial triangulation remain visible in all subsequent higher-resolution triangulation levels, leading to artifacts in renderings.

We address this problem by extending the Type-A refinement scheme to choose split points that are *not* necessarily located on the edges of a triangle being refined. We are identifying significant data sites lying inside the triangle or inside one of its neighbors. It is preferable to use original data sites whenever possible. We call this method Type-B refinement. An example of this technique is illustrated in Figure 1(b). Our overall refinement algorithm operates as follows:

- INPUT: N scattered bivariate data points; n error tolerances
- OUTPUT: n triangulations
- ALGORITHM:
  - Compute minimal point set defining the boundary polygon of the convex hull.
  - Compute initial data-dependent triangulation for the region defined by this point set.
  - Refinement. Compute n triangulations by performing the following steps:
    - While error is greater than the current tolerance ε do
      - Refine the triangle with the greatest error.
      - Perform all possible refinements and determine how they impact the error.
      - Choose a refinement that maximally decreases the error.
      - Re-calculate vertex values for those vertices affected by the re-triangulation step.

2 Related work

A data-dependent triangulation scheme adaptively generates a triangulation by considering approximation error. The techniques described in [13], [14], and [15] deal with the problem of decimating triangular surface meshes and adaptive refinement of tetrahedral volume meshes. These approaches are aimed at concentrating points in regions of high curvatures or high second derivatives. This paradigm can be used to either eliminate points in nearly linearly varying regions (*decimation*) or to insert points in highly curved regions (*refinement*). The data-dependent triangulation scheme we describe in this paper is based on the principle of refinement. Our algorithm refines a triangulation by either using existing data sites or inserting new points.

In principle, our technique is related to the idea of constructig a *multiresolution pyramid*, i.e., a data hierarchy of triangulations with increasing precision, see [10]. Figure 1 shows a multiresolution hierarchy of triangles, where the top level is a coarse triangulation, and, as we *descend* the hierarchy, finer triangulations become visible. The pyramid concept has also been extended to the adaptive construction of tetrahedral meshes for scattered scalar-valued data, see [3] and [6]. *Multiresolution methods* have been applied to polygonal (triangular) approximations of surfaces. Such approaches are described in [7], [8], and [18]. Our data-dependent technique can be viewed as a hierarchical method for representing scattered data by multiple levels of triangulations, but our approach is not based on the construction and application of orthogonal basis systems, such as *wavelet* bases.

Scarlatos and Pavlidis discuss a scheme [22] that recognizes the linear "coherence" of discontinuties. In their refinement scheme, one attempts to place a triangle edge along discontinuities in a data set. A primary difference between their work and our scheme is that we allow knots (= mesh vertices) that do not necessarily coincide with the original data sites to be introduced when there is no other option.

An alternative to constructing a triangulation hierarchy is to start with a fine mesh and decimate vertices, edges, or faces. Hoppe [16] discusses a technique for collapsing edges. In [26], an alternative scheme for collapsing faces is discussed.

Survey papers of scattered data approximation for bivariate and trivariate data are [19], [2] and [11]. In [20], various scattered data interpolation techniques (scalar-valued, trivariate case) are discussed and compared. Our scheme relies on concepts from *geometric modeling* and *computational geometry*. These are discussed in [9] and [21].

3 Adaptive triangle refinement

The input to our refinement scheme is a set of planar points with height values. The data sites do not have to lie on a regular grid though our examples all have this property.

$$S = \int ||L(x,y) - f(x,y))||\, dx\, dy + c \int \left\| \frac{\partial}{\partial x} L(x,y) - \frac{\partial}{\partial x} f(x,y) \right\| + \left\| \frac{\partial}{\partial y} L(x,y) - \frac{\partial}{\partial y} f(x,y) \right\| dx\, dy \quad (2)$$

$$E_{LSOB} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (L(x_i,y_i) - f_i)^2 + \tau_A \left(\left(\frac{\partial}{\partial x} L(x_i,y_i) - f_i^x \right)^2 + \left(\frac{\partial}{\partial y} L(x_i,y_i) - f_i^y \right)^2 \right)} \quad (3)$$

We start the adaptive refinement process by creating an initial coarse triangulation. We iteratively refine intermediate triangulations until a triangulation is obtained whose associated global approximation error is smaller than a prescribed tolerance. We use a Sobolev norm to measure the error for our approximations.

We apply triangle refinement to improve a piecewise linear approximation. In each refinement step, we identify the triangle that deviates the most from the given data and subdivide it. Refining a single triangle consists of these basic steps:

1. Identify appropriate points within the triangle and its edge neighbors. If such points exist, then use them in the refinement step.
2. If no appropriate points are found, then generate new vertices along the edges of the triangle to be split.
3. Approximate the function values for a new vertex and certain existing vertices in the neighborhood where refinement is performed.
4. Construct a new triangulation of the set of original and newly inserted vertices.
5. Compute an error estimate for the new triangulation.

These steps are iterated until a certain error tolerance is met. Our scheme is adaptive in two ways: (i) An intermediate triangulation is refined locally in regions with large errors, and (ii) the locations of vertices are chosen in order to minimize the error. We analyze all possible refinements of a triangle and compare them to determine which one leads to the best fit, i.e., leading to maximal error decrease, of the underlying data. We apply tests to guarantee that the location of vertices does not lead to overlapping triangles after subdivision.

3.1 Initial triangulation

The initial triangulation of a given scattered data set defines the domain over which the algorithm is executed. It should consist only of existing data sites of the data set. There are different possibilities to generate valid inital triangulations. We consider the boundary polygon of the convex hull of the set of given data sites as *natural* boundary of a data set. We use the *Graham's scan* algorithm [12] to compute the points defining this boundary polygon. This results in a set of points which, when triangulated, define the domain for our linear spline hierarchy.

Using the boundary polygon of the convex hull, we compute a data-dependent triangulation of the minimal point set defining the polygon. In general, one has to consider all possible triangulations of this point set and select the one that minimizes a chosen error measure. Computing all possible inital triangulations cannot be done efficiently when the boundary polygon is defined by a relatively large number of points. Therefore, we propose to construct any triangulation of the boundary points and then apply *simulated annealing* in order to obtain a better, possibly optimal, data-dependent initial triangulation, see, e.g., [17] and [23]. This is also preferable in the case of non-convex data.

Another approach proved to produce good results in certain situations: Often, it is possible to obtain a general idea of how the data set behaves in the interior of the domain by analyzing the behavior on the boundary. Following this idea, we compute *all* the points that lie on the boundary and identify the significant points of the boundary polygon. We then use the significant points of the boundary polygon to construct the initial triangulation using the Delaunay triangulation. (Thus, we apply a data-dependent point selection step on the boundary.)

Remark: For many practical applications, it might be sufficient to simply use the four vertices defining the corners of the bounding box containing all original sites. Several real-world data sets are defined on a uniform, rectilinear grid whose convex hull coincides with its bounding box.

Another practical solution is to define the start triangulation manually. This opens the possibility to concentrate on special areas of interest in a given data set.

3.2 Approximation error estimates

In this section, we describe the approximation error estimate that we use in our refinement scheme. We assume that the given scattered data in the plane and the vertices of all intermediate triangulation levels have the same convex hulls.

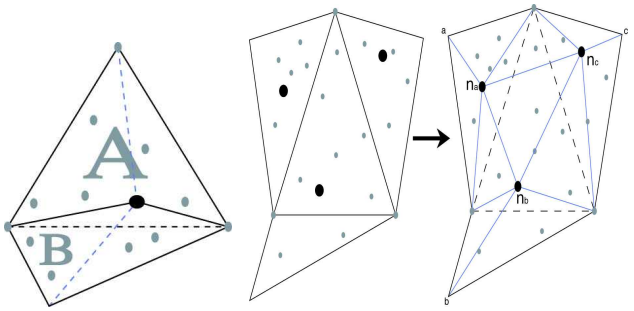
It is the objective of data-dependent triangulation to refine a triangulation in "high-detail" areas of a data set by using more and, if necessary, skinnier triangles than in "low-detail" areas. In most data sets, the significant points are points close to discontinuities or points in high-gradient regions. Thus, the error norm should assign more weight to points in those regions.

The error norm S we use is derived from the Sobolev norm [1, 25]. It is defined by Equation 2, where f denotes the original function to be approximated, L is a linear spline approximation, and $c > 0$ is a constant. The constant c can be chosen arbitrarily. After some experiments, we decided to use the area of the triangle as value for c. By considering not only the difference in function value but also in gradient value, significant areas are more readily identified and captured in the triangulations.

We compute a *local error* for each triangle and a *global error* for each triangulation. If there are m original data sites lying inside the triangle τ (including its boundary), we define the local Sobolev error E_{LSOB} as in Equation 3, where f_i is the value at a given site (x_i, y_i), f_i^x and f_i^y are the two components of the gradient at site (x_i, y_i), $L(x_i, y_i)$ is the value of the linear polynomial over the triangle containing (x_i, y_i), and τ_A is the area of the triangle containing (x_i, y_i). The global error associated with an entire triangulation is defined as the maximum of all E_{LSOB} values.

3.3 Refining a triangle

When refining a triangle, we are searching for one or three points in the triangle or in its edge neighbors being close to the edges of the triangle to be split. This leads to two different split types, shown in Figure 2. To demonstrate the basic idea we choose split points exactly on the edges. In each case, we determine a new split point within the part of the original data set that lies inside the triangles.



(a) Choosing one split point.

(b) Choosing three split points.

Figure 2: The two different split types.

If we do not find any appropriate data site, then we take the midpoint of an edge and approximate a function value for this point as described in section 3.4.

In the first case, see Figure 2(a), we are searching among the existing data sites in triangle A and its edge neighbor B for an appropriate split point. The chosen data site has do be within a certain convex region bounded by the areas of the two triangles. We have to consider the situations shown in Figure 3: Here, original data sites within the shaded regions cannot be used. The data sites have to lie in a region, that is calculated in the following way:

1. Calculate the intersection of the two lines passing through γ and δ and through α and $\beta \rightarrow S_1 =$ intersection point.
2. Calculate the intersection point S_2 in the same way.
3. If S_1 is between α and β, then use the triangle δ, S_1, β; otherwise, use the triangle α, β, δ.
4. If S_2 is between β and γ, then use the triangle β, S_2, δ; otherwise, use the triangle β, γ, δ.
5. Similar calculations have to be done to obtain the points S_3 and S_4 in the symmetrical case, shown in Figure 3, on the right-hand side.

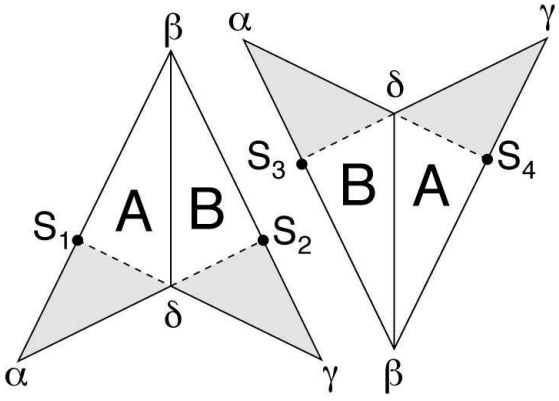


Figure 3: Generating a convex region.

Remark: To avoid very skinny triangles the distance between a chosen data site and the common edge of the two triangles has to be shorter than the distance (perpendicular) to any of the other edges of the triangles.

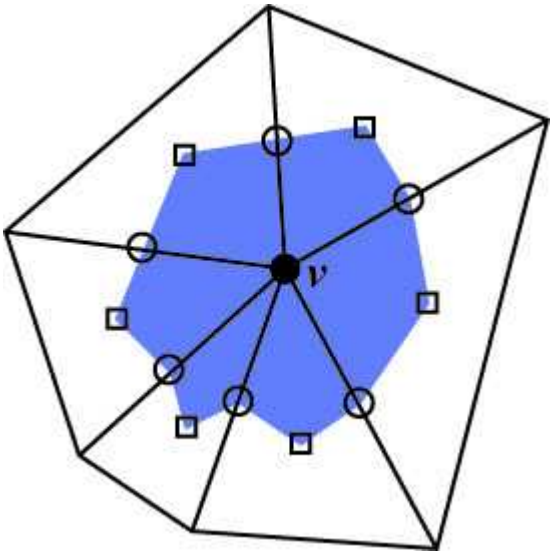


Figure 4: Construction of a tile for mesh vertex v.

Every data site satisfying the conditions described above is investigated concerning its "significance." In our current approach, we choose the data site that is approximated worst with respect to the Sobolev norm. If there exist data sites with the same deviation, we chose the one that is closer to the midpoint of the common edge of the two triangles being split. Especially in rather linear regions data sites are chosen which are positioned more in the middle of the triangles to produce more uniform triangles. On the other hand, if there is a significant data site within those two triangles, then it is chosen. In this case, the triangle may become skinnier but more appropriate in the sense of data-dependent triangulation.

If there exists no data site satisfying these conditions, then we generate a new data site that is the midpoint of the common edge. The function value of this new data site is approximated as described in section 3.4.

The second type of refinement chooses three points lying inside the triangle or inside one of its up to three edge neighbors. This is illustrated in Figure 2(b).

To get a correct triangulation we have to place the new points, called n_a, n_b, and n_c in Figure 2(b), so that none of the new edges intersect each other or the boundary polygon of the union of the triangle to be refined and its edge neighbors.

Currently, we determine a data site for each internal edge that has the closest perpendicular distance to the midpoint of that edge. If such a point does not exist or the data site has a smaller distance to any of the midpoints of the other internal edges, then we insert the midpoint of the edge as a new vertex.

3.4 Approximating function values

We approximate function values, i.e., the coefficients of our linear spline approximation, at mesh vertices using a local approximation scheme. We use a modified, localized *Shepard's method*, see [24]. We need to determine a local point set to be considered when calculating the function value at a particular vertex. The original scattered data that we use for this local approximation are the points lying within the *tile* around a particular vertex, shown in Figure 4. The tile of a vertex is constructed by connecting the midpoints of all edges emanating from the vertex and the centroids of all triangles

that share the vertex as a common vertex.

We subdivide a tile into triangles and perform an inside/outside test for this set of triangles to determine the original sites that lie inside the tile. We consider this subset of data to estimate a function value f_{app} for the central vertex v. The function value f_{app} is a weighted average defined as

$$f_{app} = \frac{\sum_{i=1}^{M} f_i/d_i^2}{\sum_{i=1}^{M} 1/d_i^2}. \quad (4)$$

Here, M is the number of original sites inside the tile, f_i is the function value associated with a given site (x_i, y_i) inside the tile, and d_i^2 is the squared Euclidean distance between the new vertex $\boldsymbol{v}$ and (x_i, y_i).

Whenever triangles are refined as a result of inserting additional vertices, we must estimate new function values for all vertices in the triangulation whose associated tiles change as a result of the refinement process. This set of vertices is given by the set of points becoming end points of new edges in the triangulation.

4 Results

We have applied our method to data sets with and without high-gradient regions and discontinuities. To demonstrate the usefulness of the chosen Sobolev norm we have also performed refinement for the same data sets using the RMS error. We have applied our method to the following data sets:

- A discrete Mount St. Helens digital-elevation model (DEM) data set, provided on a uniform rectilinear grid, which is shown in Figure 7.
- A Lake Marquette DEM, which is shown in Figure 6.

As one can see in both cases, using the RMS error leads to very skinny triangles even in low-gradient regions. Most of the refinement takes place in isolated regions. On the other hand, using our Sobolev norm leads to much improved triangulations. Even smaller features in the data sets are approximated well.

The Mount St. Helens data set demonstrates the usefulness of our approach for approximating data with narrow *cliff regions*. In this image, a drawback of using the Sobolev norm becomes apparent: The Sobolev norm tends to over-smooth the triangulation.

Considering the Lake Marquette data set, one can see how effectively our method handles data sets whith high- and low-gradient regions. In the foreground of those pictures, the lake is a low-gradient region, which is approximated by few large triangles. The fine-structured coastline is approximated by several small triangles. The higher number of triangles in the flat regions results from the use of the gradient in the error norm, as one of the edges in the initial triangulation is right on the border of the coastline.

The computational cost of our algorithm depends on the different approaches used. The initial triangulation has a complexity of $O(n \log n)$, the gradient approximation can be done in $O(n \log n)$. The individual refinement step has to check all the original data points lyning in the involved triangles, so the complexity of each step is $O(n)$. How often the iteration step is executed depends on the error value given as input. As a general rule, we can assume that no more iterations should be done than they are original data sites. So the overall complexity is $O(n^2)$.

The memory consumption is linear to the number of iterations.

5 Conclusions and future work

We have discussed a new technique for the construction of data-dependent triangulations for bivariate scattered data. Our scheme preserves high-gradient regions or potential discontinuities that might exist in a given data set by using the Sobolev norm. We have tested our method for various examples. We plan on introducing a quality measure that depends on the relative flatness of the region to prevent the generation of too many very skinny triangles. We are currently investigating local re-triangulations (through edge swapping) to eliminate the artifacts that currently result when using the Sobolev norm.

6 Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific 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, General Atomics, Silicon Graphics, Inc. and ST Microelectronics, Inc. We thank the members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

References

[1] R. A. Adams. *Sobolev spaces*. Academic press, New York, 1975.

[2] L. Alboul, G. Kloosterman, C.R. Traas, and R.M.J. van Damme. Best data-dependent triangulations. Technical Report Memorandum No. 1487, University of Twente, Faculty of Mathematical Sciences, 1999.

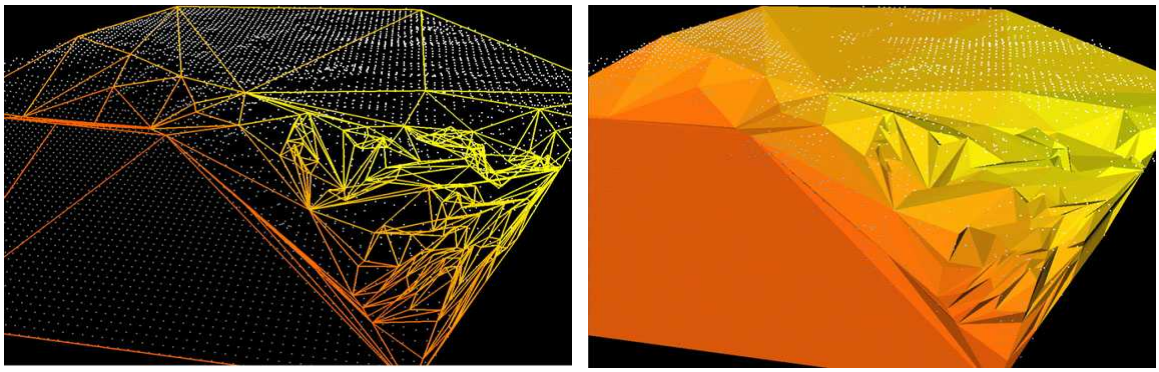
[3] M. Bertolotto, L. De Floriani, and P. Marzano. Pyramidal simplicial complexes. In C. Hoffmann and J. Rossignac, editors, *Third Symposium on Solid Modeling and Applications*, pages 153–162, New York, NY, 1995. Association for Computing Machinery, ACM Press.

[4] G. P. Bonneau. Multiresolution analysis on irregular surface meshes. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):365–378, 1998.

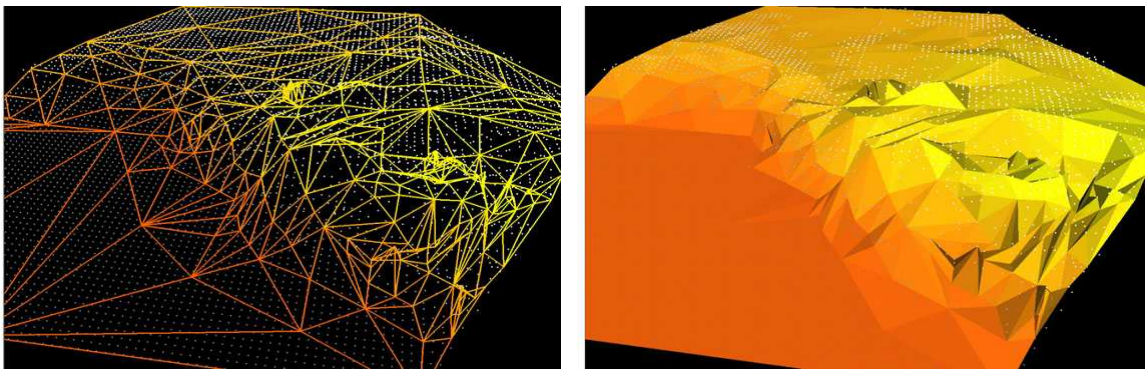
[5] G. P. Bonneau and A. Gerussi. Level-of-detail visualization of scalar data sets defined on irregular surface meshes. In D. S. Ebert;H. Hagen; H. E. Rushmeier, editor, *Proceedings of the IEEE Visualization*, pages 73–77. IEEE Computer Society Press, 1998.

[6] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In A. E. Kaufman and W. Krűger, editors, *1994 Symposium on Volume Visualization*, pages 19–26, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[7] A. D. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological shape. Technical Report 93-10-05, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 1993.



(c) Using RMS norm; 377 triangles.



(f) Using Sobolev norm; 482 triangles.

Figure 5: Lake Marquette data set (10000 sample points; 99 refinement steps).

[8] M. Eck, A.D. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In R. Cook, editor, *Proceedings of SIGGRAPH 1995*, pages 173–182, New York, NY, 1995. ACM Press.

[9] G. Farin. *Curves and Surfaces for CAGD*. Academic Press, San Diego, CA, fourth edition edition, 1997.

[10] L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics & Applications*, 9(2):67–78, 1989.

[11] M. Garland and P.S. Heckbert. Fast polygonal approximation of terrains and height fields". Technical Report TR CMU-CS-95-181, Carnegie Mellon University, School of Computer Science, 1995.

[12] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[13] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, 1994.

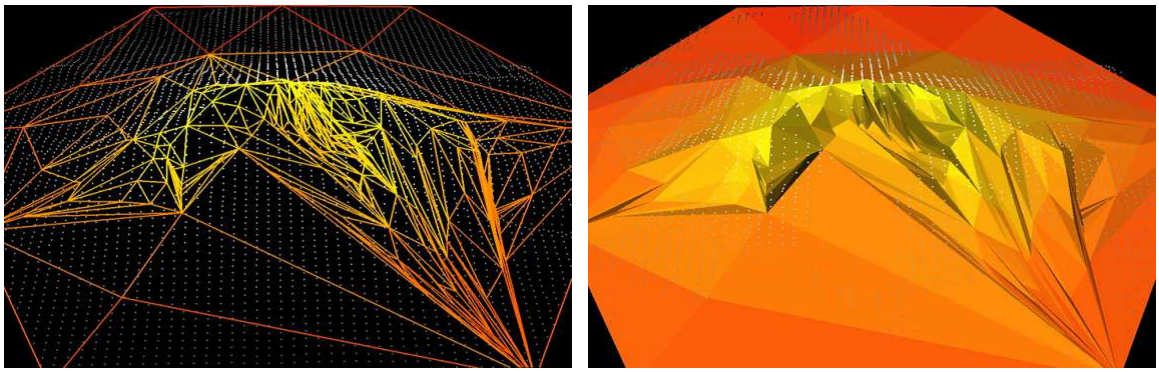
[14] B. Hamann and J. L. Chen. Data point selection for piecewise linear curve approximation. *Computer Aided Geometric Design*, 11(3):289–301, 1994.

[15] B. Hamann and J. L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11(5):477–489, 1994.

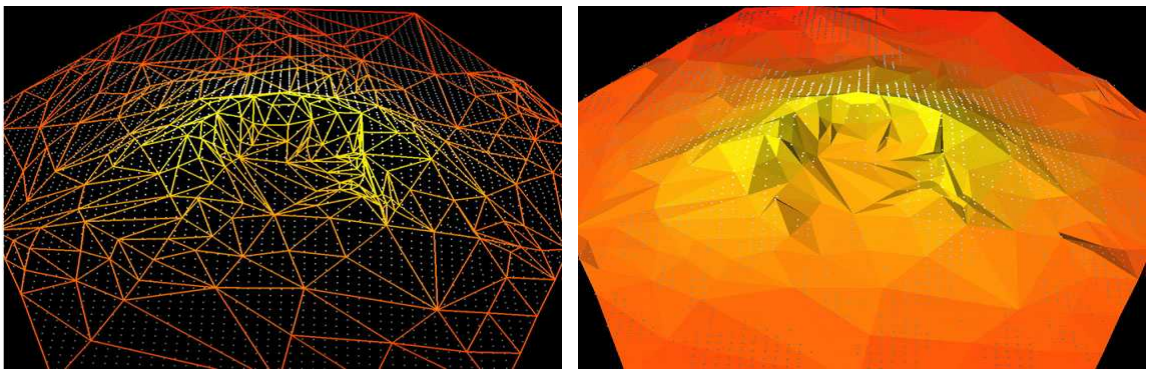
[16] H. Hoppe. Progressive meshes. In H. Rushmeier, editor, *Proceedings of SIGGRAPH 1996*, pages 99–108, New York, NY, 1996. ACM Press.

[17] O. Kreylos and B. Hamann. On simulated annealing and the construction of linear spline appoximations for scattered data. In E. Groeller, H. Loeffelman, and W. Ribarsky, editors, *Proceedings EUROGRAPHICS-IEEE TCCG Symposium an Visualization, Data Visualization '99*, pages 189–198. Springer Verlag, Vienna, Austria, 1999.

[18] M. Lounsbery. *Multiresolution analysis for surfaces of arbitrary topological shape*. Dissertation, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 1994.



(c) Using RMS norm; 402 triangles.



(f) Using Sobolev norm; 496 triangles.

Figure 6: Mount St. Helens DEM (9396 sample points; 99 refinement steps).

[19] G. M. Nielson. Scattered data modeling. *IEEE Computer Graphics*, 13(1):60–70, 1993.

[20] G. M. Nielson and J. Tvedt. Comparing methods of interpolation for scattered volumetric data. In D. F. Rogers and R. A. Earnshaw, editors, *State of the Art in Computer Graphics*, pages 67–86. Springer-Verlag, New York, NY, 1993.

[21] F. P. Preparata and M. I. Shamos. *Computational Geometry*. Springer-Verlag, New York, NY, third printing edition, 1990.

[22] L. L. Scarlatos and T. Pavlidis. Hierarchical triangulation using terrain features. In *Proceedings IEEE Conference on Visualization '90 Proceedings*, pages 168–175, 1990.

[23] L. L. Schumaker. Computing optimal triangulations using simulated annealing. *Computer Aided Geometric Design*, 10(3-4):329–345, 1993.

[24] D. Shepard. A two-dimensional interpolation function for computer mapping of irregularly spaced data. Technical Report TR-15, Harvard Univ., Center for Environmental Design Studies, Cambridge, Cambridge, MA, 1968.

[25] S. L. Sobolev. The Schwarz algorithm in the theory of elasticity. *Sokl. Acad. N. USSR*, IV (XIII):236–238, 1936.

[26] T.S. Gieng; B. Hamann; K.I. Joy; G.L. Schussman; I.J. Trotts. Constructing hierarchies for triangle meshes. *IEEE Visualization and Computer Graphics*, 4(2):145–161, 1998.

[27] B. Hamann; B.W. Jordan; D.A. Wiley. On a construction of a hierarchy of best linear spline approximations using repeated bisection. *IEEE Transactions on Visualization and Computer Graphics*, 5(1/2):30–46, 190(errata), 1999.

[28] I. J. Trotts;B. Hamann; K. I. Joy;D. F. Wiley. Simplification of tetrahedral meshes. In D. S. Ebert;H. Hagen; H. E. Rushmeier, editor, *Visualization '98*, pages 287–295. IEEE Computer Society Press, 1998.