Title
Piecewise Optimal Triangulation for the Approximation of Scattered Data in the Plane

Authors
Bertram, Martin
Barnes, James C.
Hamann, Bernd
et al.

Publication Date
2000

Peer reviewed

Piecewise Optimal Triangulation for the Approximation of Scattered Data in the Plane

Martin Bertram [a,b] James C. Barnes [b] Bernd Hamann [b]
Kenneth I. Joy [b] Helmut Pottmann [c] Dilinur Wushour [b]

[a] *Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, P.O. Box 808, L-551, Livermore, CA 94551, U.S.A.*

[b] *Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California at Davis, Davis, CA 95616-8562, U.S.A.*

[c] *Institut für Geometrie, Technische Universität Wien, Wiedner Hauptstraße 8-10, A-1040 Wien, Austria*

Abstract

We present an efficient algorithm to obtain a triangulated graph surface for scattered points $(x_i\ y_i)^T,\ i = 1...n$, with associated function values f_i. The maximal distance between the triangulated graph surface and the function values is measured in z-direction $\Big(z = f(x,y)\Big)$ and lies within a user-defined tolerance. The number of triangles is minimized locally by adapting their shapes to different second-degree least squares approximations of the underlying data. The method consists of three major steps: (i) subdividing the given discrete data set into clusters such that each cluster can be approximated by a quadratic polynomial within a prescribed tolerance; (ii) optimally triangulating the graph surface of each quadratic polynomial; and (iii) "stitching" the triangulations of all graph surfaces together. We also discuss modifications of the algorithm that are necessary to deal with discrete data points, without connectivity information, originating from a general two-manifold surface, i.e., a surface in three-dimensional space that is not necessarily a graph surface.

Key words: triangulation, data-dependent triangulation, approximation, hierarchical approximation, quadratic polynomials, clustering.

1 Introduction

Many problems in scientific visualization deal with the representation of surfaces. Examples are shock waves in three-dimensional fluid dynamic simulations, isosurfaces of certain scalar parameters of simulated hydrodynamics

data, or geographic height fields. In many cases, these surfaces are known only at certain points that either lie on a regular, uniform grid or that are randomly located without any connectivity. We refer to these points as scattered data.

Even in those cases where a continuous, analytical surface representation is available, its evaluation may be too inefficient for visualization purposes and a new discrete representation needs to be generated for the original surface. To visualize surfaces efficiently, modern graphics hardware requires a surface representation to consist of polygonal patches, preferably triangles. The number of triangles to be rendered has a major impact on the efficiency of the visualization process. In most applications, approximation error estimates should also be known for the triangulated surfaces.

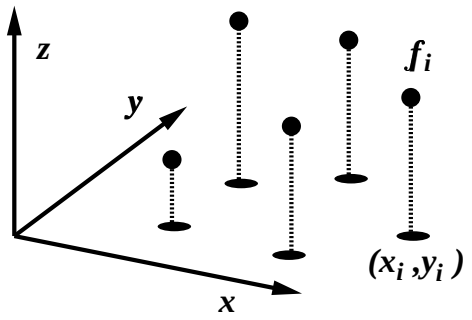

Fig. 1. Scattered points with associated function values.

The specific problem for which we provide a constructive solution can be stated as follows: Given a set of scattered points $(x_i\ y_i)^T,\ i = 1...n$, with associated function values f_i, determine a linear, triangular spline function $s(x, y)$ which approximates the given scattered data within a user-defined tolerance ε such that

$$max\ \{|f_i - s(x_i, y_i)|\}_{i=1}^{n}\ \leq\ \varepsilon, \tag{1}$$

see Figure 1.

1.1 Related Work

A variety of triangulation approaches optimizing different geometrical criteria or analytic cost functions exist. Some early approaches, e.g., [9] apply the Delaunay triangulation, which leads to low aspect ratios of the resulting triangles (the ratios between the radii of the circumscribed and inscribed circles for all triangles). However, in the case of data-dependent triangulations, see [4][7][23][28], the use of long and skinny triangles can reduce the approximation error. To obtain a global optimum for a cost function, multidimensional optimization methods, such as simulated annealing, can be applied, see [21][30].

A different class of algorithms does not rely on the original data points as triangle vertices. Nadler [24] and D'Azevedo [1] apply a coordinate transformation to generate an optimal triangulation for a certain class of analytical functions. This transformation is based on principal axes, which are also used in our approach. Quak et al. [27] apply least squares fitting to linear splines to approximate scattered data. Nielson [25] suggests the use of an affine-invariant norm to obtain a triangulation that does not depend on a specific coordinate system.

With the amount of data produced by modern computational simulations and imaging technology growing rapidly, it becomes important to handle data locally and on multiple levels of detail. A survey on multi-resolution methods in the context of view-dependent rendering is given by Heckbert [15]. A variety of view-dependent triangulation algorithms, especially useful for terrain data, are discussed in [6][19][20][31]. For more general multiresolution triangulations, we refer to [2][11][10][12][14].

Once a triangulation is constructed for a given scattered data set, the mesh can be reduced to a coarser level of detail. Mesh reduction strategies are described, for example, in [18][29]. Not only reduction of detail is possible but also synthesis. In this context, the concept of the discrete wavelet transform (DWT), see [22], which is a highly efficient method for signal processing and data compression on regular grids, has been generalized to arbitrary triangular meshes, see [5][13].

A more general problem than graph surface reconstruction from scattered points in the plane with associated function values is the following one: Given a set of scattered points in three-dimensional space, reconstruct a valid two-manifold surface that may or may not be the graph surface of a bivariate function. In this case, the surface topology needs to be recognized and mapped locally to regions that can be treated as graph surfaces. Surface reconstruction algorithms for arbitrary topology are discussed in [8][16][17].

1.2 Our Approach

Most triangulation approaches operate directly on the data. In contrast, our method is based on an intermediate data representation, which is a piecewise quadratic least squares fit of the original data. The piecewise quadratic representation significantly reduces noise that might be present in the data and improves the quality of the final triangulation.

Quadratic polynomials have the property that their graph surfaces can be approximated by a "regular" triangulation in such a way that all triangles imply the same approximation error. This property provides a method for rapidly

generating a set of optimally shaped triangles for each quadratic polynomial, which makes our approach highly efficient.

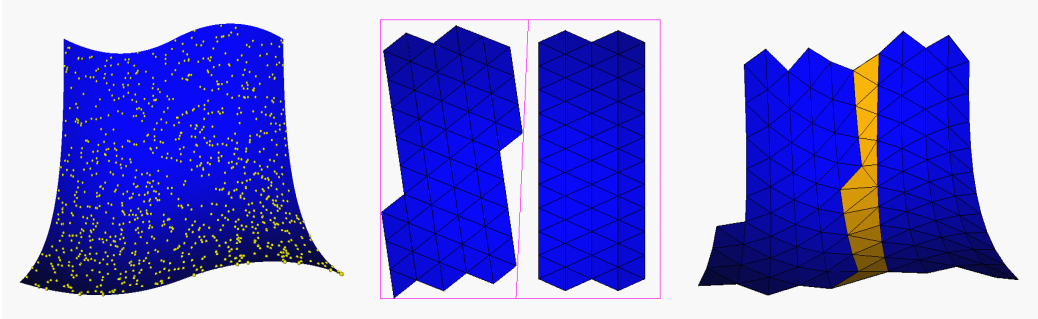

Fig. 2. Left: scattered points sampled from smooth graph surface; middle: clusters with optimal triangulations; right: final triangulation.

Our paper is organized as follows: In Section 2, we describe the clustering step, which is needed to determine the subsets of scattered data to be approximated by quadratic polynomials. This step uses a hierarchical subdivision method based on piecewise quadratic approximation. Section 3 describes the core of our method, the triangulation step used to generate the desired optimal, regular triangulations for each quadratic polynomial. We explain in detail how quadratic graph surfaces of different types (elliptic, hyperbolic, and parabolic) can be approximated by optimal triangulations. In Section 4, we discuss how to stitch the individual triangulations for the quadratic polynomials together to define a single triangulated graph surface by filling the remaining gaps with triangles. In Section 5, we provide numerical results of our method. A discussion of our algorithm and its generalization to arbitrary two-manifold surfaces is provided in Section 6. Figure 2 illustrates the data and the steps involved in the triangulation process.

2 Adaptive Clustering

Given a set of scattered points with associated function values, the first step is to subdivide these into smaller subsets that can be approximated by quadratic polynomials. The clustering method should be highly efficient to be applicable to massive data sets, and it should yield clusters of almost maximal size while maintaining a prescribed error bound. The clustering process produces a decomposition of the plane into polygonal, non-overlapping regions, which cover the entire domain of the surface that is eventually to be triangulated. These regions should be convex to simplify the boundary treatment during final stitching of the optimal triangulations corresponding to the individual least squares quadratic polynomials.

We use a subdivision approach that results in a binary space partition tree

(BSP tree). The approach initially considers the whole data set as a single cluster. The set of clusters is then recursively split into smaller clusters until each of these can be approximated, within the given error bound, by a least squares quadratic polynomial.

In the following, we summarize the splitting procedure for each cluster. First, we compute a quadratic function

$$f(x,y) \;=\; \sum_{j,\ k\geq 0,\ j+k\leq 2} c_{j,k}\ x^j\ y^k \;=\; \sum_{0\leq|\mathbf{j}|\leq 2} \mathbf{c_j}\ \mathbf{x^j} \tag{2}$$

for each cluster of data $\{(x_i\ y_i)^T, f_i\}_{i=1}^n$, employing least squares fitting, such that the residual

$$r(\mathbf{c}) \;=\; \sum_{i=1}^{n} \Big(f_i - f(x_i,y_i)\Big)^2 \tag{3}$$

is minimized among all coefficient vectors $\mathbf{c} = (c_{00}\ c_{10}\ c_{01}\ c_{11}\ c_{20}\ c_{02})^T$. For a description of least squares fitting we refer to [3]. Second, a cluster's maximal error, defined as $max\{|f_i - f(x_i,y_i)|\}_{i=1}^n$, is estimated to decide whether a cluster needs to be subdivided or not. If the maximal error is within the given tolerance, then subdivision is no longer performed.

For a cluster whose maximal error is above the given tolerance, a straight line is constructed in the xy-plane along which the cluster is subdivided. We have compared different methods to select this splitting line in order to minimize the resulting number of clusters. Principal component analysis (PCA) can be used for clustering, see [16], but, unfortunately, it does not depend on the error distribution within a cluster. In the case of a nearly parabolic graph surface, PCA produces too many clusters, since it does not generate skinny clusters. A more expensive subdivision scheme, which minimizes the approximation error for the two clusters resulting from a split, can lead to skinny clusters in the limit, even when this is not justified by the data geometry.

We now describe an efficient subdivision approach that works well for all our numerical examples. It is based on the idea of using weighted centers, called p^+ and p^-, which correspond to the positive and negative errors, respectively, given by

$$p^+ \;=\; \frac{\sum_{i=1}^n\ w_i^+\ \mathbf{x}_i}{\sum_{i=1}^n\ w_i^+}, \quad w_i^+ \;=\; \begin{cases} f_i - f(\mathbf{x}_i),\ if\ \ f_i - f(\mathbf{x}_i) > 0 \\ 0, \qquad\qquad otherwise \end{cases}$$

and

$$p^- = \frac{\sum_{i=1}^{n} w_i^- \mathbf{x}_i}{\sum_{i=1}^{n} w_i^-}, \quad w_i^- = \begin{cases} f(\mathbf{x}_i) - f_i, \; if \;\; f_i - f(\mathbf{x}_i) < 0 \\ 0, \qquad\qquad otherwise, \end{cases}$$

where $\mathbf{x}_i = (x_i, y_i)$. A cluster is subdivided along the perpendicular bisector of the line segment $\overline{p^+p^-}$ (Figure 3). If p^+ and p^- happen to be identical, the direction of subdivision will be arbitrary.

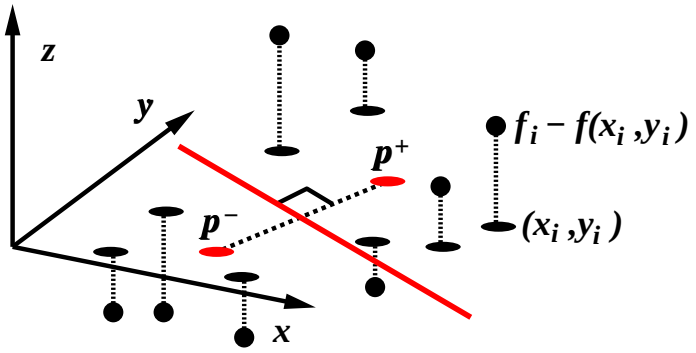

Fig. 3. Subdivision along perpendicular bisector between p^+ and p^-.

Even though the subdivision scheme generates two clusters of almost equal size after each split, it may happen that some of the final clusters become very small. If a cluster contains less than six points, or if the points are arranged such that the determinant of the least squares fitting problem vanishes, the approximating quadratic function will not be uniquely defined. In this case, a reduced set of linear basis functions is utilized for the least squares fitting step.

To reduce the computational cost of the initial clustering step, we limit the number of points which contribute to the least squares fit to a user-defined maximal number n_{max}. For a cluster containing more than n_{max} points, only a subset of n_{max} randomly selected points is considered for the quadratic approximation problem. If a cluster needs to be subdivided, the points p^+ and p^- are estimated from the same subset of selected points. If a cluster containing more than n_{max} points is not subdivided, this decision needs to be verified based on all its points. n_{max} should be selected large enough such that this case is rare.

This modification reduces the computational cost of each least squares fitting step to a constant. Since less than n cluster subdivisions are necessary, the overall cost of the least squares fitting is $O(n)$. Each cluster subdivision still requires one to process all points in the cluster, which leads to a total cost of $O(n \; log \; m)$, where m is the number of clusters generated for n data points. However, the splitting operations for building the BSP tree are very

inexpensive, and one could execute them in parallel.

3 Triangulating Quadratic Functions

Once the set of approximating quadratic polynomials has been determined, we construct an optimal triangulation for the corresponding regions in the xy-plane. We call a triangulation "optimal" if it satisfies a given approximation error with respect to the quadratic polynomial and covers the associated convex region in the domain with a minimal number of triangles. The final maximal errors for all triangulated regions are bounded by the sums of the maximal errors for the quadratic least squares approximations and the errors introduced by the optimal triangulations.

The construction of an optimal triangulation for a quadratic graph surface is discussed in [26], which provides the basis of our constructive approach. It is shown in [26] that there exists an optimal triangulation composed of congruent triangles in the xy-plane for every quadratic polynomial. The basic idea is to exploit affine invariance of the error norm to transform a quadratic graph surface to a prototype for which a regular triangulation satisfying an error bound is known. In this Section, we review the essential ideas related to the optimal triangulation step and provide necessary implementation details.

There are three fundamental observations that motivate our constructive approach:

(1) Two approximating triangles whose corresponding edge pairs have the same lengths and are parallel in the xy-plane and whose vertices have the same approximation errors (measured in z-direction) share the same maximal error, see Figure 4.
(2) The maximal error of an approximating triangle is invariant under affine transformations in the xy-plane.
(3) The ratio of two triangle areas in the xy-plane is invariant under affine transformations.

The first observation is due to the fact that the second-order partial derivatives of a quadratic graph surface are constant. This implies that a triangle can be translated in the xy-plane without changing its error profile. The second and third observations guarantee that the affine map of an optimal triangulation is also optimal. This allows us to transform the graph surface to a prototype for which an optimal triangulation consisting of congruent triangles can be constructed.

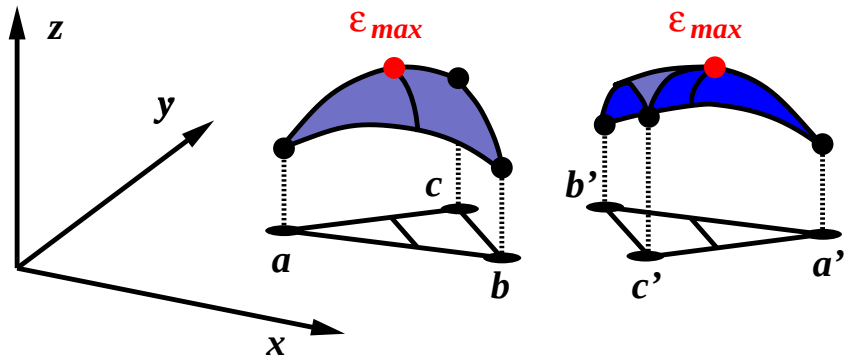

Fig. 4. Error functions for two triangles approximating a quadratic polynomial. The corresponding edges $\overline{ab}$ and $\overline{a'b'}$, $\overline{bc}$ and $\overline{b'c'}$, and $\overline{ac}$ and $\overline{a'c'}$ are parallel and have same lengths. Error profiles along corresponding, parallel line segments are the same.

3.1 Principal Axis Transformation

In matrix notation a quadratic function can be written as

$$f(x,y) \;=\; (x\ y)\begin{pmatrix} c_{20} & c_{11} \\ c_{11} & c_{02} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \;+\; (c_{10}\ c_{01})\begin{pmatrix} x \\ y \end{pmatrix} \;+\; c_{00}. \qquad (4)$$

To analyze its principal axes, i.e., the directions of minimal and maximal second derivative (or curvature), we can neglect the constant and linear terms that do not affect the approximation error. Thus, we only need to consider the quadratic form

$$(x\ y)\begin{pmatrix} c_{20} & c_{11} \\ c_{11} & c_{02} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \mathbf{x}^T C\mathbf{x} \;=\; 0. \qquad (5)$$

The graph surface's principal curvatures are implied by the eigenvalues λ_1 and λ_2 of the matrix C and the principal axes are determined by the corresponding eigenvectors $\mathbf{e_1}$ and $\mathbf{e_2}$, see [3]. Depending on the sign of the eigenvalues, we have to distinguish between three different surface types:

- The surface is elliptic if $\lambda_1\lambda_2 \;> 0$.
- The surface is hyperbolic if $\lambda_1\lambda_2 \;< 0$.
- The surface is parabolic if $\lambda_1\lambda_2 \;= 0$.

For each type, a different optimal triangulation scheme is used.

The graph surface of a quadratic function f of elliptic type is equivalent to the graph surface of the paraboloid

$$g(\overline{x},\overline{y}) \;=\; \overline{x}^2+\overline{y}^2, \tag{6}$$

i.e., there exists a linear transformation for the arguments of g that produces the same graph surface as f, neglecting linear and constant terms. In the case of positive eigenvalues, this transformation is given by

$$\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & \frac{1}{\sqrt{\lambda_2}}\, e_{21} \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & \frac{1}{\sqrt{\lambda_2}}\, e_{22} \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \;=\; G \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}, \tag{7}$$

where the matrix entries are determined by the normalized eigenvectors $\mathbf{e_1} = (e_{11}, e_{12})^T$ and $\mathbf{e_2} = (e_{21}, e_{22})^T$. This result can be verified by inserting (7) into the quadratic form (5). Considering the orthogonality of eigenvectors, G^TCG is the identity matrix, and it therefore reproduces the quadratic form of g.

We need to construct an optimal triangulation for g, which will also be optimal for f, after applying the above transformation. Due to the rotational symmetry of g, all triangles that share the same circumscribed circle of radius $\sqrt{2\varepsilon}$ have the same maximal error ε. Hence, an equilateral triangle, with arbitrary orientation in the xy-plane — like the one in Figure 5 given by the points $(-\sqrt{2\varepsilon}, 0)^T$, $(\sqrt{0.5\varepsilon}, \sqrt{1.5\varepsilon})^T$, and $(\sqrt{0.5\varepsilon}, -\sqrt{1.5\varepsilon})^T$ — has maximal area. The maximal error occurs at the center and at the vertices of the triangle.

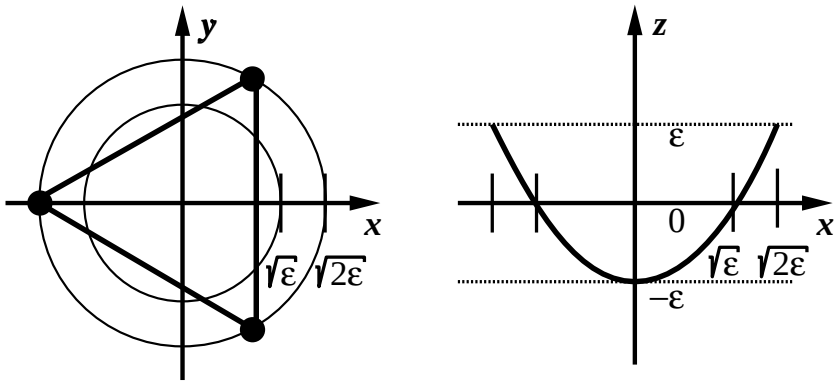

Fig. 5. Optimal triangle and error profile for $g(x,y) = x^2 + y^2$.

In the case of positive eigenvalues, the vertices must be placed below the quadratic surface by a distance ε to obtain the error profile shown in Figure 5. For negative eigenvalues, the function f can be replaced by $-f$, which implies

positive eigenvalues. The only difference in this case is that the vertices need to be placed above the surface by ε.

To obtain the final triangulation only two of the edges need to be transformed according to (7). The transformed edges define a regular triangulation that is established for the convex region corresponding to the cluster, emanating from its centroid, see Figure 2. Only triangles that lie entirely in the cluster region are generated, leaving an untriangulated gap along the cluster boundaries.

3.3 The Hyperbolic Case

In analogy to the elliptic case, we can choose a single prototype for a hyperbolic quadratic polynomial. We choose the polynomial

$$h(\overline{x}, \overline{y}) \;=\; \overline{x}^2 - \overline{y}^2. \tag{8}$$

To transform the quadratic form of h into the quadratic form of a given quadratic hyperbolic function f, with eigenvalues $\lambda_1 > 0$ and $\lambda_2 < 0$ and normalized eigenvectors $\mathbf{e_1} = (e_{11}, e_{12})^T$ and $\mathbf{e_2} = (e_{21}, e_{22})^T$, we apply the map

$$\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & \frac{1}{\sqrt{-\lambda_2}}\, e_{21} \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & \frac{1}{\sqrt{-\lambda_2}}\, e_{22} \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix} \;=\; H \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}. \tag{9}$$

The transformation is correct, since it reproduces the quadratic form of h:

$$H^T C H \;=\; \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Due to the nature of hyperbolic surfaces, it seems necessary to place the triangle vertices exactly on the surface. Offsetting a vertex may reduce the error of a single triangle, but, at the same time, it increases the error of an adjacent triangle. As shown in [26], an optimal triangle for the polynomial h has the vertices $(\sqrt{\varepsilon}, \sqrt{\varepsilon})^T$, $(\sqrt{\varepsilon}, -\sqrt{\varepsilon})^T$, and $\left((1-\sqrt{5})\sqrt{\varepsilon}, 0\right)^T$, see Figure 6.

Since all vertices lie on the surface, the maximal error occurs at the midpoint of each triangle edge. We note that the graph surface of h has zero curvature (and zero second derivatives) along the lines $x + y = 0$ and $x - y = 0$, which implies that the error inside a triangle cannot be greater than the maximal

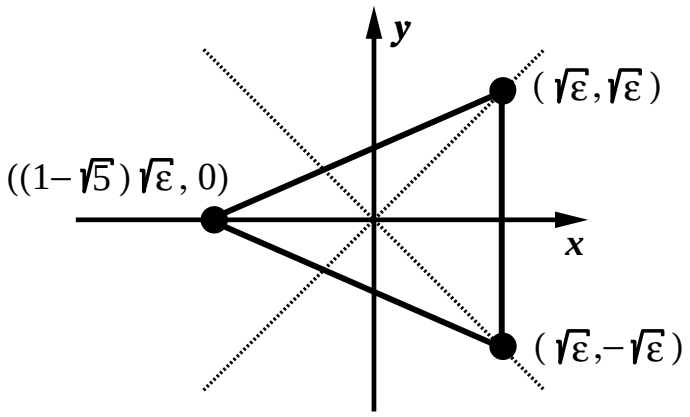

Fig. 6. Optimal triangle for hyperbolic polynomial $h(x,y) = x^2 - y^2$.

error on its edges. The maximal error for any edge $\overline{\mathbf{ab}}$ is given by

$$\varepsilon_{\mathbf{ab}} \;=\; \frac{1}{4}\left|(a_x - b_x)^2 - (a_y - b_y)^2\right|. \tag{10}$$

The final triangulation is established in the same way as in the case of elliptic surfaces, using two edges transformed according to (9). (In the case of $\lambda_1 < 0$ and $\lambda_2 > 0$, the triangulation can be obtained by considering the quadratic form of $-f$.)

3.4 The Parabolic Case

In the parabolic case, one of the eigenvalues λ_1 and λ_2, and consequently one of the principal curvatures, is zero. (In the case of two zero eigenvalues the surface is a plane, and the cluster region can be triangulated by only considering the vertices of its convex boundary polygon.) We now consider the case $\lambda_1 > 0$ and $\lambda_2 = 0$. We can transform the quadratic form of the function

$$p(\overline{x},\overline{y}) \;=\; \overline{x}^2 \tag{11}$$

into the quadratic form of f by using the map

$$\begin{pmatrix} x \\ y \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{\sqrt{\lambda_1}}\, e_{11} & 0 \\ \frac{1}{\sqrt{\lambda_1}}\, e_{12} & 0 \end{pmatrix} \begin{pmatrix} \overline{x} \\ \overline{y} \end{pmatrix}. \tag{12}$$

The error profile for a line segment is uniquely determined by its projection onto the first normalized eigenvector $\mathbf{e}_1$ of the quadratic form (Figure 7). The length of this component should be at most $\sqrt{4\varepsilon/\lambda_1}$. As in the elliptic case,

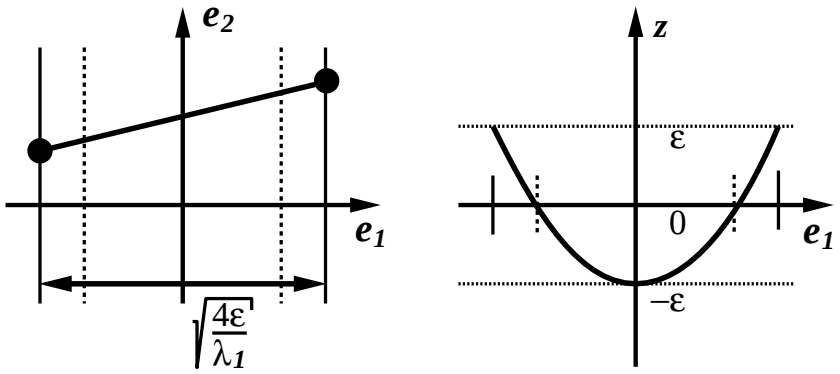

Fig. 7. Line segment with error ε on parabolic surface.

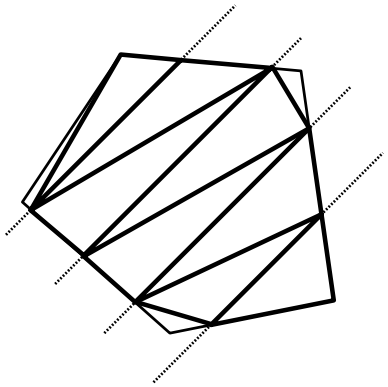

Fig. 8. Triangulation of region with parabolic graph surface.

the vertices need to be placed below the surface by the distance ε for $\lambda_1 > 0$ and above it for $\lambda_1 < 0$. In the case that $\lambda_1 = 0$ and $\lambda_2 \neq 0$ the eigenvalues and eigenvectors simply change their roles. The cluster region is finally triangulated by two, typically skinny triangles for each subregion of width $\sqrt{4\varepsilon/\lambda_1}$ (Figure 8). To leave some space at the cluster boundary for the stitching process, we shrink the cluster region by a small percentage towards its centroid.

4 Merging Triangulations

Based on the BSP tree defining the cluster hierarchy and the set of individual optimal triangulations, the final step of the algorithm stitches these triangulations together by triangulating the gaps along the cluster boundaries. This is done in the reverse order of the cluster subdivision process, which allows us to stitch exactly two triangulations at a time together along the bisecting lines that separate the corresponding cluster regions, see Figure 9.

To efficiently identify the polygon strips that need to be connected, we keep track of the boundary of each triangulation. We initially need to construct a

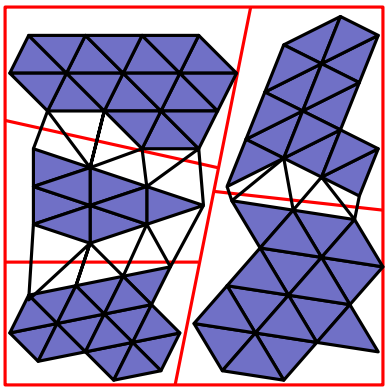

Fig. 9. "Pairs" of triangulations are stitched together along straight line cluster bisectors.

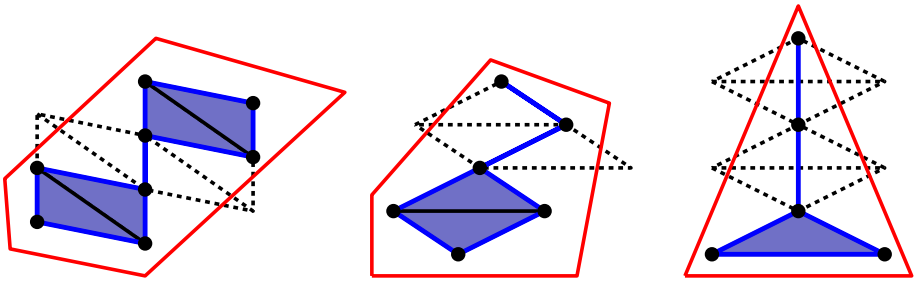

Fig. 10. Special stitching cases: All simplices — triangles, edges, and vertices — need to be enclosed by the boundary polygon.

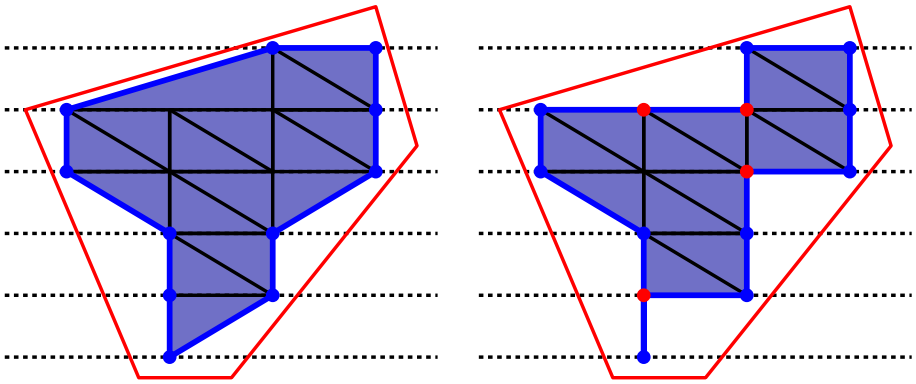

Fig. 11. Boundary construction for optimal triangulations. Left: boundary vertices for each scanline are inserted into triangulation boundary; right: additional vertices are used to minimize enclosed area.

closed boundary polygon for each optimal triangulation. This is straightforward in the case of a triangulation resulting from a parabolic or planar graph surface. In the elliptic and hyperbolic cases the triangulation inside the cluster region may be disconnected or contain line segments that are not part of any triangle. Some special cases are shown in Figure 10.

The boundary of an optimal triangulation is defined as a single loop that encloses all triangles, edges, and vertices of the optimal triangulation that are

inside the cluster region. It does not enclose areas other than optimal triangles. Thus, a vertex can be part of the boundary polygon multiple times.

To efficiently construct a triangulation boundary we use a scanline algorithm that identifies the left- and right-most vertices inside the cluster region for each scanline parallel to the longest edge in a triangulation and that inserts these vertices into the boundary polygon. As illustrated in Figure 11, additional vertices need to be added to the boundary polygon so that it encloses only triangles belonging to the optimal triangulation. This is done by traversing each scanline again and by inserting the vertices that establish the connections to the two neighboring scanlines as well as all vertices between these connecting ones.

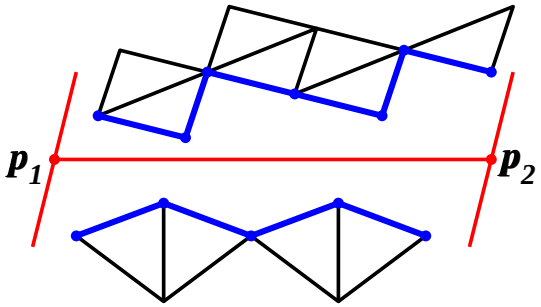

Fig. 12. Polygon strips along line segment $\overline{p_1p_2}$.

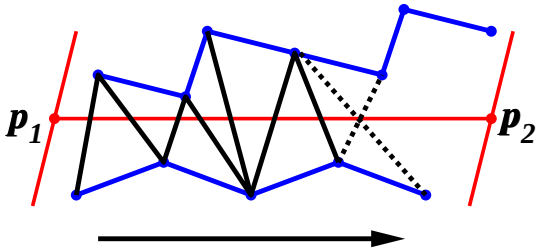

Fig. 13. Shorter edges are preferred when stitching two triangulations.

The BSP tree structure is traversed in the reverse order of cluster subdivision to merge the corresponding triangulations. We only need to discuss the stitching process for two optimal triangulations whose associated cluster regions are separated by the line segment $\overline{p_1p_2}$, see Figure 12. First, we extract from the two triangulation boundaries two polygon strips along $\overline{p_1p_2}$, starting with the vertex closest to p_1 and ending with the one closest to p_2. Second, we construct triangles by marching along the two polygon strips and by connecting vertex pairs using vertices from opposite sides of $\overline{p_1p_2}$. Considering the two possible choices for defining the next triangle, we choose an edge with minimal length, see Figure 13. One must not produce triangles with negative area, i.e., triangles that overlap others. In the case that both choices would produce triangles with negative areas, we construct a triangle from three consecutive vertices in one of the polygon stips and thus eliminate the vertex in the middle. If the stitching process would still produce negative areas for both choices, we insert a triangle from three vertices of the other polygon strip.

The triangles resulting from the stitching process are not guaranteed to satisfy the given error bound. For a rigorous treatment, the final approximation errors must be estimated for all scattered data points located inside the regions in the xy-plane defined by the triangles resulting from the stitching procedure. Triangles that do not satisfy the error bound need to be modified. This can be done by flipping an edge between two triangles or by inserting the sample with greatest approximation error as a vertex into the triangulation. The errors for all modified triangles need to be checked again. In our numerical examples we use a greater error bound for quadratic approximation than for triangulating polynomials. This implies that the total error bound is relatively loose for the triangles obtained from the stitching process and that violations of the error bound are extremely rare.

5 Numerical Examples

We have applied our algorithm to approximate scattered data sets sampled randomly from three different analytical functions and to approximate two terrain data sets sampled on regular grids. The analytical functions are defined as follows:

$$f_1(x,y) \;=\; 0.5\,x \;+\; \begin{cases} x^2 \;+\; y^2 \;if\; x < 0 \\ -x^2 \;+\; y^2 \;otherwise, \end{cases} \qquad x,\; y\,\epsilon\,[-0.5,\; 0.5]\,;$$

$$f_2(x,y) \;=\; e^{-(x^2\;+\;y^2)}, \quad x,\; y\,\epsilon\,[-2.2,\; 2.2]\,;$$

and

$$f_3(x,y) \;=\; sin(x^2)\;sin(y^2), \quad x,\; y\,\epsilon\,[-3,\; 0]\,.$$

The numbers of clusters and triangles corresponding to different error bounds are shown in Table 1. The error bounds (measured in percents of the distance between minimal and maximal function values) are split in a $4:1$ ratio between quadratic approximation and triangulation of the polynomials.

Approximation results for the two terrain data sets "St. Hellens" and "Crater Lake" are shown in Table 2. The corresponding triangulations are depicted in Figures 17 and 18. We compared the error bounds for our triangulations with the approximation errors of regular triangulations that have approximately the same number of triangles and that are obtained from a rectilinear grid. The error bounds for our method are split in a $9:1$ ratio between quadratic

Function	No. Samples	Error Bound [%]	No. Clusters	No. Triangles
f_1	1000	1.0	2	179
		0.3	3	662
f_2	3000	2.0	35	437
		0.5	73	1479
f_3	3000	10.0	38	332
		3.0	79	938

Table 1
Approximation results for sets of points sampled from analytical functions. Figures 14–16 show the resulting triangulations.

approximation and triangulation of polynomials. Since the terrain data sets are more "noisy" than samples from analytical functions used above, we need to allocate a larger fraction of the error bound for the quadratic approximation. The computation times shown in Table 2 have been obtained on a 194 MHZ MIPS R10000 processor. The computation times are split into three categories:

- (i) Adaptive Clustering.
- (ii) Constructing the Triangulation.
- (iii) Verifying the error bounds for the triangles obtained from stitching.

Dataset, No. Samples	Error [%] Our Method	Error [%] Regular Tri.	No. Clusters	No. Triangles	Comp. Time [sec] (i)	(ii)	(iii)
"St.	10.0	18.7	46	464	0.732	0.046	1.232
Hellens",	5.0	11.5	173	1,799	1.347	0.057	1.539
151,728	1.0	5.4	2,330	28,881	4.361	0.246	3.284
"Crater	10.0	22.9	114	1,149	1.029	0.054	1.464
Lake",	5.0	16.4	400	4,316	1.861	0.077	1.951
159,272	1.0	10.1	2,798	36,456	4.503	0.288	3.799

Table 2
Approximation results for terrain data sets. Figures 17 and 18 show the resulting triangulations.

In the case of tight error bounds the clustering step becomes the most expensive part of the algorithm, but it needs to be computed only once for all levels of resolution. The computation time for the triangulation (including stitching) is very small and does not depend on the number of scattered points in the initial data set. Checking the approximation errors, however, takes much com-

putation time for dense data sets. We suggest to estimate the approximation error from only a small subset of randomly selected samples.

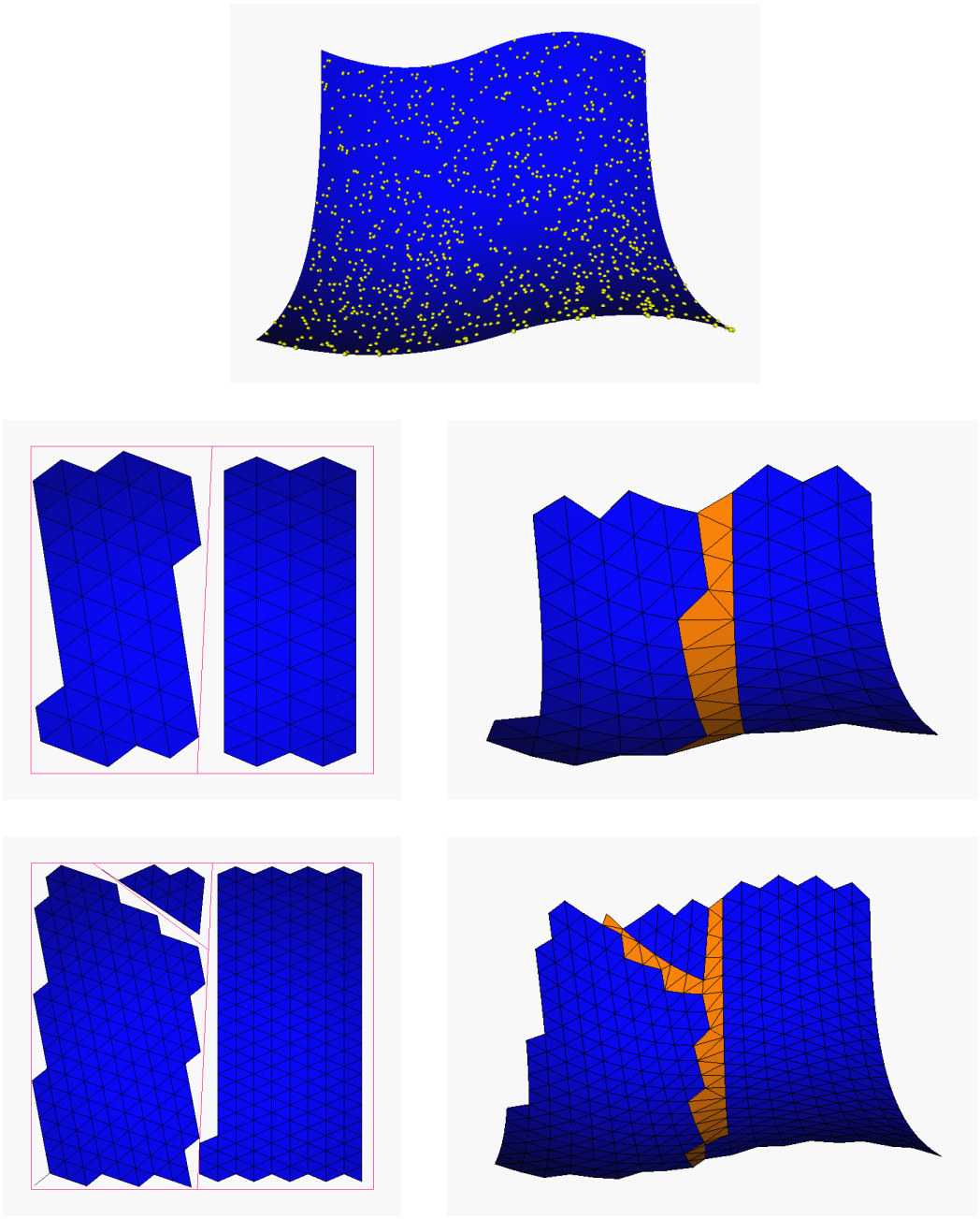

Fig. 14. Triangulations for f_1. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

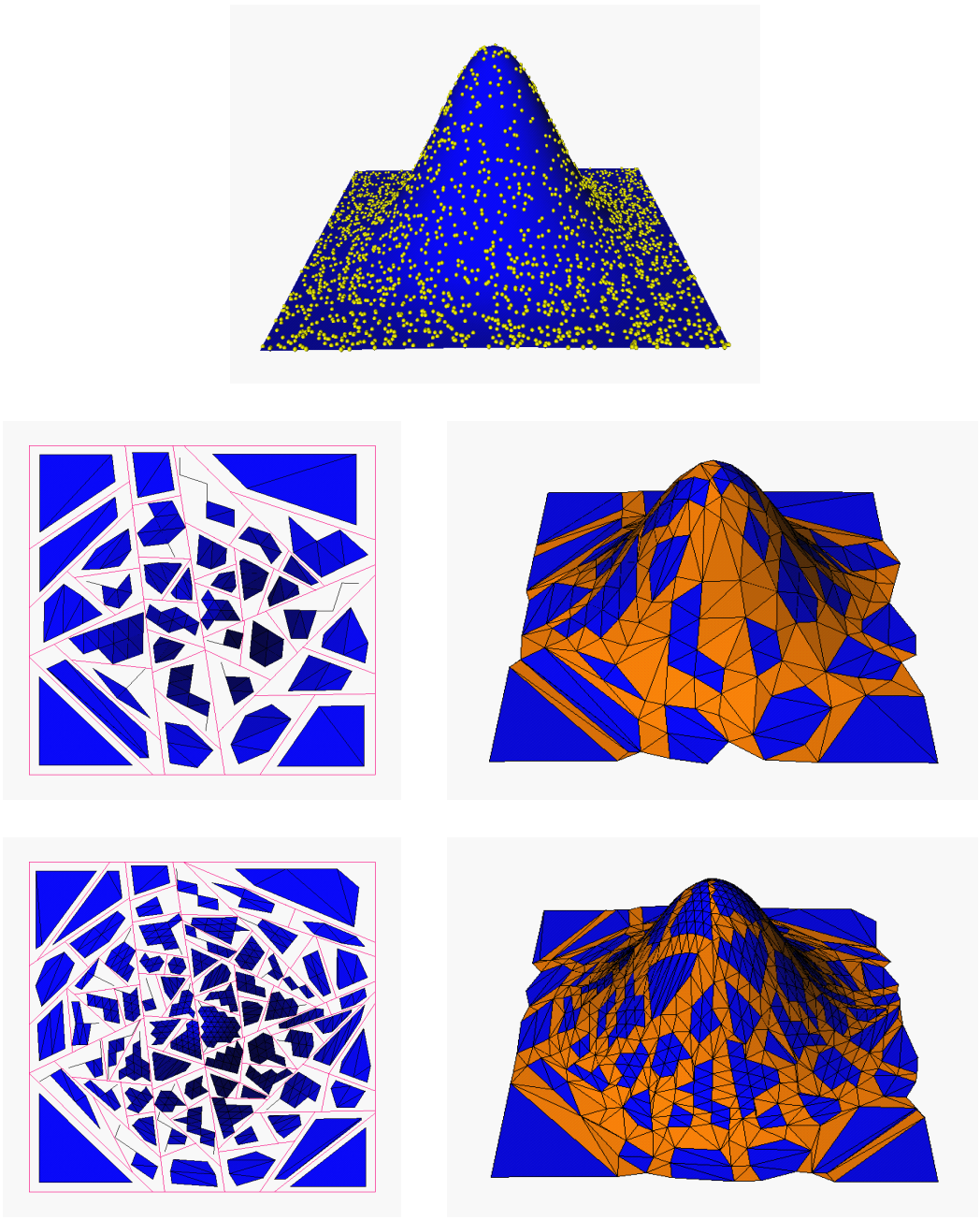

Fig. 15. Triangulations for f_2. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

6 Conclusions and Future Work

In this paper, we have introduced a new method for the construction of data-dependent triangulations for scattered bivariate data. The underlying concepts are (i) the construction of least squares quadratic polynomials locally approximating subsets of a given scattered data set within a certain tolerance; (ii)

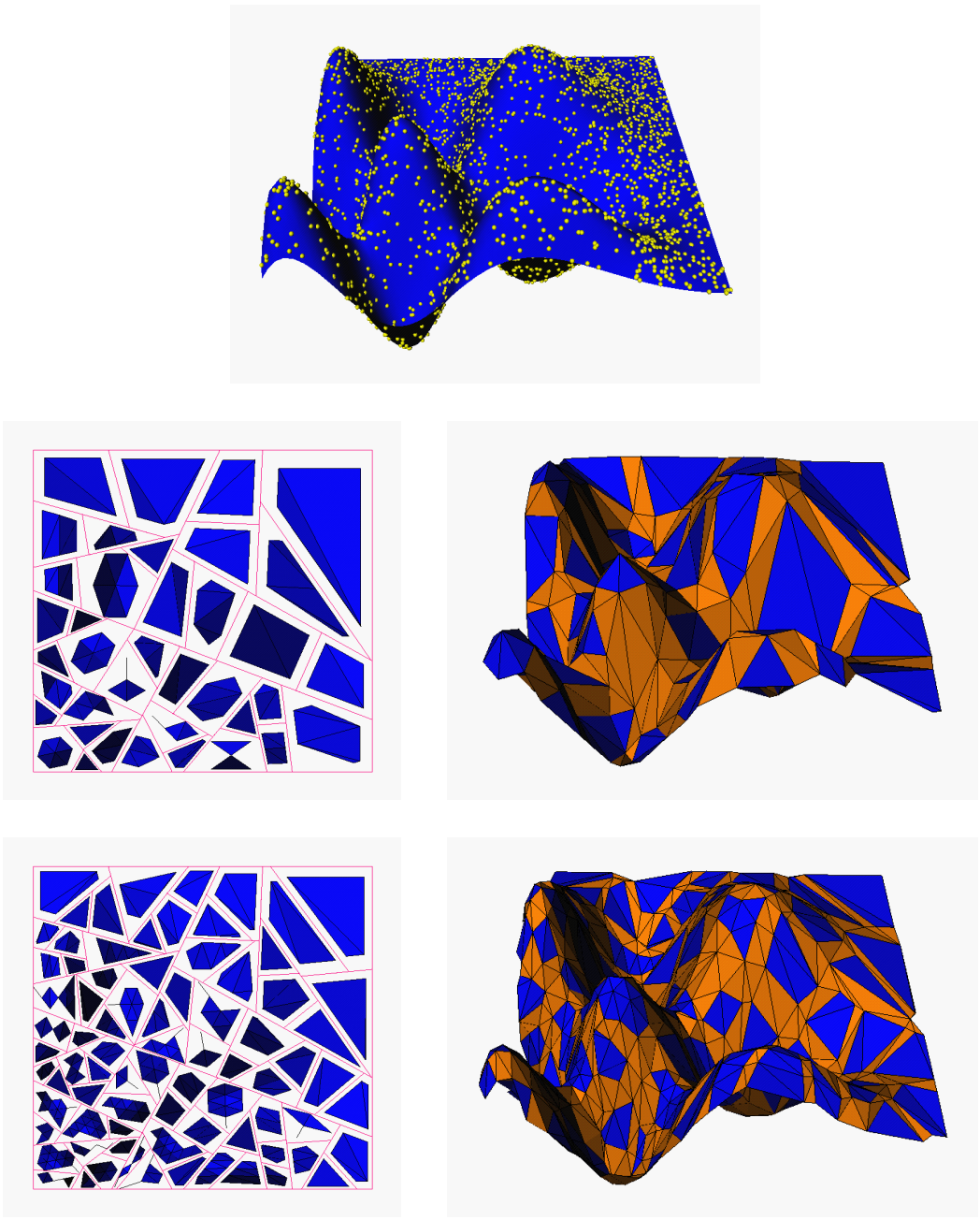

Fig. 16. Triangulations for f_3. Top: samples on original graph surface; middle: optimal cluster triangulations in xy-plane (left) and final triangulations in xyz-space (right); bottom: same for higher level of resolution.

the triangulation of the resulting quadratic graph surfaces using an optimal triangulation strategy; and (iii) merging the optimal triangulations by filling the gaps along the boundaries of the individual optimal triangulations. The computational cost for step (i) is $O(n\ log\ m)$ for n data points and m clusters. The expected computation times for steps (ii) and (iii) are linear in the number of generated triangles.

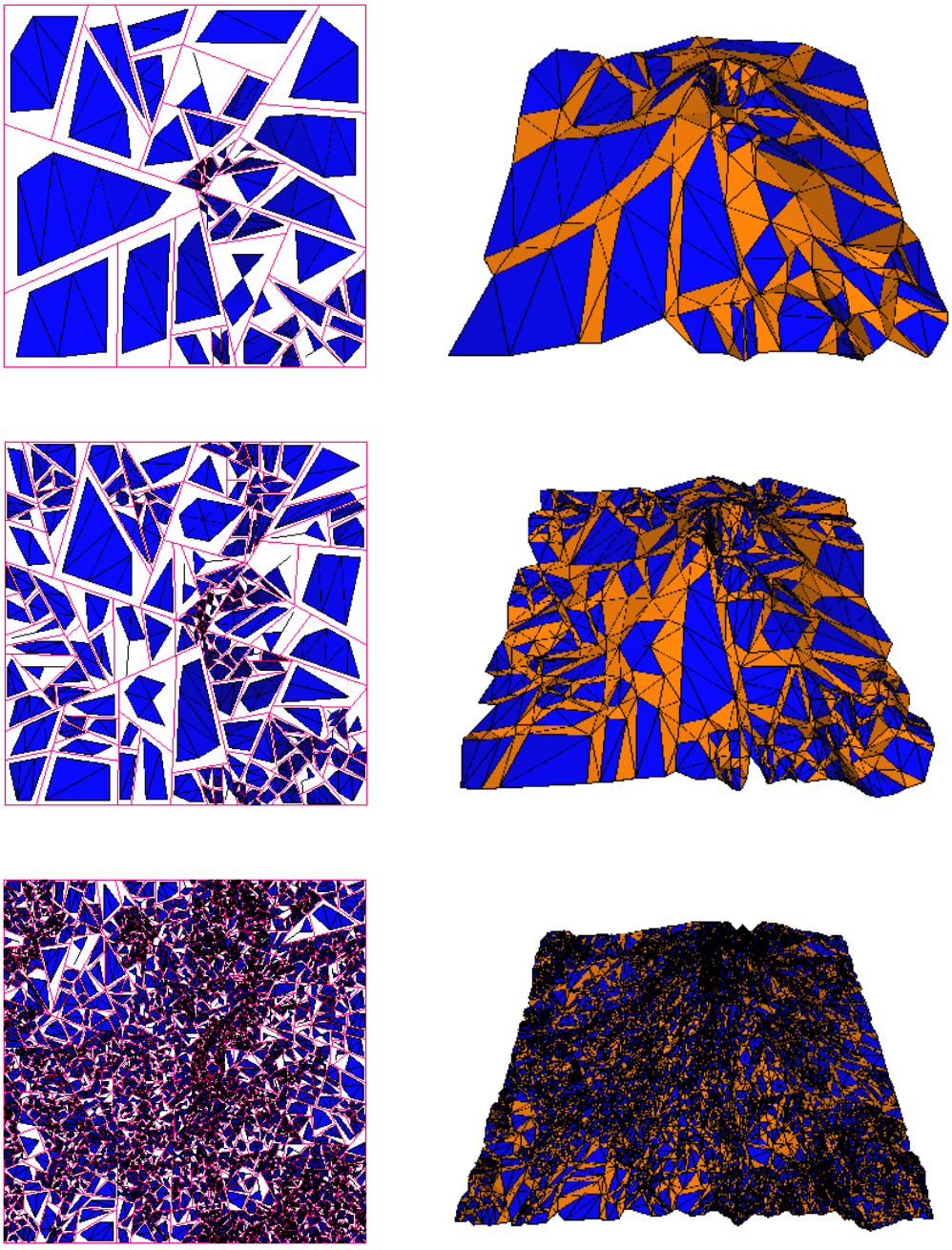

Fig. 17. Triangulations for data set "St. Hellens". Left: optimal cluster triangulations in xy-plane at three different resolutions, right: corresponding final triangulations in xyz-space.

Our algorithm is designed to efficiently generate triangulations approximating a given bivariate scattered data set at multiple levels of resolution by imposing different error thresholds. The most expensive part, generating the cluster hierarchy, is done adaptively for multiple levels of detail. For use in view-dependent rendering applications, different regions of a data set could be represented by triangulations satisfying different error bounds. The cluster

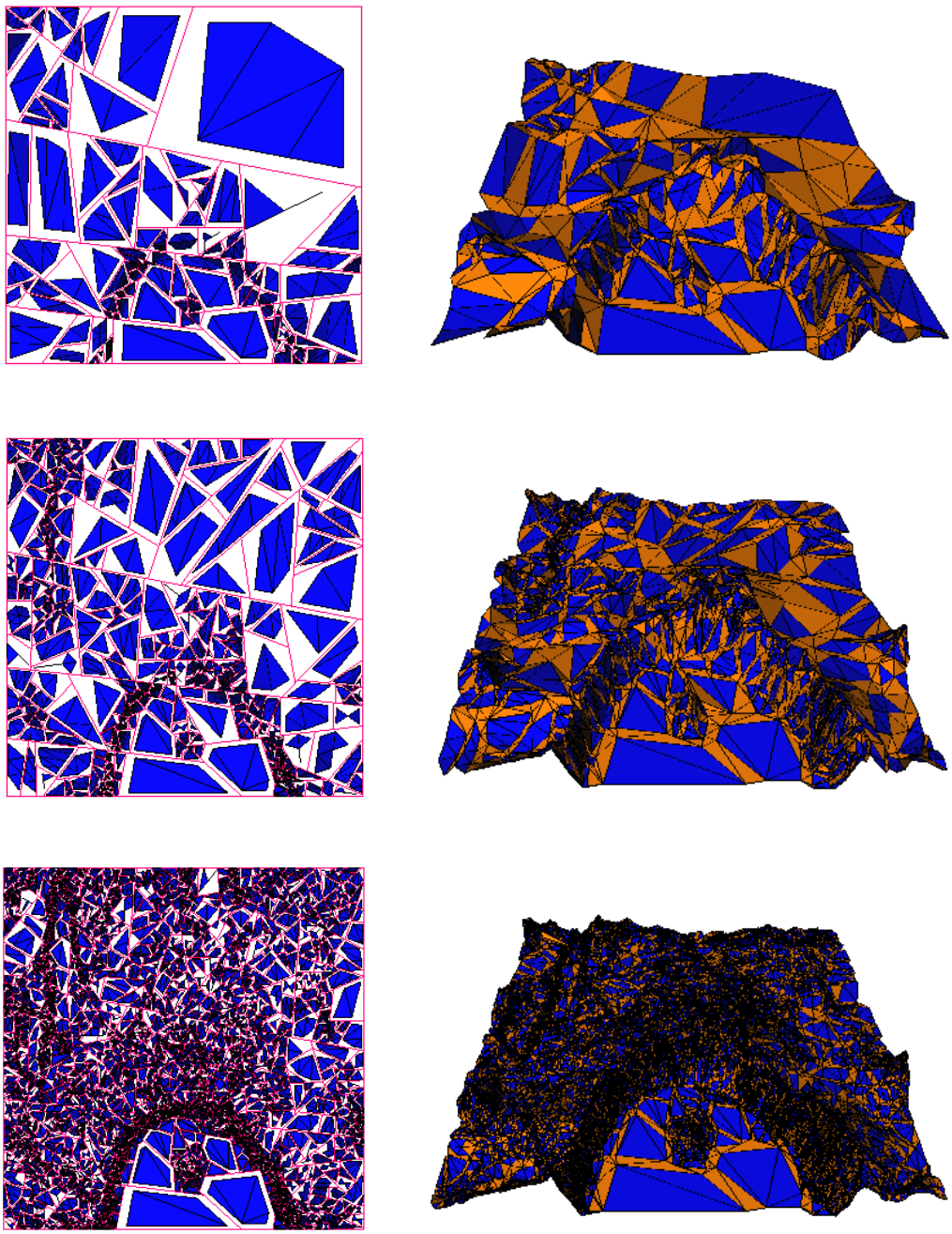

Fig. 18. Triangulations for data set "Crater Lake". Left: optimal cluster triangulations in xy-plane at three different resolutions, right: corresponding final triangulations in xyz-space.

splitting procedure needs to satisfy the tightest error bound associated with all regions that overlap the current cluster.

Considering surfaces of an arbitrary two-manifold topology, we propose to apply the adaptive clustering algorithm presented by Heckel et al. [16]. This algorithm applies PCA to subdivide point sets in three-dimensional space until

they are approximated within a prescribed tolerance by least squares planes. One can use this approach to generate hierarchical cluster representations of graph surfaces as well. The advantage of our method over the algorithm described in [16] is a smoother and more accurate surface representation using approximately the same number of triangles.

We plan to generalize our approach to approximate trivariate scalar fields defined by scattered points in three-dimensional space with associated function values by linear, tetrahedral spline functions. Several concepts generalize straightforward from the bivariate case, like building a BSP tree to define cluster regions and least squares fitting with trivariate quadratic polynomials. Other concepts are not that easily generalized, like constructing an optimal tetrahedrization for the individual trivariate polynomials that correspond to each cluster.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. This work was supported by the National Science Foundation under contracts ACI 9624034 and ACI 9983641 (CAREER Awards), 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, and the members of the Data Exploration Group at the Center for Applied Scientific Computing (CASC) at the Lawrence Livermore National Laboratory.

References

[1] E.F. d'Azevedo, *Optimal triangular mesh generation by coordinate transformation*, SIAM Journal on Scientific and Statistical Computing, Vol. 12, No. 4, July 1991, pp. 755–786.

[2] J.C. Barnes, B. Hamann, and K.I. Joy, *An edge-preserving, data-dependent triangulation scheme for hierarchical rendering*, H. Hagen, G. Nielson, and F. Post, eds., *Scientific Visualization - Methods and Applications*, IEEE Computer Society Press, Springer-Verlag, New York, to appear.

[3] W. Boehm and H. Prautzsch, *Geometric Concepts for Geometric Design*, A.K. Peters, Ltd., Wellesley, Massachusetts, 1994.

[4] J.L. Brown, *Vertex based data dependent triangulations*, Computer-Aided Geometric Design, Vol. 8, No. 3, August 1991, pp. 239–251.

[5] I. Daubechies, I. Guskov, P. Schröder, and W. Sweldens *Wavelets on irregular point sets*, Phil. Trans. R. Soc. Lond. A, to appear.

[6] M. Duchaineau, M. Wolinsky, D.E. Sigeti, M.C. Miller, C. Aldrich, and M.B. Mineev-Weinstein, *ROAMing terrain: real-time optimally adapting meshes*, Proceedings of Visualization '97, IEEE Computer Society Press, 1997, pp. 81–88.

[7] N. Dyn, D. Levin, and S. Rippa, *Algorithms for the construction of data dependent triangulations*, J.C. Mason and M.G. Cox, eds., *Algorithms for Approximation II*, Chapman and Hall, New York, 1990, pp. 185–192.

[8] H. Edelsbrunner and E.P. Mücke, *Three-dimensional alpha shapes*, ACM Transactions on Graphics, Vol. 13, No. 1, January 1994, pp. 43–72.

[9] L. De Floriani, B. Falcidieno, and C. Pienovi, *A Delaunay-based method for surface approximation*, Proceedings of Eurographics '83, Amsterdam, Netherlands, 1983, pp. 333–350.

[10] L. De Floriani, B. Falcidieno, G. Nagy, and C. Pienovi, *A hierarchical structure for surface approximation*, Computers & Graphics, Vol. 8, No. 2, 1984. pp. 183–193.

[11] T.S. Gieng, B. Hamann, K.I. Joy, G.L. Schussman, and I.J. Trotts, *Constructing Hierarchies for Triangle Meshes*, IEEE Transactions on Visualization and Computer Graphics, Vol. 4, No. 2, 1998, pp. 145–161.

[12] M.H. Gross, R. Gatti, and O. Staadt, *Fast multiresolution surface meshing*, Proceedings of Visualization '95, IEEE Computer Society Press, 1995, pp 135–42 & p. 446.

[13] I. Guskov, W. Sweldens, and P. Schröder, *Multiresolution Signal Processing for Meshes*, Computer Graphics, Proceedings of SIGGRAPH '99, ACM, 1999, pp. 325–334.

[14] B. Hamann and B. Jordan, *Triangulations from repeated bisection*, M. Daehlen, T. Lyche, and L.L. Schumaker, eds., *Mathematical Methods for Curves and Surfaces II*, Vanderbilt University Press, Nashville, Tennessee, 1998, pp. 229–236.

[15] P.S. Heckbert and M. Garland, *Multiresolution modeling for fast rendering*, Proceedings Graphics Interface '94, Canadian Inf. Process. Soc, 1994, pp. 43–50.

[16] B. Heckel, A. Uva, and B. Hamann, *Clustering-based generation of hierarchical surface models*, Late Breaking Hot Topics Proceedings, Visualization '98, IEEE Computer Society Press, 1998, pp. 41–44.

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface Reconstruction from Unorganized Points*, Computer Graphics, Proceedings of SIGGRAPH '92, ACM, 1992, pp. 71–78.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Mesh optimization*, Computer Graphics, Proceedings of SIGGRAPH '93, ACM, 1993, pp. 19–26.

[19] H. Hoppe, *Smooth view-dependent level-of-detail control and its application to terrain rendering*, Proceedings of Visualization '98, IEEE Computer Society Press, 1998, pp. 35–42 & p. 516.

[20] R. Klein, D. Cohen-Or, T. Huttner, *Incremental view-dependent multiresolution triangulation of terrain*, Journal of Visualization and Computer Animation, Vol. 9, No. 3, Wiley, 1998, pp. 129–143.

[21] O. Kreylos and B. Hamann, *On simulated annealing and the construction of linear spline approximations for scattered data*, in: E. Gröller, H. Löffelmann, and W. Ribarsky, eds., *Data Visualization '99*, Proceedings of EUROGRAPHICS-IEEE TCCG Symposium on Visualization, Springer-Verlag, Vienna, Austria, 1999, pp. 189–198 & p. 328.

[22] S.G. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No. 7, July 1989. pp. 674–693.

[23] M. Margaliot and C. Gotsman, *Piecewise-linear surface approximation from noisy scattered samples*, Proceedings of Visualization '94, IEEE Computer Society Press, 1994, pp. 61–68.

[24] E. Nadler, *Piecewise linear best l_2 approximation on triangles*, C.K. Chui, L.L. Schumaker, and J.D. Ward, eds., *Approximation Theory V*, Academic Press, 1986, pp. 499–502.

[25] G.M. Nielson and T.A. Foley, *A survey of applications of an affine invariant norm*, T. Lyche and L.L. Schumaker, eds., *Mathematical Methods in Computer Aided Geometric Design*, Academic Press, 1989, pp. 445–467.

[26] H. Pottmann, R. Krasauskas, B. Hamann, K.I. Joy, and W. Seibold, *On piecewise linear approximation of quadratic functions*, Journal of Geometry and Graphics, to appear.

[27] E. Quak and L.L. Schumaker, *Least squares fitting by linear splines on data dependent triangulations*, P.J. Laurent, A. Le Méhauté, and L.L. Schumaker, eds., *Curves and Surfaces*, Academic Press, 1991, pp. 387–390.

[28] S. Rippa, *Long and thin triangles can be good for linear interpolation*, SIAM J. Numer. Anal., 1992, Vol. 29, No. 1, pp. 257–270.

[29] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, *Decimation of triangle meshes*, Computer Graphics, Proceedings of SIGGRAPH '92, ACM, 1992, pp. 65–70.

[30] L.L. Schumaker, *Computing optimal triangulations using simulated annealing*, Computer-Aided Geometric Design, Vol. 10, No. 3–4, August 1993, pp. 329–345.

[31] J.C. Xia, A. Varshney, *Dynamic view-dependent simplification for polygonal models*, Proceedings of Visualization '96, IEEE Computer Society Press, 1996, pp. 327–334 & p. 498.