Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002)
D. Ebert, P. Brunet, I. Navazo (Editors)

Best Quadratic Spline Approximation for Hierarchical Visualization

D. F. Wiley[1], H. R. Childs[2], B. Hamann[1], K. I. Joy[1] and N. L. Max[1,3]

[1] Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science,
University of California,Davis, CA 95616-8562, U.S.A.;
e-mail: {wiley, hamann, joy}@cs.ucdavis.edu
[2] B Division, Lawrence Livermore National Laboratory, Mail Stop L-098,
7000 East Avenue, Livermore, CA 94550, U.S.A.;
e-mail: childs3@llnl.gov
[3] Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory,
7000 East Avenue, L-551, Livermore, CA 94550, U.S.A.;
e-mail: max2@llnl.gov

Abstract

We present a method for hierarchical data approximation using quadratic simplicial elements for domain decomposition and field approximation. Higher-order simplicial elements can approximate data better than linear elements. Thus, fewer quadratic elements are required to achieve similar approximation quality. We use quadratic basis functions and compute best quadratic simplicial spline approximations that are C^0-continuous everywhere. We adaptively refine a simplicial approximation by identifying and bisecting simplicial elements with largest errors. It is possible to store multiple approximation levels of increasing quality. We have tested the suitability and efficiency of our hierarchical data approximation scheme by applying it to several data sets.

Categories and Subject Descriptors (according to ACM CCS): I.4.10 [Image Processing and Computer Vision]: Hierarchical; I.4.2 [Image Processing and Computer Vision]: Approximate Methods

1. Introduction

The trend in science and engineering applications has been to produce larger data sets since computers and imaging technology are getting faster and storage space is increasing. Large amounts of data are difficult to visualize and it is impossible to directly visualize on inexpensive computers. Many visualization techniques exist that visualize certain types of large data, however, a general solution does not exist. A hierarchical method provides the foundation for a solution. Linear and quadratic decomposition elements can be used to form an approximation hierarchy representing large data; a user can then visualize this hierarchy on an inexpensive machine.

We only consider quadratic simplicial elements. In the 2D case, we use quadratic triangles whose edges are straight line segments; in the 3D case, we use quadratic tetrahedral elements whose edges are straight line segments and faces are planar triangles. We use a linear transformation to map the so-called *standard simplex* to the corresponding simplicial region in 2D/3D space. Furthermore, we use a quadratic polynomial defined over each simplicial element to locally approximate the dependent variable(s).

Our overall goal is the construction of a hierarchical data approximation over 2D or 3D domains using a best-approximation approach based on quadratic polynomials defined over the simplices defining the domain. Our approach belongs to the class of *refinement* methods. These methods are based on the principle of refining intermediate data approximations by inserting additional points or elements until a certain termination (error) criterion is satisfied. We have developed our method with a focus on massive scientific data visualization, see [13]. To enable interactive frame rates for massive data visualization, it is possible to either use low-resolution best approximations everywhere or to adaptively “insert” high-resolution approximations locally into an otherwise relatively coarse approximation. The overall approximation algorithm is based on these steps:

- **Initial simplicial domain decomposition.** We construct a coarse simplicial decomposition of the domain. (The linear transformations, mapping the standard simplex defined in *parameter space* to simplices in *physical space*, are defined by specifying corresponding point pairs in the two spaces such that one obtains a one-to-one, bijective mapping.)
- **Best approximation.** In the 2D case, each simplicial element has six associated *knots*, one knot per corner and one knot per edge. Six knots in parameter space are associated with six points in physical space, and this defines the needed mapping for a simplex. (Accordingly, the number of knots is ten in the 3D case.) For simplicity, we consider only knots that are uniformly distributed along the edges of the standard simplex. We associate a quadratic polynomial with each simplicial element that approximates the dependent variable(s) over the corresponding region in space. We represent each quadratic basis polynomial in *Bernstein-Bézier form*, see [6]. Assuming that the function to be approximated, typically a scalar- or vector-valued function, is known in analytical form, it is possible to compute the unique best quadratic spline approximation defined as a linear combination of a set of quadratic basis functions. The best approximation, understood in a least squares sense, is the result of solving the *normal equations*, see [5].
- **Adaptive bisection.** We compute a local error value for each simplicial element once a best approximation is computed. We use the L_2 norm to compute simplex-specific error values. The set of simplices is ordered according to these simplex-specific, local error values. To compute a "next-level" best quadratic approximation, we determine a certain percentage of simplices with largest error values and bisect them by splitting them at the midpoint of their longest edge. If a simplex's longest edge is not unique, we choose the edge randomly. Splitting a simplex into two simplices induces additional splits for all those simplices that share the split edge. We update a simplicial domain decomposition by considering all edge bisections and computing a new best approximation. We repeat the process of identifying simplices with largest errors, bisecting these simplices, and computing a new best approximation until we obtain an approximation for which a global approximation error is below a user-specified error threshold, or until a user-specified maximal number of simplices is reached.
- **Hierarchical data representation.** To support level-of-detail visualization we can store multiple best approximations of different resolutions. For each best approximation, we need to store the polynomial coefficients of each simplicial element. We store a fixed number of best approximations such that either the number of simplices increases in a specified fashion or the maximal simplex-specific error decreases in a certain way from one resolution to the next.

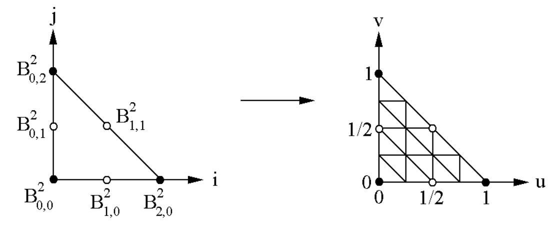


Figure 1: *Correspondence between 2D basis functions $B^2_{i,j}$ and knots (indicated by bullets and circles) in uv-parameter space.*

We discuss these steps in more detail in the following sections.

2. Previous Work

Related work in the areas of hierarchical data representation and visualization is discussed in [12, 17, 20, 27]. Simplification methods (methods that begin with a high resolution of data and then simplify by removing data) are described in [10, 14, 18, 29, 30]. Wavelet methods, in general, work well for rectilinear 2D and 3D grids and are described in [2, 3, 24]. Refinement methods (methods that begin with few data and then refine by adding more data), similar to our method, are described in [15, 16]. Data-dependent triangulation schemes, i.e., schemes concerned with the construction of piecewise linear approximations using near-optimally shaped and placed simplicial elements, are described in [22]. From a more general perspective, our work is also related to *grid generation*, and references for this area are [8, 19, 28]. Finite element methods are discussed in [31].

3. Mapping the Standard Simplex

In the 2D case, the standard simplex in parameter space is the triangle with corners $(0,0)$, $(1,0)$, and $(0,1)$. We associate a 2D quadratic Bernstein-Bézier polynomial $B^2_{i,j}(u,v)$ (abbreviated by $B^2_{i,j}$), defined as

$$\begin{aligned} B^2_{i,j}(u,v) \; &= \\ &\tfrac{2!}{(2-i-j)!\; i!\; j!}(1-u-v)^{2-i-j}\, u^i\, v^j, \\ &i,j \geq 0, \quad i+j \leq 2, \end{aligned} \tag{1}$$

with each corner and midpoint of each edge. The six basis polynomials correspond to the six knots $\mathbf{u}_{i,j} = (u_{i,j}, v_{i,j}) = \left(\frac{i}{2}, \frac{j}{2}\right)$, $i, j \geq 0$, $i+j \leq 2$ in parameter space, see Figure 1.

In the 3D case, the standard simplex is the tetrahedron with corners $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$.

We associate a 3D quadratic Bernstein-Bézier polynomial $B^2_{i,j,k}(u,v,w)$ (abbreviated by $B^2_{i,j,k}$), defined as

$$B^2_{i,j,k}(u,v,w) = \frac{2!}{(2-i-j-k)!\ i!\ j!\ k!}(1-u-v-w)^{2-i-j-k}\ u^i\ v^j\ w^k, \quad i,j,k \geq 0,\ \ i+j+k \leq 2, \tag{2}$$

with each corner and edge. The ten basis polynomials correspond to the ten knots $\mathbf{u}_{i,j,k} = (u_{i,j,k}, v_{i,j,k}, w_{i,j,k}) = \left(\frac{i}{2}, \frac{j}{2}, \frac{k}{2}\right)$, $i,j,k \geq 0$, $i+j+k \leq 2$ in parameter space.

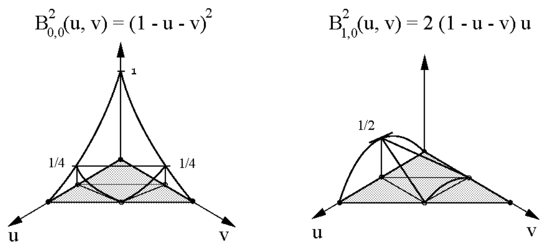
4. Initial Decomposition

The main objective driving the development of our method is the hierarchical representation of very large scientific data, where real-time and adaptive data visualization are crucial. Data sets resulting from computational simulations are typically defined on a grid, and the dependent variables are associated with either the vertices, also called *nodes* in the finite element literature, or the elements defining the grid. Either of these types of data can be approximated. Triangulating the convex hull of the original set of data sites with a coarse triangulation defines an initial decomposition of the domain. The 3D case requires us to construct a tetrahedralization of the convex hull. We decompose quadrilaterals with two triangles (2D case) hexahedra with five tetrahedra (3D case). The elements we use are geometrically linear. However, there are quadratic polynomials defined "over" them that approximate the dependent field variable(s).

5. Best Approximation

We assume that the field to be approximated over a domain is known analytically. Should this not be the case, e.g., in the case of *scattered data* (when one is given a set of randomly distributed points with associated function values without connectivity information), it is possible to construct an analytical representation by performing a prior data interpolation or approximation step, see [7, 23]. In the case that a data set is defined on a grid, the required analytical definition is given by a piecewise linear function for a simplicial (triangular or tetrahedral) grid and a piecewise bilinear/trilinear function in the case of quadrilateral/hexahedral grid cells. We denote the analytical function to be approximated over the domain by $F(\mathbf{x})$ (abbreviated by F). Based on an initial simplicial domain decomposition, we compute the corresponding best piecewise quadratic approximation of $F(\mathbf{x})$ by solving the normal equations, see [5]. The normal equations determine the set of coefficients for the desired quadratic spline representation—a best approximation in the least squares sense.

Corner vertices of simplicial elements may be shared by any number of simplices, and we denote the basis function associated with a corner vertex $\mathbf{v}_i$ by $f_i(\mathbf{x})$. An edge of a simplicial element may be shared by no more than two simplices



$B^2_{0,0}(u, v) = (1 - u - v)^2$
1
1/4
1/4
u
v
$B^2_{1,0}(u, v) = 2\ (1 - u - v)\ u$
1/2
u
v

Figure 2: *Types of basis functions: basis function associated with corner (left) and edge (right).*

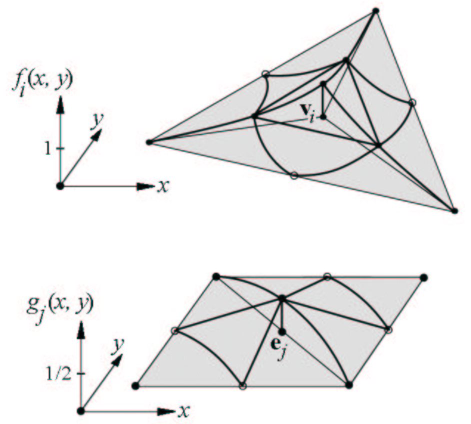


Figure 3: *Basis functions associated with the platelet of vertex $\mathbf{v}_i$ and the edge neighbors of edge $\mathbf{e}_j$.*

in the 2D case and by an arbitrary number of simplices in the 3D case. We denote a basis function associated with the midpoint of a simplex edge e_j by $g_j(\mathbf{x})$. We refer to the set of simplices sharing a common corner vertex as the *platelet* of this corner, and we call the set of simplices sharing a common edge *edge neighbors*. Thus, a set of platelet simplices defines the region in space over which a basis function associated with the corresponding corner vertex is non-zero. Edge neighbors, associated with a particular edge, define the region in space over which a basis function associated with this edge is non-zero. Figure 2 shows the two types of basis functions for the bivariate case. Figure 3 shows the basis functions associated with the platelet of a vertex and the edge neighbors of an edge.

We denote a best approximation as $a(\mathbf{x})$, and we write it as a linear combination of the basis functions associated with all distinct simplex corners ("corner basis functions" f_i) and simplex edges ("edge basis functions" g_i). Assuming that there are m distinct corners and n distinct edges, we can write a best approximation as

$$a(\mathbf{x}) = \sum_{i=1}^{m} c_i\ f_i(\mathbf{x}) + \sum_{j=1}^{n} d_j\ g_j(\mathbf{x}). \tag{3}$$

We must solve the normal equations to obtain the unknown coefficients c_i and d_j. In matrix form, the normal equations are

$$\begin{pmatrix} \langle f_1,f_1\rangle & \dots & \langle f_1,f_m\rangle & \langle f_1,g_1\rangle & \dots & \langle f_1,g_n\rangle \\ \vdots & & & & & \vdots \\ \langle f_m,f_1\rangle & \dots & \langle f_m,f_m\rangle & \langle f_m,g_1\rangle & \dots & \langle f_m,g_n\rangle \\ \langle g_1,f_1\rangle & \dots & \langle g_1,f_m\rangle & \langle g_1,g_1\rangle & \dots & \langle g_1,g_n\rangle \\ \vdots & & & & & \vdots \\ \langle g_n,f_1\rangle & \dots & \langle g_n,f_m\rangle & \langle g_n,g_1\rangle & \dots & \langle g_n,g_n\rangle \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \\ d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} \langle F,f_1\rangle \\ \vdots \\ \langle F,f_m\rangle \\ \langle F,g_1\rangle \\ \vdots \\ \langle F,g_n\rangle \end{pmatrix}, \tag{4}$$

where $\langle G,H\rangle$ denotes the inner product of the functions $G(\mathbf{x})$ and $H(\mathbf{x})$, i.e.,

$$\langle G,H\rangle = \int_{\text{Common Domain of } G \text{ and } H} G(\mathbf{x})\, H(\mathbf{x})\, d\mathbf{x}. \tag{5}$$

In our construction, we must compute inner products over the simplices. Since all simplicial elements in physical space are defined by linear mappings of the standard simplex, we can simplify integration by making use of the *change-of-variables theorem*, see [21], which relates integration in physical space to integration in parameter space. In the 2D case, integrals are computed according to the formula

$$\int_{\text{Physical Simplex}} G(x,y)\, dx\, dy = \int_{\text{Standard Simplex}} G\Big(x(u,v), y(u,v)\Big)\, J(u,v)\, du\, dv, \tag{6}$$

where $J(u,v)$ denotes the *Jacobian* associated with the mapping of the standard simplex to the corresponding simplex in physical space. The Jacobian is the determinant

$$J(u,v) = \begin{vmatrix} \frac{\partial}{\partial u}x(u,v) & \frac{\partial}{\partial v}x(u,v) \\ \frac{\partial}{\partial u}y(u,v) & \frac{\partial}{\partial v}y(u,v) \end{vmatrix}. \tag{7}$$

Thus, to effectively compute integrals of functions over triangles we only need to consider the linear transformation

$$\begin{bmatrix} x(u,v) \\ y(u,v) \end{bmatrix} = \begin{bmatrix} x_{1,0}-x_{0,0} & x_{0,1}-x_{0,0} \\ y_{1,0}-y_{0,0} & y_{0,1}-y_{0,0} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}. \tag{8}$$

This transformation maps the *standard triangle* with vertices $\mathbf{u}_0 = [0,0]^T$, $\mathbf{u}_1 = [1,0]^T$, and $\mathbf{u}_2 = [0,1]^T$ in the uv-plane to the arbitrary simplex S with corner vertices $\mathbf{v}_0 = [x_{0,0}, y_{0,0}]^T$, $\mathbf{v}_1 = [x_{1,0}, y_{1,0}]^T$, and $\mathbf{v}_2 = [x_{0,1}, y_{0,1}]^T$ in the xy-plane. (Both triangles must be oriented counterclockwise). For this linear mapping, the change-of-variables theorem yields

$$\int_S G(x,y)dxdy = J\int_{v=0}^{1}\int_{u=0}^{1-v} G\left(x(u,v),y(u,v)\right)dudv, \tag{9}$$

where the Jacobian J is given by

$$J = \det\begin{bmatrix} x_{1,0}-x_{0,0} & x_{0,1}-x_{0,0} \\ y_{1,0}-y_{0,0} & y_{0,1}-y_{0,0} \end{bmatrix}. \tag{10}$$

The 3D case is a straightforward extension; here, the Jacobian is given by

$$J = \det\begin{bmatrix} x_{1,0,0}-x_{0,0,0} & x_{0,1,0}-x_{0,0,0} & x_{0,0,1}-x_{0,0,0} \\ y_{1,0,0}-y_{0,0,0} & y_{0,1,0}-y_{0,0,0} & y_{0,0,1}-y_{0,0,0} \\ z_{1,0,0}-z_{0,0,0} & z_{0,1,0}-z_{0,0,0} & z_{0,0,1}-z_{0,0,0} \end{bmatrix}. \tag{11}$$

The matrices involved in the best-approximation step are sparse because all basis functions have local support. Several methods exist for bandwidth reduction, efficient factorization, and inversion of such sparse matrices, see [4, 9, 11, 25, 26]. We use an efficient sparse matrix representation and system solver to compute the coefficients in linear time.

The computation of the inner products appearing in the normal equations requires multi-dimensional integration over simplicial elements. While the change-of-variables theorem reduces this integration to integration over the standard simplex, we still need to perform relatively expensive numerical integration for the calculation of the inner products appearing on the right-hand side of the normal equations, i.e., the integrals of the types $\langle F, f_i\rangle$ and $\langle F, g_j\rangle$. Since $F(\mathbf{x})$ can, in general, be any analytically defined function, numerical integration can potentially become expensive. We use *Romberg integration* for the computation of these right-hand-side inner products, see [1, 16].

Once we have computed a best approximation for a particular simplicial domain decomposition, we analyze the local approximation quality to identify simplices that should be refined (bisected) to further improve approximation quality. In the following section, we discuss the principle we use for adaptive bisection.

6. Adaptive Bisection

For each simplicial element S_i in a particular domain decomposition, we compute a local approximation error e_i. We define this error as

$$e_i = \left(\int_{S_i}\Big(F(\mathbf{x}) - a(\mathbf{x})\Big)^2 d\mathbf{x}\right)^{\frac{1}{2}}. \tag{12}$$

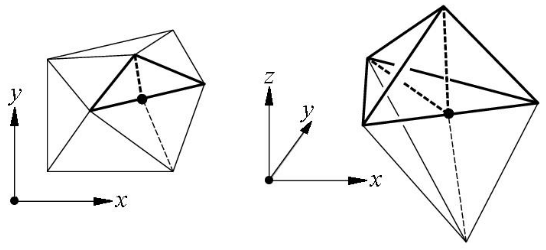


Figure 4: *Bisection of simplices in bivariate and trivariate cases. Darker simplex is the one selected for bisection.*

Selecting and bisecting simplices of maximal error are the steps used to refine the mesh. In general, we choose a certain percentage of the simplices to be refined.

We refine a simplicial element by bisecting at the midpoint of its longest edge. All simplices sharing the split edge are bisected to avoid "hanging nodes" and, therefore, to preserve a *conforming mesh*. The bisection step is shown in Figure 4. Bisection steps lead to new simplicial domain decompositions, and we must compute new best quadratic spline approximations for each one.

We continue to bisect a certain percentage of simplices in the intermediate approximations until either the number of simplices in a decomposition exceeds some user-specified maximal number or until an approximation is obtained whose global error is less than a user-specified tolerance. (We defined the global error of an approximation as the sum of all local simplex errors).

The final result of our method is a set of independent best quadratic spline approximations that can be used for the purposes of interactive and/or adaptive level-of-detail visualization.

7. Results

We have tested our method for several test data. In general, we compare a piecewise quadratic approximation to a piecewise linear approximation. We visualize quadratic simplices by tessellating them using many linear elements.

Figure 5 shows a quadratic and linear spline approximation for comparison. The quadratic approximation can, in theory, approximate this function exactly. (Numerical floating-point error is introduced in practice. For the shown example, this numerical floating-point error is on the order of 10^{-14}). The linear approximation must use a relatively large number of elements to represent this quadratic function with small error. The global error for the quadratic spline is $3.6x10^{-14}$ and the global error for the linear approximation error is $1.6x10^{-6}$. The linear approximation was computed using the method described in [16].

A comparison of a quadratic and a linear spline approximation is shown in Figure 6. The original image consists of $1536x1024$ pixels. The quadratic spline approximation—consisting of 2989 quadratic simplices—required 158 seconds of computation time while the linear approximation—consisting of 11482 linear simplices—required 536 seconds.

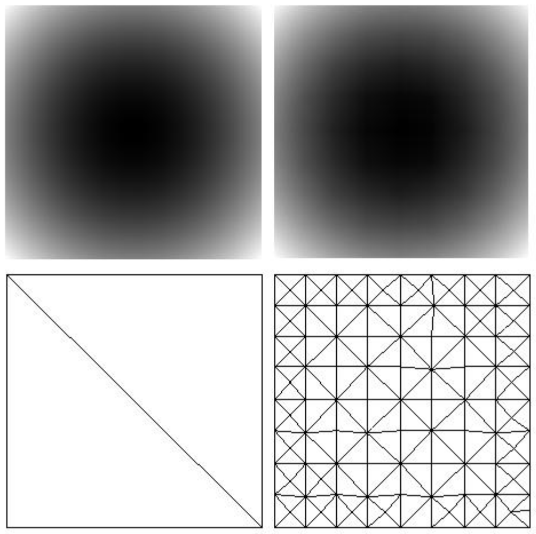


Figure 5: *Comparison between quadratic spline approximation (left) and linear spline approximation (right). Under each approximation, we show the corresponding domain decomposition. The quadratic approximation uses 9 knots and 2 simplices. The linear approximation uses 111 knots and 187 simplices. The function being approximated is* $F(x,y) = x^2 + y^2, x, y \in \left[-\frac{1}{2}, \frac{1}{2}\right]$.

A sample hierarchy of 2D quadratic spline approximations is shown in Figure 7. The original image consists of $121x121$ pixels. Global errors for the six approximations are 329.11, 106.22, 45.53, 12.85, 3.08, and 0.40. Computations times ranged from two to 200 seconds for the six approximations.

A sample hierarchy of 2D quadratic spline approximations is shown in Figure 8. The original image consists of $211x144$ pixels. Global errors for the four approximations are 37.05, 9.70, 1.86, and 0.45. Computations times ranged from six to 200 seconds for the six approximations.

A comparison of a quadratic and a linear spline approximation of a 3D skull data set is shown in Figure 9. The original data set consists of 278528 data sites. We visualize the quadratic spline approximation by tessellating each quadratic simplex with 512 linear elements and then extracting an isosurface from the linear elements. The same isosurface for the linear spline approximation was extracted directly from the linear simplices. The quadratic spline approximation has a global error of $2.15x10^{-6}$, and the linear

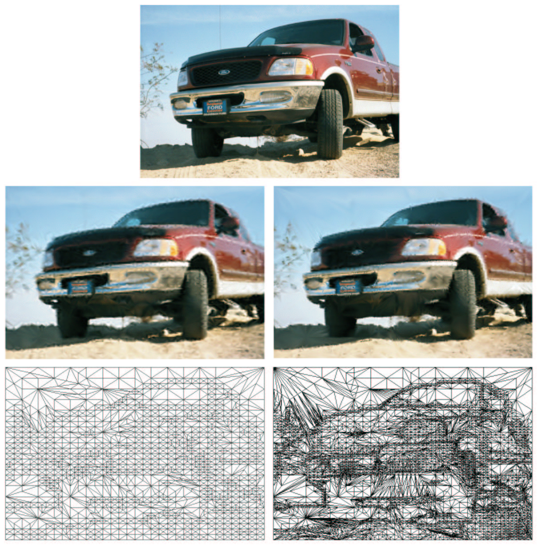


Figure 6: *Comparison between quadratic approximation (left) and linear approximation (right). Original image is shown at the top. The quadratic approximation uses 6076 knots and 2989 simplices. The linear approximation uses 5816 knots and 11482 simplices.*

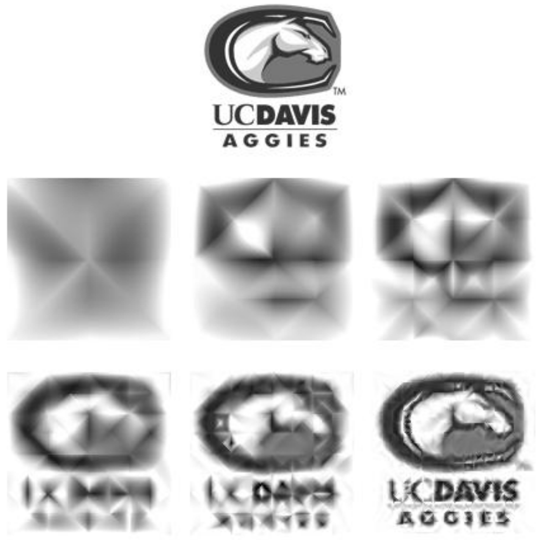


Figure 7: *Hierarchical approximations of digital image data set. Original image is shown at the top. Six approximations are shown, 8, 20, 38, 90, 225, and 633 simplices, respectively.*

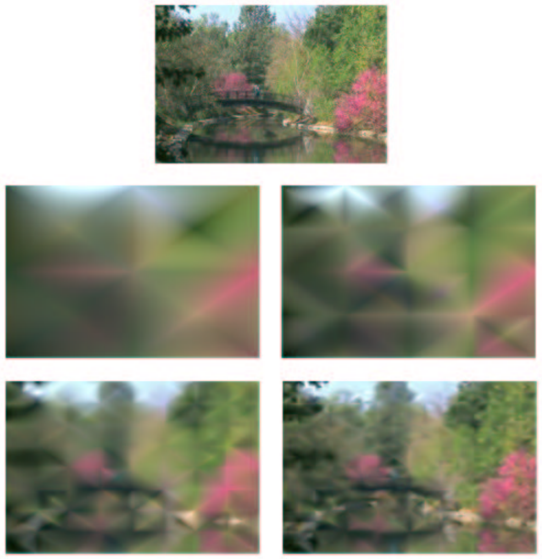


Figure 8: *Hierarchical approximations of digital image data set. Original image is shown at the top. Four approximations are shown, 16, 48, 191, and 790 simplices, respectively.*

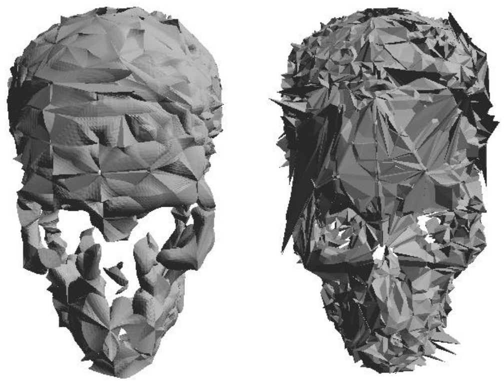


Figure 9: *Comparison between quadratic approximation (left) and linear approximation (right). The quadratic approximation uses 7487 knots and 5348 simplices. The linear approximation uses 14667 knots and 78530 simplices.*

spline approximation has a global error of $1.65x10^{-2}$. The quadratic spline approximation required about 20 hours of computation time while the linear approximation required less than three.

A sample hierarchy of 3D quadratic spline approximations for a 3D skull data set is shown in Figure 10. Global errors for the six approximations are $1.0x10^{-3}$, $4.7x10^{-4}$, $3.9x10^{-5}$, and $2.1x10^{-6}$.

All of the approximations were computed on a 1.8GHz

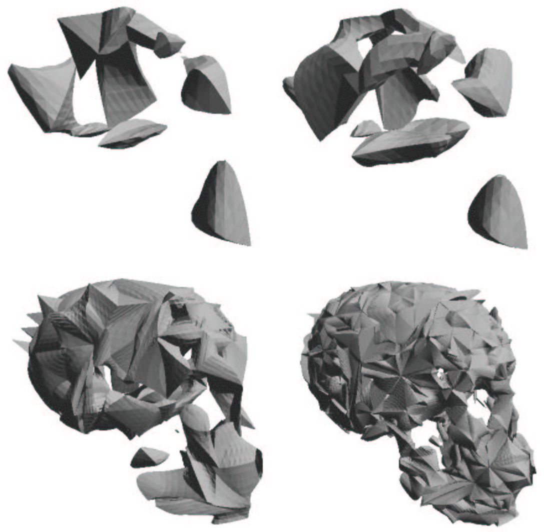


Figure 10: *Hierarchical approximations of skull data set. Four approximations are shown, 62, 125, 741, and 5348 simplices, respectively.*

Pentium IV graphics workstation with 512MB of main memory.

Linear 2D and 3D approximations were rendered at interactive frame rates. Quadratic 2D approximations required just a few seconds to render per frame. Tessellation of 3D quadratic approximations required several seconds for the highest resolutions. Once tessellated, computing and rendering a contour was at interactive frame rates.

8. Conclusions

Quadratic simplicial elements can be used to more compactly approximate data than linear simplicial elements. In general, the use of higher-order simplices should be considered as they can produce better-quality approximations, using a smaller number of simplices.

Additional research incorporating geometrically curved simplices can further improve the quality of approximations by allowing the simplices to decompose more complicated-shaped domains. A decomposition of a domain having curved boundaries would require fewer curved simplices to represent the domain well.

The generated meshes are C^0 continuous, it is also possible to produce C^1 approximations. We plan to investigate this enhancement in the future.

Higher-order simplices are growing in importance in visualization as researchers are also using them more frequently for domain decomposition in numerical simulations. Thus, visualization of these simplices is also important because of their increasing popularity. Direct higher-order visualization techniques such as contouring and volume visualization techniques, must be developed to take advantage of higher-order elements. We are currently working on such techniques.

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 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 National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2; the Army Research Office under contract ARO 36598-MA-RIP; and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159. We also acknowledge the support of ALSTOM Schilling Robotics and SGI. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis. Furthermore, we acknowledge the support received by the members of B Division, Lawrence Livermore National Laboratory.

References

1. Boehm, W. and Prautzsch, H. (1993), *Numerical Methods*, A K Peters, Ltd., Wellesley, MA. 4
2. Bonneau, G. P., Hahmann, S. and Nielson, G. M. (1996), BLaC-wavelets: A multiresolution analysis with non-nested spaces, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 43–48. 2
3. Bonneau, G. P. (1999), Optimal triangular Haar bases for spherical data, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 279–284. 2
4. Cuthill, E. and McKee, J. (1969), Reducing the bandwidth of sparse symmetric matrices, in: *Proceedings of the ACM National Conference*, Association for Computing Machinery, New York, NY, pp. 157–172. 4
5. Davis, P. J. (1975), *Interpolation and Approximation*, Dover Publications, Inc., New York, NY. 2, 3
6. Farin, G. (2002), *Curves and Surfaces for Computer Aided Geometric Design*, fifth edition, Academic Press, San Diego, CA. 2

7. Franke, R. (1982), Scattered data interpolation: Tests of some methods, Math. Comp. 38, pp. 181–200. 3

8. George, P. L. (1991), *Automatic Mesh Generation*, Wiley & Sons, New York, NY. 2

9. Gibbs, N. E., Poole, W. G. and Stockmeyer P. K. (1976), An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal. 13(2), pp. 236–250. 4

10. Gieng, T. S., Hamann, B., Joy, K. I., Schussman, G. L. and Trotts, I. J. (1998), Constructing hierarchies for triangle meshes, IEEE Transactions on Visualization and Computer Graphics 4(2), pp. 145–161. 2

11. Golub, G. H. and Van Loan, C. F. (1989), *Matrix Computations*, second edition, Johns Hopkins University Press, Baltimore, MD. 4

12. Gross, M. H., Gatti, R. and Staadt, O. (1995), Fast multiresolution surface meshing, in: Nielson, G. M. and Silver, D. eds., *Visualization '95*, IEEE Computer Society Press, Los Alamitos, CA, pp. 135–142. 2

13. Hagen, H., Müller, H. and Nielson, G. M., eds. (1993), *Focus on Scientific Visualization*, Springer-Verlag, New York, NY. 1

14. Hamann, B. (1994), A data reduction scheme for triangulated surfaces, Computer Aided Geometric Design 11(2), pp. 197–214. 2

15. Hamann, B. and Jordan, B. W. (1998), Triangulations from repeated bisection, in: Dæhlen, M., Lyche, T. and Schumaker, L. L., eds., *Mathematical Methods for Curves and Surfaces II*, Vanderbilt University Press, Nashville, TN, pp. 229–236. 2

16. Hamann, B., Jordan, B. W. and Wiley, D. F. (1999), On a construction of a hierarchy of best linear spline approximations using repeated bisection, IEEE Transactions on Visualization and Computer Graphics 5(1/2), pp. 30–46, p. 190 (errata). 2, 4, 5

17. Heckel, B., Weber, G. H., Hamann, B. and Joy, K. I. (1999), Construction of vector field hierarchies, in: Gross, M., Ebert, D. S. and Hamann, B., eds., *Visualization '99*, IEEE Computer Society Press, Los Alamitos, California, pp. 19–25. 2

18. Hoppe, H. (1997), View-dependent refinement of progressive meshes, in: Whitted, T., ed., *Proceedings of SIGGRAPH 1997*, ACM Press, New York, NY, pp. 189–198. 2

19. Knupp, P. M. and Steinberg, S. (1993), *Fundamentals of Grid Generation*, CRC Press, Boca Raton, FL. 2

20. Kreylos, O. and Hamann, B. (1999), On simulated annealing and the construction of linear spline approximations for scattered data, in: Gröller, E., Löffelmann, H. and Ribarsky, W., eds., *Data Visualization '99* (Proc. EUROGRAPHICS-IEEE TCCG Symposium on Visualization), Springer-Verlag, Vienna, Austria, pp. 189–198. 2

21. Marsden, J. E. and Tromba, A. J. (1988), *Vector Calculus*, third edition, W. H. Freeman and Company, New York, NY. 4

22. Nadler, E. (1986), Piecewise linear best L_2 approximation on triangulations, in: Ward, J. D., ed., *Approximation Theory V*, Academic Press, Inc., San Diego, CA, pp. 499–502. 2

23. Nielson, G. M. (1993), Scattered data modeling, IEEE Computer Graphics and Applications 13(1), pp. 60–70. 3

24. Nielson, G. M., Jung, I.-H. and Sung, J. (1997a), Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 143–149. 2

25. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992), *Numerical Recipes in C*, second edition, Cambridge University Press, New York, NY. 4

26. Rosen, R. (1968), Matrix bandwidth minimization, in: *Proceedings of the ACM National Conference*, ACM publication no. P-68, Brandon Systems Press, Princeton, NJ, pp. 585–595. 4

27. Staadt, O. G., Gross, M. H. and Weber, R. (1997), Multiresolution compression and reconstruction, in: Yagel, R. and Hagen, H., eds., *Visualization '97*, IEEE Computer Society Press, Los Alamitos, CA, pp. 337–346. 2

28. Thompson, J. F., Soni, B. K. and Weatherill, N. P., eds. (1999), *Handbook of Grid Generation*, CRC Press, Boca Raton, FL. 2

29. Trotts, I. J., Hamann, B., Joy, K. I. and Wiley, D. F. (1998), Simplification of tetrahedral meshes, in: Ebert, D. S., Hagen, H. and Rushmeier, H. E., eds., *Visualization '98*, IEEE Computer Society Press, Los Alamitos, California, pp. 287–295. 2

30. Xia, J. C. and Varshney, A. (1996), Dynamic view-dependent simplification for polygonal meshes, in: Yagel, R. and Nielson, G. M., eds., *Visualization '96*, IEEE Computer Society Press, Los Alamitos, CA, pp. 327–334. 2

31. Zienkiewicz, O. C. (1977), *The Finite-Element Method in Engineering Science*, McGraw-Hill, London, United Kingdom. 2