Adaptive Contouring with Quadratic Tetrahedra

Benjamin F. Gregorski[1,2] David F. Wiley[1] Henry R. Childs[3]
Bernd Hamann[1] and Kenneth I. Joy[1]
[1] Center for Image Processing and Integrated Computing(CIPIC)
Department of Computer Science, University of California Davis
[2] Center for Applied Scientific Computing(CASC)
Lawrence Livermore National Laboratory
[3] B Division Lawrence Livermore National Laboratory

Abstract

We present an algorithm for adaptively extracting and rendering isosurfaces of scalar-valued volume datasets represented by quadratic tetrahedra. Hierarchical tetrahedral meshes created by longest-edge bisection are used to construct a multiresolution C^0-continuous representation using quadratic basis functions. A new algorithm allows us to contour higher-order volume elements efficiently.

1 Introduction

Isosurface extraction is a fundamental algorithm for visualizing volume datasets. Most research concerning isosurface extraction has focused on improving the performance and quality of the extracted isosurface. Hierarchical data structures, such as those presented in [22, 2, 10], can quickly determine which regions of the dataset contain the isosurface, minimizing the number of cells examined. These algorithms extract the isosurface from the highest resolution mesh. Adaptive refinement algorithms [4, 7, 5] progressively extract isosurfaces from lower resolution volumes, and control the quality of the isosurface using user specified parameters.

An isosurface is typically represented as a piecewise linear surface. For datasets that contain smooth, steep ramps, a large number of linear elements is often needed to accurately reconstruct the dataset unless extra information is known about the data. Recent research has addressed these problems with linear elements by using *higher-order* methods that incorporate additional information into the isosurface extraction algorithm. In [9], an extended marching cubes algorithm, based on gradient information, is used to extract contours from distance volumes that contain sharp features. Cells that contain features are contoured by inserting new vertices that minimize an error function. Higher-order distance fields are also described in [12]. This approach constructs a distance field representation where each voxel has a complete description of all surface regions that contribute to the local distance field. Using this representation, sharp features and discontinuities are accurately represented as their exact locations are recorded. Ju et al. [11] describe a dual contouring scheme for adaptively refined volumes represented with Hermite data that does not have to test for sharp features. Their algorithm uses a new representation for quadric error functions to quickly and accurately position vertices within cells according to gradient information. Wiley et al. [19, 20] use quadratic elements for hierarchical approximation and visualization of image and volume data. They show that quadratic elements can effectively replace a large number of linear elements.

Higher-order elements, such as quadratic tetrahedra and quadratic hexahedra, are used in finite element solutions to reduce the number of elements and improve the quality of numerical solutions [18]. Since few algorithms directly visualize higher-order elements, they are usually tessellated by several linear elements. Conventional visualization methods, such as contouring, ray casting, and slicing, are applied to these linear elements. Using linear elements increases the number of primitives, i.e. triangles or voxels, that need to be processed. Methods for visualizing higher-order elements directly are desirable.

We use a tetrahedral mesh, constructed by longest-edge bisection as presented in [5], to create a multiresolution data representation. The linear tetrahedral elements used in previous methods are replaced with quadratic tetrahedra. The resulting mesh defines a C^0-continuous, piecewise quadratic approximation of the original dataset. This quadratic representation is computed in a preprocessing step by approximating the data values along each edge of a tetrahedron with a quadratic function that interpolates the endpoint values. A quadratic tetrahedron is constructed

from the curves along its six edges. At runtime, the hierarchical approximation is traversed to approximate the original dataset to within a user defined error tolerance. The isosurface is extracted directly from the quadratic tetrahedra.

The remainder of our paper is structured as follows: Section 2 reviews related work. Section 3 describes what quadratic tetrahedra are, and Section 4 describes how they are used to build a multiresolution representation of a volume dataset. Sections 5 describes how a quadratic tet is contoured. Our results are shown in Section 6.

2 Previous Work

Tetrahedral meshes constructed by longest-edge bisection have been used in many visualization applications due to their simple, elegant, and crack-preventing adaptive refinement properties. In [5], fine-to-coarse and coarse-to-fine mesh refinement is used to adaptively extract isosurfaces from volume datasets. Gerstner and Pajarola [7] present an algorithm for preserving the topology of an extracted isosurface using a coarse-to-fine refinement scheme assuming linear interpolation within a tetrahedron. Their algorithm can be used to extract topology-preserving isosurfaces or to perform controlled topology simplification. In [6], Gerstner shows how to render multiple transparent isosurfaces using these tetrahedral meshes, and in [8], Gerstner and Rumpf parallelize the isosurface extraction by assigning portions of the binary tree created by the tetrahedral refinement to different processors. Roxborough and Nielson [16] describe a method for adaptively modeling 3D ultrasound data. They create a model of the volume that conforms to the local complexity of the underlying data. A least-squares fitting algorithm is used to construct a best piecewise linear approximation of the data.

Contouring quadratic functions defined over triangular domains is discussed in [1, 17, 14]. Worsey and Farin [14] use Bernstein-Bézier polynomials which provide a higher degree of numerical stability compared to the monomial basis used by Marlow and Powell [17]. Bloomquist [1] provides a foundation for finding contours in quadratic elements.

In [19] and [20], quadratic functions are used for hierarchical approximation over triangular and tetrahedral domains. The approximation scheme uses the normal-equations approach described in [3] and computes the best least-squares approximation. A dataset is approximated with an initial set of quadratic triangles or tetrahedra. The initial mesh is repeatedly subdivided in regions of high error to improve the approximation. The quadratic elements are visualized by subdividing them into linear elements.

Our technique for constructing a quadratic approximation differs from [19] and [20] as we use univariate approximations along a tetrahedron's edges to define the coefficients for an approximating tetrahedron. We extract an isosurface directly from a quadratic tetrahedron by creating a set of rational-quadratic patches that approximates the isosurface. The technique we use for isosurfacing quadratic tetrahedra is described in [21].

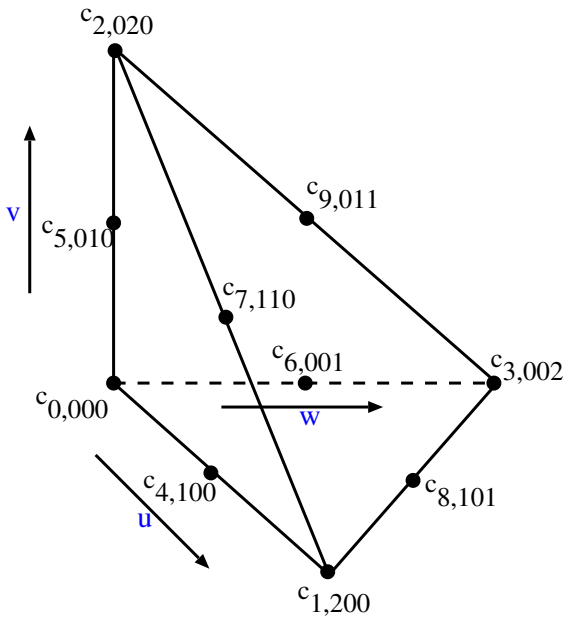

Figure 1. Indexing of vertices and parameter space configuration for the ten control points of a quadratic tetrahedron.

3 Quadratic Tetrahedra

A linear tetrahedron $T_L(u, v, w)$ having four coefficients f_i at its vertices $\mathbf{V}_i$ is defined as

$$\begin{aligned} T_L(u,v,w) &= f_0 u + f_1 v + f_2 w + \\ &\quad f_3(1-u-v-w). \end{aligned} \tag{1}$$

The quadratic tetrahedron $T_Q(u, v, w)$ (called T_Q) that we use as our decomposition element has linearly defined edges such that its domain is completely described by four vertices (the same as a conventional linear tetrahedron). The function over T_Q is defined by a quadratic polynomial. We call this element a *linear-edge quadratic tetrahedron* or *quadratic tetrahedron*. The quadratic polynomial is defined, in Bernstein-Bézier form, by ten coefficients c_m, $0 \leq$

$m \leq 9$, as

$$T_Q(u,v,w) = \sum_{k=0}^{1} \sum_{j=0}^{2-k} \sum_{i=0}^{2-k-j} c_{ijk} B^2_{ijk}(u,v,w) \quad (2)$$

The Bernstein-Bézier basis functions $B^2_{ijk}(u,v,w)$ are

$$B^2_{ijk} = \frac{2!}{(2-i-j-k)!i!j!k!} (1-u-v-w)^{2-i-j-k} u^i v^j w^k \quad (3)$$

The indexing of the coefficients is shown in Figure 1.

4 Constructing a Quadratic Representation

A quadratic tetrahedron T_Q is constructed from a linear tetrahedron T_L with corner vertices V_0, V_1, V_2, and V_3, by fitting quadratic functions along the six edges of T_L. Since a quadratic function requires three coefficients, there is an additional value associated with each edge.

4.1 Fitting Quadratic Curves

Given a set of function values $f_0, f_1 \ldots f_n$ at positions $x_0, x_1 \ldots x_n$, we create a quadratic function that passes through the end points and approximates the remaining data values.

The quadratic function $C(t)$ we use to approximate the function values along an edge is defined as

$$C(t) = \sum_{i=0}^{2} c_i B^2_i(t) \quad (4)$$

The quadratic Bernstein polynomial $B^2_i(t)$ is defined as

$$B^2_i(t) = \frac{2!}{(2-i)!i!} (1-u)^{2-i} u^i \quad (5)$$

First we parameterize the data by assigning parameter values $t_0, t_1 \ldots t_n$ in the interval $[0,1]$ to the positions $x_0, x_1 \ldots x_n$. Parameter values are defined with a chord-length parameterization as

$$t_i = \frac{x_i - x_0}{x_n - x_0}$$

Next, we solve a least-squares approximation problem to determine the coefficients c_i of $C(t)$. The resulting overdetermined system of linear equations is

$$\begin{bmatrix} (1-t_0)^2 & 2(1-t_0)t_0 & {t_0}^2 \\ (1-t_1)^2 & 2(1-t_1)t_1 & {t_1}^2 \\ \vdots & \vdots & \vdots \\ (1-t_n)^2 & 2(1-t_n)t_n & {t_n}^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}. \quad (6)$$

Constraining $C(t)$, so that it interpolates the endpoint values, i.e. $C(0) = f_0$ and $C(1) = f_n$, leads to the system

$$\begin{bmatrix} 2(1-t_1)t_1 \\ 2(1-t_2)t_2 \\ \vdots \\ 2(1-t_{n-1})t_{n-1} \end{bmatrix} [c_1] = \begin{bmatrix} f_1 - f_0(1-t_1)^2 - f_n {t_1}^2 \\ f_2 - f_0(1-t_2)^2 - f_n {t_2}^2 \\ \vdots \\ f_{n-1} - f_0(1-t_{n-1})^2 - f_n {t_{n-1}}^2 \end{bmatrix} \quad (7)$$

for the one degree of freedom c_1.

4.2 Approximating a Dataset

A quadratic approximation of a dataset is created by approximating the data values along each edge in the tetrahedral mesh with a quadratic function as described in Section 4.1. Each linear tetrahedron becomes a quadratic tetrahedron. The resulting approximation is C^1-continuous within a tetrahedron and C^0-continuous on shared faces and edges. The approximation error e_a for a tetrahedron T is the maximum difference between the quadratic approximation over T and all original data values associated with points inside and on $T's$ boundary.

In tetrahedral meshes created by longest-edge bisection, each edge E in the mesh, except for the edges at the finest level of the mesh, is the split edge of a *diamond* D, see [5], and is associated with a split vertex SV. The computed coefficient c_1 for the edge E is stored with the *split vertex* SV. The edges used for computing the quadratic representation can be enumerated by recursively traversing the tetrahedral mesh and examining the refinement edges. This process is illustrated for the 2D case in Figure 2. Since quadratic tetrahedra have three coefficients along each edge, the leaf level of a mesh with quadratic tetrahedra is one level higher in the mesh than the leaf level for linear tetrahedra, see Figure 3.

In summary, we construct a quadratic approximation of a volume data set as follows:

1. For each edge of the mesh hierarchy, approximate the data values along the edge with a quadratic function that passes through the endpoints.

2. For each tetrahedron in the hierarchy, construct a quadratic tetrahedron from the six quadratic functions along its edges.

3. Compute the approximation error e_a for each tetrahedron.

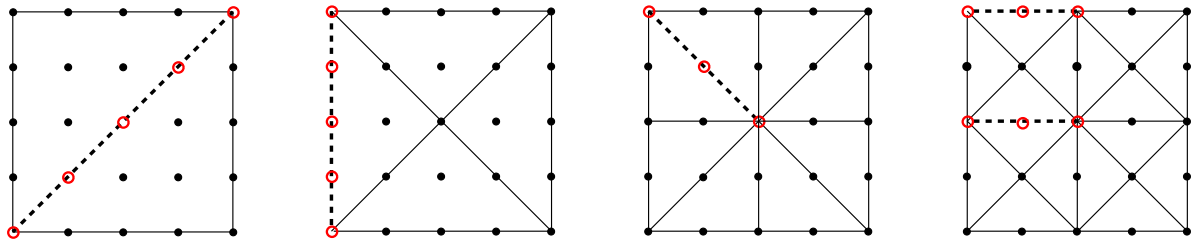

Figure 2. Enumeration of edges for constructing quadratic approximation using longest-edge bisection. Circles indicate original function values used to compute approximating quadratic functions along each edge.

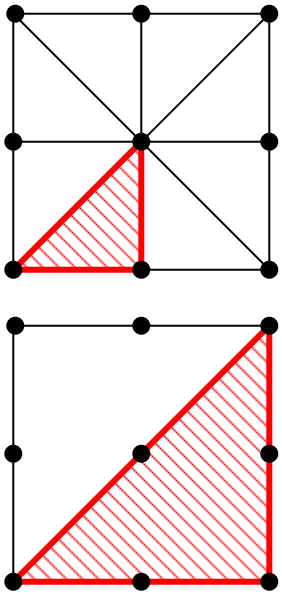

Figure 3. Top: leaf tetrahedra for a mesh with linear tetrahedra. Bottom: leaf tetrahedra for a mesh with quadratic tetrahedra.

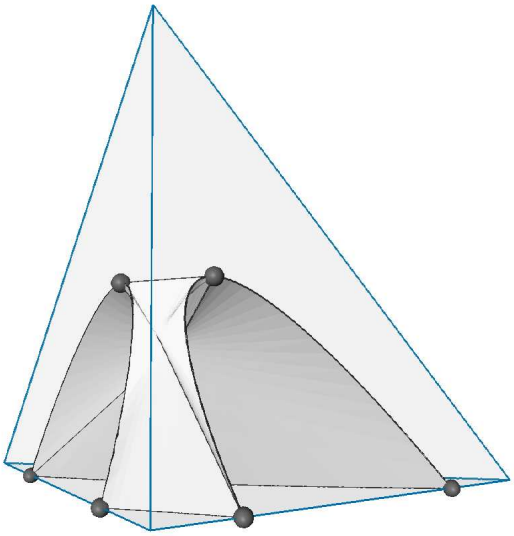

Figure 4. Isosurface bounded by six face-intersection curves. Dark dots indicate endpoints of the curves.

5 Contouring Quadratic Tetrahedra

We use the method described in [21] to extract and represent isosurfaces of quadratic tetrahedra. We summarize the main aspects of the method here. First, the intersection of the isosurface is computed with each face of the quadratic tetrahedron forming *face-intersection curves*. Next, the face-intersection curves are connected end-to-end to form groups of curves that bound various portions of the isosurface inside the tetrahedron, see Figure 4. Finally, the face-intersection groups are "triangulated" with rational-quadratic patches to represent the various portions of the isosurface inside the quadratic tetrahedron.

Since intersections are conic sections [14], the intersections between the isosurface and the faces produce rational-quadratic curves. We define a rational-quadratic curve $Q(t)$ with control points p_i and weights w_i, $0 \leq i \leq 2$, as

$$Q(t) = \frac{\sum_{i=0}^{2} w_i \mathbf{p}_i B_i^2(t)}{\sum_{i=0}^{2} w_i B_i^2(t)} \tag{8}$$

By connecting the endpoints of the N face-intersection curves $Q_j(t),\ 0 \leq j \leq N-1$, we construct M rational-quadratic patches $Q_k(u,v),\ 0 \leq k \leq M-1$, to represent the surface. We define a rational-quadratic patch $Q(u,v)$ with six control points p_{ij} and six weights w_{ij} as

$$Q(u,v) = \frac{\sum_{j=0}^{2} \sum_{i=0}^{2-j} w_{ij} p_{ij} B_{ij}^2(u,v)}{\sum_{j=0}^{2} \sum_{i=0}^{2-j} w_{ij} B_{ij}^2(u,v)} \tag{9}$$

Dataset	Size	Error	Tets	Patches	Tris
Buckyball	256^3	2.0	8560	4609	6060
Buckyball	256^3	1.3	23604	10922	14387
Buckyball	256^3	0.7	86690	32662	43152
H-Atom	128^3	1.23	8172	3644	3480
H-Atom	128^3	0.57	20767	7397	9196

Table 1. Error values, number of quadratic tetrahedra used for approximation, number of quadratic patches extracted, and number of triangles extracted.

A patch $Q(u, v)$ is constructed from two or three face-intersection curves by using the control points of the curves as the control points for $Q(u, v)$. Four or more face-intersection curves require the use of a "divide-and-conquer" method that results in multiple patches, see [21].

6 Results

We have applied our algorithm to various volume datasets. The datasets are all byte-valued, and the quadratic coefficients along the edges are stored as signed shorts. In all examples, the mesh is refined to approximate the original dataset, according to the quadratic tetrahedra approximation, within a user specified error bound e_u. The resulting mesh consists of a set of quadratic tetrahedra which approximates the dataset within e_u. The isosurface is extracted from this mesh. For comparison purposes, we extract an isosurface using linear elements constructed from the corner coefficients of the quadratic elements. This allows us to compare isosurfaces generated using the same number of elements. Table 1 summarizes the results. It shows the error value, the number of quadratic tetrahedra needed to approximate the dataset to within the specified error tolerance, and the number of quadratic patches and linear triangles extracted from the mesh.

As discussed in Section 4.2, the error value indicates the maximum difference between the quadratic representation and the actual function values at the data points. On the boundaries, our quadratic representation is C^0 continuous with respect to the function value and discontinuous with respect to the gradient; thus the gradients used for shading are discontinuous at patch boundaries. This fact leads to the creases seen in the contours extracted from the quadratic elements. The patches which define the contour are tessellated and rendered as triangle face lists. The ability to vary a patch's tessellation factor allows us to balance isosurface quality with rendering speed.

The storage requirements of the linear and quadratic representations are summarized in Table 2. Storage costs of linear and quadratic representations with and without precomputed gradients are shown. When gradients are precomputed for shading, a gradient must be computed at each data location regardless of representation. When rendering linear surfaces, gradients are often precomputed and quantized to avoid the cost of computing them at runtime. For quadratic patches, gradients do not need to be precomputed because they can be computed from the analytical definition of the surface. However, if gradients are precomputed, they can be used directly.

The difference between the leaf levels of linear and quadratic representations, as described in Section 4.2, implies that there are eight times as many diamonds in the linear representation than there are in the quadratic representation. We represent the quadratic coefficients with two bytes; however, in most cases, they can be represented using fewer bits. For example, the quadratic coefficients for the Buckyball dataset shown in Figures 5 and 6 lie in the range [-88,390] and can be represented in a single byte. The representation of error, min, and max values is the same for both representations. They can be stored as raw values or compressed to reduce storage costs. The quadratic representation essentially removes three levels from the binary tree of the tetrahedral refinement.

The first dataset is a Buckyball dataset made from Gaussian functions. Figure 5 compares contours extracted using quadratic and linear tetrahedra against the full resolution surface. The isosurfaces are extracted from the same mesh which consists of 86690 tets; it yields 32662 patches and 43152 linear triangles respectively. Figure 6 shows three isosurfaces of the Buckyball from the same viewpoint at different resolutions. The images are created by refining the mesh using a view-dependent error bound. Thus, the middle image, for an error of 1.3 has more refinement in the region closer to the viewpoint and less refinement in the regions further from the viewpoint. For the Buckyball dataset, the patches are tessellated with 28 vertices and 36 triangles. These pictures show how the quadratic representation approximates the dataset better than the linear representation when the same number of elements is used.

The second dataset is the Hydrogen Atom dataset obtained from **www.volvis.org**. The dataset is the result of a simulation of the spatial probability distribution of the electron in a hydrogen atom, residing in a strong magnetic field. Figure 7 shows the surfaces generated from quadratic and linear tetrahedra and the tetrahedral mesh from which the contours are extracted. While the linear and quadratic representations both capture the overall shape of the contour, the quadratic representation leads to a better reconstruction when using the same number of elements. Figure 8 is a closeup view of the dataset's interior. It shows a thin "hourglass-like" feature emanating from the probability lobe visible on the right. For the Hydrogen Atom dataset, the patches are tessellated with 15 vertices and 16 triangles.

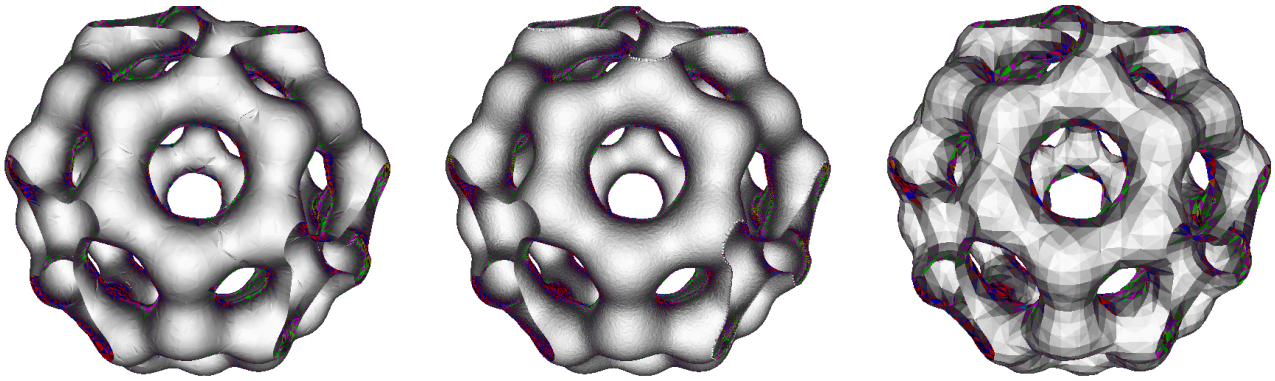

Figure 5. Left: Isosurface of quadratic patches extracted using quadratic tetrahedra. Middle: Full resolution isosurface (1798644 triangles). Right: Isosurface of triangles extracted from the same mesh. Isovalue = 184.4, Error = 0.7.

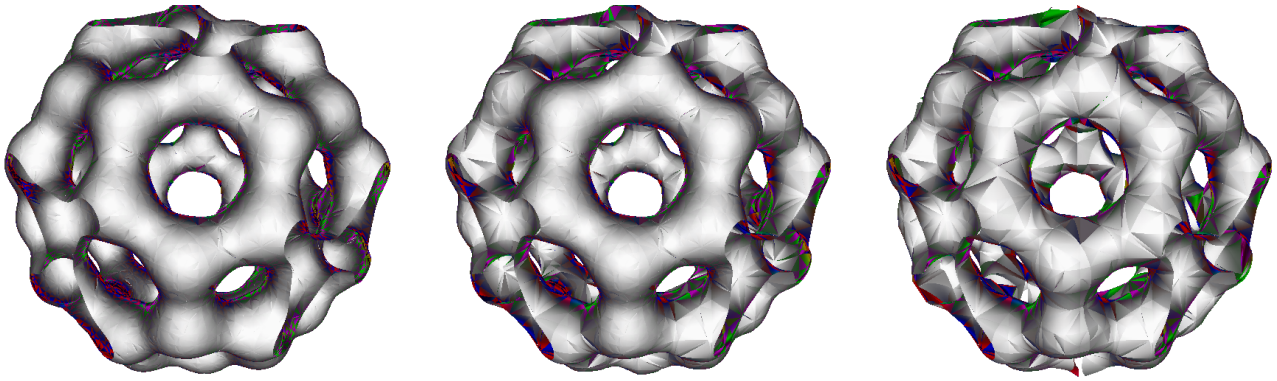

Figure 6. Isosurfaces extracted using quadratic tetrahedra at different error bounds. Left to Right: Error = 0.7, 1.2, and 2.0. Number of Quadratic Patches = 32662, 10922, 4609.

7 Conclusions

We have presented an algorithm for approximating and contouring datasets with quadratic tetrahedra. Our algorithm uses hierarchically defined tetrahedral meshes to construct a multiresolution representation. This representation is used to approximate the dataset within a user specified error tolerance. Quadratic tetrahedra are created from this multiresolution mesh by constructing approximating quadratic functions along edges and using these functions to form quadratic tetrahedra. We have improved previous methods for visualizing quadratic elements by showing how to directly contour them without splitting them into a large number of linear elements. Comparisons of datasets represented with quadratic and linear elements show that quadratic elements can represent datasets with a smaller number of elements and without a large storage overhead.

Future work is planned in these areas:

- **Improving the quality and speed of the contour extraction and comparing the quality of the surfaces to those generated from linear tetrahedra.** Currently, our algorithm generates some small thin surfaces that are undesirable for visualization. Additionally we are working on arbitrary slicing and volume rendering of quadratic elements.
- **Improving the computation of the quadratic representation.** Our current algorithm, while computationally efficient, fails to capture the behavior of the dataset within a tetrahedron, and yields discontinuous gradients at the boundaries. It is desirable to have an approximation that is overall C^1-continuous or C^1-continuous in most regions and C^0 in regions where discontinuities exist in the data. A C^1-continuous approximation might improve the overall approximation quality, allowing us to use fewer elements, and would improve the visual quality of the extracted contours.

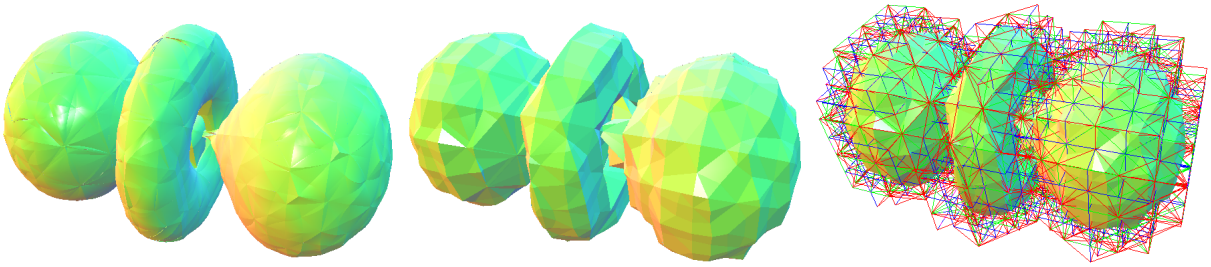

Figure 7. Isosurface through the Hydrogen Atom dataset. From left to right: the isosurface using quadratic patches, linear triangles, and the tetrahedra from which the contours were extracted. Isovalue = 9.4, Error = 1.23, Number of patches = 3644, Number of triangles = 3480.

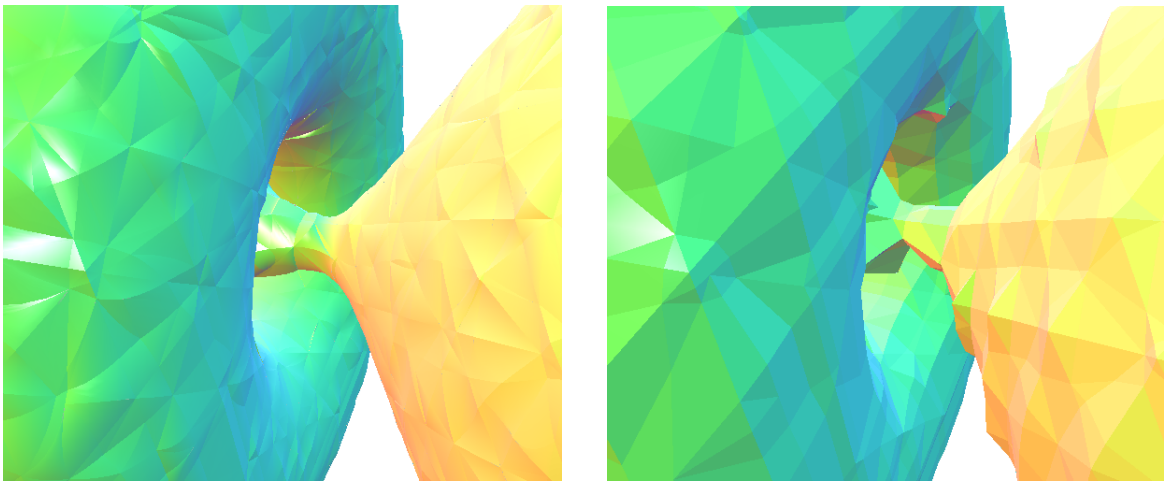

Figure 8. Closeup view of hydrogen atom dataset rendered with quadratic patches(left) and triangles(right). Isovalue = 9.4, Error = 0.566.

Acknowledgments

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 also supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI) and a large Information Technology Research (ITR) grant; the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the National Science Foundation; and the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 memorandum agreement B347878, and agreements B503159 and B523294; 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.

References

[1] B.K. Bloomquist, Contouring Trivariate Surfaces, Masters Thesis, Arizona State University, Computer Science Department, 1990

[2] P. Cignoni and P. Marino and C. Montani and E. Puppo and R. Scopigno Speeding Up Isosurface Extraction Using Interval Trees *IEEE Transactions on Visualization and Computer Graphics 1991*, 158–170

[3] P. J. Davis Interpolation and Approximation Dover Publications, Inc., New York, NY. 2, 3

Type	Data	Gradients	Bézier Coeffs	Error/Min/Max	Total
Linear	L	0	0	RL	$L(1+R)$
Linear	L	GL	0	RL	$L(1+G+R)$
Quadratic	$\frac{L}{8}$	0	CL	$R\frac{L}{8}$	$L\frac{1+8C+R}{8}$
Quadratic	$\frac{L}{8}$	GL	CL	$R\frac{L}{8}$	$L\frac{1+8G+8C+R}{8}$

Table 2. Storage requirements(bytes) for linear and quadratic representations for a dataset with 2^{3n} points. The linear representation consists of $L = 2^{3n}$ diamonds, and the quadratic representation consists of $\frac{L}{8} = 2^{3(n-1)}$ diamonds. R is the number of bytes used to store the error, min, and max values of a diamond, G is the number of bytes used to store a gradient, and C is the number of bytes used to store a quadratic coefficient.

[4] Klaus Engel and Rudiger Westermann and Thomas Ertl Isosurface Extraction Techniques For Web-Based Volume Visualization *Proceedings of IEEE Visualization 1999*, 139–146

[5] Benjamin Gregorski, Mark Duchaineau, Peter Lindstrom, Valerio Pascucci, and Kenneth I. Joy Interactive View-Dependent Extraction of Large Isosurfaces *Proceedings of IEEE Visualization 2002*, 475–482

[6] T. Gerstner Fast Multiresolution Extraction Of Multiple Transparent Isosurfaces, *Data Visualization 2001 Proceedings of VisSim 2001*

[7] Thomas Gerstner and Renato Pajarola, Topology Preserving And Controlled Topology Simplifying Multiresolution Isosurface Extraction, *Proceedings of IEEE Visualization 2000*, 259–266

[8] T. Gerstner and M. Rumpf, Multiresolution Parallel Isosurface Extraction Based On Tetrahedral Bisection, *Volume Graphics 2000*, 267–278

[9] Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel Feature-Sensitive Surface Extraction From Volume Data *SIGGRAPH 2001 Conference Proceedings*, 57–66

[10] Y. Livnat and C. Hansen View Dependent Isosurface Extraction *Proceedings of IEEE Visualization 1998*, 172–180

[11] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren Dual contouring of hermite data *SIGGRAPH 2002 Conference Proceedings*, 339–346

[12] Jian Huang, Yan Li, Roger Crawfis, Shao-Chiung Lu, and Shuh-Yuan Liou A Complete Distance Field Representation *Proceedings of Visualization 2001*, 247–254

[13] Gerald Farin, Curves and Surfaces for CAGD, Fifth edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2001

[14] A.J. Worsey and G. Farin, Contouring a bivariate quadratic polynomial over a triangle, *Computer Aided Geometric Design* 7 (1–4), 337–352, 1990

[15] B. Hamann, I.J. Trotts, and G. Farin *On Approximating Contours of the Piecewise Trilinear Interpolant using triangular rational-quadratic Bézier patches*, *IEEE Transactions on Visualization and Computer Graphics*, 3(3), 315–337 1997

[16] Tom Roxborough and Gregory M. Nielson, Tetrahedron Based, Least Squares, Progressive Volume Models With Application To Freehand Ultrasound Data", *In Proceedings of IEEE Visualization 2000*, **9**3–100

[17] S. Marlow and M.J.D. Powell, A Fortran subroutine for plotting the part of a conic that is inside a given triangle, Report no. R 8336, Atomic Energy Research Establishment, Harwell, United Kingdom, 1976

[18] R. Van Uitert, D. Weinstein, C.R. Johnson, and L. Zhukov Finite Element EEG and MEG Simulations for Realistic Head Models: Quadratic vs. Linear Approximations Special Issue of the Journal of Biomedizinische Technik, Vol. 46, 32–34, 2001.

[19] David F. Wiley, H.R. Childs, Bernd Hamann, Kenneth I. Joy, and Nelson Max, Using Quadratic Simplicial Elements for Hierarchical Approximation and Visualization, *Visualization and Data Analysis 2002*, *Proceedings*, SPIE - The International Society for Optical Engineering, 32–43, 2002

[20] David F. Wiley, H.R. Childs, Bernd Hamann, Kenneth I. Joy, and Nelson Max, Best Quadratic Spline Approximation for Hierarchical Visualization, *Data Visualization 2002, Proceedings of VisSym 2002*

[21] D. F. Wiley, H. R. Childs, B. F. Gregorski, B. Hamann, and K. I. Joy Contouring Curved Quadratic Elements *Data Visualization 2003, Proceedings of VisSym 2003*

[22] Jane Wilhelms and Allen Van Gelder Octrees for Faster Isosurface Generation *ACM Transaction in Graphics*, 201–227, July 1992