Isosurface Generation for Large-Scale Scattered Data Visualization

Christopher S. Co, Kenneth I. Joy

Institute for Data Analysis and Visualization (IDAV)
University of California, Davis
Email: {co, joy}@cs.ucdavis.edu

Abstract

We present an isosurface generation algorithm for large-scale volumetric scattered data. Rather than construct a global tessellation of the data points, we define a set of local tetrahedrizations that are guaranteed to cover the domain. Inside each local tetrahedrization, a point-based isosurface contouring algorithm is applied to generate isosurface geometry. Our work differs from previous work in that we make very few assumptions about the density of the point distribution, and we construct visual representations of the data directly without global tessellation or resampling. The merit of such an approach is a fairly general method that produces faithful renderings of large-scale scientific data.

1 Introduction

As data acquisition and scientific simulation become increasingly more sophisticated, there is a growing need for new visualization algorithms that support meshless data representations—data defined without an underlying mesh. Data acquisition technology is rapidly advancing to the point where acquired volumetric data sets can consist of an irregular lattice of sample points. Such data sets in the past have been referred to as *scattered* or *unstructured* data sets, or *point-clouds*. One attractive feature of methods designed for scattered data is that they can be applied to regular and hierarchical mesh and grid data without specialization.

Visualization algorithms for scattered data have followed two primary paradigms: global meshing and resampling. In the global meshing approach, a global set of tetrahedra representing the domain is computed. Once these tetrahedra are obtained, tetrahedra-based visualization algorithms—such as volume splatting [15], cell projection [19], or marching tetrahedra [26]—can be applied. The primary drawback of such an approach is its lack of scalability, since global mesh generation "remains one of the most time-consuming techniques" [16]. Another drawback of this general approach is that many tetrahedral mesh visualization techniques assume linear interpolation inside a given tetrahedron and cannot incorporate application-specific interpolation methods.

In a resampling approach, the scalar field is resampled on a rectilinear grid using a scattered data interpolant. Given the resulting grid, grid-based visualization algorithms can be applied. A severe disadvantage of this approach is that an additional interpolation step—e.g., trilinear interpolation—must be applied in rendering the grid data. This interpolation of the resampled data can produce unmeasurable error evidenced by artifacts in the rendering, see Figure 1. These artifacts are not only distracting but also unacceptably unfaithful to the data. Visualization should not introduce biases that would affect the analytic conclusions based on the data. Therefore, it is desirable to apply methods that directly visualize the original data, not a resampling of it. These issues motivate the development of methods that bypass the generation of a global mesh or a resampling grid.

To directly visualize the scattered data, we use a point-sampling isosurface extraction approach. We make use of an initial linear approximation (in the form of a tetrahedrization) to the potentially complex volumetric function. This tetrahedrization allows isosurface triangles approximating the surface of interest to be produced. The triangles are point-sampled, and these points are projected to the desired isosurface defined by the original interpolating function. Because the input scattered data may consist of a large number of data points, we avoid the generation of a global mesh by creating overlapping local meshes covering the domain of the global mesh. We refer to the problem of adequately fill-

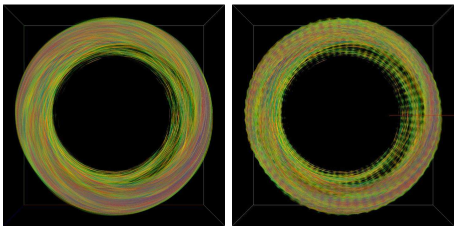

Figure 1: Using the original data and a resampling of the data. Note the rendering artifacts in the resampled version of this gyrokinetic simulation data set. (The resampled version uses trilinear interpolation.)

ing a domain with local meshes the *domain cover* problem. The focus of this paper is solving domain cover, as it is a prerequisite to applying point-based isosurfacing methods.

The major contribution of our work is a new localized approach to visualize scattered data that uses direct sampling. Specifically, we extract isosurface geometry by directly point-sampling the scalar field. Our algorithm has several desirable features. First, the locality of our method reduces the amount of memory necessary during processing and allows a parallel implementation. Second, since no resampling step is employed in the extraction of isosurface geometry, rendering artifacts are avoided. Finally, the sampling and interpolation procedures are decoupled, allowing application-specific interpolations to be incorporated without modifying the sampling procedure. This paper introduces a novel way of approaching the visualization of scattered data and represents a significant step to building effective visualization tools for scattered data sets. Figure 2 illustrates our method for isosurface extraction from scattered data, highlighting our solution to the domain cover problem.

In the following sections, we describe an automated algorithm to solve the domain cover problem and how this algorithm is applied to extract isosurfaces from scattered data. This process begins with stencil creation, where the point set data is partitioned into subsets. An octree aids in the process of *regularization* of the distribution to compute such *stencil point sets.* The union of local tetrahedrizations constructed from these stencil point sets provides a domain cover, and we use the local tetrahedrizations to generate continuous isosurface geometry.

2 Related Work

Numerous publications have been devoted to the study of scattered data interpolation. Traditional methods for scattered data interpolation include Shepard's method, radial basis functions (RBFs), tessellation-based methods, tensor-product methods, and blended local methods [10]. RBFs [4, 11, 13] and local methods have proven effective, particularly in the reverse engineering and surface reconstruction communities. There has been renewed interest in the application of moving least-squares [1, 3, 17] as a local method for fitting smooth approximations to point cloud data. Partition-of-unity blending functions [12, 20, 24] have promoted the use of local approximations used to construct a globally continuous approximation.

Meshing algorithms for point clouds play an important role in computational science, parameterization, and surface reconstruction. In particular, the Delaunay tetrahedrization [5, 9], and its dual, the Voronoi tessellation, are standard tools for natural neighbor interpolation [23, 24] and 3D surface reconstruction [2]. However, global tessellations often become undesirable in practice as data sets grow in size and complexity.

Researchers exploring point-based methods have studied the use of raw point samples without explicit mesh connectivity information. The point-based rendering community has focused primarily on the display and manipulation of complex surfaces represented by point primitives [21, 22]. Isosurfaces have been extracted in a point-based manner [6, 25, 18]. The iso-splatting technique was adapted to extract continuous point-based isosurfaces from multiblock data [8].

Our work is a visualization system in which various interpolation methods can be incorporated. Rather than construct a global tessellation of the data points, we define a set of local tetrahedrizations that are guaranteed to cover the domain. Inside each local tetrahedrization, point-based contouring is applied to extract isosurface geometry. Our work differs from previous work in that we make very few assumptions about the density of the point distribution, and we construct direct visualizations of the data without global tessellation or resampling. The contribution of this direct approach is a relatively general method that produces faithful scientific visualizations.

3 Regularization

To facilitate the construction of local tetrahedral meshes, we first regularize the point distribution. We add data points to the data set such that the resulting point distribution is quasi-uniform. We create a regular grid around the data points and distribute the points among the cells of this grid. Empty cells are filled with a single data point at the center of the cell whose function value is determined by interpolating. (Interpolation is discussed in Section 5.) These additional points are used for creating local tetrahedrizations and are not considered when evaluating the approximation. We refer to the resulting grid as the *binning grid*.

The properties of a desirable binning grid balance two competing desires. First, the grid should be coarse enough to minimize the number of evaluations of the scattered data interpolant. Since the interpolant is evaluated at points added to empty cells of the binning grid, the number of interpolant evaluations is equal to the number of empty cells in the binning grid. Second, the grid should be fine enough to minimize computation time of the local tetrahedrizations.

The properties of the binning grid are determined by using an octree to balance these competing constraints. The data points are rescaled to fit inside an axis-aligned unit cube, the root node of the octree. This cube is refined adaptively until each leaf cell contains at most one data point of the data set. The binning grid is a set of cells resulting from one full level of octree subdivision. We refer to this level as the *binning grid level* l_b, which corresponds to a $2^{l_b} \times 2^{l_b} \times 2^{l_b}$ grid. The binning grid is computed in two steps. In the first step, l_b is determined by traversing the octree in a depth-first fashion. In the second step, the binning grid is obtained by trimming or augmenting the $2^{l_b} \times 2^{l_b} \times 2^{l_b}$ grid such that a single layer of cells of the binning grid surrounds the axis-aligned bounding box of the data points.

To compute l_b, the octree is traversed in a depth-first fashion until a node is reached that contains at least one empty child node. The level of this node is appended to a list. If a leaf cell is encountered, its level is also added to the list. (A leaf node is a node containing *only* empty children.) The median level of the resulting list of node levels is selected as l_b. It is possible to choose the average level of this list, however we have found that the resulting binning grid is often *too fine*.

4 Local Tetrahedral Mesh Generation

Points inside a given $3 \times 3 \times 3$ stencil of binning grid cells are used to construct a local tetrahedrization. The stencils are centered in alternate cells to avoid excessive tetrahedrization overlap, see Figure 3. We use Delaunay methods to tetrahedrize the stencil point set. We note that the tetrahedrization provides starting point information for sampling the isosurface and not a globally consistent interpolating function.

Constructing local tetrahedrizations in this manner solves the domain cover problem. Consider that points are added to the outer layer of binning grid cells. The convex hull of these outer layer points form an axis-aligned box that contains the domain of the data points. After regularization, each cell of the binning grid contains at least one point. The union of the stencil tetrahedrizations cover the box formed by the outer layer points, since these tetrahedrizations span the space covered by the grid cells. This remains true for non-stencil-center cells since the tetrahedrizations either conform or overlap at their boundaries. Thus, since the union of the stencil triangulations cover the outer layer grid, and the outer layer grid contains the domain of the scattered data, the domain of the scattered data is appropriately covered.

Figure 3 shows 2D examples of conforming and overlapping local triangulations. Our algorithm naturally produces meshes that conform. However, we note that it is not necessary for the local meshes to conform to achieve domain cover. The local tetrahedrizations do not need to be computed *a priori*, but rather are generated on-the-fly and later discarded when the stencil is no longer needed. This greatly reduces the memory requirement for storing tetrahedrizations and enables a distributed implementation.

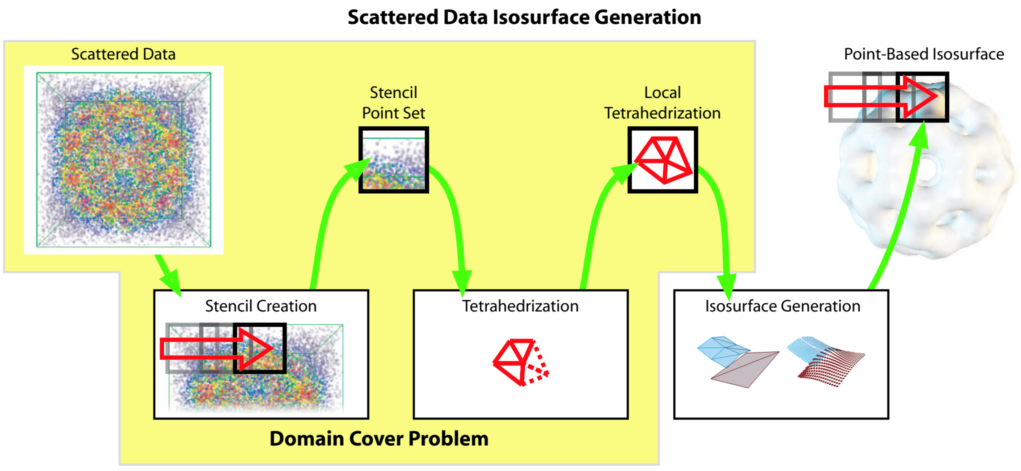

Figure 2: Overview of our algorithm. Given a scattered data set, our method constructs stencil point sets. A tetrahedrization is computed over each stencil point set. The local tetrahedrizations provide a domain cover. The portion highlighted in yellow (left) represents the portion that solves the domain cover problem. Isosurface generation is accomplished using point-based techniques.

5 Interpolation

For interpolation, we reuse the octree data structure computed for regularization. Our interpolation method uses the algorithmic structure of the multi-level partition-of-unity (MPU) implicits method of Ohtake et al. [20]. This method constructs a global approximation by blending local quadric approximations placed with the aid of an octree. Instead of using piecewise quadric approximations at octree nodes, we use local radial basis function (RBF) interpolants. During octree construction, we compute at each node information about the number of points contained in each cell as well as the radius of a sphere of influence centered at the centroid of the cell. The radius is computed to be αd, where d denotes the length of the octree node's diagonal. (We use $\alpha = 0.75$, which was shown to be an effective choice of α in the MPU implicits method.)

We use Hardy's multiquadric RBF method [14], since it offers high interpolation quality for a relatively low computational cost. Given a set of sample points S defined by a set of points $\{\mathbf{p}_i\}$, $i = 1, 2, \ldots, |S|$, let f_i be the scalar value associated with $\mathbf{p}_i$. Let $\mathbf{x}$ be an arbitrary location at which we wish to evaluate the interpolant. We construct Hardy's interpolant

$$H(\mathbf{x}) = \sum_{i}^{|S|} \lambda_i \phi_i(\mathbf{x}),$$

where

$$\phi_i(\mathbf{x}) = \sqrt{|\mathbf{x} - \mathbf{p}_i|^2 + c^2}.$$

Here, c denotes Hardy's constant, which we have set to 0.025 in our implementation, a value we have observed to be good in previous experimentation for data sets we have used. In order to construct a local interpolant, we build S from the set of points inside the sphere of influence of an interpolation node in the octree.

We use partition-of-unity weight functions to construct a global approximation of the scalar field by blending together local RBF interpolants. The local RBF interpolants are placed adaptively using our octree by associating them with the centroids of selected octree nodes, called *interpolation nodes*. We partition the data points into clusters manageable by the matrix solver [8]. At a given node, we collect the points inside the node's sphere of influence for the local interpolant. If the number of collected points is above a user-defined threshold N, we traverse the octree deeper until the matrix problem size is less than or equal to N. It is possible

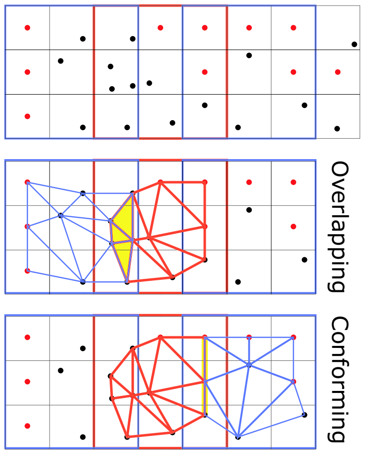

Figure 3: Illustration of boundary conditions in stencil triangulations. *Top*: Local triangulations are constructed from points inside 3×3 stencil of cells, shown in blue and red. *Middle*: An example of two neighboring stencil triangulations that overlap. Three triangles from the blue triangulation overlap three triangles of the red triangulation; the overlap is shaded in light orange. *Bottom*: An example of two neighboring stencil triangulations conforming at their boundaries, shaded in light orange. The fact that local triangulations either overlap or conform guarantee domain cover.

for the sphere of a leaf node in the octree to contain "too many" points, in which case, we augment the tree with further subdivisions until the threshold is satisfied. (In our implementation, we chose N to be 100 because the linear system can be solved relatively quickly while still produce high-quality results.)

Evaluation of the approximation is performed using a recursive routine in $O(log\ m)$ time, where m is the number of interpolation nodes in the octree [20]. The contribution of each local interpolation is calculated, the sum of the weights is accumulated, and the final value is normalized by dividing the sum of the contributions by the sum of the weights. It is important to note that only the original sample points are used for interpolation.

6 Isosurface Contouring

To construct isosurfaces from scattered data sets, we adapt a point-based, meshless isosurface generation method [8]. Our adaptation uses *marching tetrahedra* to obtain an initial triangle approximation for the isosurface of interest. These triangles are decomposed into points and projected to the actual isosurface as defined by the interpolation scheme. Since the triangles are derived from data points using the original data set, we believe the triangles serve as a reasonable initial approximation to the final surface.

The resulting surfaces are continuous point-based representations. The initial set of triangles resulting from the local tetrahedrization may be discontinuous. Once these triangles are decomposed into points and projected to the implicit surface defined by the interpolating function and the isovalue, the discontinuities disappear, and the surface defined by the interpolating function can be seen.

Isosurfaces are accurately generated in our system. Point projection is achieved through the use of Newton-Raphson iteration. Convergence of the iteration scheme is achieved when the displacement of the point between iterations is less than a user-defined threshold. (In our results, this threshold was set to 0.01.) In this way, isosurfaces accurate to within a user-defined threshold are produced.

The isosurface generation process is performed in a streaming fashion, such that only the current stencil tetrahedrization needs to be stored in main memory at any given time. A *marching stencil* generates the tetrahedrization, the triangles approximating the isosurface, and the resulting point-based isosurface on-the-fly.

7 Results

We implemented a parallel version of the isosurface contouring method, where the machines concurrently generate isosurface fragments while marching through separate stencils. We implemented a simple master-slave architecture, where a single master monitors the progress of slaves generating the isosurface geometry. We used a cluster of eleven desktop PCs communicating through MPI over a 100 Mb/s line. We employed a hybrid computing network consisting of computers with processors ranging from 2.80 GHz to 3.20 GHz and mem-

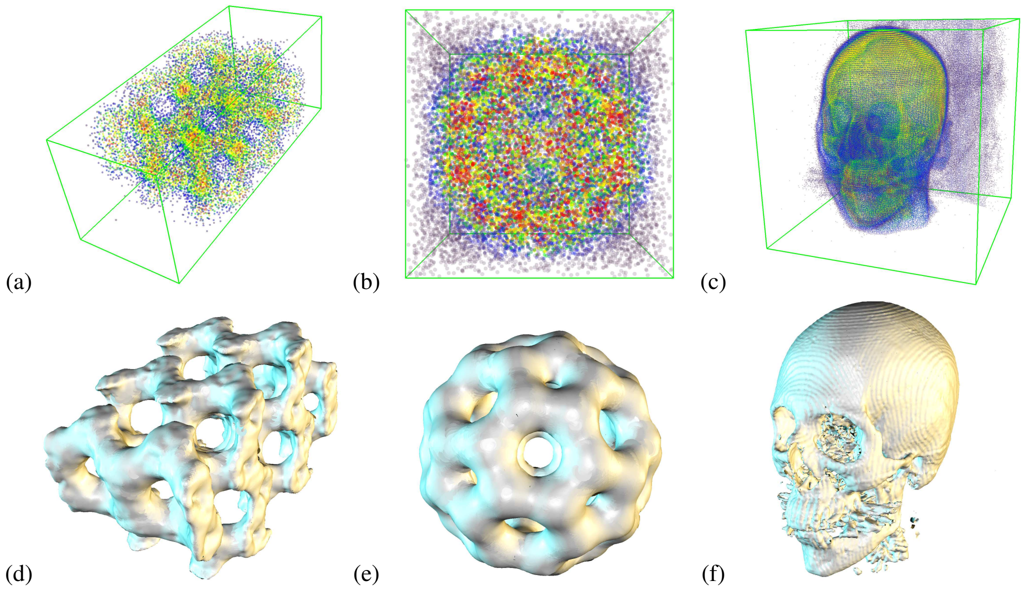

Figure 4: Visualizations of the irregular data point distributions for (a) a silicium, (b) buckyball, and (c) HighResHead data set. Point-based isosurface extracted from each scattered data set, (d), (e), (f).

Data set	Octree construction	Local RBF construction	Regular-ization	Preprocess Total	Isosurface generation
silicium	0.1 s	3.2 s	0.1 s	3.4 s	176.3 s
buckyball	0.2 s	3.1 s	0.1 s	3.4 s	69.2 s
HighResHead	1.9 s	48.8 s	45.5 s	96.2 s	375.1 s

Data set	Number of data points	Number of local RBFs	Number of empty cells
silicium	17,532	4,180	2,519
buckyball	20,000	4,096	31
HighResHead	345,452	92,022	1,758,408

Table 1: Summary of data sets and results of the timing experiments.

ory ranging from 1 GB to 4 GB. Given this configuration, we performed isosurface generation for a variety of data sets, measuring preprocessing time (octree construction, local RBF construction, and regularization) to assess the effectiveness of the approach. Timing results for the preprocessing are given in Table 1.

To test the method, we utilized three data sets from a variety of sources. The buckyball data set was obtained by sampling a 128^3 buckyball data set for 20,000 points using a uniform random distribution. For many of the data sets, we used levels of detail obtained from a meshless multiresolution hierarchy [7]. Data points in the hierarchy are determined using an iterative refinement process based on principal component analysis and binary space partitioning. While the original data sets are regular grid structures, the point distributions resulting from the meshless hierarchy are irregularly spaced. We used the meshless multiresolution hierarchy to generate the silicium data set and "HighResHead" data set. Table 1 provides a summary of the size of each data set. Visualizations of the data distributions as well as of the point-based isosurfaces extracted in the timing experiments are shown in Figure 4. We note that the bumpiness in the isosurfaces of Figure 4f is part of the data and not an artifact of our surface reconstruction.

In the case of HighResHead, regularization increases the number of data points by over a factor of five. This observation is indicative of the high density regions of data points focused around fine features in the data (see Figure 4), which influence the binning grid level in favor of a higher resolution binning grid. One remedy to avoid storing a large number of additional samples is the generation of samples in the empty cells on-the-fly. Nevertheless, regularization allows our system to easily generate local tetrahedrizations that are guaranteed to cover the domain. This is an important advantage of our algorithm, since it opens up the possibility to exploit the computational power of several machines by processing local tetrahedrizations in parallel.

8 Discussion

Octree construction time is $O(n \ logn)$ where n is the number of points in the data set. The time complexity of the local RBF construction depends heavily on the distribution of the data points. Regularization is proportional to the number of cells in the binning grid that are empty. The stencil containing the largest number of data points determines an upper bound for the construction of the stencil tetrahedrizations. While pathological point distributions can be artificially generated causing the tetrahedrization of a possibly large number of points, our method works well for data sets commonly encountered in practice.

Our system's performance is sensitive to the computational requirements of the interpolation method, which in the case of scattered data is *slow*. However, the quality of the images is better. While the speed advantage of resampling techniques and trilinear interpolation may justify the potential lack of accuracy in the visualized result in certain applications, we attempt to retain the information and the related context of the original data to the greatest degree possible and not introduce any biases into the visualization. Our technique scales to large parallel computational systems and can be easily modified to utilize new interpolation methods as they become available.

9 Conclusion and Future Work

We have presented an algorithm to extract isosurfaces from large-scale scattered data using a set of local tetrahedrizations. We compute a binning grid that balances the need for fine cells where the data distribution is dense against the need for coarse cells where the distribution is sparse. The distribution is regularized using the binning grid, and local tetrahedrizations are constructed using a *marching stencil* approach. Our system is an attractive solution to data exploration because it is a local method that requires a fairly modest memory footprint and lends itself naturally to parallel implementation.

Future work will focus on further improvement of our method. We have developed a system to handle point clouds representing a volume, where the domain is assumed to be the convex hull of the point set. Thus, adjustments to our system must be made to accommodate data sets consisting of non-convex domains. Although heuristics have been described to reduce the development of holes in the resulting surface [8], holes may still appear. We plan to investigate the use of surface splatting algorithms, which are able to fill small holes during rendering.

Acknowledgments

This work was supported by the National Science Foundation under contracts ACR 9982251 and ACR 0222909; the Lawrence Livermore National Laboratory under contract B523818, and by Lawrence Berkeley National Laboratory. We thank the members of the Visualization and Graphics Group of the Institute for Data Analysis and Visualization (IDAV) at UC Davis. The visualizations of the gyrokinetic data set are courtesy of Min-Yu Huang of IDAV. We thank Yutaka Ohtake for many useful discussions with respect to his work with MPU implicits. Tetrahedrizations were accomplished using the Computational Geometry Algorithms Library (CGAL) (http://www.cgal.org/). The silicium data set may be obtained online at http://www.volvis.org/. The buckyball data set is courtesy of Oliver Kreylos.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point Set Surfaces. In *IEEE Visualization '01 (VIS '01)*, pages 21–28, Washington - Brussels - Tokyo, October 2001. IEEE.

[2] N. Amenta, M. Bern, and M. Kamvysselis. A New Voronoi-Based Surface Reconstruction Algorithm. In M. Cohen, editor, *Proceedings of SIGGRAPH 98*, Annual Conference Series, Addison Wesley, pages 415–422. Addison Wesley, 1998.

[3] N. Amenta and Y.J. Kil. Defining point set surfaces. In *ACM*, volume 23 of *ACM Transactions on Graphics (TOG)*, pages 264–270, 1515 Broadway, New York, NY 10036, 8 2004. ACM, ACM Press.

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *Proceedings ACM SIGGRAPH 2001*, pages 67–76, Los Angeles, CA, August 2001. ACM SIGGRAPH.

[5] P. Cignoni, C. Montani, and R. Scopigno. DeWall: A Fast Divide and Conquer Delaunay Triangulation Algorithm in E^d. *Computer-Aided Design*, 30(5):333–341, 1998.

[6] C. S. Co, B. Hamann, and K. I. Joy. Iso-splatting: A Point-based Alternative to Isosurface Visualization. In J. Rokne, W. Wang, and R. Klein, editors, *Proceedings of the Eleventh Pacific Conference on Computer Graphics and Applications - Pacific Graphics 2003*, pages 325–334, October 8–10 2003.

[7] C. S. Co, B. Heckel, H. Hagen, B. Hamann, and K. I. Joy. Hierarchical Clustering for Unstructured Volumetric Scalar Fields. In G. Turk, J. J. van Wijk, and R. Moorhead, editors, *Proceedings of IEEE Visualization 2003*, pages 325–332. IEEE, October 19–24 2003.

[8] C. S. Co, S. D. Porumbescu, and K. I. Joy. Meshless Isosurface Generation from Multiblock Data. In O. Deussen, C. D. Hansen, D. A. Keim, and D. Saupe, editors, *Proceedings of VisSym 2004*. Eurographics, May 19–21 2004. (to appear).

[9] B. Delaunay. Sur la sphère vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[10] T. Foley and H. Hagen. Advances in Scattered Data Interpolation. *Surveys on Mathematics for Industry*, 4:71–84, 1994.

[11] R. Franke and H. Hagen. Least Squares Surface Approximation Using Multiquadrics and Parametric Domain Distortion. *Computer Aided Geometric Design*, 16:177–196, 1999.

[12] R. Franke and G. Nielson. Smooth Interpolation of Large Sets of Scattered Data. *International Journal for Numerical Methods in Engineering*, 15(11):1691–1704, 1980.

[13] H. Hagen, R. Franke, and G. Nielson. Repeated Knots in Least Squares Multiquadric Functions. *Computing Suppl. 10*, pages 177–187, 1995.

[14] R. L. Hardy. Multiquadric Equations of Topography and Other Irregular Surfaces. *Journal of Geophysical Research*, 76:1906–1915, 1971.

[15] M. Hopf and T. Ertl. Hierarchical Splatting of Scattered Data. In G. Turk, J. J. van Wijk, and R. Moorhead, editors, *Proceedings of IEEE Visualization 2003*, pages 433–440. IEEE, October 19–24 2003.

[16] S. R. Idelsohn, E. Oñate, N. Calvo, and F. Del Pin. The meshless finite element method. *International Journal for Numerical Methods in Engineering*, 58:893–912, 2003.

[17] D. Levin. The Approximation Power of Moving Least-Squares. *Mathematics of Computation*, 67(224):1517–1531, October 1998.

[18] Y. Livnat and X. Tricoche. Interactive Point-Based Isosurface Extraction. In H. Rushmeier, G. Turk, and J. J. van Wijk, editors, *Proceedings of IEEE Visualization 2004*, pages 457–464. IEEE, October 10–15 2004.

[19] K.-L. Ma and T. W. Crockett. A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data. In *Proceedings of the IEEE symposium on Parallel rendering*, pages 95–ff. ACM Press, 1997.

[20] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel. Multi-level Partition of Unity Implicits. In J. Hart, editor, *Siggraph 2003, Computer Graphics Proceedings*, volume 22, pages 463–470, July 2003.

[21] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape Modeling with Point-Sampled Geometry. In J. Hart, editor, *Siggraph 2003, Computer Graphics Proceedings*, volume 22, pages 641–650, July 2003.

[22] H. Pfister, J. van Baar, M. Zwicker, and M. Gross. Surfels: Surface elements as rendering primitives. In S. Hoffmeyer, editor, *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 335–342, New York, July 23–28 2000. ACMPress.

[23] R. Sibson. A Brief Description of the Natural Neighbour Interpolant. Technical report, Math Department, U. of Bath, 1980.

[24] N. Sukumar. Meshless Methods and Partition of Unity Finite Elements. In V. Brucato, editor, *Proceedings of the Sixth International ESAFORM Conference on Material Forming*, pages 603–606, Salerno, Italy, April 2003.

[25] B. von Rymon-Lipinski, N. Hanssen, T. Jansen, L. Ritter, and E. Keeve. Efficient Point-Based Isosurface Exploration Using the Span-Triangle. In H. Rushmeier, G. Turk, and J. J. van Wijk, editors, *Proceedings of IEEE Visualization 2004*, pages 441–448. IEEE, October 10–15 2004.

[26] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings of the 8th conference on Visualization '97*, pages 135–ff. IEEE Computer Society Press, 1997.