Title
Meshless Isosurface Generation from Multiblock Data

Authors
Co, Christopher S.
Porumbescu, Serban D.
Joy, Ken

Publication Date
2004

Peer reviewed

Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

Meshless Isosurface Generation from Multiblock Data

Christopher S. Co Serban D. Porumbescu Kenneth I. Joy

Center for Image Processing and Integrated Computing
Department of Computer Science
University of California, Davis
{co,porumbes,joy}@cs.ucdavis.edu

Abstract

We propose a meshless method for the extraction of high-quality continuous isosurfaces from volumetric data represented by multiple grids, also called "multiblock" data sets. Multiblock data sets are commonplace in computational mechanics applications. Relatively little research has been performed on contouring multiblock data sets, particularly when the grids overlap one another. Our algorithm proceeds in two steps. In the first step, we determine a continuous interpolant using a set of locally defined radial basis functions (RBFs) in conjunction with a partition of unity method to blend smoothly between these functions. In the second step, we extract isosurface geometry by sampling points on Marching Cubes triangles and projecting these point samples onto the isosurface defined by our interpolant. A surface splatting algorithm is employed for visualizing the resulting point set representing the isosurface. Because of our method's generality, it inherently solves the "crack problem" in isosurface generation. Results using a set of synthetic data sets and a discussion of practical considerations are presented. The importance of our method is that it can be applied to arbitrary grid data regardless of mesh layout or orientation.

1. Introduction

Isosurfaces play an important role in scientific visualization. Many important applications, such as geometric modeling and volume segmentation, are based on isosurface extraction. Much of the research in isosurface generation has focused on regular grid data sets, but relatively little research has addressed isosurfacing data represented by multiple grids of arbitrary layout and orientation. We refer to such data as *multiblock* data sets. Multiblock data representations arise in many important applications. In particular, flow simulations used in computational fluid dynamics (CFD) very frequently make use of multiple grids (see Figure 1). Moreover, these grids often overlap one another, a situation not addressed by standard isosurfacing techniques. Multiblock data sets appear in the form of hierarchical data representations as well, such as octrees [28] or adaptive mesh refinement (AMR) [27] grids. Volumetric data represented by such adaptive grid structures can be thought of as a specific class of multiblock data sets.

A major issue encountered when contouring multiblock data is the appearance of undesirable cracks or self-intersections, which result from a naive application of methods designed for single grid data sets. There are several reasons why it is important to resolve these discontinuities in the surface. Generally, the original object being sampled is continuous, and the cracks appear only as an artifact of a poor reconstruction. Moreover, many applications require watertight surface models. This issue has been treated to some extent in the study of adaptive grids for representing volume data, but such techniques frequently do not generalize to arbitrary multiblock data sets. The fundamental problem is two-fold. First, interpolation across the boundaries of these grids is often not well-defined or not easily dealt with. For instance, it is not well-defined how a trilinear interpolant defined in one grid should incorporate data samples from another grid overlapping it (see Figure 2(d)). We refer to this as the *interpolation problem*. Second, it is not clear how to extract continuous isosurface geometry across the grid boundaries. In particular, a direct application of the Marching Cubes algorithm [16] to each separate grid will often produce a triangle-based surface containing triangles that do not meet up (forming cracks) and triangles that intersect

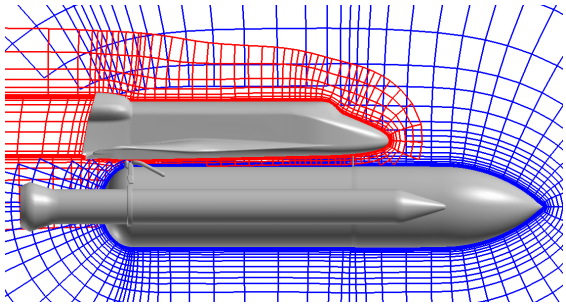

Figure 1: *An example of a multiblock data representation used in the simulation of flow around a space shuttle launch vehicle. (The data set is available online at http://www.nas.nasa.gov/Research/. Visualization courtesy of Edward A. Mayda of the Department of Mechanical and Aeronautical Engineering at UC Davis.)*

one another. We refer to this as the *isosurface generation problem*.

Meshless techniques for scientific computation have been introduced to alleviate mesh generation requirements by supplying robust interpolation methods that do not rely heavily on mesh connectivity. We use principles of meshless methods to address the interpolation problem by defining a continuous interpolation across grid boundaries. This continuous interpolation is used in conjunction with a flexible point-based surface sampling approach to solve the isosurface generation problem.

Our contribution is a general approach for the construction of isosurfaces from arbitrary multiblock data through the use of meshless techniques. We define a continuous interpolation scheme based on a set of radial basis functions, discussed in Section 3. The isosurface generation problem is addressed by first computing the triangles of a Marching Cubes surface in each grid. Inside the generated triangles, points are sampled and then projected onto the isosurface of interest using our continuous interpolant and principles of the iso-splatting method [4]. Isosurface generation is discussed in Section 4. What results is a point set representation suitable for rendering and geometry processing.

2. Related Work

Lorensen and Cline [16] greatly popularized isosurface visualization by introducing the Marching Cubes technique. Their method generates triangles to approximate an isosurface of interest for regular grid data sets using a case-driven approach. The elegance and relative simplicity of the Marching Cubes method make it a desirable technique to apply to data sets composed of hexahedral cells, such as multiblock data sets. However, few have successfully managed to extract continuous isocontours from arbitrary multiblock data. Although few have addressed this issue directly, several have treated a similar problem in attempting to reduce the amount of geometry generated by the standard Marching Cubes approach. Namely, the use of adaptive resolution volumetric representations [24, 22, 28, 13] often introduces discontinuities in the interpolation that ultimately result in surface cracks that must be patched. This area of research essentially addresses a specific instance of a broader class of multiblock isosurfacing problems. Solutions to the "crack problem" resolve discontinuities in situations where two grids *semi-conform* at the interface where the grids meet (see Figure 2(b)).

Several strategies exist to address the crack problem. Polygons residing in the plane of the crack can be computed to cover the hole [24]. Cracks can be patched by aligning vertices of triangles obtained from the high-resolution grid to the contour of the lower-resolution grid [22]. Coarse triangles can be replaced with an alternative triangulation that stitches the triangles from the two resolutions together [28, 13]. Another option is to avoid the generation of a volumetric mesh that would introduce a discontinuity in the volume [27, 8]. In this way, the interpolation and isosurface generation remain continuous.

These methods work well for conforming and semi-conforming grids, where all or some of the sample points along the interface between the grids are shared. These methods do not address problems with non-conforming or overlapping grids, sometimes called *overset grids* in computational mechanics. The fundamental issue here is that grid-based interpolants are not well defined in these domains, or, in the case of overlapping grids, it is not clear how to extract continuous geometry from this irregularly sampled subdomain. Our method specifically addresses non-conforming and overlapping grids and generalizes to conforming and semi-conforming grids without modification. Figure 2 shows various possible grid layouts for multiblock data.

Meshless methods for processing scientific data are becoming popular, particularly in the world of computational mechanics. Idelsohn et al. [12] pointed out that "3D mesh generation remains one of the most time-consuming techniques," and therefore, techniques that are independent of a mesh are attractive as an alternative. Floater and Reimers [5] have developed a meshless parameterization scheme suitable for surface reconstruction. Sukumar [26] advocates the use of meshless methods based on natural neighbor interpolation for finite element analysis. Many of these meshless techniques are based on scattered data interpolants, such as Sibson's interpolant [25], which is based on an underlying Voronoi decomposition of the domain. Radial basis functions (RBFs) have been popular for hole fixing and surface reconstruction [3], surface approximation [6, 10], and surface modeling [17]. Hardy's multiquadric interpolant [11] belongs to the class of RBF scattered data interpolants and has been successfully applied in scientific visualization.

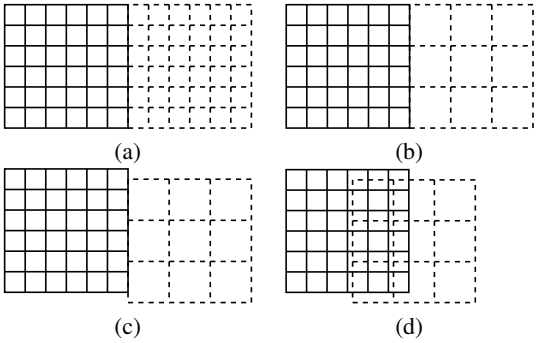

Figure 2: *Grid layouts in multiblock data sets. Many techniques exist to generate isocontours from (a) conforming grids and (b) semi-conforming grids, where some sample points are shared among grids at the interface. Relatively few techniques exist to generate isocontours from (c) non-conforming grids or from (d) overlapping grids. In these environments, common methods for interpolation are not well defined, and it is unclear how to extract a continuous isosurface representation.*

Scattered data interpolation schemes can have global and local formulations. Franke and Nielson [7] and Renka [20] discuss strategies for combining locally defined interpolants to obtain a globally smooth approximating function.

A field related to meshless methods is the area of point-based computer graphics. Researchers exploring this area have studied the use of raw point samples without explicit mesh connectivity information as an alternative to other popular geometric primitives. The point-based rendering community has focused primarily on the display and manipulation of complex surfaces represented by point primitives [15, 9, 19, 21, 1, 29, 2, 18]. Isosurfaces have been extracted in a point-based manner using the iso-splatting technique [4]. Much of this research has benefited from work in the meshless community, such as the use of moving least squares [14, 1, 18] to define the surface described by an arbitrary set of possibly noisy point samples.

We overcome the interpolation problem by disregarding the grid connectivity and defining a continuous RBF interpolant over the data points. In order to generate geometry to represent the isosurface, we extend principles from the iso-splatting method [4] to obtain a set of oriented points. The resulting point set is rendered using a surface splatting algorithm [21, 29, 2, 18] and is suitable for further geometry processing. The simplicity and generality of our method make it an attractive solution for obtaining high-quality isosurfaces from arbitrary grid data.

3. Interpolation

Radial basis functions (RBFs) provide effective and high-quality tools for interpolating arbitrary multivariate data. We first discuss the RBF interpolation scheme we use in Section 3.1. How we build and later evaluate these local interpolants to form a single global function is described in Section 3.2.

3.1. RBF Interpolation

Given a set of sample points S defined by a set of points $p_i = (x_i, y_i, z_i),\ i = 1, 2, \ldots, |S|$, let F_i be the scalar value associated with p_i. We interpolate a value H at $p = (x, y, z)$ using a multiquadric method [11, 10, 6].

To compute a function value, we utilize an interpolation of the form

$$H(x,y,z) = \sum_{j=1}^{N} a_j B_j(x,y,z), \tag{1}$$

where each B_j is a multiquadric function defined as

$$B_j(x,y,z) = [\,(x_j - x)^2 + (y_j - y)^2 + (z_j - z)^2 + R^2\,]^{1/2},$$

subject to the constraints

$$H(x_j, y_j, z_j) = F_j \quad j = 1, 2, \ldots, N. \tag{2}$$

The gradient of this function is given by

$$\nabla H(x,y,z) = \left(\frac{\delta H}{\delta x}, \frac{\delta H}{\delta y}, \frac{\delta H}{\delta z}\right),$$

where the partial derivative $\frac{\delta H}{\delta x}$ is computed as

$$\frac{\delta H}{\delta x} = \sum_{j=1}^{N} \frac{x - x_j}{B_j(x,y,z)}. \tag{3}$$

The partial derivative of $H(x, y, z)$ with respect to y and z take a similar form as in Equation (3). We note that Equations (1) and (3) make use of the evaluation of the basis functions B_j, and so the function value and gradient can be computed simultaneously.

We call R the *multiquadric parameter* and a_j the *blending coefficients*. The multiquadric parameter is usually fixed. The appropriate value for R is still an open problem, however, we have found that using a value of 0.025 works very well for many of the volumetric scalar fields we have used. The blending coefficients are determined by solving a linear system of equations given by the constraints of Equation (2). The value of N is the number of data points used for value approximation. Classically, a global fit is performed by setting N to be the same as $|S|$, which can be computationally expensive for large data sets. A local scheme can be adopted by considering only the N points around p within a certain neighborhood.

A local interpolation defined in a piecewise fashion is not guaranteed to be continuous and thus must be blended. Partition of unity methods [7, 20, 26, 17] are usually applied to blend locally defined interpolation functions into one smooth global interpolant. Partition of unity functions ϕ_k have the

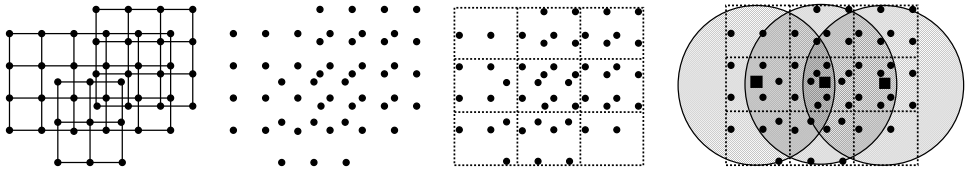

Figure 3: *Local RBF Construction. (a) The input is multiblock data. (b) The grid connectivity is disregarded. (c) Data points (shown as black dots) are binned into RBF grid cells. (d) Local RBFs are computed with centers placed at the center of the cells. An RBF center is shown here as a square point, and its radius of influence is shown as a shaded circle. To avoid visual clutter, only the three middle RBFs are shown.*

feature that across the domain of the function

$$\sum_k \phi_k(x,y,z) \equiv 1.$$

Given M nonnegative compactly supported functions W_q, we can define a set of partition of unity functions by

$$\phi_k(x,y,z) = W_k(x,y,z) \Big/ \sum_{q=1}^{M} W_q(x,y,z).$$

Our choice of W_k is discussed in Section 3.2. To define a globally continuous interpolant $F(x,y,z)$, we define

$$F(x,y,z) = \sum_{k=1}^{M} \phi_k(x,y,z) H_k(x,y,z),$$

where H_k is a local RBF defined as in Equation (1).

3.2. Local RBF Construction

To define a local interpolation H_k, we first disregard the connectivity of the grid and consider only the data points. These data points are then binned into a global regular grid, which we call the *RBF grid*, defined by the bounding box around the data points. We associate a local interpolant H_k with each cell center point c_k of this grid. This point is often referred to as an *RBF center*. We define a radius of influence R_w around each RBF center to be

$$R_w = 2\sqrt{{s_x}^2 + {s_y}^2 + {s_z}^2},$$

where (s_x, s_y, s_z) are the dimensions of a single cell in the RBF grid. The coefficients of the local RBF associated with c_k are computed by considering all points within the sphere defined by the RBF center c_k and the radius of influence R_w. The points necessary to compute these parameters are obtained by checking each point in the neighboring 26 cells– the eight vertex neighbors, twelve edge neighbors, and six face neighbors–to see if the point is inside the sphere. If this region happens to be empty, we do not construct an RBF center there. The RBF construction process is illustrated in Figure 3.

In creating the RBF grid, we attempt to make the cells as close to cubes in shape as possible. This is achieved by computing the RBF grid's dimensions $n_x \times n_y \times n_z$ using

$$\begin{aligned} s &= \left(m \frac{w_x w_y w_z}{n}\right)^{1/3}, \\ n_x &= round\left(\frac{w_x}{s}\right), \\ n_y &= round\left(\frac{w_y}{s}\right), \\ n_z &= round\left(\frac{w_z}{s}\right), \end{aligned}$$

where n is the number of data points in the entire data set, (w_x, w_y, w_z) is the size of the bounding box enclosing all the data grids, and m is a user-defined parameter roughly specifying the number of data points to place in each cell. We refer to m as the *binning constant*. These equations are commonly used to create a grid for the accelerated ray tracing of complex scenes [23].

When evaluating the function at an arbitrary point $p = (x,y,z)$, we determine in which RBF grid cell the point p resides to obtain the local RBF interpolant H_k. We collect the 26 neighboring local RBF interpolants adjacent to this cell in the RBF grid. At the point p, these 27 local interpolants are evaluated and blended together using a partition of unity function to obtain a final function value and gradient. We utilize the inverse distance functions

$$\begin{aligned} W_k(x,y,z) &= \left(\frac{(R_w - d_k)_+}{R_w d_k}\right)^2 \\ (R_w - d_k)_+ &= \begin{cases} R_w - d_k & \text{if } d_k < R_w \\ 0 & \text{if } d_k \geq R_w \end{cases} \end{aligned}$$

as weight functions to generate our partition of unity function, where d_k is the Euclidean distance between the RBF center point c_k and the point p [20]. If a cell does not have an associated local interpolant, we do not consider it for our calculation.

4. Isosurface Generation

To generate geometry to represent the isosurface, we first compute triangles inside each cell of the original multiblock

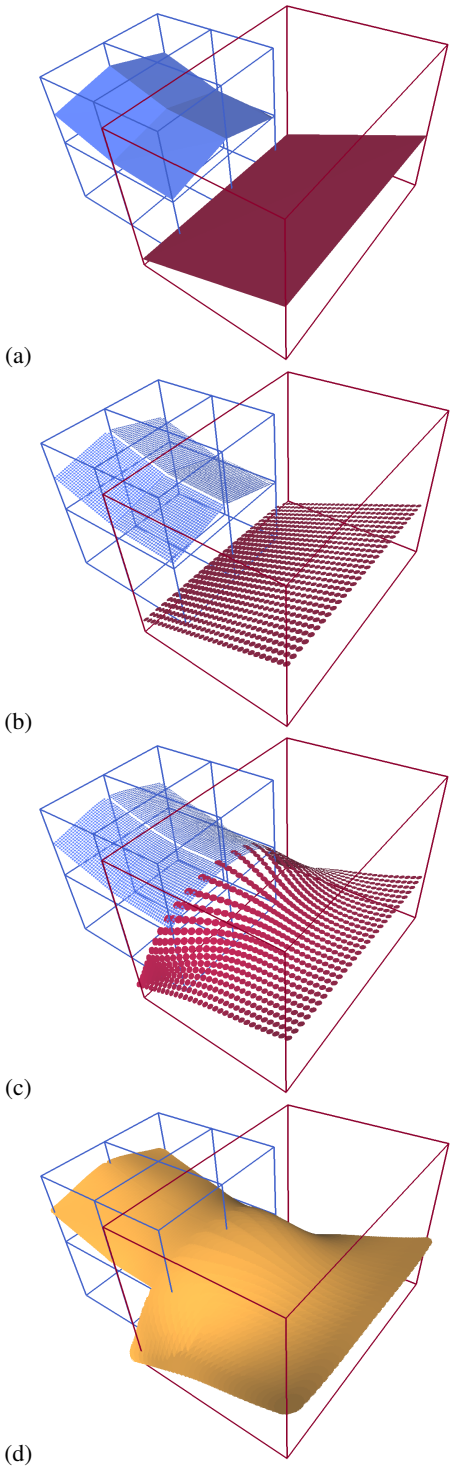

Figure 4: *Illustration of our isosurface generation method. (a) Marching Cubes triangles are generated inside each cell. (b) The point samples are generated in each triangle and then (c) projected onto the isosurface. (d) The point set is rendered using a surface splatting algorithm.*

grids according to the Marching Cubes algorithm. Inside these triangles, we compute a set of points at which we sample our volume for function value and gradient using our interpolation scheme. We generate a uniform lattice of samples inside the triangle using barycentric coordinates and determine the number of samples to generate by specifying the number of samples to compute in the u- and v-directions. We take care not to sample along the edge of a given triangle to avoid redundant computation, since the triangle's edge neighbor could duplicate the same exact points. Each sample point, its function value, and associated gradient are used as input to a Newton-Raphson root-finding procedure to project the point onto the isosurface of interest. Newton-Raphson iteration finds approximate roots of a function by using function value and first derivative information to move an initial guess closer to the actual solution. The Newton-Raphson procedure converges to the isosurface quadratically when given a reasonably good initial guess, which the triangles inside cells intersected by the isosurface provide. The result is a set of points and normals which collectively define the isosurface. Figure 4 illustrates our isosurface generation scheme for a small example.

For rendering and geometry processing, we define the radius of each point sample to be proportional to the area of the triangle in which it was sampled and the distance it traveled from the original point sample taken. Let A denote the area of the triangle, n be the number of samples taken uniformly in the u- and v-directions inside the triangle, l be the length of the diagonal of the cell in which the triangle exists, and d be the Euclidean distance between the original sample point and its final computed location. The number of points sampled inside a triangle is $\frac{n(n+1)}{2}$. The radius r of a point sample is computed to be

$$r = 2\left(1+\frac{d}{l}\right)\sqrt{\frac{2A}{\pi n(n+1)}} \quad .$$

We effectively compute the size of each point to be large enough so that if no displacement of the points occur, the original triangle is rendered without holes. However, when the point is displaced from the original sample location, we scale the size of the point linearly with respect to the distance it travels from that point. The premise is that as a point moves further from its original sample location, the surface sampling becomes less dense, and thus the point must be made larger to compensate for gaps evolving between it and the neighboring points.

5. Results

To test the effectiveness of our approach, we used a set of synthetic data sets created by sampling available scalar data sets on multiple grids. The "bucky ball" data set was sampled using semi-conformal grids. The "fuel injection" and "neghip" data sets were sampled using multiple overlapping grids. Our test machine was a 2.8 GHz P4 PC with 2 GB of

memory. We fixed the multiquadric parameter to 0.025. The binning constant was set to 5. The RBF construction computation is dominated by the linear system solver for computing the blending coefficients. The time complexity of this computation is $O(n^3)$, where n is the number of data points used to compute the local interpolant. Each local RBF used an average of approximately 150-300 data points. Information about the data sets as well as the RBF construction performance are provided in Table 1.

Figure 5 demonstrates how our method can be used on semi-conformal multiblock data sets, such as those which result from octree subdivisions. Again, the cracks exhibited by a naive application of Marching Cubes (seen in the top and middle images) are not realized in the isosurface extracted using our approach (bottom-most image).

Figure 6 shows the neghip data set sampled by two overlapping grids. The rendering of the isosurfaces obtained from a naive application of Marching Cubes are color-coded to match the color of the grids. Cracks and self-intersections in the isosurface are observed, since the blue surface intersects the pink one, and vice versa. This shows that the trilinear interpolant of one grid does not agree with the other grid as to the location of the isosurface. The bottom images of Figure 6 provide a side-by-side quality comparison of Marching Cubes versus our method.

Figure 7 illustrates our approach applied to the fuel injection data set consisting of several overlapping grids. The isosurfaces are color-coded to match the grid in which the geometry was sampled. Note that the Marching Cubes result (middle image) also contains several self-intersections and cracks. Again, this is an indication of a discontinuous interpolation scheme combined with an isosurface generation method that makes it difficult to extract a continuous surface. The same isosurface using our approach (right-most image) offers improved representation quality by using a continuous interpolation scheme along with a flexible geometry generation method.

6. Discussion and Future Work

Our method has several advantages and offers many interesting avenues for future work. Specifically, the use of points for sampling our implicit surface is well-suited to the case of overlapping grids. Also, one can imagine many other interesting ways to generate point samples on the isosurface of interest.

One major advantage to sampling the isosurface using points is that the point samples are not adversely affected by the overlap of nearby samples. For instance, triangles generated in one grid may be extremely close to triangles generated in an overlapping grid. When we sample these triangles for points and project them onto the isosurface, it is likely that many points will end up overlapping one another significantly. While this usually causes problems when using other geometric primitives, such as triangles, point geometry is less sensitive to this redundancy. Since all of these points snap to the same surface, redundancy is not a problem. As long as there is consistency among the point samples as to where the isosurface exists, no cracks or self-intersections will appear. This consistency is provided by our continuous interpolation scheme. The right-most image of Figure 7 shows an example where significant point sample overlap occurs because of overlapping grids.

Another major advantage to sampling the isosurface using points is that we can adjust our sampling in a flexible manner to meet application-specific requirements. The surface onto which we wish to map points is implicitly known and can be sampled less densely, for instance, in areas of low curvature and more densely in areas of high curvature. Similarly, it is possible to steer the generation of point samples based on viewing parameters, as in the point-set-surface method by Alexa et al. [1], where a moving least squares surface is dynamically sampled to meet screen space resolution. While this feature is not a major focus of our work, we feel that this added flexibility is a key advantage that makes our approach usable in a variety of application domains.

Our future work will focus on more effective strategies for isosurface generation. In certain situations, uniformly sampling the triangles using barycentric coordinates may produce more point samples than is really necessary to represent the desired isosurface. On the other hand, although Marching Cubes triangles offer an excellent framework within which to sample our continuous isosurface, there is still no guarantee that these triangles will provide enough initial samples to adequately represent the desired isosurface. Other methods for inserting initial guesses for the placement of isosurface geometry must be explored. We intend to investigate alternative sampling patterns and methods that analyze surface characteristics to achieve a more optimal distribution of surface samples.

Other future work will include investigating optimal placement of the RBF centers and the application of our technique to data sets consisting of other types of cells. We use a uniform placement of RBF centers in our work. A more optimal placement of RBF centers may be achieved by using, for instance, octrees and error estimates to guide the process. We have primarily discussed the application of our method to hexahedral data sets. However, our method has its foundation in scattered data analysis, and thus easily generalizes to various types of volumetric meshes, such as tetrahedral meshes. The use of other scattered data interpolation schemes and more efficient methods for evaluating these interpolants are issues of great importance and deserve a thorough investigation.

7. Conclusion

We have presented the first technique for generating continuous isosurfaces from arbitrary multiblock data. Using

Data Set	# of Grids	# of Data Points	# of RBF Centers	RBF Construction Time
bucky ball	2	21,114	4,096	1 min 42 s
fuel injection	5	2,688	496	5 s
neghip	2	16,000	3,060	50 s

Table 1: *RBF construction statistics performed on a 2.8 GHz P4 with 2 GB of memory.*

local radial basis functions and partition of unity methods for interpolation, we defined a continuous interpolant across grid boundaries, regardless of grid resolution, grid overlap and regardless of whether or not the grids conform. We use Marching Cubes as a framework for generating point samples near the isosurface. These point samples are projected onto the isosurface through Newton-Raphson iteration. The result is a point set describing a continuous surface. The main contribution of this work is that arbitrary grid data can be handled in a general way, making our method applicable to a wide variety of data sets without any specialization.

Acknowledgments

This work was supported by the National Science Foundation under contracts ACR 9982251 and ACR 0222909, through the National Partnership for Advanced Computing Infrastructure (NPACI); the Lawrence Livermore National Laboratory under contract B523818, and by Lawrence Berkeley National Laboratory. We thank N. Sukumar of the Department of Civil and Environmental Engineering at UC Davis for many useful discussions with respect to meshless methods for scientific computation. We are grateful to Edward A. Mayda of the Department of Mechanical and Aeronautical Engineering at UC Davis for his help understanding issues in computational mechanics. We thank Oliver Kreylos of the Center for Image Processing and Integrated Computing (CIPIC) at UC Davis for supplying the bucky ball data set. The neghip and fuel injection data sets may be obtained online at http://www.volvis.org/. The space shuttle lauch vehicle data set can be obtained online at http://www.nas.nasa.gov/Research/. We thank the members of the Visualization and Graphics Group of the Center for Image Processing and Integrated Computing (CIPIC) at UC Davis.

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. 3, 6

[2] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In S. Gibson and P. Debevec, editors, *Proceedings of the 13th Eurographics Workshop on Rendering (RENDERING TECHNIQUES-02)*, pages 53–64, Aire-la-Ville, Switzerland, June 26–28 2002. Eurographics Association. 3

[3] 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. 2

[4] C. S. Co, B. Hamann, and K. I. Joy. Iso-splatting: A Point-based Alternative to Isosurface Visualization. In *Proceedings of the Eleventh Pacific Conference on Computer Graphics and Applications - Pacific Graphics 2003*, pages 325–334, October 8–10 2003. 2, 3

[5] M. S. Floater. Meshless parameterization and surface reconstruction. *Computer Aided Geometric Design*, 18:77–92, 2001. 2

[6] R. Franke and H. Hagen. Least squares surface approximation using multiquadrics and parametric domain distortion. *Computer Aided Geometric Design*, 16:177–196, 1999. 2, 3

[7] 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. 3

[8] B. F. Gregorski, M. A. Duchaineau, P. Lindstrom, V. Pascucci, and K. I. Joy. Interactive view-dependent rendering of large isosurfaces. In *Proceedings of the IEEE Visualization 2002*. IEEE, IEEE, 10 2002. 2

[9] J. P. Grossman and W. J. Dally. Point sample rendering. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, Eurographics, pages 181–192. Springer-Verlag Wien New York, 1998. 3

[10] H. Hagen, R. Franke, and G. Nielson. Repeated knots in least squares multiquadric functions. *Computing Suppl. 10*, pages 177–187, 1995. 2, 3

[11] R. L. Hardy. Theory and applications of the multiquadric-biharmonic method: 20 years of discovery 1968–1988. *Computers and Mathematics with Applications*, 19:163–208, 1990. 2, 3

[12] 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. 2

[13] R. S. Laramee and R. D. Bergeron. An Isosurface Continuity Algorithm for Super Adaptive Resolution Data. In J. Vince and R. Earnshaw, editors, *Advances in Modelling, Animation, and Rendering: Computer Graphics International (CGI 2002)*, pages 215–237, Bradford, UK, July 1-5 2002. Computer Graphics Society, Springer. 2

[14] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, October 1998. 3

[15] M. Levoy and T. Whitted. The use of points as a display primitive. Technical Report 85-022, University of North Carolina at Chapel Hill, 1985. 3

[16] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface reconstruction algorithm. In M. C. Stone,

editor, *Siggraph 1987, Computer Graphics Proceedings*, volume 21, pages 163–169. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, July 1987. 1, 2

[17] 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. 2, 3

[18] 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. 3

[19] 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. 3

[20] R. J. Renka. Multivariate interpolation of large sets of scattered data. *ACM Transactions on Mathematical Software*, 14(2):139–148, June 1988. 3, 4

[21] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In K. Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 343–352. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000. 3

[22] R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In R. Yagel and G. M. Nielson, editors, *Proceedings of the Conference on Visualization*, pages 335–344, Los Alamitos, October 27–November 1 1996. IEEE. 2

[23] P. Shirley. *Realistic Ray Tracing*. AK Peters Limited, first edition, 2000. 4

[24] R. Shu, C. Zhou, and M. S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11(4):202–217, 1995. ISSN 0178-2789. 2

[25] R. Sibson. A vector identity for the dirichlet tessellation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 87(1):151–155, 1980. 2

[26] 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. 2, 3

[27] G. H. Weber, O. Kreylos, T. J. Ligocki, J. M. Shalf, H. Hagen, B. Hamann, and K. I. Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In D. S. Ebert, J. M. Favre, and R. Peikert, editors, *Data Visualization 2001 (Proceedings of "VisSym '01")*, pages 25–34, Vienna, Austria, 2001. Springer-Verlag. 1, 2

[28] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, 15(2):100–111, 1999. 1, 2

[29] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In E. Fiume, editor, *Siggraph 2001, Computer Graphics Proceedings*, pages 371–378. ACM Press / ACM SIGGRAPH, 2001. 3

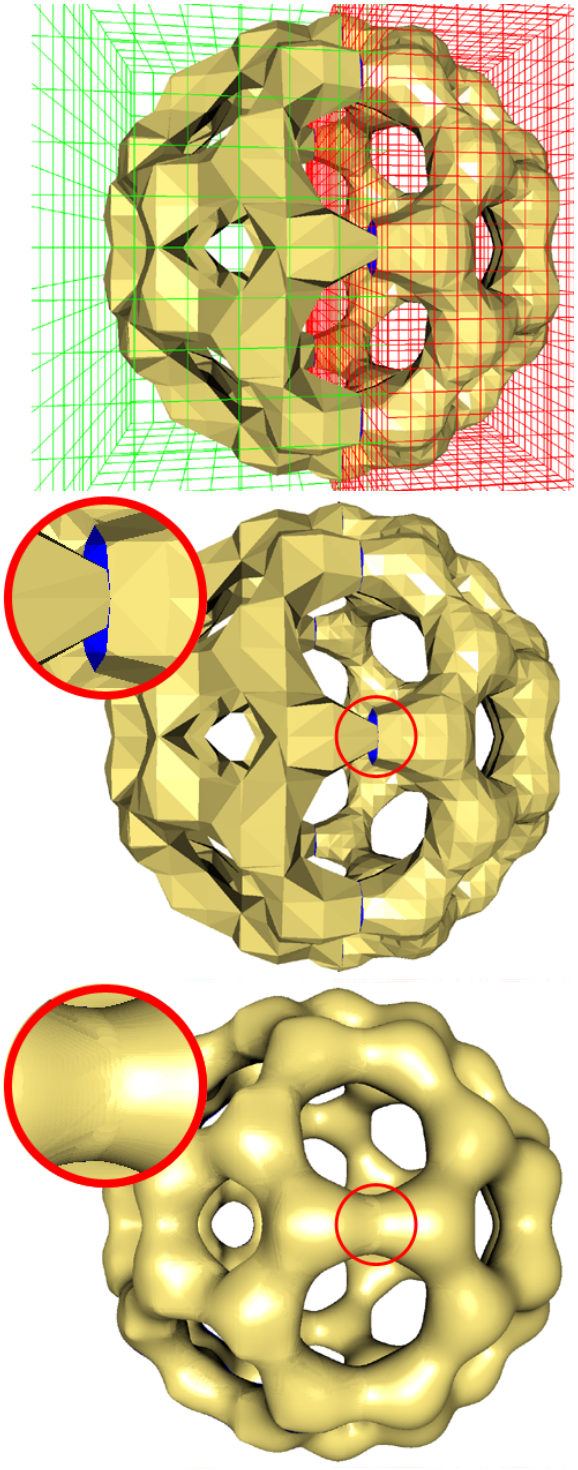

Figure 5: *Isosurface visualization of a semi-conformal multiblock representation of the bucky ball data set. Top: View showing two grid resolutions. Middle: Marching Cubes results in cracks at the coarse-fine boundary. Bottom: Our method applied to the same data set produces a continuous closed isosurface representation.*

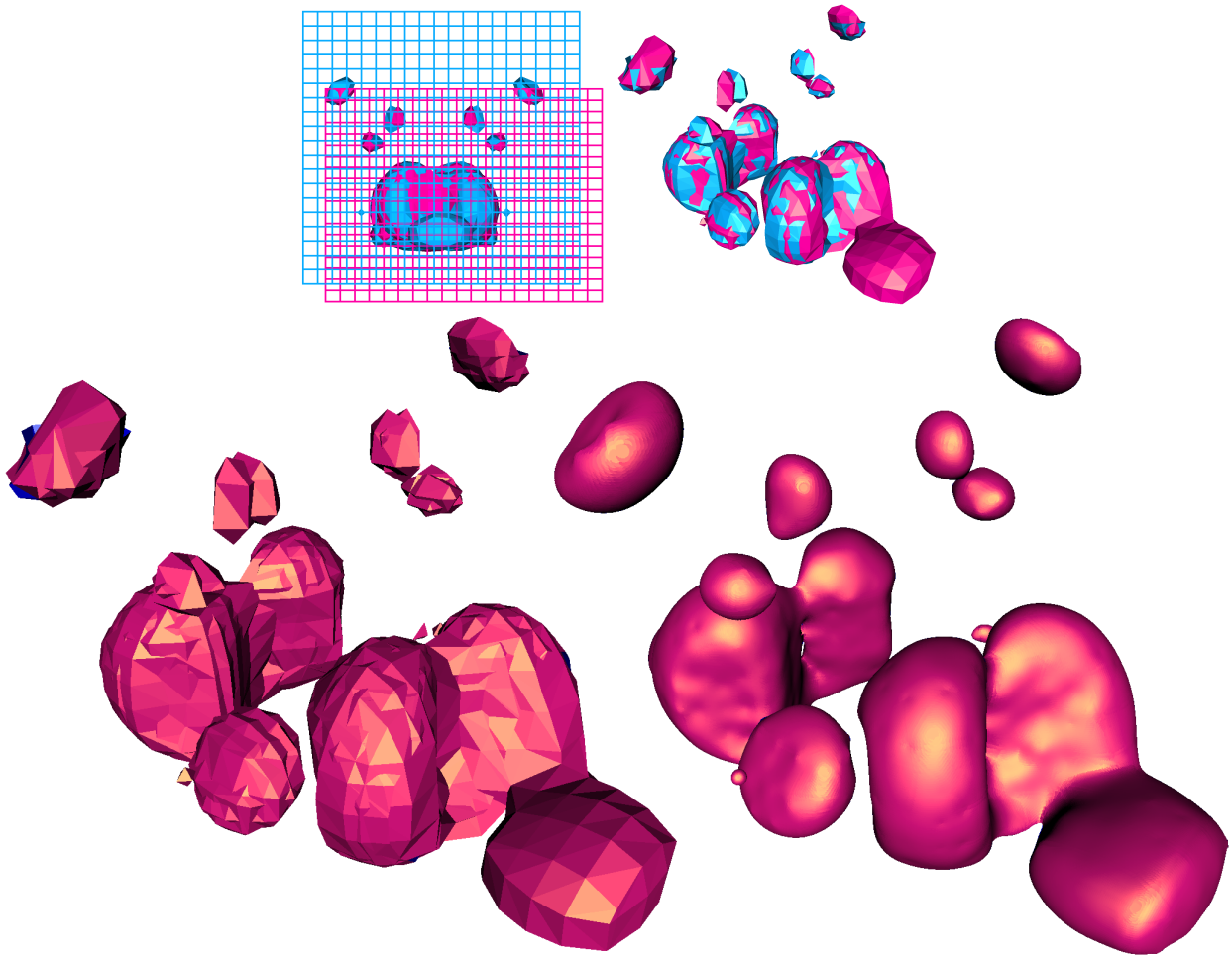

Figure 6: *Isosurface visualization of a multiblock representation of the neghip data set. Top-left: Axis-aligned view showing two overlapping grids representing the data. Top-right: Marching Cubes triangles color-coded by grid. Notice the self-intersections and cracks in the isosurface. Bottom: Side-by-side comparison of Marching Cubes versus our approach.*

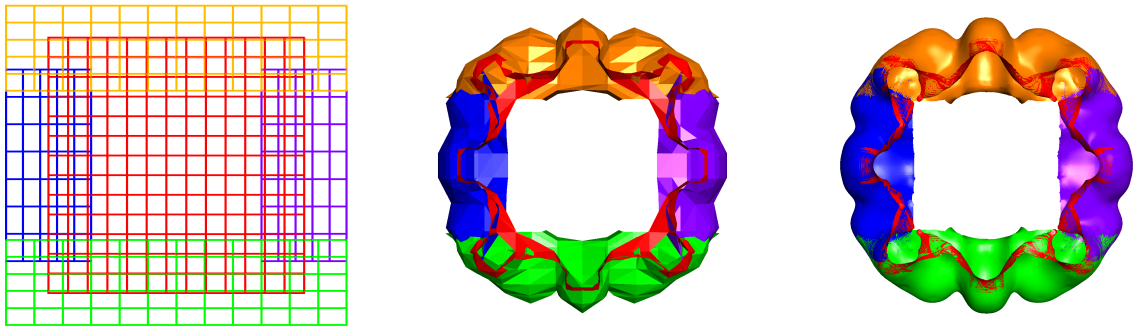

Figure 7: *A fuel injection data set represented by five overlapping grids. Left: A visualization of the five grids. Middle: A naive application of Marching Cubes produces several cracks and self-intersections in regions of overlap. Right: Our method generates a single continuous surface. Overlapping points are not a problem, as they project to a globally defined isosurface.*