Particle Systems for Efficient and Accurate High-Order Finite Element Visualization

Miriah Meyer, *Student Member*, *IEEE*, Blake Nelson,
Robert M. Kirby, *Member*, *IEEE*, and Ross Whitaker, *Member*, *IEEE*

Abstract—Visualization has become an important component of the simulation pipeline, providing scientists and engineers a visual intuition of their models. Simulations that make use of the high-order finite element method for spatial subdivision, however, present a challenge to conventional isosurface visualization techniques. High-order finite element isosurfaces are often defined by basis functions in *reference space*, which give rise to a *world-space* solution through a coordinate transformation, which does not necessarily have a closed-form inverse. Therefore, world-space isosurface rendering methods such as marching cubes and ray tracing must perform a nested root finding, which is computationally expensive. We thus propose visualizing these isosurfaces with a particle system. We present a framework that allows particles to sample an isosurface in reference space, avoiding the costly inverse mapping of positions from world space when evaluating the basis functions. The distribution of particles across the reference space isosurface is controlled by geometric information from the world-space isosurface such as the surface gradient and curvature. The resulting particle distributions can be distributed evenly or adapted to accommodate world-space surface features. This provides compact, efficient, and accurate isosurface representations of these challenging data sets.

Index Terms—Particle systems, high-order finite elements, isosurface visualization.

1 Introduction

The method of finite elements [14] is a common spatial subdivision scheme used by scientists and engineers to reduce large simulation domains to sets of small subdomains over which physical simulations can be computed robustly and efficiently. Although traditional finite element methods utilize only low-order linear basis functions for representing data over the elements, they provide considerable flexibility for handling complex geometries. The geometric flexibility is aided by the transformation of individual elements constructed as identical cubes in *reference space* into unique *world-space* elements, which can have not only rectangular faces, but also triangular faces. In world space, the spatial extent of each element is defined by characteristics of the domain and simulation, such as boundary conditions and features of interest. The mapping functions responsible for the transformations can distort the elements by stretching, skewing, or even collapsing the faces of the reference space cubes, as illustrated in Fig. 1.

A number of researchers have developed methods to improve the convergence properties of finite elements through the use of high-order functions for the representation of the data, as well as the element transformations. Today, high-order finite element techniques have reached a level of sophistication such that they are commonly applied to a broad range of engineering problems [9], [17], [32]. Although there exist some high-order finite element methods that do not rely on reference space transformations, the use of curvilinear coordinate transformations is of increasing interest [15]. This paper is addressing the problem of finite element methods that rely on higher order (higher than linear) basis functions for the solutions, as well as the coordinate transformations.

Conventional approaches to finite element isosurface visualization assume that linear data representations can be adapted to accommodate low-order finite elements. This strategy, however, faces a number of challenges when considering high-order data sets. First, the data must be finely subsampled to ensure that features are adequately captured with linear approximation schemes. Second, there is, in general, no closed form expression for the inverse of high-order mapping functions. Numerical inversion schemes are required to transform world-space locations into the reference space when sampling the data, creating a nested-root-finding problem when locating an isosurface. Furthermore, determining which reference-space element in which to invert a particular point in world space adds to the computation.

Computational scientists who wish to visualize high-order finite element solutions will require visualization algorithms that are flexible enough to accommodate these constraints. These algorithms will need to have variable degrees of freedom so that users can easily control the trade-off between visualization quality and speed. For efficiency, these computations must be locally adaptive, allowing computational power to be applied to regions of the solutions that exhibit the most complexity (that is, *h-r adaptivity* in finite element terms). Furthermore, these algorithms will need to achieve the appropriate balance of computations in world space, where the metrics for adaptivity are defined, and reference space, where there are closed-form expressions for the

• *The authors are with the Scientific Computing and Imaging Institute, University of Utah, 50 S. Central Campus Drive, Room 3490, Salt Lake City, UT 84103. E-mail: {miriah, bnelson, kirby, whitaker}@sci.utah.edu.*

Manuscript received 8 Sept. 2006; revised 23 Jan. 2007; accepted 21 Feb. 2007; published online 20 Mar. 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0162-0906.
Digital Object Identifier no. 10.1109/TVCG.2007.1048.

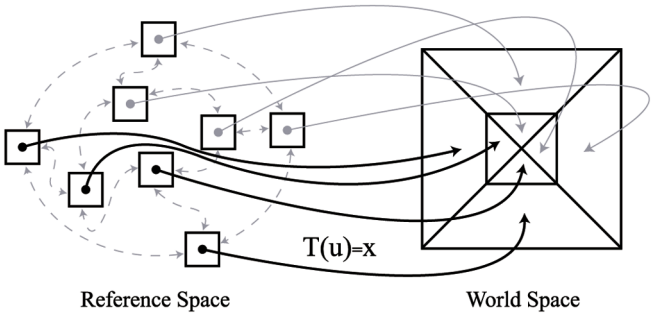


Fig. 1. A 2D schematic of the finite element subdivision scheme. In *reference space*, elements are defined as identical squares. These squares are transformed into *world space* by a mapping function T that can stretch, skew, shrink, or even collapse edges.

associated geometric quantities. To address these issues, we are proposing an isosurface visualization technique that relies on a particle system, exhibited in Fig. 2. The particles are constrained to an isosurface and exert repulsive forces on each other, resulting in even distributions across the surface.

Previously, we showed how these types of system can be made robust and controllable [21]. Efficient visualization of high-order finite element data, however, requires some novel extensions of the particle system framework. First, to reduce the number of repulsive force computations between particles, we have redefined how adaptivity is added into the system. We allow particles to maintain individual, changeable radii, and adapt interparticle forces by scaling the distance between particles based on the particles' radii. This mechanism maintains a limited number of influential neighbors around a particle, even in areas of high adaptivity. The adaptivity is based on curvature of the isosurface in world space, which is a function of not only the high-order basis functions, but also the mapping function. Our second contribution is the derivation of isosurface curvature in the presence of curvilinear coordinate transformations, including the reduction of the isosurface Hessian from a rank-three tensor contraction into a series of standard vector-matrix computations. Third, we manipulate the particle positions *in reference space* to avoid a numerical inversion of the coordinate transformation while computing the particle interactions and adaptivity in world coordinates.

The resulting particle system allows for a series of *forward* computations to obtain desirable distributions of samples over the world-space surface. The proposed modifications, along with several optimization strategies, create a system that produces accurate and compact point samples of finite element isosurfaces. The resulting distribution of particles—a process that may take anywhere from a few seconds to minutes—can be rendered interactively as either simple point sprites or a water-tight *splat* surface on the graphics processing unit (GPU), allowing a scientist or engineer to quickly explore their data from any camera location. Furthermore, the generality of this system can be broadly applied to any type of data representation that makes use of a reference space and a mapping function.

2 Background

Until recently, much of the work in finite-element simulations has focused on linear elements. In the simplest case where the finite elements form a regular grid in world space, conventional methods like marching cubes [19] and direct volume rendering [20] are applicable. In general, however, the finite elements produce an irregular grid in world space that is incompatible with the assumptions these methods make about the regularity of the grid. Early work by Shirley and Tuchman [30] and Williams [39] propose a volume rendering approach for tetrahedral elements, and Bunyk et al. [3] propose a generalized ray-casting algorithm for irregular grids. Doi and Koide [10] present the Marching Tetrahedra method for triangulating isosurfaces defined over tetrahedral elements. More recently, work has moved the volume rendering [36], [4] and isosurface generation [25], [23] algorithms onto the GPU to obtain faster rendering speeds.

Applying these low-order linear methods to high-order finite elements, however, presents several challenges. First, high-order basis functions represent features in the data with far fewer grid elements than an equivalent low-order representation. Thus, visualization methods that rely on linear interpolation must first finely subdivide the domain to ensure that features in the data are not missed. This increase in grid resolution can have an explosive effect on not only the storage requirements for the visualization, but also on the computation required to sufficiently sample the elements.

The second problem stems from the need to compute an inverse of the mapping function to evaluate the data in the

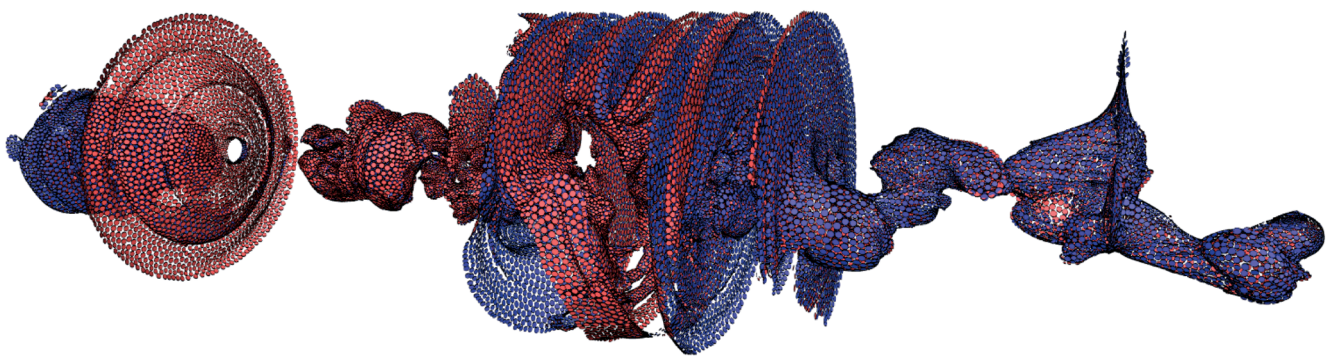


Fig. 2. An isosurface of a finite element fluid simulation pressure field sampled with a particle system. The color indicates the relative direction of the surface normal at the particle (blue indicates *outward* and red indicates *inward*).

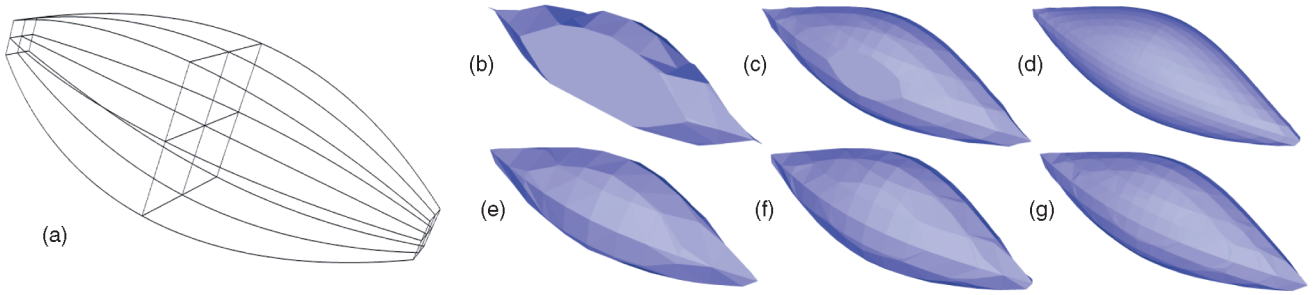


Fig. 3. Marching cubes surfaces of a sphere mapped through quadratic b-spline functions: (a) the transformed elements, (b) $5 \times 5 \times 13$ grid after one level of adaptive subdivision, requiring 6.7 seconds and 0.6 million forward evaluations, (c) $9 \times 9 \times 25$ grid after two levels of adaptive subdivision, requiring 25 seconds and 2.4 million forward evaluations, (d) $17 \times 17 \times 49$ grid after three levels of adaptive subdivision, requiring 82 seconds and 7.7 million forward evaluations, (e) $5 \times 5 \times 13$ grid with Newton-Rhapson root finding, requiring 19 seconds and 1.7 million forward evaluations, (f) $7 \times 7 \times 17$ grid with Newton-Rhapson root finding, requiring 35 seconds and 3.2 million forward evaluations, and (g) $9 \times 9 \times 25$ grid with Newton-Rhapson root finding, requiring 73 seconds and 6.8 million forward evaluations.

world space. Let $F^*(\mathbf{u})$ be the functional representation of the finite element solution, which is defined in the reference space, and let T be the coordinate transformation that maps a reference space point $\mathbf{u}$ into a world-space point $\mathbf{x}$, that is, $T(\mathbf{u}) = \mathbf{x}$. The world-space representation of the solution is therefore $F(\mathbf{x}) = F^*(T^{-1}(\mathbf{x}))$. There is generally no closed form expression for T^{-1}, causing world-space evaluations to require iterative numerical schemes for the computation of $\mathbf{u} = T^{-1}(\mathbf{x})$. Thus, determining the location of isosurfaces in high-order finite element data becomes a nested-root-finding problem—$F(\mathbf{x})$ (and its derivatives) must be iteratively evaluated to determine the position of the isosurface, with each evaluation of $F(\mathbf{x})$ requiring an iterative, numerical inversion of T.

There is yet one more challenge for the general problem of visualizing high-order finite elements. The data and coordinate transformations are valid for only a single element and, in practice, another layer of computation is required to determine which reference element contains the point $\mathbf{u} = T^{-1}(\mathbf{x})$. Although efficient element lookups based on regular grids [27], [11] or low-order curves [38] can be applied to elements with planar or quadratic curved faces respectively, there is no closed form solution for the general problem. Spatial partitioning schemes can be utilized to reduce the grid ambiguity to among a few elements [22], but the inverse mapping $\mathbf{u} = T^{-1}(\mathbf{x})$ will (generally) require multiple iterations across multiple elements. The problem is becoming increasingly more difficult as results from the scientific computing literature extend the finite element methodology to more general frameworks. For instance, recent work by Hughes et al. [15] proposes spline-based functions for finite elements, which produces yet another class of curvilinear mappings between the reference and world domains.

Adapting marching cubes to accommodate high-order finite elements elucidates these three challenges. In Fig. 3, we present the results of the isosurface extraction technique applied to a sphere that is transformed through a $2 \times 2 \times 2$ set of quadratic b-spline functions. To generate these results, a regular world-space grid is first created. The world-space location of each grid node is numerically inverted within each potential element until the associated reference space location is determined, and the basis functions can be evaluated. Accurately finding the zeros of the high-order data along the grid is then accomplished via a root-trapping mechanism.

We have incorporated two different root-trapping methods into a marching cubes framework. The first is a grid refinement strategy that uses an adaptive subdivision scheme to be as efficient as possible, recursively subdividing only the grid cells that contain zero crossings. Care has been taken in the implementation of the subdivision scheme to ensure coherence across neighboring cells, avoiding redundant sampling of the data. The second root-trapping approach uses the Newton-Raphson method along grid edges to determine the zero crossings.

Ensuring that point samples of the isosurface—for any sampling scheme—lie on the surface to within a small error tolerance is important for generating accurate surface approximations. In Figs. 3b, 3c, and 3d, the vertex locations are computed using linear interpolation over progressively more refined grids. These results indicate that using low-order interpolation schemes requires a very finely subdivided grid to accurately determine the zeros of the data and capture the geometry of the surface (Fig. 3d). In Figs. 3e, 3f, and 3g, the vertex locations are computed using a Newton-Raphson root-finding method. Although the grids in these images are relatively coarse, the zeros of the data are more accurately computed, generating more precise approximations of the surface.

Although Fig. 3 illustrates that capturing the geometry of high-order data is possible with low-order schemes, the results come at the cost of lengthy compute times. The computations are dominated by the large number of mapping function evaluations, which we call *forward evaluations*. The surfaces in Figs. 3 require a numerical inversion of each grid node, and the surfaces in Figs. 3e, 3f, and 3g also incur nested root-finding evaluations along the grid edges. On a P4 3.2-GHz CPU with 2.0 Gbytes of memory, Fig. 3b requires 6.7 seconds and 0.6 million forward evaluations, Fig. 3c requires 25 seconds and 2.4 million forward evaluations, Fig. 3d requires 82 seconds and 7.7 million forward evaluations, Fig. 3e requires 19 seconds and 1.7 million forward evaluations, Fig. 3f requires 35 seconds and 3.2 million forward evaluations, and Fig. 3g requires 73 seconds and 6.8 million forward evaluations. It is interesting to note that Figs. 3e, 3f, and 3g out perform Fig. 3d, indicating that root-finding along

coarse grid edges is more efficient than linearly interpolating along refined grid edges.

Other researchers have also noted the challenges of efficiently adapting low-order visualization methods to high-order functions, and some work has been done to specifically address the problem of visualizing high-order finite element data. Wiley et al. [38], [37] formulate ray casting for curved-quadratic elements, and Brasher and Haimes [2] propose a GPU-based method for color mapping cut planes of quadratic and cubic elements. A method to subdivide elements containing high-order basis functions so that low-order visualizing methods can be used is proposed [28], [35]. Coppola et al. [6] address the issue of vector visualization with high-order representations by formulating the particle advection problem on high-order basis functions. Similar to our framework, this work tracks particle advection in the reference space to avoid the inverse mapping problem. More recently, Nelson and Kirby [22] present an algorithm for ray tracing high-order spectral/*hp* elements. Their method uses a world-space approximation of the composition of the coordinate transformation and the reference space basis functions. It assumes multilinear mappings (linear element boundaries in world-space) and includes a quantification of the approximation and root-finding error. They show that the image-space method compares favorably with marching cubes in compute time when the tolerances on surface position are sufficiently high.

This work proposes the use of a particle system to sample high-order finite element data. The literature on using particle systems to represent and manipulate surfaces spans more than a decade—here, we give only a brief summary. The oriented particle system of Szeliski et al. [34], [33] builds deformable surfaces with particle dynamics modeled from the molecular dynamics literature. De Figueiredo et al. [8] propose a polygonalization method that models particle interactions with a mass-spring system. Building on these ideas, Witkin and Heckbert [40] present a novel physically-based system that uses repulsive particles to evenly sample implicit surfaces for modeling and visualization. Several works [13], [12], [31] describe modifications and applications of this method. Research on geometric adaptivity in such particle systems [26], [7], [16] has shown that the underlying formulation has limitations in this regard. Recent work by Meyer et al. [21] describes a new class of potential functions that provide more control over particle density. This paper utilizes potential functions of the type described in [21] to sample and render isosurfaces in high-order finite element data.

3 Particle System Formulation

In this section, we provide the mathematic framework for distributing particles adaptively across a finite element isosurface. Our notation is given as follows: boldface variables denote column vectors such as $\mathbf{x} = [x\ y\ z]^{\mathrm{T}}$; boldface subscripts denote partial derivatives of the function with respect to each component of the subscripted column vector such as $F_{\mathbf{x}} = [\frac{\partial F}{\partial x}\ \frac{\partial F}{\partial y}\ \frac{\partial F}{\partial z}]^{\mathrm{T}}$; $\mathbf{x}_i$ specifies the position of the ith particle, and other nonbold subscripts denote the evaluation of a scalar function at a specific particle's location such as E_i. We stray from this convention only in (10), (11), (12), (13), (14), (15), (16), (17), and (18), where we present the Einstein notation convention for our derivations of the world-space geometry.

3.1 Particle Potentials

To distribute the particles across an isosurface, we use the framework proposed in [21], which builds on the work of Witkin and Heckbert [40]. In this framework, a set of n particles are first constrained to a level set (isosurface) of the function $(F(\mathbf{x}_i) = C)$ using a Newton-Raphson approximation scheme for finding the roots of F, where C is the isovalue for the surface of interest. For the rest of the discussion, we assume, for simplicity, that $C = 0$. The particle positions $\mathbf{x}_i$ are iteratively refined until all the particles lie within an error threshold ϵ_T of the surface:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - F(\mathbf{x}_i)\frac{F_{\mathbf{x}}(\mathbf{x}_i)}{F_{\mathbf{x}}^{\top}(\mathbf{x}_i)\ F_{\mathbf{x}}(\mathbf{x}_i)}, \tag{1}$$

where $F_{\mathbf{x}}(\mathbf{x}_i)$ is the gradient of the implicit function at $\mathbf{x}_i$.

For each particle on the surface, we associate a compact potential energy kernel, which decreases monotonically with the distance from the particle—the results in this paper use a modified cotangent energy function [21]. These kernels give rise to a computation of the energy at a particle, E_i, based on the euclidean distances to the m neighboring particles:

$$E_i = \sum_{j=1, j\neq i}^{m} E_{ij} = \sum_{j=1, j\neq i}^{m} E\left(\frac{|\mathbf{r}_{ij}|}{\sigma}\right), \tag{2}$$

where $\mathbf{r}_{ij} = \mathbf{x}_i - \mathbf{x}_j$, and σ defines the extent of the kernel such that when $|\mathbf{r}_{ij}| > \sigma$, $E_{ij} = 0$. Euclidean distance is a fast approximation to the more accurate geodesic distance and fails to cull spatially close neighbors that lie on adjacent surfaces. To account for this problem, we discard potential neighbors with normals that are more than a 90-degree difference from the normal of the particle at $\mathbf{x}_i$.

The derivative of the energy at a particle with respect to the particle's position results in a repulsive force that defines the velocity, $\mathbf{v}_i$, that moves the particle to a locally lower energy state:

$$\mathbf{v}_i = -\frac{\partial E_i}{\partial \mathbf{x}_i} = -\sum_{j=1, j\neq i}^{m} \frac{\partial E_{ij}}{\partial |\mathbf{r}_{ij}|}\frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|}. \tag{3}$$

The particle positions are then updated using the projection of the repulsive velocity onto the local tangent plane:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \left(I - \frac{F_{\mathbf{x}}(\mathbf{x}_i)\ F_{\mathbf{x}}^{\top}(\mathbf{x}_i)}{F_{\mathbf{x}}^{\top}(\mathbf{x}_i)\ F_{\mathbf{x}}(\mathbf{x}_i)}\right)\mathbf{v}_i, \tag{4}$$

where I is the identity matrix. Equation (4) is the Lagrangian formulation of the constrained optimization that keeps particles on the zeroset of F. Movements in the tangent plane, however, can push particles off the surface, especially in areas of high curvature. Therefore, each update must be followed with a reprojection to the surface using (1) to ensure that the particles are within ϵ_T of the surface. By iteratively moving particles along the potential

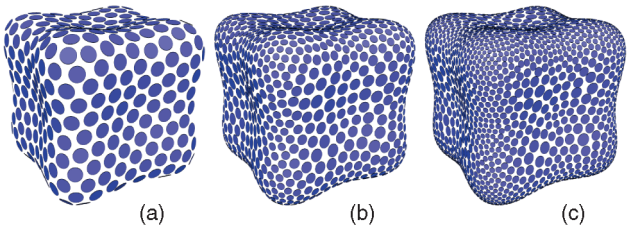


Fig. 4. The adaptivity of the particle system is modified with (a) $\rho = 0$, (b) $\rho = 7$, and (c) $\rho = 15$.

energy gradients, the system distributes particles evenly across the surface.

To increase the efficiency of the system when adapting the distributions, we have modified the framework to scale the *distance* between particles based on the local geometry of the surface, instead of scaling the energy and force functions. For instance, if we allow surface curvature to increase the effective distance between particles, areas of high curvature will have a higher density of particles. This adaptivity mechanism maintains a small number of influential neighbors around a particle even in regions of high curvature—a distinct change in the particle system framework—which allows for more effective optimization strategies (to be discussed in Section 4.1). For results in this paper, $\sigma = 1$, and the vector between two particles is

$$\mathbf{r}_{ij} = \alpha_{ij}(\mathbf{x}_i - \mathbf{x}_j) = -\mathbf{r}_{ji}. \tag{5}$$

With this formulation, when $|\mathbf{r}_{ij}| > 1$, we have $E_{ij} = 0$, and $|\mathbf{v}_{ij}| = 0$. Equation (5) also ensures that the energy and force between two particles is symmetric, which is important for stability in the system. The scaling factor α could be defined as a function based on any geometric property of the surface, and we have developed a formulation based on the curvature magnitude, D (root sum of squares of the principle curvatures):

$$\alpha_{ij} = \alpha_{ji} = \frac{1 + \rho D_{ij}\left(\frac{s}{2\pi}\right)}{s\beta}, \tag{6}$$

where s and ρ are user-defined variables that specify the distance between particles on a planar surface and the density of particles per unit angle over a curved surface, respectively, and D_{ij} is the average of the curvature magnitudes at $\mathbf{x}_i$ and $\mathbf{x}_j$. The principle curvatures are computed analytically from the gradient and Hessian of the implicit function, as described in [18]. The parameter $\beta = 0.5/\cos(\frac{\pi}{6})$ is derived from an ideal hexagonal packing of particles where the region of influence of the energy kernel at the center particle ends at the two ring of neighbors.

For surfaces that contain singularities, the curvature magnitudes can become arbitrarily large at critical points and cause extremely high densities of particles. To curb this effect, a maximal bound can be placed on α_{ij}. This constraint still allows particles to get very close to singularities, but does not guarantee that the critical point will be exactly sampled.

To illustrate the adaptivity of the particle system, Fig. 4 presents three examples of particle distributions over a quartic implicit function with varying values of the angular density parameter, ρ.

3.2 Isosurface Geometry in Finite Elements

To adaptively distribute particles across a finite element isosurface, we must formulate the gradient and Hessian of the world-space implicit function, F, in terms of the reference space specifications that are given by the finite element basis and mapping functions. As mentioned in Section 2, the implicit function representing the simulation data, F^* is defined over a set of finite elements in reference space and is transformed into world space through a mapping function $T(\mathbf{u}) = \mathbf{x}$. The gradient and Hessian of the world-space implicit function are thus defined not only by F^*, but also by T and, therefore, T must be included in all of the derivative calculations.

Used in the computation of the world-space gradient and Hessian is the Jacobian of the mapping function, which describes how the space around a reference space position is stretched or squashed by the mapping function:

$$J(\mathbf{u}) = \frac{\partial T(\mathbf{u})}{\partial \mathbf{u}} = \frac{\partial \mathbf{x}}{\partial \mathbf{u}} = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial w} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial w} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial w} \end{pmatrix}. \tag{7}$$

Also, to simplify the following derivations, we denote the inverse of the Jacobian as

$$K = J^{-1} \tag{8}$$

and provide a linear algebra identity for a matrix $\mathbf{M}$:

$$\frac{d\mathbf{M}^{-1}}{d\mathbf{z}} = -\mathbf{M}^{-1}\frac{d\mathbf{M}}{d\mathbf{z}}\mathbf{M}^{-1}, \tag{9}$$

where $\mathbf{z}$ denotes some Cartesian basis vector.

Formulating the expressions for the world-space gradient and Hessian requires meticulous derivations of the derivatives of the coordinate transformation, T. Care must be taken to correctly determine the order of the component multiplications, as well as which vectors and matrices need to be transposed. Furthermore, the derivation of the Hessian includes a vector multiplication with a rank-three tensor, which is the result of computing the second derivative of the vector-valued coordinate transformation. To clarify these derivations, we use the Einstein notation convention, as described in [1]. Developed for dealing with curved spaces in physics, Einstein notation identifies relationships often hidden by conventional linear algebra notation such as transposition and order of operations.

To begin, we define the world space as X, the reference space as U, and the mapping function as $T : U \mapsto X$; let $\mathbf{x} \in X$ and $\mathbf{u} \in U$. Using the Einstein notation, we define $\mathbf{x} = x^i$ and $\mathbf{u} = u_i$, utilizing upper indices for world-space components and lower indices for reference space components. The number of indices of a variable indicates the rank of the tensor—v_i is a rank-one tensor (or vector), M_{ij} is a rank-two tensor (or matrix), T_{ijk} is a rank-three tensor, and so on. In this convention, repeated indices in a term indicates a summation over the range of index values. For example,

$$\underbrace{\mathbf{a}^\top \cdot \mathbf{b}}_{vector} = \underbrace{\sum_i a_i b_i}_{summation} = \underbrace{a_i b_i}_{Einstein}. \tag{10}$$

Parenthesis are also dropped on functional variables such as from the basis functions and the mapping function:

$$F(\mathbf{x}) = F^i x^i, \tag{11}$$

$$F^*(\mathbf{u}) = F_i^* u_i, \tag{12}$$

$$T(\mathbf{u}) = T_j^i u_j = x^i. \tag{13}$$

For brevity, however, we will notate (11) and (12) as simply F and F^*, respectively, emphasizing the role of the equations as scalar functions throughout the derivations. Finally, Einstein notation also uses indices to represent partial derivatives such as for the Jacobian of the mapping function:

$$J(\mathbf{u}) = \frac{\partial}{\partial u_j} T_k^i u_k = \frac{\partial x^i}{\partial u_j} = J_j^i. \tag{14}$$

The inverse of the Jacobian is then

$$K_j^i = (J_j^i)^{-1}. \tag{15}$$

The world-space gradient, F^i, is

$$F^i = \frac{\partial F}{\partial x^i} = \frac{\partial F^*}{\partial u_j}\frac{\partial u_j}{\partial x^i} = F_j^* K_j^i. \tag{16}$$

The expression of the Hessian includes the derivative of the inverse Jacobian which, using (9), is written

$$K_{jk}^i = \frac{\partial (J_j^i)^{-1}}{\partial u_k} = -K_j^l J_{mk}^l K_m^i. \tag{17}$$

Notice that the second derivative of the mapping function, J_{mk}^l, is a rank-three tensor. Using (16) and (17), the Hessian, F^{ij}, is defined as

$$\frac{\partial}{\partial x^i}\left(\frac{\partial F}{\partial x^j}\right) = \frac{\partial}{\partial u_k}\left(\frac{\partial F}{\partial x^j}\frac{\partial u_k}{\partial x^i}\right) = F_{lk}^* K_l^j K_k^i - F_l^* K_l^m J_{kn}^m K_n^j K_k^i. \tag{18}$$

By carefully matching Einstein notation indices of each component in (16) and (18), the gradient and Hessian expressions are reduced to a series of standard vector-matrix operations. Introducing $[\cdot|\cdot|\cdot]$ to express the concatenation of three column vectors into a matrix, the expressions for the world-space gradient and Hessian in standard notation are

$$F_{\mathbf{x}} = K^\top F_{\mathbf{u}}^* \tag{19}$$

$$F_{\mathbf{xx}} = (F_{\mathbf{uu}}^* K)^\top K - \left(\left[F_{\mathbf{x}}^\top \frac{\partial J}{\partial u} | F_{\mathbf{x}}^\top \frac{\partial J}{\partial v} | F_{\mathbf{x}}^\top \frac{\partial J}{\partial w}\right] K\right)^\top K. \tag{20}$$

The expressions conveyed in (19) and (20) allow us to achieve world-space adaptivity using reference space evaluations of the basis functions, mapping functions, and their derivatives, along with any standard vector-matrix library. This formulation has a general applicability and could, for instance, be used to add curvature dependencies in other applications that make use of reference space regularity such as mesh generation [29].

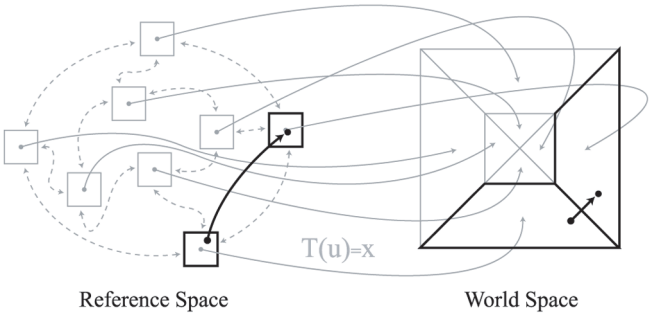


Fig. 5. Particles move from element to element by utilizing neighboring element information (dashed lines) and linear interpolation of the reference space element vertices to determine the coordinates of the particle in the neighboring element.

3.3 Reference Space Particles

To avoid the computations associated with T^{-1}, we have developed a strategy that maintains particle positions in reference space while controlling the particle distributions with geometric information from the world space. This strategy is similar to the guided search algorithm proposed by Coppola et al. [6] for particle advection. Maintaining reference space positions allows us to sample the basis functions that define F^* using closed form expressions. Thus, the curvature calculations and projection of particles onto the isosurface (1) can be computed directly. We ensure that the particle distributions are even and adaptive in the world space by using world-space positions (forward mapping) when computing the repulsive force velocities in (3). We then transform these velocities via the derivative of the mapping function to obtain an approximate reference space velocity, $\mathbf{v}_i^*$:

$$\mathbf{v}_i^* = K\mathbf{v}_i. \tag{21}$$

The reference space positions are then updated using (4) with reference space components. We note that (21) is a first-order approximation and thus assumes updates that are small relative to the isosurface and the curvature of the coordinate transformation. We are not interested in the precise motion of particles but rather that they move to lower energy states; thus, we adapt the time steps to accommodate this first-order approximation, as will be described in Section 4.

To effectively distribute particles in the reference domain, particles must also be able to move from element to element (Fig. 5). Finite element data sets typically contain reference space information describing which elements abut one another in the world space, as shown by the dashed lines in Fig. 5. Because reference space elements are identical cubes, we can easily determine when a particle's positional update causes it to leave an element, and we use the adjacency information to compute the particle's new element and new position within that element. When a particle moves from one element into the next, we use linear interpolation—based upon the reference space coordinates of the adjacent elements' shared world-space vertices—to determine the particle's location in the neighboring element's local coordinate system.

4 Implementation

For all of the results presented in this paper, we initialized the system with several hundred particles at random locations, evenly distributed among all finite elements. The s and ρ parameters are user-defined, where s is in world-space units. Also specified is an error tolerance, ϵ_T, for projecting the particles onto the isosurface—all of the images in this paper were generated using $\epsilon_T = 10^{-6}$. The particles maintain reference space positions and corresponding world-space positions, as well as a time step parameter, λ_i, that is used in an adaptive gradient descent algorithm described below.

First, the particles are projected onto the isosurface by performing several iterations of (1). We have found that five iterations is usually sufficient for moving particles to within ϵ_T of the isovalue. Next, the system iterates using a Gauss-Seidel updating scheme until the particle distribution converges by repeating the following steps:

1. For each particle:
   a. Compute $\mathbf{v}_i$ and the current E_i by summing the repulsive forces and energies of all the neighboring particles in world space (see Section 4.1 for an optimization of this step). Transform the velocity into reference space with (21).
   b. Temporarily move the particle in reference space with (4) using $\mathbf{v}_i^* = \lambda_i \mathbf{v}_i^*$, then reproject back onto the surface with (1). If the new position is outside of the element, determine which element the new position is in and convert the position into the new local coordinate system.
   c. Compute E_i^{new} at the new location (by mapping the new position into world space with T). If $|\mathbf{x}_i - \mathbf{x}_i^{new}| > s$, or $E_i^{new} > E_i$, or $F_i^{new} > \epsilon_T$, or $\mathbf{x}_i^{new}$ is outside of the simulation domain, set $\lambda_i = 0.1\lambda_i$ and return to Step 1b. Otherwise, update the particle's position, and if this is the first time through Step 1, set $\lambda_i = 10\lambda_i$.
2. Decide whether the system is at a steady state. There are numerous metrics to determine steady state, and we have chosen to use the difference of the system energy (the sum of the energy at all the particles) from one iteration to the next. When the system energy difference is less than a small fraction of the total energy (we use 0.15 percent for the results presented in this paper, although a range of values would produce similar results), we deduce that particles have reached a steady state. Otherwise, repeat Step 1.
3. Check whether the configuration of particles is desirable. We compare each particle's energy against an ideal energy, E^{ideal}, which is defined by a hexagonal packing of neighbors on a flat surface with interparticle distances of s. We bias E_i with a random value on the interval [0, 1] to eliminate mass splitting or dying, then split particles with $E_i < 0.35E^{ideal}$, and delete particles with $E_i > 1.75E^{ideal}$. Alternatively, if a constant number of particles is desired, the planar separation variable s could be modified to move the system energy toward the ideal system energy. Although we have provided specific values for this step, we have found in practice that varying the values by up to 20 percent produces visually similar results, although with different convergence times.
4. If the energy of the particles is acceptable, stop iterating.

In our implementation of the proposed reference space sampling method, we eliminate divisions by zero by adding a small (machine precision) value to all denominators.

4.1 Computational Optimizations

We have implemented several strategies to increase the computational efficiency of the distribution process. First and foremost, any type of Lagrangian scheme suffers from an inherent lack of explicit spatial relationships. In the case of the particle system described in this paper, the problem manifests in the computation of repulsive forces from local particle interactions. Nominally, the particle-to-particle interaction problem is $O(N^2)$ for each iteration. The use of compact energy kernels, however, leaves the energy and force computations null for all but a small subset of neighboring particles. Thus, we have implemented a spatial binning structure [7] that lessens the subset of potential interactions. The size of the bins is based upon the maximum possible extent of the energy kernel (which, derived from (6), is $s\beta$), and each bin maintains a list of resident particles. Neighboring particle queries for computing forces and energies is then reduced to computing interactions with particles that reside in the $3 \times 3 \times 3$ block of bins surrounding the querying particle's bin. As such, every time a particle is moved in Step 1 above, the binning structure particle lists are updated.

Although the binning structure dramatically decreases the system convergence time, particles will still compute distances with many nonneighbor particles, especially as the adaptivity of the system is increased (that is, for large values, for ρ). By not allowing particles to move further than s in any one iteration (Step 1b), however, the neighborhood configurations change very little from iteration to iteration. Taking advantage of this observation, we store a list of neighbors with each particle that is updated only after several iterations (every five iterations for the results in this paper). Although the lists may be out of date for intermediate iterations, the iterative convergence method smooths out these errors. We have found that this approach is particularly effective as the system approaches a steady state and the particle neighborhoods become stable. Storing the lists of neighbors typically decreases the computation time by a factor of 2-3, and even more so with increasing adaptivity.

We have also observed that the particle systems quickly eliminate the high-frequency errors in their configurations by taking large movements during the first few iterations, followed by many iterations of small movements to eliminate the low-frequency errors. To exploit this trend, we have developed a hierarchical distribution mechanism. We first distribute coarse-level particles across the system using a large s value, then split each of these particles into four evenly spaced particles and reduce s by half. This process is repeated until s reaches the user defined value. We have found that an initial value of s based on the approximate size of the surface features works most effectively.

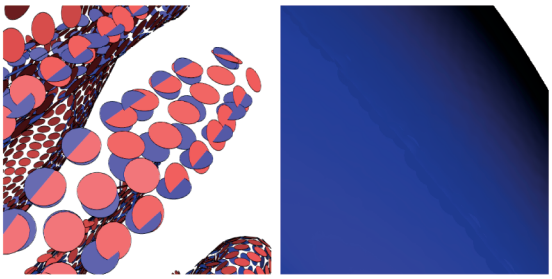


Fig. 6. Artifacts due to boundary discontinuities where (a) disks or (b) splats intersect each other.

4.2 Rendering

Rendering the particle systems with oriented disks provides the user with a visualization of accurate surface samples. The disks allow the user to infer the topology of the surface, aided by the ability to rotate and translate the surface in 3D. However, the oriented disks cannot express subtle shading cues or connectivity as effectively as rendering methods that generate watertight surface visualizations.

Over the last several years, work on point set surfaces has established points as a popular, and effective, geometric primitive. One common approach for rendering point sets is the splatting method, first introduced by Pfister et al. as *surfels* [24], with numerous extensions for improving the quality of the rendering [41], [42], as well as the speed [5]. Splatting entails an additive blend of oriented, alpha-channel Gaussian kernels at each point, followed by a normalization of each pixel's color to ensure an even intensity across the surface. The normals of the points can also be included in the projection and blended to allow for smooth shading of the surface. We have implemented the basic splatting algorithm on the GPU, achieving interactive rendering speeds. The adaptivity of the particle system allows the splat surfaces to appear smooth with relatively sparse point samples (sparse compared to typical point set surfaces that contain millions of points).

4.3 Boundary Discontinuities

Surfaces defined over finite elements are usually guaranteed to be only C^0 continuous at the element boundaries, allowing for cusps on a surface in areas that are analytically flat. This potentially results in lessened interparticle forces due to the tangent plane projection of a particle's neighborhood force field. The iterative nature of the convergence mechanism, however, smooths out this effect.

In practice, we have found the discontinuities present a problem only when rendering the particles. The particles' radii do not adapt to these discontinuous features, causing artifacts to sometimes appear when disks or splats intersect each other—Fig. 6 illustrates the phenomenon along boundaries. To more accurately capture these surface features, the particles can be adapted in (6) by not only the curvature computed from the Hessian, but also by the particles' proximity to element boundaries. This adaptation will not remove the visual presence of the cusps, but will make the discontinuities appear smoother.

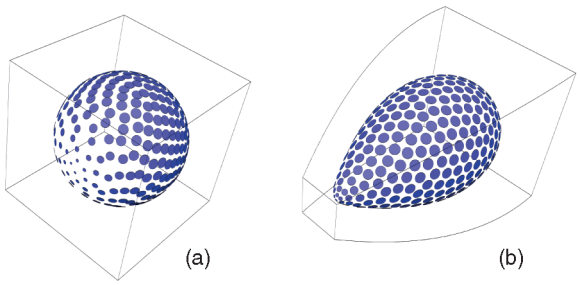


Fig. 7. (a) A sphere defined in reference space is mapped to (b) a teardrop in world space (b). The mapping function induces a curvature variation to the surface and the particles adapt accordingly.

5 Results and Discussion

We begin with a demonstration of the method on a curvilinear mapping of a simple implicit function. Fig. 7 shows a quadratic b-spline coordinate mapping function, which maps a reference space sphere into a world-space teardrop, introducing a curvature variation across the surface. Our mathematical formulation of the world-space curvature correctly accounts for the effects of the mapping function.

Fig. 8 demonstrates the particle system achieving an even distribution on a sphere in the world space, despite the irregular reference space geometry. Fig. 7 asserts that the system can correctly adapt to the world-space curvature introduced by the mapping function. All of these results are produced by manipulating the positions of particles in the reference space while computing interparticle distances and curvatures using world-space geometries.

The particle system visualizations in Fig. 9 are generated from the data set used in the marching cubes example shown in Fig. 3. These examples are provided as a comparison of what the particle system achieves in approximately the same amount of time as marching cubes with root finding. Using the same computing resources specified in Section 2, Fig. 9a requires 16 seconds and 1.1 million forward evaluations, Fig. 9b requires 36 seconds and 2.5 million forward evaluations, and Fig. 9c requires 70 seconds and 4.8 million forward evaluations. The particle system is able to capture the sharp tips of the surface faithfully due to the adaptivity mechanism, even at coarse resolutions—this is a significant difference from the polygonal surfaces. Furthermore, the splat renderings produce

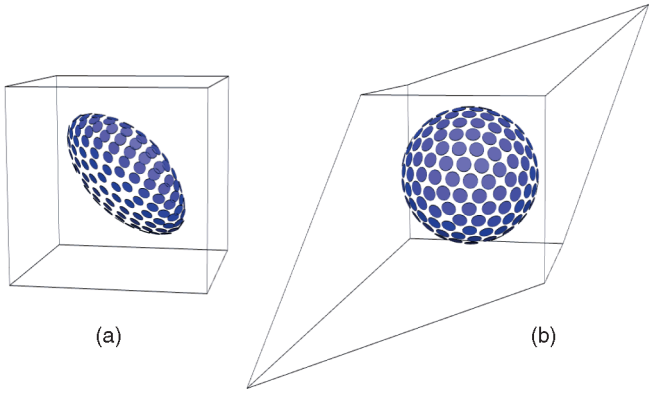


Fig. 8. (b) The particle system converges to an even distribution in world space regardless of (a) the shape of the surface in reference space.

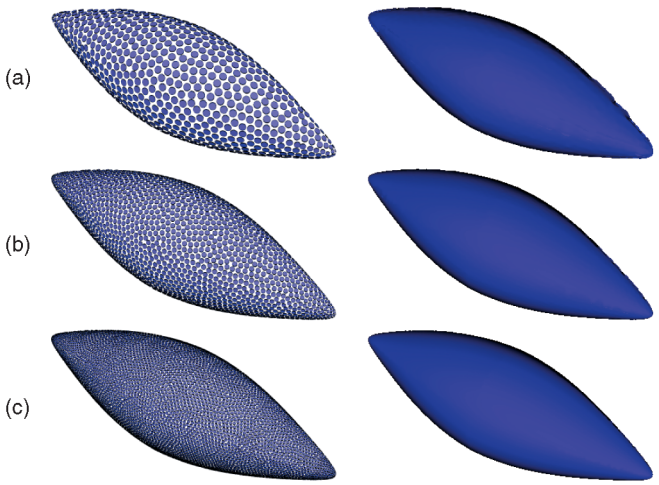


Fig. 9. The data set in Fig. 3 sampled with a particle system. The *right* column images are splat renderings of the particles in the *left* column: (a) 1,280 particles distributed in 16 seconds with 1.1 million forward evaluations, (b) 3,232 particles distributed in 36 seconds with 2.5 million forward evaluations, and (c) 8,647 particles distributed in 70 seconds with 4.8 million forward evaluations.

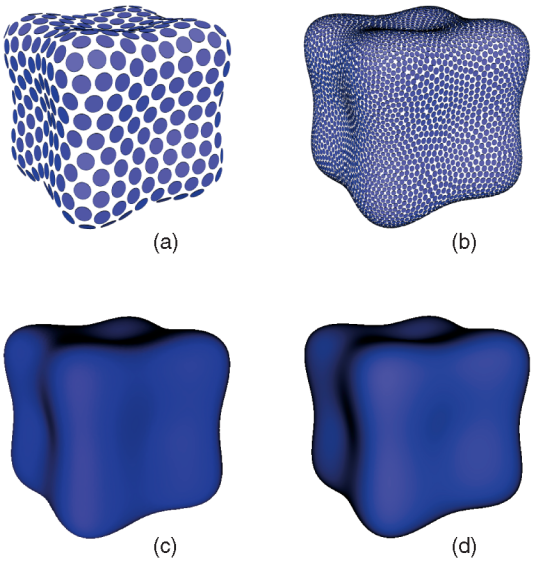


Fig. 10. The zeroset of an eighth-order implicit function defined within a single hexahedral element: (a) 500 particles, with a distribution time of 4 seconds, (b) 6,800 particles, with a distribution time of 3 minutes, and (c) a GPU-based splat rendering of the 6,800 particles that is visually indistinguishable from the 512 $\times$ 512 ray traced image in (d).

much smoother watertight surface approximations than the marching cubes results.

These results allow us to compare directly against marching cubes, elucidating the advantage of the particle system. First, notice that the particle system produces higher quality results in similar amounts of time with *fewer* forward evaluations (compare Figs. 3e, 3f, and 3g with Fig. 9). This is in part due to marching cubes evaluating grid nodes in regions where the surface does not exist but also due to the numerous iterations at each grid intersection point to determine the correct placement of the vertex, a chore that consumes over half of the compute time. In contrast, the particle system spends most computation time evaluating the data (by these numbers, about 70 percent) and little time in the overhead of the particle system, that is, computing particle-particle interactions. These results, like those described in the literature [22], demonstrate that generating *accurate* visualizations of high-order data is inherently expensive due to the cost of evaluating the high-order solutions and coordinate transformations. By simultaneously finding the roots of F and distributing the surface samples in a sensible way, the particle system makes very effective use of these costly evaluations.

The ray tracing method described in [22] generates nicely shaded and watertight images of high-order finite element isosurfaces but with the image-space drawbacks inherent to all ray tracing methods. Data exploration is computationally expensive because each viewpoint requires the isosurface to be resampled, and the computational cost is also associated with the resolution of the resultant image. Furthermore, the accuracy of the ray-isosurface intersections are related to the degree of the polynomial used to approximate the implicit function along the ray in world space. This relationship scales with computation time as p^2 to p^3, where p is the degree of the polynomial.

Conversely, the particle system allows users the freedom to explore the data by interactively moving a distributed set of particles in space. The accuracy of the particle positions with respect to the isosurface are controlled on a per particle basis by the error threshold, ϵ_T, and is independent of the sampling density of the particle system, unlike the ray tracing method. Thus, very coarse, fast distributions of particles will be guaranteed to lie within ϵ_T of the isosurface. The accuracy of the visualization produced by the particle system is instead related to the *inferred topology* of the isosurface by the viewer. Course level distributions provide insight to the gross geometry of the isosurface, whereas finer distributions provide increasingly more detailed representations of the underlying geometry. Accuracy thus relates to the amount of detail that can be visualized with a specific resolution of the particle system—a relationship that scales linearly with time.

Figs. 10a and 10b illustrate the results of two different resolutions of the particle system with 500 and 6,800 particles, respectively. In Fig. 10c, the particle distribution from Fig. 10b has been rendered with a GPU-based splat algorithm and is virtually indistinguishable from a ray traced image at 512$\times$ 512 resolution. Moreover, the isosurface in Fig. 10c can be rotated at interactive frame rates (greater than 30 frames per second). The GPU-based splat algorithm runs on an Nvidia GeForce 6800 GT card with Pixel Shader 3.0. The isosurfaces in Fig. 10 reside within a single hexahedral element and are the zeroset of an eighth-order polynomial implicit function. The 500 particles in Fig. 10a took 4 seconds to converge, and the 6,800 particles in Fig. 10b took 3 minutes to converge. The 512 $\times$ 512 ray traced image with a 25th-order reconstruction polynomial required 6 minutes to render. We note that the particle system implementation uses the same finite element evaluation code as the ray tracer, which is also the implementation used for the results in [22]. We have found that sampling the finite element implicit function takes, on average, an order of magnitude longer than computing the interparticle forces. This is consistent with our early observation, that basis-element evaluations dominate the computation.

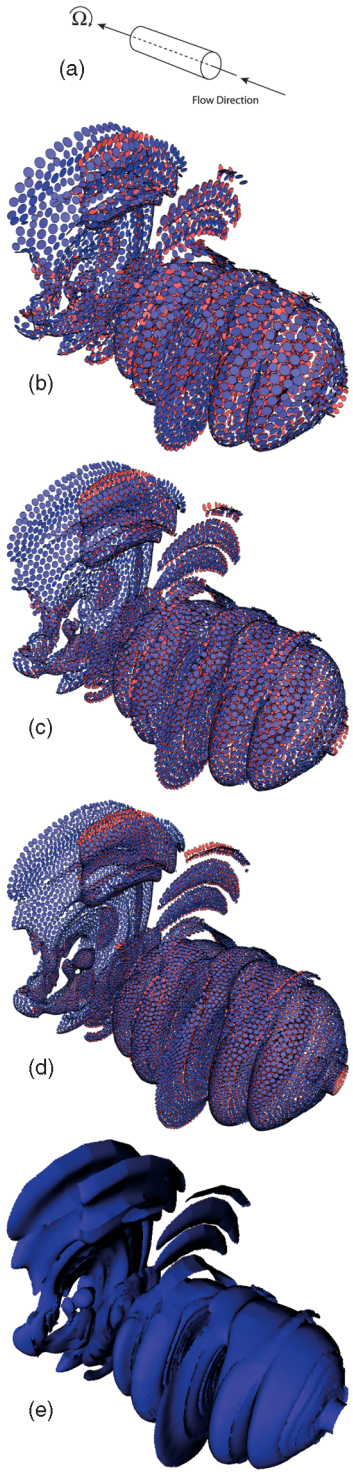


Fig. 11. The isosurface of pressure $C = 0$ for a CFD simulation over 5,736 elements with a third-order polynomial implicit function in each element: (a) schematic for the fluid simulation, (b) 5,000 particles, 55 seconds, (c) 13,000 particles, 3.4 minutes, (d) 28,000 particles, 15 minutes, and (e) 59,000 particles, 39 minutes.

We demonstrate the capabilities of our proposed particle system by visualizing pressure field isosurfaces of two computational fluid dynamics (CFD) simulations. In the first example, shown in Figs. 2, 11, and 12, we examine the wake of a rotating canister traveling through an incompressible fluid. The finite element mesh consists of 5,040 hexahedra and 696 prisms with the computational fluid mechanics problem being solved with third-order polynomials per element. In the second example, shown in Fig. 13, we visualize the flow past a block with an array of splitter plates placed downstream of the block. This example contains 3,360 hexahedra and 7,644 prisms, again with the computational fluid mechanics problem being solved with third-order polynomials per element. The color of the disks in Figs. 2, 11, and 13 indicates the relative direction of the surface normal at the particle (blue indicates *outward* and red indicates *inward*). In Fig. 12, color specifies the size of the particles.

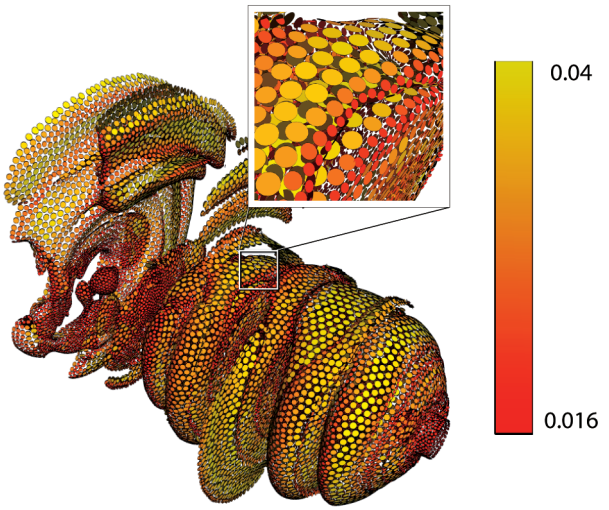


Fig. 12. The distribution from Fig. 11d, color-mapped based on the radii of the particles. The color bar provides the range of values of the radii.

6 Conclusions and Future Work

We have presented a general and robust method for visualizing isosurfaces of high-order finite element data sets that would allow scientists and engineers to efficiently explore simulation data. By sampling isosurfaces with a

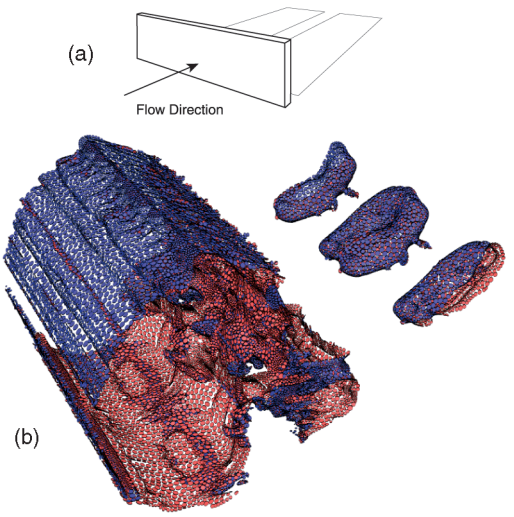


Fig. 13. The isosurface of pressure $C = -0.1$ for a CFD simulation over 11,004 elements with a third-order polynomial implicit function in each element: (a) schematic for the fluid simulation and (b) 43,000 particles, 25 minutes.

particle system, the method produces compact and adaptive visualizations that can be viewed at a variety of resolutions. Furthermore, the proposed system is general and easily adaptable to a broad range of finite element representations, from low-order linear elements to complex spline-based elements. We are planning to extend this work toward the generation of more accurate finite element meshes.

There are numerous avenues to improve the proposed system. Although we have presented several optimization strategies, more work can be done to increase the efficiency of the distribution process. First, the development of an adaptive spatial binning structure would reduce the number of potential neighbors at highly adaptive particles. Second, further exploitation of the system's tendency to quickly eliminate high-frequency error in the distribution process could be done by posing the system in a hierarchical framework such as multigrid. These advancements would help to enable visualization of dynamic data sets, an avenue of work we are interested in pursuing.

As mentioned in Section 3.2, the discontinuities in the derivatives at the element boundaries can cause features in the surface that cannot be analytically detected through computation of surface curvature. These undetectable features will contain a very sparse sampling of particles, which create artifacts when the particles are splat. One solution would be to implement a more sophisticated splatting algorithm that clips splats along these boundaries [43].

ACKNOWLEDGMENTS

The authors wish to thank Joe Kniss for the helpful discussions on GPU rendering techniques and Nathan Galli for the creation of the diagrams in this paper. The authors also thank the anonymous reviewers for their constructive suggestions. This research was supported by US National Science Foundation (NSF) Grant CCR-0310705, ARO Grant DAAD19-01-10013, ARO Grant W911NF-05-1-0395, and NIH NCRR Project 2-P41-RR12553-07.

REFERENCES

[1] R.L. Bishop and S.I. Goldberg, *Tensor Analysis on Manifolds.* Dover Publications, 1980.

[2] M. Brasher and R. Haimes, "Rendering Planar Cuts through Quadratic and Cubic Finite Elements," *Proc. Conf. Visualization (VIS '04),* pp. 409-416, 2004.

[3] P. Bunyk, A. Kaufman, and C.T. Silva, "Simple, Fast, and Robust Ray Casting of Irregular Grids," *Proc. Scientific Visualization Conf.,* pp. 30-36, 2000.

[4] S.P. Callahan, M. Ikits, J.L.D. Comba, and C.T. Silva, "Hardware-Assisted Visibility Ordering for Unstructured Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 11, no. 3, pp. 285-295, May/June 2005.

[5] W. Chen, L. Ren, M. Zwicker, and H. Pfister, "Ardware-Accelerated Adaptive EWA Volume Splatting," *Proc. IEEE Visualization Conf.,* Oct. 2004.

[6] G. Coppola, S.J. Sherwin, and J. Peiro, "Nonlinear Particle Tracking for High-Order Elements," *J. Computational Physics,* vol. 172, no. 1, pp. 356-386, 2001.

[7] P. Crossno and E. Angel, "Isosurface Extraction Using Particle Systems," *Proc. Eighth Conf. Visualization,* pp. 495-498, 1997.

[8] L. Henrique de Figueiredo, J. de Miranda Gomes, D. Terzopoulos, and L. Velho, "Physically-Based Methods for Polygonization of Implicit Surfaces," *Proc. Conf. Graphics Interface,* pp. 250-257, 1992.

[9] M.O. Deville, E.H. Mund, and P.F. Fischer, *High Order Methods for Incompressible Fluid Flow.* Cambridge Univ. Press, 2002.

[10] A. Doi and A. Koide, "An Efficient Method of Triangulating Equivalued Surfaces by Using Tetrahedral Cells," *IEICE Trans. Comm., Electronics, Information and Systems,* vol. E74, no. 1, pp. 214-224, Jan. 1991.

[11] A.S. Glassner, "Space Subdivision for Fast Ray Tracing," *IEEE Computer Graphics and Applications,* vol. 4, no. 10, pp. 15-22, 1984.

[12] J.C. Hart, E. Bachta, W. Jarosz, and T. Fleury, "Using Particles to Sample and Control More Complex Implicit Surfaces," *Proc. Shape Modeling Int'l Conf. (SMI '02),* p. 129, 2002.

[13] P. Heckbert, "Fast Surface Particle Repulsion," *Proc. ACM SIGGRAPH '97,* New Frontiers in Modeling and Texturing Course, pp. 95-114, Aug. 1997.

[14] T.J.R. Hughes, *The Finite Element Method.* Prentice Hall, 1987.

[15] T.J.R. Hughes, J.A. Cottrell, and Y.B. Azilevs, "Isogeometric Analysis: Cad, Finite Elements, Nurbs, Exact Geometry and Mesh Refinement," Technical Report 04-50, ICES, Univ. of Texas, Austin, Oct. 2004.

[16] T. Karkanis and A. James Stewart, "Curvature-Dependent Triangulation of Implicit Surfaces," *IEEE Computer Graphics and Applications,* vol. 22, no. 2, pp. 60-69, Mar. 2001.

[17] G. Em Karniadakis and S.J. Sherwin, *Spectral/HP Element Methods for CFD.* Oxford Univ. Press, 1999.

[18] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller, "Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications," *Proc. IEEE Visualization Conf.,* pp. 513-520, Oct. 2003.

[19] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proc. ACM SIGGRAPH '87,* pp. 163-169, 1987.

[20] N.L. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 1, no. 2, pp. 99-108, 1995.

[21] M.D. Meyer, P. Georgel, and R.T. Whitaker, "Robust Particle Systems for Curvature Dependent Sampling of Implicit Surfaces," *Proc. Int'l Conf. Shape Modeling and Applications (SMI '05),* pp. 124-133, June 2005.

[22] B. Nelson and M. Kirby, "Ray-Tracing Polymorphic Multi-Domain Spectral/*hp* Elements for Isosurface Rendering," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 1, pp. 114-125, 2006.

[23] V. Pascucci, "Isosurface Computation Made Simple: Hardware Acceleration, Adaptive Refinement and Tetrahedral Stripping," *Proc. IEEE TVCG Symp. Visualization,* pp. 293-300, 2004.

[24] H. Pfister, M. Zwicker, J. van Baar, and M. Gross, "Surfels: Surface Elements as Rendering Primitives," *Proc. ACM SIGGRAPH '00,* pp. 335-342, 2000.

[25] S. Roettger, M. Kraus, and T. Ertl, "Hardware-Accelerated Volume and Isosurface Rendering Based on Cell-Projection," *Proc. IEEE Conf. Visualization,* pp. 109-116, Oct. 2000.

[26] A. Rösch, M. Ruhl, and D. Saupe, "Interactive Visualization of Implicit Surfaces with Singularities," *Proc. Eurographics Computer Graphics Forum,* vol. 16, no. 5, pp. 295-306, 1996.

[27] S.M. Rubin and T. Whitted, "A 3-Dimensional Representation for Fast Rendering of Complex Scenes," *Proc. ACM SIGGRAPH '80,* pp. 110-116, 1980.

[28] W.J. Schroeder, F. Bertel, M. Malaterre, D. Thompson, P.P. Pébay, R.M. O'Bara, and S. Tendulkar, "Framework for Visualizing Higher-Order Basis Functions," *Proc. IEEE Conf. Visualization,* 2005.

[29] S.J. Sherwin and J. Peiro, "Mesh Generation in Curvilinear Domains Using High-Order Elements," *Int'l J. Numerical Methods in Eng.,* vol. 53, no. 1, pp. 207-223, 2002.

[30] P. Shirley and A. Tuchman, "A Polygonal Approximation to Direct Scalar Volume Rendering," *Proc. San Diego Workshop Volume Visualization,* vol. 24, no. 5, pp. 63-70, Nov. 1990.

[31] W.Y. Su and J.C. Hart, "A Programmable Particle System Framework for Shape Modeling," *Proc. Int'l Conf. Shape Modeling and Applications 2005 (SMI' 05),* pp. 114-123, 2005.

[32] B.A. Szabó and I. Babuška, *Finite Element Analysis.* John Wiley & Sons, 1991.

[33] R. Szeliski, D. Tonnesen, and D. Terzopoulos, "Modeling Surfaces of Arbitrary Topology with Dynamic Particles," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR '93),* pp. 140-152, June 1993.

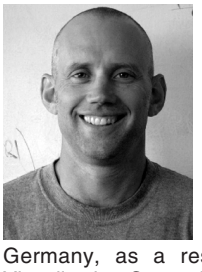
[34] R. Szeliski and D. Tonnesen, "Surface Modeling with Oriented Particle Systems," *Proc. 19th Ann. Conf. Computer Graphics and Interactive Techniques,* pp. 185-194, 1992.

[35] D.C. Thompson and P.P. Pebay, "Visualizing Higher Order Finite Elements: Final Report," Technical Report SAND2005-6999, Sandia Nat'l Laboratories, 2005.
[36] M. Weiler, M. Kraus, M. Merz, and T. Ertl, "Hardware-Based Ray Casting for Tetrahedral Meshes," *Proc. IEEE Conf. Visualization '03,* pp. 333-340, Oct. 2003.
[37] D.F. Wiley, H. Childs, B. Hamann, and K.I. Joy, "Ray Casting Curved-Quadratic Elements," *Proc. Eurographics/IEEE TCVG, ACM Siggraph Data Visualization 2004,* pp. 201-209, 2004.
[38] D.F. Wiley, H.R. Childs, B.F. Gregorski, B. Hamann, and K.I. Joy, "Contouring Curved Quadratic Elements," *Proc. Symp. Visualization (VisSym '03) Workshop Data Visualization,* 2003.
[39] P.L. Williams, "Visibility-Ordering Meshed Polyhedra," *ACM Trans. Graphics,* vol. 11, no. 2, pp. 103-126, Apr. 1992.
[40] A.P. Witkin and P.S. Heckbert, "Using Particles to Sample and Control Implicit Surfaces," *Proc. ACM SIGGRAPH '94,* pp. 269-277, 1994.
[41] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Surface Splatting," *Proc. ACM SIGGRAPH '01,* pp. 371-378, 2001.
[42] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "EWA Splatting," *IEEE Trans. Visualization and Computer Graphics,* vol. 8, no. 3, pp. 223-238, 2002.
[43] M. Zwicker, J. Rosnen, M. Botsch, C. Dachsbacher, and M. Pauly, "Perspective Accurate Splatting," *Proc. 2004 Conf. Graphics Interface (GI '04),* pp. 247-254, 2004.

Miriah Meyer received the BS degree in astronomy and astrophysics from the Pennsylvania State University in 1999. From 2000 to 2002, she worked as a software engineer at the Raytheon Corporation. Currently, she is a PhD student in the School of Computing, University of Utah, where she works in the Scientific Computing and Imaging Institute. Her research interests include visualization, scientific computing, and computer graphics. She is a student member of the IEEE.

Blake Nelson received the BS and MS degrees in computer science from the University of Utah. He is currently a PhD candidate in the Computer Science Department at the University of Utah. His research interests include graphics, scientific visualization, algorithms, and software architecture.

Robert M. Kirby received the ScM degrees in computer science and in applied mathematics and the PhD degree in applied mathematics from Brown University. He is an assistant professor of computer science at the University of Utah's School of Computing and is a member of the Scientific Computing and Imaging Institute at the university. His research interests lie in scientific computing and visualization. He is a member of the IEEE.



Ross Whitaker received the BS degree (summa cum laude) in electrical engineering and computer science from Princeton University in 1986 and the PhD degree in computer science in 1994 from the University of North Carolina (UNC)-Chapel Hill. At UNC, he received the Alumni Scholarship Award. From 1986 to 1988, he worked for the Boston Consulting Group. From 1994 to 1996, he worked at the European Computer-Industry Research Centre in Munich, Germany, as a research scientist in the User Interaction and Visualization Group. From 1996 to 2000, he was an assistant professor in the Department of Electrical Engineering, University of Tennessee. Since then, he has been at the University of Utah where he is an associate professor in the School of Computing and a faculty member of the Scientific Computing and Imaging Institute. He teaches image processing, computer vision, and scientific visualization. His research interests include computer vision, image processing, medical image analysis, surface modeling, and visualization. He is a member of IEEE.