

Isosurface Extraction Using Particle Systems

Patricia Crossno¹

Edward Angel²

Department of Computer Science, University of New Mexico

ABSTRACT

We present a new approach to isosurface extraction from volume data using particle systems. Particle behavior is dynamic and can be based on laws of physics or artificial rules. For isosurface extraction, we program particles to be attracted towards a specific surface value while simultaneously repelling adjacent particles. The repulsive forces are based on the curvature of the surface at that location. A birth-death process results in a denser concentration of particles in areas of high curvature and sparser populations in areas of lower curvature. The overall level of detail is controlled through a scaling factor that increases or decreases the repulsive forces of the particles. Once particles reach equilibrium, their locations are used as vertices in generating a triangular mesh of the surface.

Advantages of our approach include: vertex densities are based on surface features rather than on the sampling rate of the volume; a single scaling factor simplifies level of detail control; meshing is efficient because it uses neighbor information that has already been generated during the force calculations.

1. INTRODUCTION

Isosurface extraction produces a geometric model representing a subset of volume data. For a number of years, the marching cubes algorithm [2] has been the conventional method used to generate surface models from volume data. However, it suffers from the drawback that it generates a large number of triangles, many of which do not contribute relevant detail to the resulting model. Rendering time increases with model size. This problem has led to data reduction schemes such as triangle decimation [6], and retiling polygonal surfaces [8]. These algorithms seek to keep greater numbers of triangles in those regions of the model that have sharp edges or high degrees of curvature while reducing the number of triangles in flatter regions. Unfortunately, a great deal of work is expended in building up a polygonal model, much of which ends up being discarded.

We sought to develop a new approach to finding isosurfaces that would concentrate vertices in areas of high curvature as an intrinsic part of the model construction process. We chose to use particle systems as the basis for our new algorithm. The literature contains a number of relevant papers.

Szeliski and Tonneson combined deformable surface modeling and oriented particle systems to model free-form surfaces [7]. Figueiredo et al. modeled implicit surfaces using a particle repulsion approach combined with Delaunay triangulation [1]. Pang applied particle systems and behavioral animation to scientific visualization via the metaphor of a spray can as the user interface for "painting" volume data sets with different particle types [3][4]. Witkin and Heckbert used particle systems to display and interactively "sculpt" implicit surface models [9]. In their adaptive repulsion scheme, each particle has its own repulsion radius and decides individually whether to split or die.

Although our work was originally based on Witkin and Heckbert's, it differs from theirs in a number of respects. Like Figueiredo they modeled implicit surfaces for which they have analytically defined functions. We do not know the underlying functions that we are trying to model. We have only sampled data and must approximate function values and derivatives. We are limited by the domain of the volume data set and must constrain particles to remain within certain coordinates. We have differentiated the particles that move along volume boundaries

from those that move in the interior. Interactive manipulation of the model is not of interest, so we have no control points.

Witkin and Heckbert used a uniform distribution of particles and an $O(n^2)$ calculation in evaluating repulsion forces between particles. In our system, we have adapted Witkin and Heckbert's birth/death scheme to accommodate a nonuniform distribution of particles by having each particle seek and maintain its own desired repulsion radius, rather than some global value. The desired repulsion radius, and hence particle density, is based on local curvature estimates multiplied by a scaling factor. This enables us to easily control the level of detail in the model.

After a model is created, we can change the level of detail by adjusting the scaling factor and allowing the existing particles to redistribute themselves based on the new density level. We use a hybrid 3D bin and distance-ordered list scheme to reduce the repulsion force calculations to a local neighborhood around each particle. Surface normals are used to identify disjoint particles. Once equilibrium is reached, the neighborhood lists are used to reduce the search space needed in triangulating the points.

Here is a pseudo-code description of our algorithm:

```
Read Volume Data Set
Calculate Gradients at All Sample Points
Initialize Particles
Create Neighbor Web
Repeat Until Particles Reach Equilibrium
  Split or Kill Particles
  Update Web Connections
  Compute Repulsion Forces
  Adjust Repulsion Radii
  Adjust Bin Size
  Calculate Velocities
  Update Particle Positions
  Update Desired Repulsion Radii
Triangulate
```

In this paper we use the following notational conventions. Vectors appear in bold face type, scalars and functions in italics. Partial differentiation is denoted by subscripts. Superscripts of i or j denote members of a collection of objects, whereas other superscripts are exponents. A dot over a letter indicates a derivative with respect to time.

2. SYSTEM INITIALIZATION

An isosurface is defined to be all the points in the volume that are equal to some selected value, which we will refer to as the surface value, s . It is assumed that although none of the sample points may exactly equal s , if there are values above and below s , the underlying function must equal s somewhere in between. Since we do not know the underlying function, we use trilinear interpolation between the eight neighboring sample points (those forming the nearest cube around the particle location) to obtain an approximation of the function at that location. We define our surface as $F(\mathbf{x}) = G(\mathbf{x}) - s = 0$, where $G(\mathbf{x})$ is the interpolated value at the location specified by \mathbf{x} . We subtract s from $G(\mathbf{x})$ to provide the functional evaluation at a point, F^i .

When the volume data is read in, we estimate the gradient at each sample point using central differences. Later, these vectors are trilinearly interpolated component-by-component using the eight neighboring gradients whenever the normal or derivative at a point, F_x^i , is required. Boundary checking must be done to insure that particles are not outside the volume domain before estimating F^i or F_x^i .

¹crossno@cs.unm.edu ²angel@cs.unm.edu

The number and placement of particles at startup is calculated while computing the gradients at the sample points. A particle is placed between any adjacent samples whose values are on opposite sides of the surface value. The particle location is found by linear interpolation between the samples. In this way we guarantee that all of the disconnected components (that are within a voxel in size) contain a seed point. However, the density of the particle placement is based on the sampling rate of the volume data set, rather than the local curvature of the surface. At this point, all of the particles have the same radius of repulsion and are sorted into a three-dimensional lattice of bins. The bin dimensions are initialized to twice the initial radius of repulsion. As the particles redistribute themselves based on the repulsive forces between them, the radii of repulsion will be gradually adjusted based on the local surface curvature and the local particle population.

3. CALCULATING REPULSION FORCES

The repulsive force on a particular particle by all of its neighbors, \mathbf{P}^i , is defined as:

$$\mathbf{P}^i = (\sigma^i)^2 \sum_{j=1}^n \left(\frac{\mathbf{r}^{ij}}{(\sigma^i)^2} E^{ij} + \frac{\mathbf{r}^{ji}}{(\sigma^j)^2} E^{ji} \right)$$

where σ^i is the repulsion radius of particle i , σ^j is the repulsion radius of particle j , \mathbf{r}^{ij} is the vector between particles i and j , E^{ij} is the energy of particle i due to particle j , and E^{ji} is the energy of particle j due to particle i . This definition is essentially the same as used by Witkin and Heckbert [9], though we have changed the meaning of n from all of the particles in the system to the number of particles in a limited region around particle, thus avoiding an $O(n^2)$ force calculation. The energy term is also from Witkin and Heckbert and consists of the Gaussian

$$E^{ij} = \alpha \exp\left(-\frac{|\mathbf{r}^{ij}|^2}{2(\sigma^i)^2}\right)$$

where α is a global constant for repulsion amplitude. Typically $\sigma^i \neq \sigma^j$ and thus E^{ij} is generally not equal to E^{ji} .

Ideally, each of a bin's three dimensions is close to the size of a particle's spherical repulsion diameter, 2σ . Then if the particle is positioned anywhere within the bin, its repulsive range never extends beyond the $3 \times 3 \times 3$ block of bins centered on the bin containing the particle. This scheme limits the number of particles that are checked during the repulsion force calculation to just those particles that are in the block of twenty-seven bins centered about that particle. This works well so long as the radii of repulsion are fairly uniform. However, once curvature-based repulsion is started, bin size must be adjusted to keep up with the largest radius in the system. Otherwise, adjacent particles are not seen and a particle's radius can grow very large due to the lack of forces acting on it. But increasing bin size reduces the efficiency of the system in handling smaller particles since large numbers of particles may then share the same bin.

Another difficulty with the bin scheme is that particles that share a bin but are on physically disconnected components will interact with one another. Instead, we want only particles on the same surface to be included in the repulsion force calculations for one another. We determine whether two particles are on the same surface by comparing the normals. If the normals differ by more than 90 degrees, the particles are assumed to either be on a disconnected component, or to lie on a fold of the same surface that is not directly connected. The bin scheme does not facilitate storing these distinctions in the neighbor relationships.

Our solution is to use a linked list of neighbors (ordered by distance) for each particle, while still retaining the bins. In combination, these lists form a web of proximity relationships for

each disconnected component of the surface. Because the energy from a particular particle drops off rapidly with distance, only particles within a limited region around a particle are included in that particle's neighborhood list.

Initially, each particle's list is created by searching the block of twenty-seven bins. Particle j is included in particle i 's neighbor list if the distance between the two particles is less than two times the radius of repulsion of the largest of the two particles and the angle between the normals of the particles is less than ninety degrees.

Later, with every iteration, neighboring particles' neighbor lists are searched for particles that have moved into range. Once these new particles have been added to the list and the distances of the earlier neighbors updated, the end of the list is examined for the removal of particles that have drifted out of range. We are seeking a hexagonal packing to equalize the aspect ratio of triangles in the resulting triangulation, so if the list contains less than six neighbors, it is left alone. Beyond the six, we remove only those particles that are out of range. If there are less than three neighbors, the bins are searched. Early on, there are typically large numbers of deaths among particles in areas of low curvature, leaving particles isolated and disconnected from neighboring web sections. By keeping the bins, we are able to find neighbors later on when the particle's radius of repulsion has grown sufficiently to bring them into range.

4. PARTICLE DENSITY

The use of a changeable radius of repulsion and a birth-death process allows us to balance between local issues such as distributing particles in a region and global issues such as achieving a desired overall density. For a uniform distribution of particles across the surface, all of the particles in the system would seek to reach a common radius of repulsion defined as a single global parameter, $\hat{\sigma}$. This radius of repulsion would be achieved by all of the particles when the system reaches equilibrium. If a particle is near equilibrium and its radius of repulsion exceeds the desired repulsion radius, it will split. If a particle is near equilibrium and its radius of repulsion is less than some fraction of the desired repulsion radius, it will die.

We prefer a nonuniform particle distribution, so that we can concentrate particles in areas where there is high curvature. For there to be a nonuniform particle distribution, each particle must have its own desired repulsion radius, $\hat{\sigma}^i$, which can be independently adjusted based on the curvature of the local surface. We initially set each particle's desired repulsion radius to the same value. Then the desired repulsion radius for each particle is gradually changed to one based on the (estimated) curvature of the surface at that point.

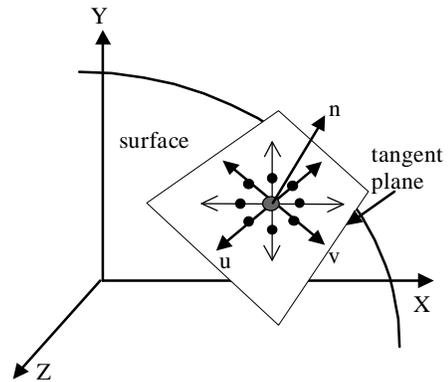


Figure 1: Coordinates tangent to surface at particle position.

Curvature can be thought of as the radius of the largest sphere that will fit against the surface at the particle's position without

passing through the surface at any other point [8]. Note that the curvature has an inverse relationship with the size of the sphere, so that regions of high curvature fit only small spheres, and regions of low curvature fit large spheres. In order to estimate the curvature for each particle (assumed to be on an isosurface), we construct a coordinate system tangent to the estimated surface at the position of the particle, as shown in Figure 1.

We normalize F_x^i to be a unit vector, \mathbf{n} . Then we rotate \mathbf{n} to form a set of orthonormal coordinate axes, \mathbf{u} and \mathbf{v} , in the plane tangent to the surface at the particle's position. In the figure, the particle is the gray point at the center of the \mathbf{n} , \mathbf{u} , \mathbf{v} axes. Additional derivatives are taken at positions that are plus and minus a small distance ϵ from the particle position along \mathbf{u} , \mathbf{v} , and both diagonals between \mathbf{u} and \mathbf{v} (shown as small black dots in Figure 1). After the derivatives are normalized, the dot product is computed between each of them and \mathbf{n} . The minimum of these dot products is saved.

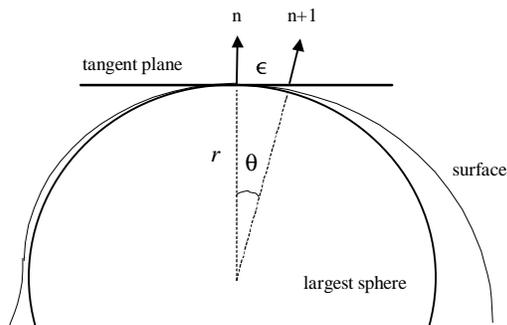


Figure 2: Calculating the radius of repulsion.

In Figure 2, the tangent plane and the surface are viewed in cross section. We take the arccosine of the minimum of the dot products to derive the angle, θ , between \mathbf{n} and the normalized derivative. This angle is then used to compute the radius, r , of the smallest inscribed sphere:

$$r = (\epsilon / |\tan \theta|) + e$$

The estimated curvature is set to the natural log of $r - 1$. The natural log is used as a scaling mechanism to prevent the radius from becoming infinitely large in the case of a flat surface. However, for input values of less than one, the natural log returns a negative value, which is unacceptable for a desired repulsion radius. Consequently, we add e to r prior to taking the log. We subtract out one from the curvature to shift the section of the logarithmic curve to one with a smaller slope.

The change in the desired repulsion radius is calculated as $\dot{\sigma}^i = -(\dot{\sigma}^i - c^i)$, where for particle i , $\dot{\sigma}^i$ is the change in the desired repulsion radius, σ^i is the current desired repulsion radius, and c^i is the curvature at particle i 's current position.

The desired repulsion radius should always have some minimal value. Therefore, the potential new desired repulsion radius is checked against a minimum value before permitting the update to occur. If the update would make the desired repulsion radius too small, the update is not done. Otherwise, the desired repulsion radius is updated using Euler's method:

$$\hat{\sigma}^i(t + \Delta t) = \hat{\sigma}^i(t) + \Delta t \dot{\hat{\sigma}}^i$$

5. PARTICLE MOVEMENT

We compute particle velocity using a modification of the velocity equation in Witkin and Heckbert [9]:

$$\dot{\mathbf{p}}^i(t) = \mathbf{P}^i - \frac{F_x^i \cdot \mathbf{P}^i + \phi(t) F_x^i}{F_x^i \cdot F_x^i} F_x^i$$

where \mathbf{P}^i is the sum of the repulsive forces on particle i , F_x^i is the normal or derivative of the unknown function $F(\mathbf{x})$ at the

current location of particle i , F^i is the value of $F(\mathbf{x})$ at particle i , and $\phi(t)$ is a feedback coefficient that increases with time.

Particle movement is calculated using an adaptive step-size embedded fourth-order Runge-Kutta solver for accuracy and stability [5]. Instead of treating the entire particle system as a unit, we operate on each particle independently, with each one maintaining its own step-size variable. This choice permits us to take smaller steps in areas of high curvature and larger steps elsewhere. Also, we can treat particles on volume boundaries differently from interior particles by zeroing out the velocity components that would take them outside the volume domain. Particle positions are not updated until new positions for all particles have been calculated. Also, repulsion forces are calculated for every particle prior to calculating any velocities.

However, once we added the adaptive step-size solver, we found that if we kept particles very close to the surface, they lost mobility. Particle mobility is fundamental to achieving a good distribution. Yet, if we allowed sufficient error tolerances to ensure mobility, then particles would jitter back and forth after they reached the desired distribution. Jittering made termination a problem since the size of the steps taken were large enough to be indistinguishable from valid movement during distribution. Termination is evaluated by monitoring the changes in radii of repulsion, particle position, and particle population. If the changes are less than some small amount, we consider equilibrium has been reached.

By modifying the feedback coefficient to change over time, we produce a damping effect on particle motion and eliminate jittering. Increasing $\phi(t)$ increases the particle's movement toward the surface along the normal and decreases movement in the direction the particle is being pushed by the repulsive forces. We tried a variety of functions for $\phi(t)$, but found that simply incrementing $\phi(t)$ with each step of the simulation worked best.

6. TRIANGULATION

Given that for each particle we have a neighbor list sorted by distance and distinguishing between disconnected components, we decided to develop our own triangulation algorithm to capitalize on this information. We start with a particle and its nearest neighbor. We create an edge list with the edge between the two points. Using one of the particles as pivot vertex, we sort all of the neighboring particles by angle into a ring around it. We have a beginning and ending angle based on the edges leading to and from the pivot point. With the first pivot point the ending angle is 2π .

Then we evaluate ring vertices for removal by looking at each one's placement with respect to the two adjacent vertices in the ring. If the vertex considered for removal falls inside the triangle formed by its neighbors and the pivot vertex, it must be kept. Otherwise, the aspect ratios of the alternative triangulations with and without the current ring vertex are evaluated. The alternative with the higher aspect ratio is selected. Once the entire ring has been evaluated, the pivot vertex is used to triangulate the ring in a star configuration and the outside edges of the star are substituted into the edge list for the two edges touching the pivot vertex. As each remaining ring particle is used in the triangulation, it is marked as "used". After triangulation, pivot vertices are marked as "done", since once encircled, no further triangles can use them. Then a new pivot vertex is selected from the edge list by traversing it counterclockwise.

Whenever the surface encounters a volume boundary, a different kind of edge element is used to distinguish it. These are not removed from the list until all of the interior edges have been removed. Then the edge list is freed and a new one begun using an "unused" particle. The bins are systematically searched for a starting particle for a new edge list. Once all particles have been used, triangulation is complete.

Figure	Marching Cubes & Decimation		Particle System (Scale Factor = 1)	
	Vertices	Triangles	Vertices	Triangles
Fig 4: Hyperboloid	350	660	354	619
Fig 5: Blast Wave	2070	4055	2180	4292
Fig 6: Hydrogen	1787	3566	1780	3546

Table 1: Vertex and triangle counts for Figures 4 through 6.

7. RESULTS

In Figures 4 through 6 in the color plate, we present three examples of the results of our algorithm compared with the output from our own implementation of Marching Cubes combined with triangle decimation. In each case, we used the combination of decimation parameters (distance to plane, distance to edge, feature angle, and aspect ratio) that reduced the number of vertices in the Marching Cubes surface to closely match the number of vertices in the particle system surface. In this way, a fair comparison can be made of how the two methods distribute a similar numbers of points. The vertex and triangle counts for each figure are given in Table 1. Note that because of the way decimation parameters work, it is difficult to control both the number of vertices and the aspect ratio of the triangles.

In each figure, (a) shows the surfaces rendered with the triangles outlined in white and (b) shows the surfaces rendered using Gouraud shading. The Marching Cubes outputs are colored pink, and the particle system outputs are colored red.

In Figure 4, we show the results of running the particle system over a 32^3 volume generated from a function. It is a hyperboloid of two sheets in which the separation between the lobes occurs within a single cell. This provides a test case where particles have close spatial proximity, but belong on disjoint components. Although both methods correctly separate the sheets, notice that our method concentrates a greater number of points near the tip, producing smoother curves in both sheets near the point of separation. The Marching Cubes example displays noticeable artifacts both in the tip of the larger sheet and in the flattening of the lobe on the other side.

In Figure 5, the two methods are compared on a 64^3 volume data set that is a single time step of a simulation of a blast wave hitting a barrier. This example demonstrates how our method handles sharp corners. We view it from this angle to display the sharpness of the ridges at the bottom of the curl. In (a), the sizes of the triangles in this region are so small that they appear as a mass of white. In (b), this concentration of points results in a sharp clean ridge in the surface created by our particle system. The Marching Cubes surface displays artifacts in this region.

In Figure 6, we present the results from a 64^3 volume data set of a hydrogen atom. We included this data set to demonstrate the effect of different surface curvatures on particle density distribution. In the particle system example, the concentration of particles on the ring contrasts with the more sparsely populated balls and clearly shows the curvature-based repulsion radii at work. Notice the smoothness of the ring compared to the squared edges of the ring in the Marching Cubes example. Also, the points on the two balls of the model are smoothly distributed using our method, whereas decimation tends to clump the points. The particle system produces a smoother surface.

8. CONCLUSION AND FUTURE WORK

We have shown that particle systems provide an alternate method for extracting isosurfaces from volumetric data. The principal advantage of this approach is that we directly generate the desired vertex density concentrating particles in regions where there is high curvature. By building the model this way, we are able to produce smoother surfaces with fewer artifacts

than decimated Marching Cubes. This advantage is even more pronounced when the level of detail is reduced.

Another advantage of our approach is that the scaling factor provides a simple means of controlling the level of detail that preserves good aspect ratios in the resulting triangulation. Changes in the scaling factor do not require recalculation of the surface from scratch. Instead the system starts from the current particle configuration and adjusts the repulsive forces between particles over several iterations. The particles redistribute themselves and adjust their population to accommodate the increase or decrease in density that is required.

There are many possible directions to take this work now. Because particle systems are inherently parallel, we can easily parallelize our algorithm to run in a massively parallel environment. Due to the object-oriented approach we used in our implementation of the particle system, extending it to operate on irregular grids would only require replacing the code that evaluates F^i and F_x^i . Or perhaps the evaluations of F^i and F_x^i could be modified to interact with multiple volumes simultaneously so that surfaces representing the intersection or union of surfaces could be generated. It would also be good to be able to quantify the differences between the surfaces generated by Marching Cubes and our method using some sort of error measure from a known surface.

9. ACKNOWLEDGMENTS

The DOE Mathematics, Information, and Computer Science Office funded this research. The work was performed at Sandia National Laboratories, operated for DOE under contract No. DE-AC04-76DP00789. We thank Brian Wylie, Dino Pavlakos and Claudio Silva for their help and input.

10. REFERENCES

- [1] Figueiredo, L. d., et al. Physically-Based Methods for Polygonization of Implicit Surfaces. in Graphics Interface '92. 1992.
- [2] Lorensen, W. and H. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, in Computer Graphics (SIGGRAPH '87 Proceedings). 1987. p. 163-169.
- [3] Pang, A. and K. Smith. Spray Rendering: Visualization Using Smart Particles. in Visualization '93. 1993. San Jose, California: IEEE Computer Society Press.
- [4] Pang, A., Spray Rendering. IEEE Computer Graphics and Applications, 1994. 14(5): p. 57-63.
- [5] Press, W., et al., Numerical Recipes in C, Second Ed. Cambridge University Press, Cambridge, England, 1992.
- [6] Schroeder, W., J. Zarge, and W. Lorensen, Decimation of Triangle Meshes. Computer Graphics (SIGGRAPH '92 Proceedings), 1992. 26(2): p. 65-70.
- [7] Szeliski, R. and D. Tonnesen, Surface Modeling with Oriented Particle Systems, in Computer Graphics. 1992. p. 185-194.
- [8] Turk, G., Re-Tiling Polygonal Surfaces. Computer Graphics (SIGGRAPH '92 Proceedings), 1992. 26(2): p. 55-64.
- [9] Witkin, A. and P. Heckbert, Using Particles to Sample and Control Implicit Surfaces. Computer Graphics Proceedings, Annual Conference Series, 1994, 1994: p. 269-277.

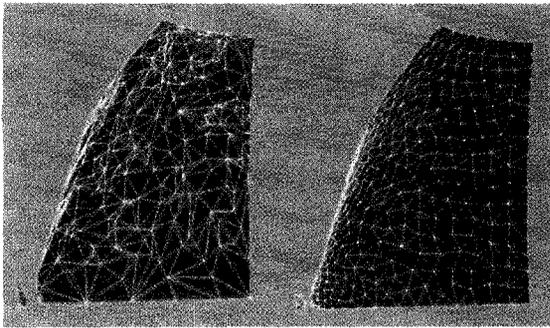


Figure 4(a): Hyperboloid of 2 sheets with outlined triangulation. Marching Cubes combined with decimation is on the left in pink, and the particle system triangulation is on the right in red.



Figure 5(a): Blast wave (viewed from behind) with outlined triangulation. Marching Cubes combined with decimation is on the top in pink, and the particle system triangulation is on the bottom in red.

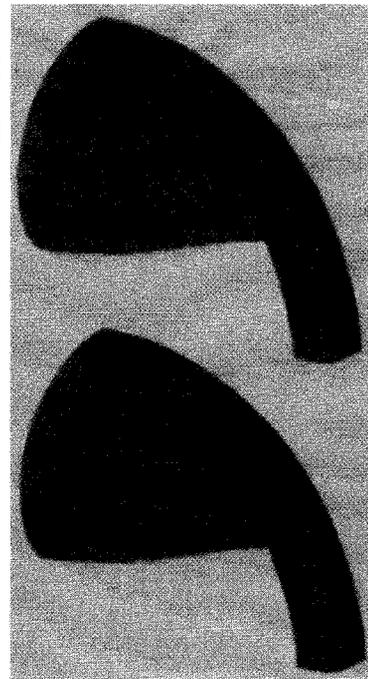


Figure 5(b): Blast wave (viewed from behind) with Gouraud shading. Marching Cubes combined with decimation is on the top in pink, and the particle system surface is on the bottom in red.

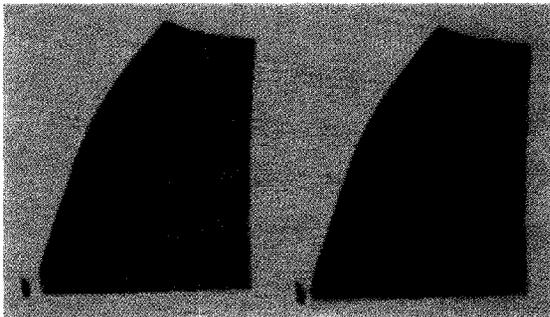


Figure 4(b): Hyperboloid of 2 sheets with Gouraud shading. Marching Cubes combined with decimation is on the left in pink, and the particle system surface is on the right in red.

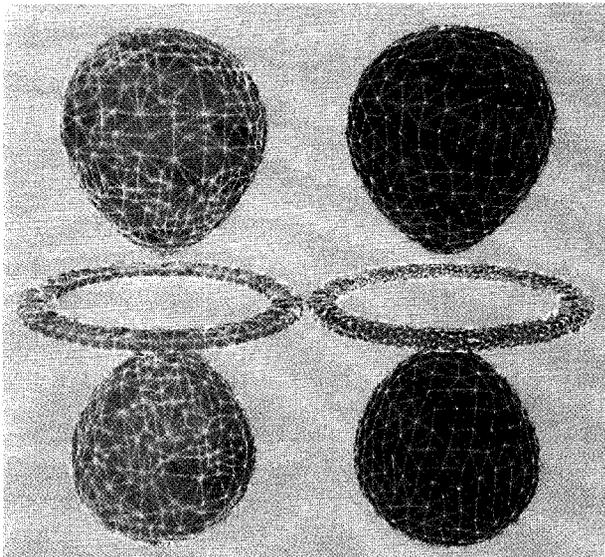


Figure 6(a): Hydrogen atom with outlined triangulation. Marching Cubes combined with decimation is on the left in pink, and the particle system triangulation is on the right in red.

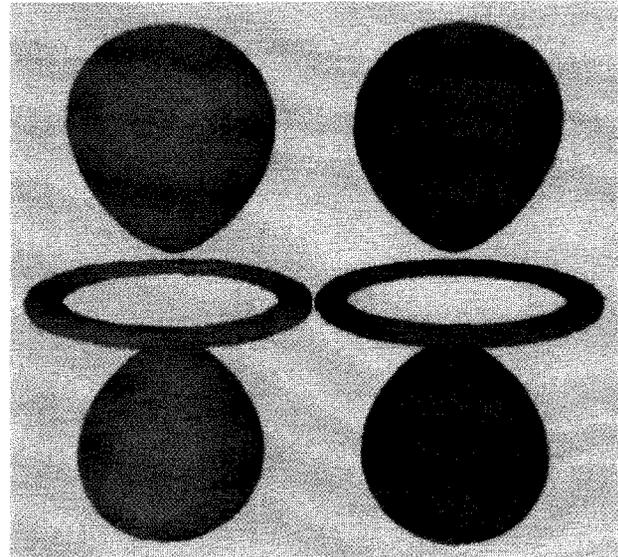


Figure 6(b): Hydrogen atom with Gouraud shading. Marching Cubes combined with decimation is on the left in pink, and the particle system surface is on the right in red.

Isosurface Extraction Using Particle Systems

Patricia Crossno, Edward Angel