The Visual Computer

Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces

John C. Hart

School of EECS, Washington State University,
Pullman, WA 99164-2752, USA
e-mail: hart@eecs.wsu.edu

Sphere tracing is a new technique for rendering implicit surfaces that uses geometric distance. Sphere tracing marches along the ray toward its first intersection in steps guaranteed not to penetrate the implicit surface. It is particularly adept at rendering pathological surfaces. Creased and rough implicit surfaces are defined by functions with discontinuous or undefined derivatives. Sphere tracing requires only a bound on the magnitude of the derivative, robustly avoiding problems where the derivative jumps or vanishes. It is an efficient direct visualization system for the design and investigation of new implicit models. Sphere tracing efficiently approximates cone tracing, supporting symbolic-prefiltered antialiasing. Signed distance functions for a variety of primitives and operations are derived.

Key words: Distance – Implicit surface – Lipschitz condition – Ray tracing – Solid modeling

1 Introduction

Whereas a parametric surface is defined by a function that, given a tuple of parameters, indicates a corresponding location in space, an implicit surface is defined by a function that, given a point in space, indicates whether the point is inside, on or outside the surface.

The most commonly studied form of implicit surfaces is the algebraic surface, defined implicitly by a polynomial function. For example, the unit sphere is defined by the second-degree algebraic implicit equation

$$x^2 + y^2 + z^2 - 1 = 0 \tag{1}$$

as the locus of coordinates of which the hypotenuse (squared) is unity.

Alternatively, using a distance metric, one can represent the unit sphere geometrically by the implicit equation

$$\|\boldsymbol{x}\| - 1 = 0 \tag{2}$$

as the locus of points of unit distance from the origin. Here $\boldsymbol{x} = (x, y, z)$ and $\|(x, y, z)\|$ denotes the Euclidean magnitude $\sqrt{x^2 + y^2 + z^2}$. The implicit surface of Eq. 2 agrees with that of Eq. 1, though their values differ at almost every other point in $\mathbb{R}^3$. Specifically, Eq. 1 returns *algebraic distance* (Rockwood and Owen 1987) whereas Eq. 2 returns *geometric distance*.

A comparison of geometric versus algebraic representations of quadric surfaces led us to prefer the geometric representation (Goldman 1983). The parameters of a geometric representation are independent of the coordinates, and are more robust and intuitive than algebraic coefficients. Distance-based functions like Eq. 2 provide one method for representing implicit surfaces geometrically.

Distance-based models can be found in a variety of areas. *Offset* surfaces have become valuable in computer-aided geometric design for their use of distance to model the physical capabilities of machine cutting tools (Barnhill et al. 1992). *Skeletal* models, which in computer graphics simulate articulated figures such as hands and dinosaurs, are equivalent to offset surfaces. Computer vision's medial-axis transform converts a given shape to its skeletal representation (Ballard and Brown, 1982). *Generalized cylinders*, began as a geometric representation in computer vision (Agin and

The Visual Computer (1996) 12:527–545

Binford 1976), but have also matured into a standard modeling primitive in computer graphics (Bloomenthal and Wyvill 1990) – special ray-tracing algorithms were developed for their rendering in (Wijk 1984; Bronsvoort and Klok 1985).

1.1 Previous work

Several methods exist for rendering implicit surfaces. Indirect methods polygonize the implicit surface to a given tolerance, allowing the use of existing polygon-rendering techniques and hardware for interactive inspection (Wyvill et al. 1986; Bloomenthal 1988). Although polygonization transforms implicit surfaces into a representation easily rendered and incorporated into graphics systems, polygonizations are typically not guaranteed and may not accurately detect disconnected or detailed sections of the implicit surface. Production-rendering systems tend to polygonize surfaces, resulting in large time and memory overheads to represent accurately an otherwise simple implicit model.

In an effort to combine speed and accuracy, Sederberg and Zundel (1989) developed a direct scanline method to render more accurately algebraic implicit surfaces at interactive speeds.

Although slower, ray tracing provides a direct, accurate, and elegant method for investigating a much larger variety of implicit surfaces. Let

$$\boldsymbol{r}(t) = \boldsymbol{r}_0 + t\boldsymbol{r}_d \tag{3}$$

parametrically define a ray anchored at $\boldsymbol{r}_0$ in the direction of the unit vector $\boldsymbol{r}_d$. Plugging the ray equation $\boldsymbol{r}: \mathbb{R} \to \mathbb{R}^3$ into the function $f: \mathbb{R}^3 \to \mathbb{R}$ that defines the implicit surface produces the composite real function $F: \mathbb{R} \to \mathbb{R}$ where $F = f \circ \boldsymbol{r}$ so that the solutions to

$$F(t) = 0 \tag{4}$$

correspond to ray intersections with the implicit surface. Implicit surface ray-tracing algorithms simply apply one of the multitude of numerical root-finding methods to solve Eq. 4.

When $f(\boldsymbol{x}) = 0$ implicitly defines an algebraic surface, Eq. 4 is a polynomial equation, and it can be solved by DesCartes' rule of signs (Hanrahan 1983), Sturm sequences (Wijk 1984), and Laguerre's method (Wyvill and Trotman 1990).

Ideally, the root-finding procedure should only need the ability to evaluate the function at any point. However, one can always construct a pathological function that will cause such a "blind" technique to miss one or more roots by inserting an arbitrarily thin region between samples where the function zips off to zero and back [a point reiterated from Kalra and Barr (1989) and Von Herzen et al (1990)]. Hence, any robust root finder needs more information than simple function evaluation.

The "hypertexture" system was a brute-force, blind, ray-marching scheme, using only function evaluation (Perlin and Hoffert 1989). This supported the design of implicit surfaces without regard to the analytic properties of the defining functions. Freed from such constraints, fractal and hairy surfaces were modeled by implicit surfaces with functions that contained procedural elements. The high frequencies produced by these geometric textures required fine sampling along the ray, resulting in a rendering speed so slow that it necessitated parallel implementation.

Guaranted ray intersection requires extra information, which in most cases is produced by the derivative of the function. Interval analysis finds ray intersections by defining the function and its derivative on intervals instead of single values (Mitchell 1990).

The (LG) surfaces method imposed the Lipschitz condition on f to create an efficient octree partitioning guaranteed to contain the implicit surface, and imposed the Lipschitz condition on F' to find ray intersections within each octree cell (Kalra and Barr 1989).

1.2 Overview

Sphere tracing is a guaranteed technique for ray tracing implicit surfaces. Unlike LG surfaces or interval analysis, it does not require the ability to evaluate the derivative of the function. Instead, it requires only a bound on the magnitude of the derivative – that the function be continuous and Lipschitz. Thus, the derivative of the function need not be continuous, nor even defined.

Sphere tracing benefits from this relaxation by using the continuous but nondifferentiable

minimum and maximum operations for constructive solid geometry (CSG) instead of the commonly used Roth diagrams (Roth 1982). Unlike typical ray tracers, sphere tracing finds the first ray intersection, the least-positive solution t to Eq. 4. Typically, all ray intersections must be determined for constructive solid geometry (Roth 1982). Sphere tracing overcomes this requirement by using maximum and minimum operations to model the entire-scene with a single, nondifferentiable function. This also supports the blending of nondifferentiable CSG results.

Sphere tracing allows the efficient visualization of a wider range of implicit surfaces than previously possible, including creased, rough, and fractal surfaces. Like the slower, brute force, rendering approach of the hypertexture system (Perlin and Hoffert 1989), sphere tracing frees the implicit surface designer from many concerns regarding the analytic behavior of the defining function, fostering more diverse implicit formulations. Moreover, a structure in mathematics is often specified as the locus of points that satisfy a particular condition. Sphere tracing visualizes such structures, regardless of smoothness, extent, and connectedness, given only a bound on the rate of the condition's continuous changes over space. Sphere tracing provides a direct and flexible visualization tool for the development of new implicit models.

Sphere tracing also approximates cone tracing (Amanatides 1984) to eliminate aliasing artifacts and simulate soft shadows.

2 Sphere tracing

Sphere tracing capitalizes on functions that return the distance to their implicit surfaces (Sect. 2.1) to define a sequence of points (Sect. 2.2) that converges linearly to the first ray-surface intersection (Sect. 2.3). Section 2.4 incorporates CSG into sphere tracing at the model level. Section 2.5 describes several enhancements to sphere tracing to hasten convergence.

2.1 Distance surfaces

This section defines and discusses functions that measure or bound the geometric distance to their implicit surfaces. Such functions implicitly define *distance surfaces*, as mentioned by Bloomenthal and Shoemake (1991). The appendices derive functions that measure or bound distances for a variety primitives and operations.

Let the function f be a continuous mapping $f: \mathbb{R}^n \to \mathbb{R}$ that implicitly describes the set $A \subset \mathbb{R}^n$ as the locus of points

$$A = \{\boldsymbol{x} : f(\boldsymbol{x}) \leq 0\}\,. \tag{5}$$

The continuity of f implies that it returns zero on the boundary ∂A, which forms the *implicit surface* of f. If f is strictly negative over the interior $\mathring{A}$, then the multivalued function image $f^{-1}(0)$ concisely represents the implicit surface of f. Even if f is continuous, it need not be strictly negative over the interior. For example, the set A may be validly represented by a continuous function that returns zero for every point in $\mathring{A}$.

Definition 1. The point-to-set distance defines *the distance from a point* $\boldsymbol{x} \in \mathbb{R}^3$ *to a set* $A \subset \mathbb{R}^3$ as the distance from $\boldsymbol{x}$ to the closest point in A,

$$d(\boldsymbol{x}, A) = \min_{\boldsymbol{y} \in A} \|\boldsymbol{x} - \boldsymbol{y}\|\,. \tag{6}$$

Given a set A, the point-to-set distance $d(\boldsymbol{x}, A)$ implicitly defines A (from the outside) (Kaplansky 1977). Here, we are interested in the converse: *given an implicit function, what is the point-to-set distance to its surface?*

Definition 2. A function $f: \mathbb{R}^3 \to \mathbb{R}$ is a *signed distance bound of its implicit surface* $f^{-1}(0)$ if and only if

$$|f(\boldsymbol{x})| \leq d(\boldsymbol{x}, f^{-1}(0))\,. \tag{7}$$

If the equality holds for Eq. 7, then f is a *signed distance function.*

Table 1 lists the primitives and operations for which the appendices contain signed distance functions and bounds.

The Lipschitz constant is a useful quantity for deriving signed distance bounds to complex shapes. Lipschitz constants have been used in computer graphics for collision detection (Von Herzen and Barr 1987) and rendering implicit surfaces (Kalra and Barr 1989).

Table 1. Directory of signed distance functions and bounds

Primitive/operation	Signed distance function	Signed distance bound
Plane	Appendix A	
Sphere	Appendix A	
Ellipsoid	(Hart 1994)	Appendices A and E
Cylinder	Appendix A	
Cone	Appendix A	
Torus	Appendix A	
Superquadrics		Appendix B
Generalized cylinder	Appendix C	
Union	Section 2.4	
Intersection		Section 2.4
Complement	Section 2.4	
Soft objects		Section D
Pseudonorm blend	(Rockwood 1989)	Appendix D
Isometry	Appendix E	
Uniform scale	Appendix E	
Linear transformation		Appendix E
Taper		Appendix E
Twist		Appendix E
Hypertexture		Appendix F
Fractals		Appendix F

Definition 3. *A function* $f: \mathbb{R}^3 \to \mathbb{R}$ *is Lipschitz over a domain D* if and only if for all $\boldsymbol{x}, \boldsymbol{y} \in D$, there is a positive, finite, constant λ such that

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \lambda \|\boldsymbol{x} - \boldsymbol{y}\| . \tag{8}$$

The value λ is called the *Lipschitz constant*. The function $\operatorname{Lip} f$ returns the minimum Lipschitz constant λ satisfying Eq. 8.
A Lipschitz constant of the sum of two functions results from the sum of the functions' Lipschitz constants. By the chain rule, a Lipschitz constant of the composition of functions results from the product of the component functions' Lipschitz constants.
The following theorem shows how to turn a Lipschitz function into a signed distance bound, allowing sphere tracing to render any implicit surface defined by a Lipschitz function.

Theorem 1. *Let f be Lipschitz with Lipschitz constant* λ. *Then the function* f/λ *is a signed distance bound of its implicit surface.*

Proof. Given a point $\boldsymbol{x}$, let $\boldsymbol{y} \in f^{-1}(0)$ be one of the points such that

$$\|\boldsymbol{x} - \boldsymbol{y}\| = d(\boldsymbol{x}, f^{-1}(0)) . \tag{9}$$

Then by Eq. 8 and $f(\boldsymbol{y}) = 0$ it follows that

$$|f(\boldsymbol{x})| \leq \lambda d(\boldsymbol{x}, f^{-1}(0)) . \tag{10}$$

Hence, $\lambda^{-1} f(\boldsymbol{x})$ is a signed distance bound for any Lipschitz function f. [Compare Eq. 8 of Kalra and Barr (1989).] □

If $\lambda = \operatorname{Lip} f$, then an optimal signed distance bound results.

2.2 Ray intersection

One intersects a ray $\boldsymbol{r}(t)$ with the implicit surface defined by the signed distance bound $f(\boldsymbol{x})$ by finding its least positive root (the *first* root) of $F(t)$. This root is the limit point of the sequence defined by the recurrence equation

$$t_{i+1} = t_i + F(t_i) \tag{11}$$

and the initial point $t_0 = 0$. The sequence converges if and only if the ray intersects the implicit surface. This sequence forms the kernel of the geometric, implicit surface-rendering algorithm in Fig. 1.
The convergence test ε is set to the desired precision. The maximum distance D corresponds to the radius of a viewer-centered yonder clipping sphere and is necessary to detect nonconvergent sequences.
The absolute value of the signed distance function can be considered the radius of a sphere guaranteed not to penetrate any of the implicit surface. This sphere was called an *unbounding* sphere by Hart et al. (1989) who used a distance bound to define and visualize 3D deterministic fractals implicitly because the implicit surface is contained in the closed complement of this sphere. Unlike a bounding volume that surrounds an object, an unbounding volume surrounds an area of space not containing the object. The name "sphere tracing" arose from the property that ray intersections are determined by sequences of unbounding spheres.
As did Ricci (1974), sphere tracing uses the minimum and maximum functions for CSG. These operations crease the implicit surface locally, so that the defining function remains continuous in value, but not in derivative. Derivative

The signed distance bound f, ray $\mathbf{r}(t)$ and maximum ray traversal distance D are given.

```
Initialize t = 0
While t < D
    Let d = f(r(t))
    If d < ε then return t — intersection
    Increment t = t + d
return ∅ — no intersection
```

Fig. 1. Pseudocode of the geometric implicit surface rendering algorithm

discontinuity can cause problems with root finders, which must find all roots of the function and resolve the CSG operation by using a Roth diagram (Roth 1982). Sphere tracing operates independently of the derivative, given its bound, and need converge only to the first root, even for CSG models.

2.3 Analysis

Root refinement methods, such as Newton's method, converge quadratically to simple roots (where the ray penetrates the surface) and linearly to multiple roots (where the ray grazes the surface) (Gerald and Wheatley 1989). Root isolation methods that divide and conquer, such as LG surfaces (Kalra and Barr 1989) and interval analysis (Mitchell 1990), converge linearly, since the widths of the intervals are reduced by a factor of one-half at each iteration. Root isolation methods are allowed to converge only in the event of a multiple root; otherwise they pass control to a faster root-refinement method the moment they find a monotonic region straddling the t-axis.

Theorem 2. *Given a function $F:\mathbb{R}\to\mathbb{R}$ with the Lipschitz bound $\lambda \geq Lip\, F$ and an initial point t_0, sphere tracing converges linearly to the smallest root greater than t_0.*

The sphere-tracing sequence can be written as:

$$t_{i+1} = g(t_i) = t_i + \frac{|F(t_i)|}{\lambda}. \tag{12}$$

In this form, the similarities of Eq. 12 to Newton's method are more visible. Let r be the smallest root greater than the initial point t_0. Since $F(r) = 0$, then $g(r) = r$, and at any nonroot $|F|/\lambda$ is positive. Hence Eq. 12 converges to the first root.
Without loss of generality, F is assumed to be non-negative in the region of interest, which eliminates the need for the absolute value. The Taylor expansion of $F(t_i)$ about the root r is:

$$g(t_i) = g(r) + (t_i - r)g'(r) + \frac{(t_i - r)^2}{2} g''(\tau) \tag{13}$$

for some $\tau \in [t_i, r]$ and $g'(r) = 1 + F'(y)/\lambda$. The error term becomes:

$$\begin{aligned} e_{i+1} &= t_{i+1} - r = g(t_i) - g(r) \\ &= g'(r)e_i + \text{higher-order terms}. \end{aligned} \tag{14}$$

Since $g'(r)$ is constant in the iteration Eq. 12 converges linearly to y. □

Corollary 2.1. *Sphere tracing converges quadratically if and only if the function is steepest at its first root.*

In the event that $F'(r) = -\lambda$, the linear term of the error of Eq. 13 drops out, leaving the quadratic and higher-order terms. □

2.4 Constructive solid geometry (CSG)

Following Ricci (1974), the minimum and maximum operations on functions result in union and intersection operations on their implicit surfaces. In the following equations, let f_A, f_B be signed distance functions of sets A and B respectively. If f_A or f_B is a signed distance bound, then the resulting CSG implicit function will be also be a bound.
The distance to the union of A and B is the distance to the closer of the two.

$$d(\mathbf{x}, A \cup B) = \min f_A(\mathbf{x}), f_B(\mathbf{x}). \tag{15}$$

Similarly, the distance to a list of objects is the smallest of the distances to each of the component objects.
The distance to the complement of A takes advantage of the signed nature of the distance function

$$d(\boldsymbol{x}, \mathbb{R}^3 \setminus A) = -f_A(\boldsymbol{x}). \tag{16}$$

Although DeMorgan's theorem defines an intersection as the complement of the union of complements, the minimum operators used in the union are not complemented properly. Instead, the distance to the intersection is bound by the distance to the farthest component.

Theorem 3. *The distance from a point* $\boldsymbol{x}$ *to the intersection of two implicit surfaces* $A = f_A^{-1}(0)$ *and* $B = f_B^{-1}(0)$ *defined by signed distance bounds* f_A, f_B *is bounded by*

$$d(\boldsymbol{x}, A \cap B) \geq \max f_A(\boldsymbol{x}), f_B(\boldsymbol{x}). \tag{17}$$

Proof. The theorem can be proved by parts, as illustrated on a sample intersection in Fig. 3.

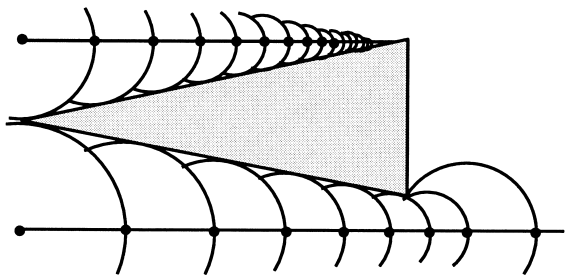
Case 1. $\boldsymbol{x} \in A \cap B$. Both f_A and f_B are negative, and the larger of the two indicates the (negative) distance to the closest edge of the intersection.

Case 2. $\boldsymbol{x} \in A, \boldsymbol{x} \notin B$. The function f_A is negative, whereas f_B is positive, hence the greater of the two. The closest point on B to $\boldsymbol{x}$ may not be in the intersection, but no point in the intersection can be closer.

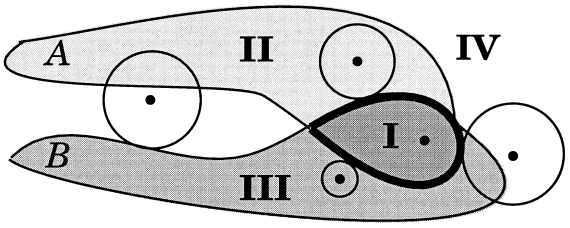
Case 3. $\boldsymbol{x} \notin A, \boldsymbol{x} \in B$. This is symmetric with case 2.

Case 4. $\boldsymbol{x} \notin A \cup B$. As before, the closest point in the intersection $A \cap B$ can be no closer than the farther of the closest point in A and the closest point in B. □

From its definition, set subtraction $A - B$ may be simulated as $A \cap (\mathbb{R}^3 \setminus B)$, though it yields only a signed distance bound due to the intersection operator.
The union and intersection operators are demonstrated in Fig. 9 in Sect. 4.2.



2



3

Fig. 2. A hit and a miss

Fig. 3. Sample points illustrate a bound on the distance to the intersection between two sets

2.5 Enhancements

The following enhancements increase the efficiency of sphere tracing by reducing unnecessary distance computations, which can be quite expensive and even iterative in some cases. The enhancements are evaluated and analyzed empirically in Sect. 4.3.

2.5.1 Image coherence

An algorithm similar to sphere tracing has been developed for rendering discrete volumetric data. It uses the 3D *distance transform* (Zuiderveld et al. 1992). The distance transform takes a binary "filled/unfilled" voxel array to a numerical voxel array so that each voxel contains the distance to the closest "filled" voxel under a given metric. We have also extended the concept of Lipschitz constants to volume rendering (Stander and Hart 1994), trading the distance transform for an octree

of local Lipschitz constants as in Kalra and Barr (1989), allowing distance-based, accelerated, volume rendering of arbitrary isovalued surfaces, while eliminating the need to recompute the preprocessed data structure for each change in the threshold.

One enhancement by Zuiderveld et al. (1992) keeps track of the smallest distance encountered by a ray that misses the object. Under an orthogonal projection, this smallest distance defines the radius of a disk of guaranteed empty pixels surrounding the sample point. Under a perspective projection, the minimum *projected* distance must be computed (requiring ray-sphere intersection), and this enhancement becomes less efficient. Initial tests have shown that this enhancement degrades performance in the perspective case for typical implicit surfaces.

Initialize $d = \infty$.

Repeat

  Let d be the lesser of d or the distance to
  the contents of the bounding volume
  at the top of the heap.

  Remove the top of the heap and re-heap.

  Let d_h be the distance to the bounding volume
  now at the top of the heap.

Until $d < d_h$ or the heap is empty.

return d.

Fig. 4. An efficient algorithm for finding the closest object of a collection using bounding volumes

2.5.2 Bounding volumes

Bounding volumes provide a useful mechanism to cull processing of intricate geometries that are irrelevant to the current task. Beyond their typical benefit of avoiding the casting of rays that miss an object, they also help sphere tracing avoid distance computations for objects farther away than others. The overhead of quick distance checks by bounding volume is, in most cases, a small price to pay for the benefit of avoiding many expensive, but useless, distance computations.

First, the distances to each bounding volume in a union or collection of objects is computed. Then, in order of increasing bounding volume distance, the distance to the contents of each bounding volume is computed until a content's distance is less than the smallest bounding volume distance. This distance is then the point-to-set distance to the collection of objects. This process is sketched in Fig. 4.

A Lagrange multiplier method for finding the bouding parallelepiped of an implicit surface appears in Kay and Kajiya (1986). The signed distance bound has properties that might yield an alternative algorithm for the bounding volume of an implicit surface, but this topic is left for further research.

2.5.3 The triangle inequality

When computing the shortest distance between a point and a collection of objects, one need not compute the distance to an object if its distance evaluation from the previous iteration minus the distance traversed along the ray since the previous iteration exceeds the minimum distance measured in the current iteration. This triangle inequality enhancement is implemented in Fig. 5.

2.5.4 Octree partitioning

Eliminating empty space certainly aids rendering efficiency, but the major benefit of partitioning is that it allows the imposition of local bounds on the Lipschitz constants yielding more accurate signed distance bounds. Octree partitioning has been used in the polygonization (Bloomenthal 1988) and ray tracing (Kalra and Barr 1989) of implicit surfaces. Sphere tracing reaps the same benefits from spatial partitioning as did the root finding method of Kalra and Barr, who used the Lipschitz constant to cull octree

```
Initialize d_last = 0 and t = 0.

For each object o ∈ O initialize o_d = 0.

Repeat.

    Reset d_min = ∞.
    For each object o ∈ O.
        If o_d − d_last > d_min then
            Update o_d = o_d − d_last.
        Otherwise
            Let d = d(r(t), o).
            Reset o_d = d.
            Update d_min = min(d_min, d).
        End if.
        Let d_last = d_min.
    End for.
    Update t = t + d_min.

Until d_min < ε or t > D.
```

Fig. 5. Triangle inequality algorithm for avoiding unnecessary distance computations

nodes guaranteed not to intersect the implicit surface.

Ray intersection with an implicit surface defined by a signed distance bound is penalized by the section of the domain where the gradient magnitude is greatest. Chopping an object into the union of smaller chunks allows each chunk to be treated individually, and it is penalized only by the largest gradient within its bounds. Since Kalra and Barr's partitioning algorithm requires only a bound on the Lipschitz constant of the function, the use of this octree in no way restricts the domain of functions available for sphere tracing. Octree partitioning further enhances sphere tracing of unions and lists by optionally storing an index to the object closest to the cell. An object is closest to an octree cell if and only if it is the closest object to every point in the cell. Under this definition, some cells may not have a closest object. By the triangle inequality, an object is closest to a cell if the distance from the cell's centroid to the object, plus the distance from the centroid to the cell corner, is still less than the distance from the centroid to any other object.

2.5.5 Convexity

Knowing that an object is convex can make sphere tracing more efficient by increasing the step size along the ray.

Theorem 4. *Let $A \subset \mathbb{R}^3$ be a convex set defined implicitly by the signed distance function f. Then, given a unit vector $\boldsymbol{v} \in \mathbb{R}^3$, the line segment*

$$\left[\boldsymbol{x}, \frac{f(\boldsymbol{x})}{-\boldsymbol{v} \cdot \nabla f(\boldsymbol{x})}\,\boldsymbol{v}\right] \tag{18}$$

does not intersect A, except possibly at its second end point.

Proof. The gradient of a signed distance function ∇f has the following properties on the complement of a convex set $\mathbb{R}^3 \setminus A$: (1) it is continuous, (2) its magnitude is one (the change in the function equals the change in the distance), and (3) its direction points directly away from the closest point on the implicit surface. Hence, for any $\boldsymbol{x} \in \mathbb{R}^3 \setminus A$, we know the closest point in A, and its surface normal points toward $\boldsymbol{x}$. Since A is convex, it cannot penetrate the tangent plane at the closest point in A.

The intersection of a ray anchored at $\boldsymbol{x}$ and direction $\boldsymbol{v}$ with the tangent plane normal to the vector $\nabla f(\boldsymbol{x})$ a distance of $f(\boldsymbol{x})$ from $\boldsymbol{x}$ is given by the second end point of Eq. 18. □

Corollary 4.1. *If $\nabla f(\boldsymbol{x}) \cdot \boldsymbol{v} \geq 0$, then the ray anchored at $\boldsymbol{x}$ and direction $\boldsymbol{v}$ does not intersect the implicit surface of f.*

Theorem 4 allows sphere tracing to take larger steps toward convex objects, and Corollary 4.1 allows sphere tracing to avoid computing the distance to convex objects it has stepped beyond. The convexity enhancement likely causes sphere tracing to converge quadratically because of its similarity to Newton's method, which also converges quadratically.

Bounding volumes are usually convex, and combining these two techniques can further reduce the computation of unnecessary distances.

Knowledge of convexity becomes a necessity for rendering scenes with a horizon line. Consider a ground plane and a ray parallel to it. Sphere tracing will step along this ray at fixed intervals looking for an intersection that never happens. Corollary 4.1 avoids this situation, whereas Theorem 4 hastens convergence of rays nearly parallel to the ground plane.

3 Antialiasing

Tracing cones instead of rays results in an area-sampling antialiasing method in Amanatides (1984). Cone tracing computes the intersection of cones with spheres, planes, and polygons to prefilter an image symbolically, eliminating the aliasing artifacts that result from point sampling. Sphere tracing can detect and approximate cone intersections with any implicit surface defined by a signed distance function. One must still implement the details of the cone-tracing algorithm to determine the shape of the cones as they bounce around a scene, but may rely on unbounding spheres to increase the efficiency of computing cone intersections.

At some point along a grazing ray, the sequence of unbounding spheres shrinks, falling within the bounds of the cone; then enlarges, escaping the bounds of the cone. This poses the problem of "choosing a representative" (Amanatides 1984) – a location to take a sample to approximate the shading of the cone's intersection with the surface. A cover is a pixel-radius offset bounding an implicit surface on the inside and outside such that a ray-cover intersection indicates a cone-object intersection (Thomas et al. 1989). Given an implicit surface defined by the signed distance function $f(\boldsymbol{x})$, its outer cover is the global offset surface implicitly defined by $f(\boldsymbol{x}) - r_p$, and its inner cover is the global offset surface implicitly defined by $f(\boldsymbol{x}) + r_p$, where r_p is the radius of a pixel [one-half of the diameter of a pixel (Hart and DeFanti 1991)]. In other words, the outer cover is the surface $f^{-1}(r_p)$, and the inner cover is the surface $f^{-1}(r_p)$. Instead of sphere tracing the implicit surface of $f(\boldsymbol{x})$, the antialiasing algorithm sphere traces the inner cover – the implicit surface of $f(\boldsymbol{x}) + r_p$.

The development of covers proposes that the most representative choice for silhouette antialiasing would be the point along the section of the ray closest to the surface. Hence, of the unbounding spheres inside the cone, the center of the smallest sphere (with respect to pixel size) becomes the representative sample. Though this sample is off the implicit surface, one assumes a reasonable level of continuity in the gradient of the distance function to define a usable surface normal. The sequence along the ray of unbounding spheres is related to a cone as shown in Fig. 6.

For smooth implicit surfaces, one may assume local planarity. Hence the implicit surface is assumed to cover the cross section of the cone with a straight edge of the given distance from the cone's center. The amount of influence this shaded point has, with respect to the points the ray intersects further on, depends on the signed distance function evaluated at the representative $f(\boldsymbol{x})$ (the radius of the closest unbounding sphere) to the implicit surface. The fraction of coverage of a disk of radius r_p by an intersecting half-plane of signed distance $f(\boldsymbol{x})$ from its center is given by:

$$\alpha = \frac{1}{2} - \frac{f(\boldsymbol{x})\sqrt{r_p^2 - f(\boldsymbol{x})^2}}{\pi r_p^2} - \frac{1}{\pi}\arcsin\frac{f(\boldsymbol{x})}{r_p} \qquad (19)$$

and is derived by Thompson (1990).

Ray traversal proceeds in steps of $f(\boldsymbol{x}) + r_p$ (which may take it through the surface). The percentage of coverage α represents the cone intersection of the grazing ray. It is treated as an opacity; it is accumulated and used to blend the shading of the current representative $\boldsymbol{x}$ with the shading resulting from further near misses and intersections. The standard rules of image compositing (Porter and Duff 1984) are used.

For intersection edges, one must keep track of all signed distance functions having unbounding spheres that fit within the bounds of the cone. Upon ray-intersection approximation, the signed distance functions of each of the intersecting surfaces provide the proportions for the proper combination of their shading properties. The representative for intersection is the last point of the ray-traversal sequence, the point that satisfies the convergence test.

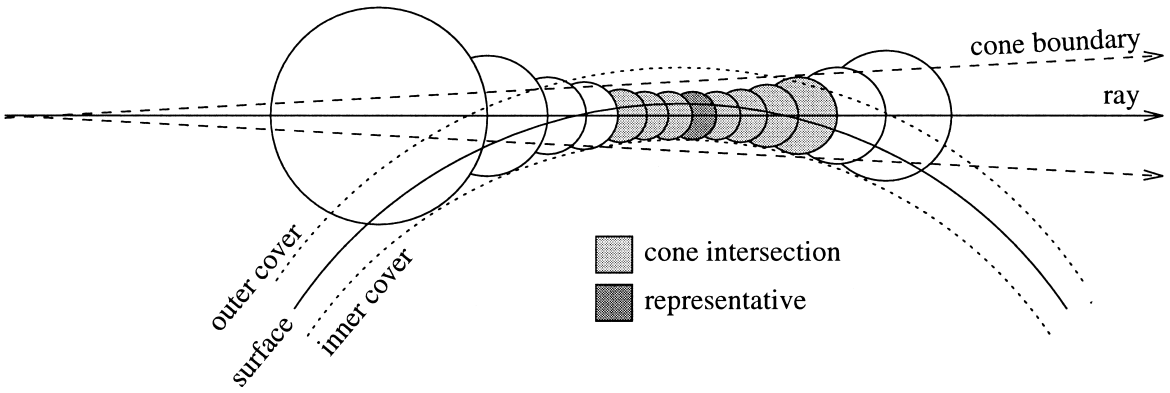


Fig. 6. Sphere tracing approximates a cone intersection. The ray intersects the original surface but misses its inner cover. This cone intersection accounts for more than half of the pixel's illumination

Often the signed distance function is too expensive to compute efficiently, and a signed distance bound is used. A bound may return unbounding spheres with radii that prematurely shrink below the radius of a pixel, resulting in incorrect cone intersections. In this case, a separate distance approximation may be useful. For example, Pratt (1987) and Taubin (1994) estimate the distance to the implicit surface of f with the first-order approximation $f/\|\nabla f\|$. In general, this approximation is not necessarily a distance bound. Taubin's (1994) Lemma 1 asserts that this approximation is asymptotic to geometric distance as one approaches the surface. Cone intersections can hence be more accurately determined by this approximation than by the signed distance bound.
Cone tracing inhibits texture aliasing by filtering the texture based on the radius of the cone at intersection and extends directly to the sphere tracing method.

4 Results

Sphere tracing simplifies the implementation of an implicit surface ray tracer, and runs at speeds comparable to other implicit surface-rendering algorithms.

4.1 Implementation

Sphere tracing has been implemented in a rendering system called zeno. Inclusion of an implicit surface into zeno requires the definition of two functions: a signed distance function for ray intersection and a surface normal function for shading. A new primitive or operation can be incorporated into zeno with no more than a distance bound. The negative part of the signed distance bound is only necessary for some CSG and blending operations. It is not needed for the visualization of functions that are zero-valued inside the implicit surface. The surface normal function can be avoided by a general six-sample numerical gradient approximation of the distance bound gradient. Since most of the time is spent on ray intersection, the inefficient numerical gradient approximation has a negligible impact on rendering performance.
The simplicity with which implicit surfaces are incorporated in zeno makes it useful for visualization of mathematical tasks and investigation of new implicit surfaces. For example, a homotopy that removes a 720° twist from a ribbon without moving either end formed the basis for the animated short '*Air on the Dirac strings*' (Sandin et al. 1993), for which zeno rendered a segment. This homotopy is based heavily on interpolated quaternion rotations and was easily incorporated into zeno as a domain transformation after a quick search and analysis of the most extreme deformation in the homotopy (Hart et al. 1993).

4.2 Exhibition

The three tori in Fig. 7 are combined with the superelliptic blend described in Appendix D.2.

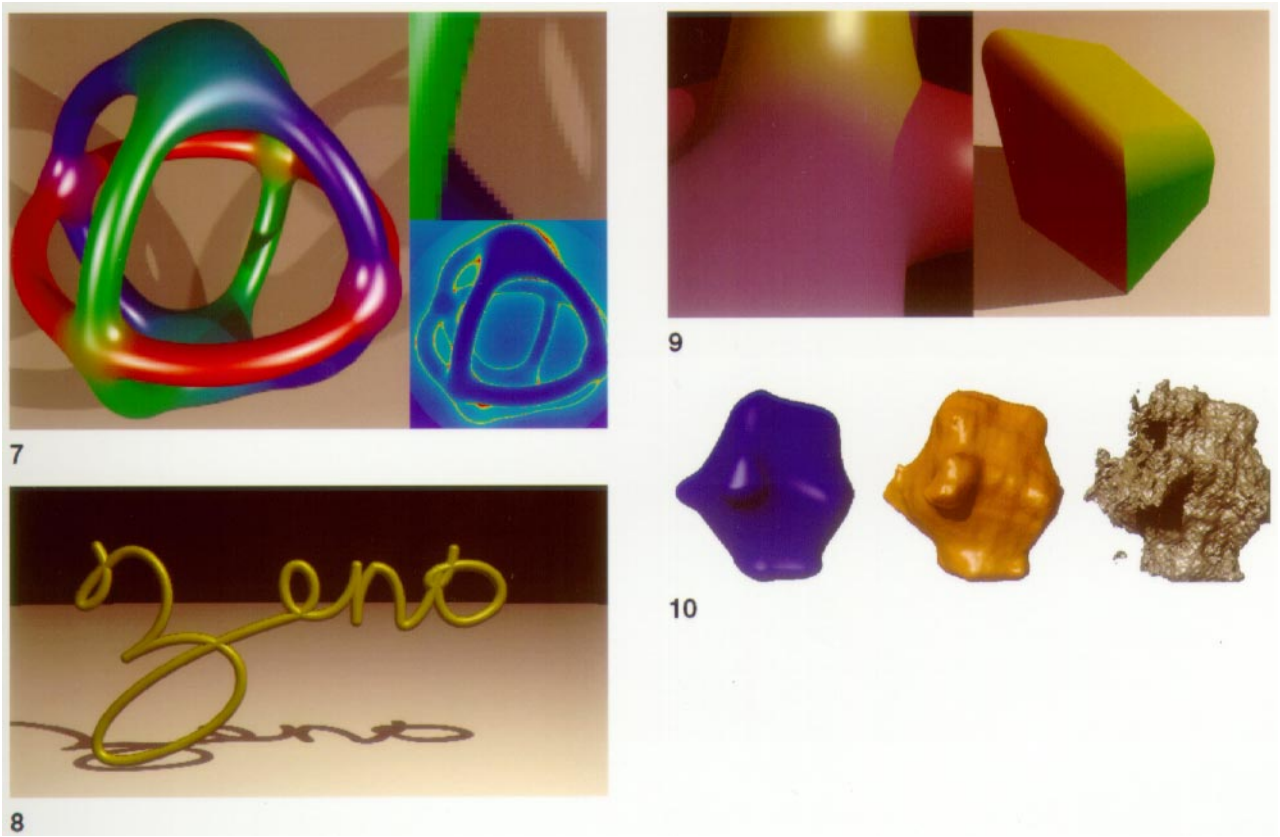


Fig. 7. Three blends of tori (left), blow-up (upper right), and work image (lower right)

Fig. 8. A logo for zeno

Fig. 9. Creases created by blended edges

Fig. 10. "Lava" (**a**) modeled as a sphere deformed by the noise function, "Muscle" (**b**) modeled with $\beta = 2$ noise, and "Rock" (**c**) modeled with $\beta = 1$ noise

The tori are all of major radius one, and minor radius one-tenth. The blue-green blend is quadratic, extending along the tori a radius of 0.5 from their intersection. The red-green blend also has the radius 0.5, but is degree eight. The red-blue blend is also degree eight, but has a radius of only 0.2.

Sphere tracing rendered Fig. 7 (left) in 12:47 at a resolution of only 256×256, using prefiltering to avoid the severe aliasing that ordinarily accompany such low sampling rates. Experiments on the difference of execution using point sampling and area sampling show that the increased execution time due to area sampling is negligible.

Although the superelliptic blend is implemented in zeno as a signed distance bound, it returns an underestimated distance of no less than 70% of the actual distance, which adequately indicated cone intersections as the enlargement demonstrates in Fig. 7 (upper right).

The work image in Fig. 7 (lower right) shows that sphere tracing concentrates on silhouette edges. Blue areas converge after ten iterations; green, around 50; and red, over 100.

Figure 8 demonstrates a generalized cylinder, from Appendix C, that has a skeleton consisting of a space curve modeled with 14 Bézier control polygons. Sphere tracing can render this scene in as little as 5:30 using bounding spheres to

eliminate unnecessary distance computations. The curved horizon is an artifact of the yonder clipping sphere of radius 1000 used to terminate ray stepping.

Figure 9 demonstrates the robustness of sphere tracing on creased surfaces. Both images were rendered with prefiltering at a resolution of 512×512, and in 16:48 for the cylinders, 12:36 for the cube.

The creases were created as CSG unions and intersections, defined implicitly by the continuous, but nondifferentiable, minimum and maximum operations from Sect. 2.4. The resulting edge was then merged into a third object with the aid of the pseudonorm blend from Appendix D.2. Such creased surfaces appear periodically in a variety of shapes, particularly in the modeling of biological forms.

Figure 10 illustrates the "noise" range deformation described in Appendix F. The left image uses a single octave of noise, whereas the next two use six octaves. Their amplitudes were scaled by $1/f^2$ and $1/f$, respectively, yielding a muscle texture and a rocky surface. The three images were each rendered at a resolution of 256×256 in (from left to right) approximately 5 min, 30 min, and 2 h. The great variation of distance estimates prohibited prefiltering the results of the noise function.

4.3 Analysis

Sphere tracing convergence is entirely linear, whereas other general root finders, such as interval analysis, have a linearly convergent root-isolation phase, followed by a quadratically convergent root-refinement stage. Work images, such as Fig. 7 (lower right), show that ray intersection is most costly at silhouette edges. During the sphere tracing of these edges, the distance to the surface is only a fraction of the distance to the ray intersection, which slows convergence. For other methods like interval analysis, silhouettes are double roots (which prevent root refinement), and their neighborhoods consist of closely spaced pairs of roots. Such root pairs are costly to separate with root-refinement methods that use midpoint subdivision since the distance between the two roots can be several orders of magnitude smaller than the initial interval.

Table 2. Comparison of execution times for enhanced sphere tracing of various scenes

Scene	Execution time (m:s)	Relative time (%)	Enhancement
Single sphere	2:00	100	None
	1:23	69	Convexity
Nine spheres/ plane	2:53	100	None
	1:42	59	Convexity
	1:19	46	Triangle inequality
	1:10	40	Both
Zeno logo	26:29	100	None
	19:23	73	Triangle inequality
	5:28	21	Bounding spheres
Lava	4:37	1	(Single noise)
Muscle	33:52	7.3	($1/f^2$ Noise)
Rock	2:06:56	27.5	($1/f$ Noise)

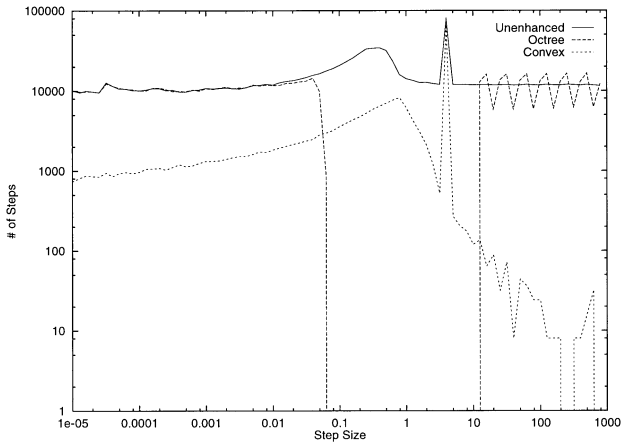
The convexity enhancement hastened convergence by 31% as shown in Table 2. With more primitives, this same table shows that the triangle inequality enhancement more than doubles the convergence rate, and when combined with convexity, it enhances ordinary sphere tracing by 60%.

Table 2 also compares various enhanced rendering times for the zeno logo. The fact that all 14 Bézier curves were nearly equidistant from the eye prevented the triangle inequality from significantly reducing unnecessary distance evaluations until sphere tracing had traversed much of each ray.

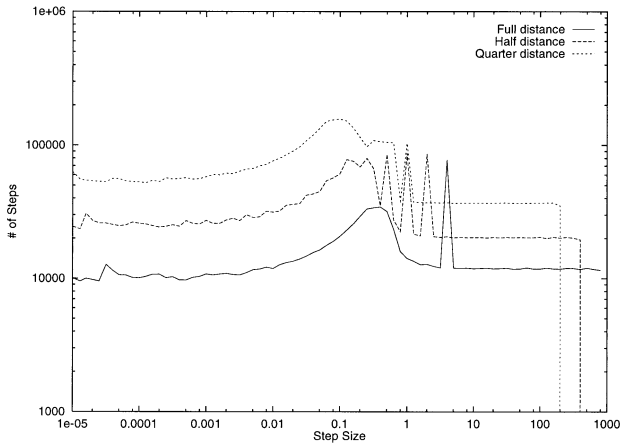
Figure 11 reveals the distribution of step sizes used in sphere tracing a ball. This histogram counted only the distance evaluations used to intersect primary (eye) rays.

Unimproved sphere tracing is evenly distributed, with a small hump in the middle. An octree replaces the increased distance computation in this humped area with octree parsing overhead (which this histogram does not measure). Echoes of the octree bounds cause the oscillations at the high end of its spectrum, whereas the low end adheres to the unenhanced performance. Experiments on simple scenes failed to demonstrate any increased performance from the octree enhancement, although more complicated scenes are likely to benefit from its use.

The convex histogram demonstrates the power of this enhancement. Its slope on the left confirms the expectation from Sect. 2.5.5 that it provides



11



12

Fig. 11. Histogram of step sizes for sphere tracing a ball

Fig. 12. Halving step sizes doubles the convergence time

sphere tracing a faster order of convergence. The right side of this histogram is significantly reduced, due to the cessation of stepping after moving beyond the sphere.

The spike in the unenhanced and convex graphs indicates the distance from the eye to the ball, which is the first step taken by every ray emanating from the eye point. One can remove these spikes from the graph by measuring this distance once and refering to it as the first step for rays emanating from the eye point, and likewise for the light sources. This "head start" barely improved the performance in the experiments.

Similar histograms given by Zuiderveld et al. (1992) measure performance logarithmically in the number of steps, but linearly in step size. As a result, their graphs are more logarithmically shaped than Fig. 11.

The accuracy of the distance estimate is directly proportional to the rate of convergence. Experiments on a sphere show that half the distance doubles the number of steps. The step-size histograms in Fig. 12 reveals the effects of distance underestimation.

The relationship between distance accuracy and sphere-tracing performance suggests that in certain cases a slower signed distance function may perform better than a fast distance underestimate. For example, consider the distance to an ellipsoid with major axes of radii 100, 100, and 1 modeled as a nonuniform scale transformation of the unit sphere. Appendix E yields a signed distance bound that returns at best the distance to the ellipsoid, and at worst 1% of the distance, in closed form, whereas Hart (1994) yields a signed distance function that returns the exact distance at the expense of several Newton iterations. In this case, the signed distance function would likely result in better performance.

Finally, the Lipschitz constants of the noise functions are 3 for single noise, 6 for $1/f^2$ noise, and 18 for $1/f$ noise (six octaves). The timings in Table 2 corresponding to the images in Fig. 10 show that the $1/f^2$-noise-rendering time was actually 7.3 times (instead of the expected value of twice) the single noise time. The likely reason is that the $1/f^2$ noise invokes the noise function six times more than the single noise function (yielding an expected value of 12 times). The $1/f$-noise-rendering time was 27.5 times longer than that of single noise (less than the expected 36 times), and 3.75 times longer than the $1/f^2$ noise (slightly larger than the expected value of 3).

5 Conclusion

Sphere tracing provides a tool for investigating a larger variety of implicit surfaces than previously possible.

With its enhancements and prefiltering, sphere tracing becomes a competitive implicit surface renderer of presentation quality. In particular, the convexity enhancement greatly increases

rendering speeds, and the triangle inequality is quite effective for large assortments of objects. Bounding volumes also increase the rendering performance as expected. However, techniques based on image coherence and space coherence (octree) do not perform as well.

Whereas sphere tracing was significantly slower than standard ray tracing on simple objects consisting of quadrics and polygons, it excelled at rendering the results of sophisticated geometric modeling operations.

The geometric nature of sphere tracing adapts it to symbolic prefiltering, supporting antialiasing at a nominal overhead.

In lieu of direct experimental comparison, several theoretical arguments show sphere tracing as a viable alternative to interval analysis and LG surfaces.

5.1 Further research

Sphere tracing demonstrates the utility of signed distance functions in the task of rendering geometric implicit surfaces. We expect that these functions will similarly enhance other applications, particularly in the area of geometric processing. As geometric distance becomes more important in computer-aided geometric design and other areas of modeling, the demand for more efficient geometric distance algorithms will increase.

In retrospect, the use of the Euclidean distance metric seems an arbitrary choice for sphere tracing. The linear nature of the chessboard and Manhatten metrics may result in more efficiently computed distances and ray intersection. "Cube-tracing" and "octahedron-tracing" algorithms are left as further research.

Acknowledgements. Thanks go to T. DeFanti and L. Smarr for procuring the support for the first year of this research. This research was subsequently supported by the National Science Foundation (NSF) under the Research Initiation Award No. CCR-9309210, and its implementation and distribution supported by a Research Experience for Undergraduates extension. The research was performed at the Imaging Research Laboratory, which is supported in part by NSF grants No. CDA-9121675 and No. CDA-9422044.

Special thanks go to A. Norton who, in 1989, encouraged me to apply sphere tracing to nonfractal models. I was greatly helped with the task of tracking down and deriving the distances in the appendix by conversations with A. Barr, C. Bajaj, C. Gunn, P. Hanrahan, J. Kajiya, D. Mitchell, and A. Rockwood. I also thank the anonymous reviewers, and also J. Bloomenthal, for many insightful comments and recommendations. B. Wyvill and J. Bloomenthal deserve special mention for inspiring this research by putting together an excellent course on implicit surfaces at SIGGRAPH '90.

Appendix A. Distance to natural quadrics and torus

These appendices derive signed distance functions, bounds and Lipschitz constants, and bounds for a variety of primitives and operations in the hope that they will aid in the implementation of sphere tracing. They may also serve as a tutorial in developing signed distance functions, bounds and Lipschitz constants, and bounds for other primitives and operations.

Distances to the standard solid modeling primitives are listed. The geometric rendering algorithm is not as efficient as the standard closed-form solutions. However, these distances are useful when the primitives are used in higher-order constructions such as blends and deformations.

Plane. The signed distance to a plane P with a unit normal $\boldsymbol{n}$ intersecting the point $r\boldsymbol{n}$ is

$$d(\boldsymbol{x}, P) = \boldsymbol{x} \cdot \boldsymbol{n} - r\,. \tag{20}$$

Sphere. A sphere is defined as the locus of points a fixed distance from given point. The distance to the unit sphere S about at the origin hence given by:

$$d(\boldsymbol{x}, S) = \|\boldsymbol{x}\| - 1\,. \tag{21}$$

Through domain transformations (Appendix E), the radius and location of the sphere may be changed. The sphere may even become an ellipsoid, though this reformulates the signed distance function into one requiring the solution to a sixth degree polynomial (Hart 1994). Through alternate distance metrics (Appendix B), the sphere can become a superellipsoid. These techniques generalize the rest of the basic primitives as well.

Cylinder. The distance to a unit-radius cylinder centered about the z-axis is found by projecting into the xy-plane and measuring the distance to

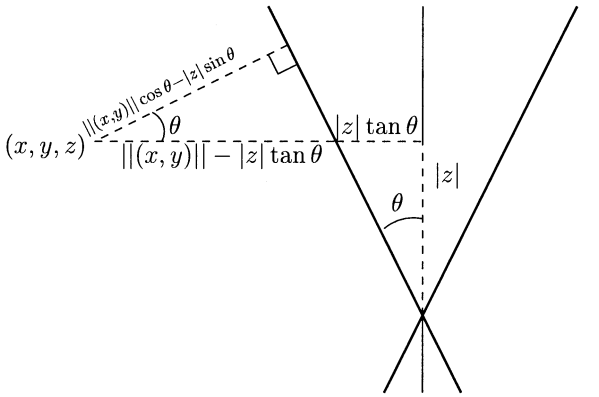


Fig. 13. Geometry for the distance to a cone

the unit circle

$$d(\boldsymbol{x}, Cyl) = \|(x, y)\| - 1\,. \tag{22}$$

Note that, in Eq. 22, and throughout the rest of the appendix, $\boldsymbol{x} = (x, y, z)$.

Cone. The distance to a cone centered at the origin oriented along the z-axis is

$$d(\boldsymbol{x}, Cone) = \|(x, y)\| \cos\theta - |z| \sin\theta\,, \tag{23}$$

where θ is the angle of divergence from the z-axis. The trigonometry behind its derivation is illustrated by Fig. 13.

Torus. The torus is the product of two circles, and its distance is evaluated as such

$$d(\boldsymbol{x}, T) = \|(\|(x, y)\| - R, z)\| - r \tag{24}$$

for a torus of major radius R and minor radius r, centered at the origin and spun about the z-axis.

Appendix B. Distance to superquadrics

Superquadrics (Barr 1981) result from the generalization of distance metrics. Distance to the basic primitives all used the $\|\cdot\|$ operator. In two dimensions, this operator generalizes to the p-norm

$$\|(x, y)\|^p = (|x|^p + |y|^p)^{1/p}\,. \tag{25}$$

which, when $p = 2$, becomes the familiar euclidean metric whose *circle* is a round circle. The Manhattan metric ($p = 1$) has a diamond for its *circle*. Taking the limit as $p \to \infty$ results in the chessboard metric

$$\|(x, y)\|^\infty = \max x, y\,, \tag{26}$$

where a square forms its *circle*. The other intervening values for p produce rounded variations on these basic shapes, and setting $0 < p < 1$ produces pinched versions. Generalized spheres, so-called *superellipsoids*, are produced by a pq-norm as

$$\|(x, y, z)\|^{pq} = \|(\|(x, y)\|^p, z)\|^q\,. \tag{27}$$

The natural quadrics now generalize to superquadrics, and tori likewise become supertori, and their distances are measured in the appropriate metric. One unifying metric space must be used for the distances to be comparable. Hence, pq-norm distances must be converted into euclidean distances.

Let $f(\boldsymbol{x})$ return a pq-norm distance to its implicit surface. This distance defines the radius of an unbounding superellipsoid. The radius of the largest euclidean sphere r_e inscribed within the pq-norm superellipsoid of radius r_s (in the pq-norm metric) is given by:

$$r_e = \begin{cases} r_s \Big/ \left\| \left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3} \right) \right\|^{pq} & \text{if } p \text{ or } q < 2 \\ r_s & \text{otherwise}\,. \end{cases} \tag{28}$$

Appendix C. Distance to offset surfaces

Given some closed skeleton geometry $S \subset \mathbb{R}^3$, the *global* offset surface is defined geometrically by the implicit equation

$$d(\boldsymbol{x}, S) - r = 0\,. \tag{29}$$

We define local offsets parametrically, using the normal of the skeleton geometry. Global offsets are the more desirable representation (Hoffman 1989), and in particular avoid interior surfaces that can cause problems in ray tracing and CSG (Wijk 1984).

The offset of an algebraic implicit surface is algebraic, though of higher degree in general. Several techniques have been developed to approximate offset surfaces with lower-degree representations. Treating offset surfaces geometrically overcomes the problems of dealing with high-degree algebraic representations and the loss of precision of low-degree approximations.

One useful skeletal model is the generalized cylinder, such as the fixed-radius global offset surface of a Bézier curve. Define the space curve parametrically as the image of the function $\boldsymbol{p}: \mathbb{R} \to \mathbb{R}^3$. Without loss of generality, assume the point from which we want to find the distance to the space curve is the origin.

Let $\boldsymbol{p}(u)$ define a cubic Bézier space curve. The point on the space curve closest to a given point $\boldsymbol{x}$ occurs either at one of the end points, or at point $\boldsymbol{p}(u)$ on the space curve such that

$$(\boldsymbol{x} - \boldsymbol{p}(u)) \cdot \boldsymbol{p}_u(u) = 0\,. \tag{30}$$

Equation 30 can be converted into a degree-five 1D Bézier curve (Schneider 1990), and can be solved efficiently using a technique described in (Rockwood et al. 1989). Such a generalized cylinder is demonstrated in Fig. 8 in Sect. 4.2.

Appendix D. Distance to blended objects

Blends smoothly join nearby objects and have found applications in image synthesis and computer-aided geometric design.

D.1 Soft metablobbies

Blinn (1982) uses a gaussian distribution function to produce a blending function that has come to be known as the "blobby" model. "Soft" objects approximate gaussian distribution with a sixth degree polynomial to avoid exponentiation and localize the blends (Wyvill et al. 1986). "Metaballs" approximate gaussian distributions with piecewise quadratics to avoid exponentiation and iterative root finding (Nishimura et al. 1985). Following (Wyvill et al. 1986), the following piecewise cubic in distance r

$$C_R(r) = \begin{cases} 2\dfrac{r^3}{R^3} - 3\dfrac{r^2}{R^2} + 1 & \text{if } r < R\,, \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

approximates a gaussian distribution.

Reformulating this function to accommodate the implicit surface definitions in this paper, Eq. 31 forms the basis for *a soft* implicit surface consisting of n key points $\boldsymbol{p}_i$ with radii R_i, and threshold T, defined by the function

$$f(\boldsymbol{x}) = T - \sum_{i=1}^{n} C_{R_i}(\|\boldsymbol{x} - \boldsymbol{p}_i\|)\,. \tag{32}$$

Negative key points are incorporated into the model by negating the value returned by $C_{R_i}()$.

Theorem 5. *The distance to the implicit blend B defined by Eq.* 32 *is bounded by*

$$d(\boldsymbol{x}, B) \geq \frac{2}{3} f(\boldsymbol{x}) \sum_{i=1}^{n} R_i\,. \tag{33}$$

Proof. Repeated differentiation of Eq. 31 produces

$$C'(r) = 6\frac{r^2}{R^3} - 6\frac{r}{R^2} \tag{34}$$

$$C''(r) = 12\frac{r}{R^3} - \frac{6}{R^2}\,. \tag{35}$$

Solving $C''(r) = 0$ yields the maximum slope, which occurs at the midpoint $r = R/2$. Its Lipschitz constant is given by

$$\operatorname{Lip} C(r) = |C'(R/2)| = \frac{3}{2R}\,. \tag{36}$$

The Lipschitz constant of a sum is bounded by the sum of the Lipschitz constants, which gives the result just noted. □

In practice, local Lipschitz bounds may be used for tighter distance bounds by taking the first summation in Eq. 33 over key points i with non-zero contributions. Additional efficiency results from the use of bounding volumes of radius R_i surrounding the key points $\boldsymbol{p}_i$, as detailed by Wyvill and Trotman (1990).

D.2 Superelliptic blends

Rockwell and Owen's (1987) pseudonorm blend returns the p-norm distance to the blended union of implicit surfaces of signed distance functions. Hence, with the techniques from Appendix B, sphere tracing can render pseudonorm-blended surfaces, as demonstrated in Figs. 3 and 5 in Sect. 4.2.
The pseudonorm blend creases the space surrounding the blend (Rockwood and Owen 1987). Such gradient discontinuities can be disastrous for some root finders, but do not impact sphere tracing.

Appendix E. Distances to transformed objects

Implicit surfaces are transformed by applying the inverse transformation to the space before applying the function. Let $\boldsymbol{T}(\boldsymbol{x})$ be a transformation and let $f(\boldsymbol{x})$ define the implicit surface. Then the transformed implicit surface is defined as the implicit surface of

$$f(\boldsymbol{T}^{-1}(\boldsymbol{x})) = 0\,. \tag{37}$$

The Lipschitz constant of the composition is no greater than the product of the component Lipschitz constants. We are concerned with the Lipschitz constant of the transformation inverse, which is not necessarily the inverse of the Lipschitz constant of the transformation.

Isometry. Isometries are transformations that preserve distances. If $\boldsymbol{I}$ is an isometry, the distance returned by f needs no adjustment.

$$d(\boldsymbol{x}, \boldsymbol{I} \circ f^{-1}(0)) = d(\boldsymbol{I}^{-1}(\boldsymbol{x}), f^{-1}(0))\,. \tag{38}$$

Isometries include rotations, translations, and reflections.

Uniform scale. A uniform scale is a transformation $\boldsymbol{S}(\boldsymbol{x})$ of the form

$$\boldsymbol{S}(\boldsymbol{x}) = s\boldsymbol{x}\,, \tag{39}$$

where s is the scale factor. The inverse $\boldsymbol{S}^{-1}$ is a scale by $1/s$. Hence, the distance to a scaled implicit surface is

$$d(\boldsymbol{x}, \boldsymbol{S}(f^{-1}(0))) = sd(\boldsymbol{S}^{-1}(\boldsymbol{x}), f^{-1}(0)) \tag{40}$$

and the Lipschitz constant of the inverse scale is $1/s$.

Linear deformation. The distance to the linear image of an implicit surface is found by determining the Lipschitz constant of the linear transformation's inverse, which is also a linear transformation.
The Lipschitz constant of an arbitrary linear transformation is found by the power method, which iteratively finds the largest eigenvalue of a matrix (Gerald and Wheatley 1989).

Taper. The taper deformation scales two axes by a function $r(\cdot)$ of the third axis (Barr 1984). The taper is defined as:

$$\text{taper}(\boldsymbol{x}) = (r(z)x, r(z)y, z)\,, \tag{41}$$

whereas its inverse differs only by using $r^{-1}(\cdot)$ instead of $r(\cdot)$. The Lipschitz constant of the inverse deformation is

$$\text{Lip taper} = \min_{z \in \mathbb{R}} r^{-1}(z)\,. \tag{42}$$

In other words, the Lipschitz constant of the inverse taper is the amount of its "tightest" tapering.

Twist. The twisting deformation rotates two axes by a linear function $a(\cdot)$ of the third axis. Twisting is defined as:

$$\text{twist}(\boldsymbol{x}) = \begin{pmatrix} x\cos a(z) - y\sin a(z), \\ x\sin a(z) + y\cos a(z), \\ z \end{pmatrix}, \tag{43}$$

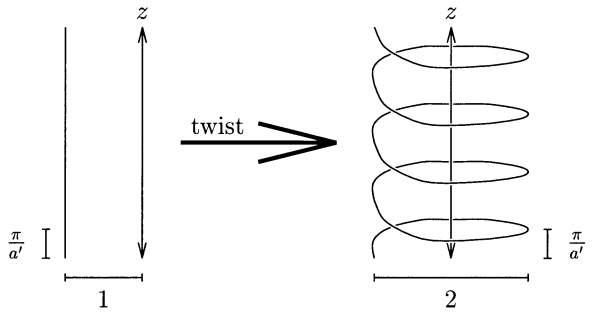


Fig. 14. Geometric calculation of the Lipschitz constant of the bounded twist deformation

whereas its inverse differs only by using $a^{-1}(\cdot)$ instead of $a(\cdot)$. Twisting is not Lipschitz on $\mathbb{R}^n$, since for any Lipschitz bound λ one can find two points $\mathbb{R}^n$ at a great distance from the twisting axis that are transformed farther apart by a ratio greater than λ. Thus, twisting must be constrained to a domain where it satisfies the Lipschitz criterion. One such domain is the unit cylinder oriented along the twisting axis. The Lipschitz constant of the twist is computed from the worst-case scenario within the bounds of the unit cylinder as illustrated in Fig. 14:

$$\operatorname{Lip}\text{twist} = \sqrt{4 + \left(\frac{\pi}{a'}\right)^2}\,. \tag{44}$$

Appendix F. Distance to hypertextures

The use of sophisticated noise functions has greatly increased the power of procedural models for making existing geometric representations more realistic. The recent work has applied stochastic textures directly to the geometry instead of altering the shading (Lewis 1989; Perlin and Hoffert 1989).

The original "hypertexture" system formulated implicit models for a variety of surface phenomena, including hair and fire. This appendix focuses on incorporating hypertexture's model of noise into sphere tracing, though the same techniques can be used to adapt the other hypertexture models as well.

"Hypertexture" treats solid procedural noise as a deformation, and it was designed for use with implicit surfaces. Its original ray-tracing algorithm stepped along the ray in fixed intervals. Determining a distance bound on a "hypertextured" shape allows sphere tracing to render its result more efficiently.

Band-limited solid noise results from the smooth interpolation of a lattice of random unit vectors. Condensing (Perlin and Hoffert 1989), the noise function is given by

$$\begin{aligned}\operatorname{noise}(x,y,z) = \sum_{k=\lfloor z\rfloor}^{\lfloor z\rfloor+1}\sum_{j=\lfloor y\rfloor}^{\lfloor y\rfloor+1}\sum_{i=\lfloor x\rfloor}^{\lfloor x\rfloor+1} & C_1(|x-i|)\\ & \times C_1(|y-j|)C_1(|z-k|)\Gamma(i,j,k)\\ & \cdot(x-i,y-j,z-k),\end{aligned} \tag{45}$$

where C_R is the cubic gaussian approximation (Eq. 31) used for soft objects, and Γ is an array of random unit vectors. From Theorem 5, we know that $\operatorname{Lip} C_1 = 3/2$. Two opposing vectors can be neighbors in Γ, so $\operatorname{Lip}\Gamma = 2$. Hence, their composition results in $\operatorname{Lip}\text{noise} = 3$.

Fractal noise is formed by summing scaled versions of the noise function

$$\operatorname{noise}_\beta(\boldsymbol{x}) = \sum_{i=0}^{n-1}\frac{\operatorname{noise}(2^i\boldsymbol{x})}{2^{\beta i}} \tag{46}$$

over n octaves (Perlin and Hoffert 1989).

For $\beta = 1$, the amplitude decreases proportionately to the increase in frequency, so its Lipschitz constant equals the sum of the individual noise functions:

$$\operatorname{Lip}\operatorname{noise}_{\beta=1} = 3n\,. \tag{47}$$

Thus $\beta = 1$ noise is not Lipschitz, but its band-limited form for finite n is.

For $\beta = 2$ noise, the amplitude decreases geometrically as the frequency increases, resulting in

$$\operatorname{Lip}\operatorname{noise}_{\beta=2} = 3\left(2 - \frac{1}{2^{n-1}}\right) \le 6\,. \tag{48}$$

Hence, Brownian motion is Lipschitz (this can also be derived from the definition of brownian motion as the integral of white noise).

Sphere tracings of noise-textured spheres appear in Fig. 10.

References

Agin GJ, Binford TO (1976) Computer description of curved objects. IEEE Trans Comput C-25:439–449

Amanatides J (1984) Ray tracing with cones. Comput Graph 18:129–135

Ballard DH, Brown CM (1982) Comput Vision. Prentice-Hall, Englewood Cliffs, NJ

Barnhill RE, Frost TM, Kersey SN (1992) Self-intersections and offset surfaces. In: Barnhill RE (ed) Geometry processing for design and manufacture, SIAM, Philadelphia, pp 35–44

Barr AH (1981) Superquadrics and angle-preserving transformations. IEEE Comput Graph Appl 1:11–23

Barr AH (1984) Global and local deformations of solid primitives. Comput Graph 18:21–30

Blinn JF (1982) A generalization of algebraic surface drawing. ACM Trans Graph 1:235–256

Bloomenthal J (1988) Polygonization of implicit surfaces. Comput Aided Geom Design 5:341–355

Bloomenthal J, Wyvill B (1990) Interactive techniques for implicit modeling. Comput graph 24:109–116

Bloomenthal J, Shoemake K (1991) Convolution surfaces. Comput Graph 25:251–256

Bronsvoort WF, Klok F (1985) Ray tracing generalized cylinders. ACM Trans Graphs 4:291–303

Gerald CF, Wheatley PO (1989) Applied numerical analysis. Addison-Wesley, Reading, Mass

Goldman RN (1983) Two approaches to a computer model for quadric surfaces. IEEE Comput Graph Appl 3:21–24

Hanrahan P (1983) Ray tracing algebraic surfaces. Comput Graph 17:83–90

Hart JC (1994) Distance to an ellipsoid. In: Heckbert P (ed) Graphics Gems IV. Academic Press, New York, pp 113–119

Hart JC, DeFanti TA (1991) Efficient antialiased rendering of 3-D linear fractals. Comput Graph 25:91–100

Hart JC, Sandin DJ, Kauffman LH (1989) Ray tracing deterministic 3-D fractals. Comput Graph 23:289–296

Hart JC, Francis GK, Kauffman LH (1993) Visualizing quaternion rotation. ACM Transaction on Computer Graphics 13:256–276

Hoffman CM (1989) Geometric and solid modeling. Morgan Kaufmann, San Mateo, CA

Kalra D, Barr AH (1989) Guaranteed ray intersections with implicit surfaces. Comput Graph 23:297–306

Kaplansky I (1977) Set theory and metric spaces. Chelsea, New York

Kay TL, Kajiya JT (1986) Ray tracing complex scenes. Comput Graph 20:269–278

Lewis JP (1989) Algorithms for solid noise synthesis. Comput Graph 23:263–270

Mitchell DP (1990) Robust ray intersection with interval arithmetic. Proceedings of Graphics Interface '90, Conference held in Halifax Nova Scotia. Morgan Kauffman, Palo Alto, CA, pp 68–74

Nishimura H, Hirai M, Kawai T, Kawata T, Shirakawa I, Omura K (1985) Object modeling by distribution function and a method of image generation (in Japanese). Proceedings of the Electronics Communication Conference '85, pp 718–725

Perlin K, Hoffert EM (1989) Hypertexture. Comput Graph 23:253–262

Porter T, Duff T (1984) Compositing digital images. Comput Graph 18:253–259

Pratt V (1987) Direct least-squares fitting of algebraic surfaces. Comput Graph 21:145–152

Ricci A (1974) A constructive geometry for computer graphics. Comput J 16:157–160

Rockwood AP (1989) The displacement method for implicit blending surfaces in solid models. ACM Trans Graph 8:279–297

Rockwood AP, Owen JC (1987) Blending surfaces in solid modeling. In: Farin G (ed) Geometric modelling. SIAM, Philadelphia, pp 367–383

Rockwood A, Heaton K, Davis T (1989) Real-time rendering of trimmed surfaces. Comput Graph 23:107–116

Roth SD (1982) Ray casting for modeling solids. Comput Graph Image Processing 18:109–144

Sandin DJ, Kauffman LH, Francis GK (1993) Air on the Dirac strings. SIGGRAPH Video Review 93 (Animation)

Schneider PJ (1990) Solving the nearest-point-on-curve problem. In: Glassner AS (ed) Graphics Gems I. Academic Press, Boston, pp 607–611

Sederberg TW, Zundel AK (1989) Scan line display of algebraic surfaces. Comput Graph 23:147–156

Stander BT, Hart JC (1994) A Lipschitz method for accelerated volume rendering. Proceedings of the Volume Visualization Symposium '94, IEEE CS, Piscataway, NJ, pp 107–114

Taubin G (1994) Distance approximations for rasterizing impicit curves. ACM Trans Graph 13:3–42

Thomas D, Netravali AN, Fox DS (1989) Antialiased ray tracing with covers. Comput Graph Forum 8:325–336

Thompson K (1990) Area of intersection: circle and a half-plane. In: Glassner AS (ed) Graphics Gems I. Academic Press, Boston

Von Herzen B, Barr AH (1987) Accurate triangulations of deformed, intersecting surfaces. Comput Graph 21:103–110

Von Herzen B, Barr AH, Zatz HR (1990) Geometric collisions for time-dependent parametric surfaces. Comput Graph 24:39–48

van Wijk J (1984) Ray tracing objects defined by sweeping a sphere. Proceedings of Eurographics '84, Elsevier, North-Holland, pp 73–82

Wyvill G, Trotman A (1990) Ray tracing soft objects. Proceedings of Computer Graphics International '90. Springer, Berlin Heidelberg New York, pp 467–476

Wyvill G, McPheeters C, Wyvill B (1986) Data structure for soft objects. Visual Comput 2:227–234

Zuiderveld KJ, Koning AHJ, Viergever MA (1992) Acceleration of ray-casting using 3-D distance transforms. Proceedings of Visualization in Biomedical Computing. 1808:324–335

JOHN C. HART is an assistant professor in the School of Electrical Engineering and Computer Science at Washington State University, and a member of its Imaging Research Laboratory. His research focuses on fractal geometry, implicit surfaces, natural modeling and scientific visualization. He received his M.S. in 1989 and Ph.D. in 1991 in computer science from the Electronic Visualization Laboratory at the University of Illinois at Chicago, and his B.S. in 1987 from Aurora University. Hart is a member of SIGGRAPH, the ACM and the IEEE Computer Society, and is currently the SIGGRAPH Director for Communications.