Ray Tracing Implicit Surfaces *

John C. Hart
School of EECS
Washington State University
Pullman, WA 99164-2752
hart@eecs.wsu.edu

Abstract

This is an overview of methods for ray tracing implicit surfaces, including the primary problem of intersecting a ray with an implicit surface, and the secondary problems of determining surface normals, and incorporating CSG operations and deformations.

1 Introduction

An implicit surface is defined by a function $f : \mathbf{R}^3 \to \mathbf{R}$ that assigns a scalar value to each point in space. The surface is the set of points $\mathbf{x} \equiv (x, y, z) \in \mathbf{R}^3$ such that $f(\mathbf{x}) = 0$. Let A be a closed solid described by the implicit function f. Then, for uniformity, we assume

$$\begin{aligned} \mathbf{x} \in \overset{\circ}{A} &\Leftrightarrow f(\mathbf{x}) < 0 & (1) \\ \mathbf{x} \in \partial A &\Leftrightarrow f(\mathbf{x}) = 0 & (2) \\ \mathbf{x} \in \mathbf{R}^3 - A &\Leftrightarrow f(\mathbf{x}) > 0. & (3) \end{aligned}$$

This states, via point-set topology, that the implicit function is negative inside the solid, zero on its surface, and positive outside.

(This standard is the most common in recent implicit surface literature. Other examples include: [Ricci, 1974], where implicit functions were always positive, and unit-valued on the surface, less than one inside and greater than one outside; and [Blinn, 1982], where implicit functions were zero on the surface, negative outside and positive inside.)

Points on a ray are defined parametrically by a function $\mathbf{r} : \mathbf{R} \to \mathbf{R}^3$ as

$$\mathbf{x} = \mathbf{r}(t) = \mathbf{x}_o + t\mathbf{x}_d \qquad (4)$$

*Also appeares in the SIGGRAPH '93 Course Notes: Design, Visualization and Animation of Implicit Surfaces

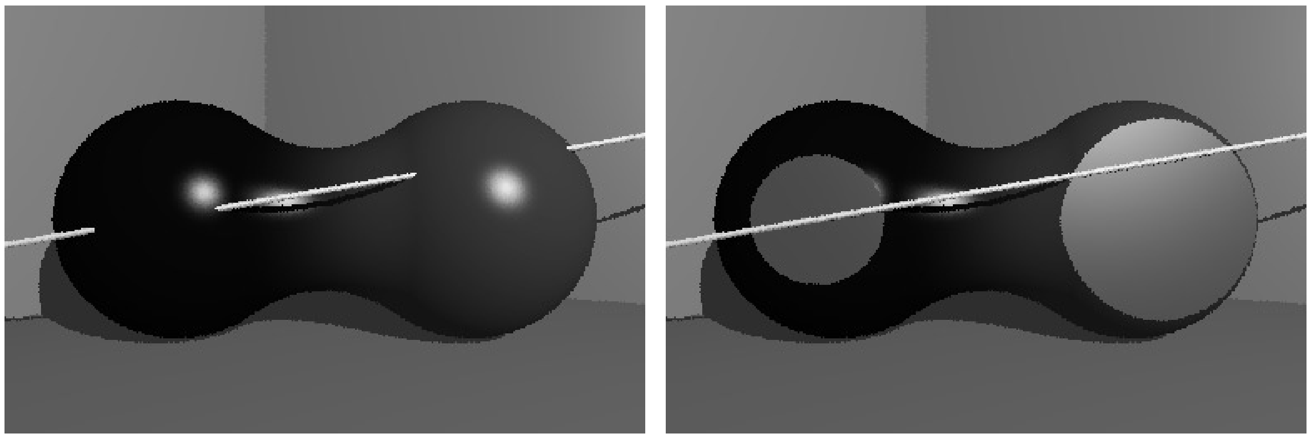

Figure 1: Two blended spheres (left) pierced by the ray shown in the cut-away (right).

where $\mathbf{x}_o$ is the anchor, and $\mathbf{x}_d$ is the direction, of the ray. The direction $\mathbf{x}_d$ is typically but not necessarily unit length. In the following examples, the ray is anchored at the origin point $\mathbf{x}_o = (-4.91, -0.67, 0.364)$ and extends parallel to the unit direction vector $\mathbf{x}_d = (0.981, 0.174, 0.0872)$.

Consider two spheres of unit radius, centered at $(\pm 1.2, 0, 0)$. These spheres are implicitly described by the algebraic functions

$$f(x, y, z) = (x - 1.2)^2 + y^2 + z^2 - 1 \tag{5}$$

and

$$g(x, y, z) = (x + 1.2)^2 + y^2 + z^2 - 1. \tag{6}$$

A third function h produces a yet another implicit surface by blending the implicit surfaces of f and g. (See [Woodwark, 1987] for an excellent survey of the variety of implicit surface blending methods.) The following examples find the intersection of a ray with two blended spheres, as shown in Figure 1

Consider the implicit surface defined by f and the ray defined by $\mathbf{r}$. The intersection of the ray with the surface is found by finding the smallest positive value t such that $\mathbf{x} = \mathbf{r}(t)$ is a point such that $f(\mathbf{x}) = 0$. Composing f with $\mathbf{r}$ produces the composite real function $F : \mathbf{R} \to \mathbf{R}$ as

$$F(t) = f \circ \mathbf{r}(t). \tag{7}$$

The intersections of the ray defined by $\mathbf{x} = \mathbf{r}(t)$ with the implicit surface defined by $f(\mathbf{x})$ are given by the values of t which cause $F(t) = 0$.

For example, the specific values given above produce the quadratic function

$$\begin{aligned} F(t) &= f \circ \mathbf{r}(t) && (8)\\ &= f\left((-4.905 + 0.981t, -0.67 + 0.174t, 0.364 + 0.0872t)\right) && (9)\\ &= (-6.105 + 0.981t)^2 + (-0.67 + 0.174t)^2 + (0.364 + 0.0872t)^2 - 1 && (10)\\ &= t^2 - 12.1t + 36.9 && (11) \end{aligned}$$

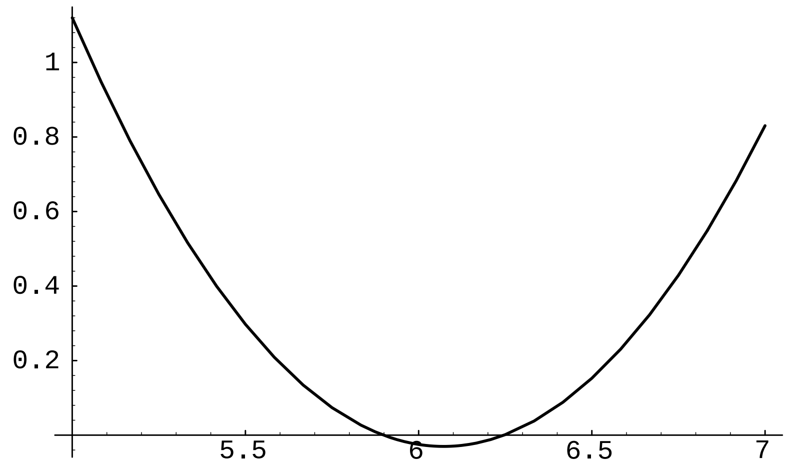

Figure 2: The graph of $F(t)$. The ray intersects the sphere when the graph intersects the x-axis.

which is plotted in Figure 2. The two zeros of this simple parabola can be found via the quadratic equation.

2 Blinn's Blobs

In [Blinn, 1982], spheres were represented as the implicit surfaces of Gaussian distributions, and their blend is the sum of these distributions. In this case, given two implicit surfaces defined by functions f and g, their blend is the implicit surface defined by the function

$$h(\mathbf{x}) = 1 - e^{-f(x)} - e^{-g(x)}. \tag{12}$$

(According to the parameters in [Blinn, 1982], (12) was constructed with *threshhold:* $T = 1$ and *blobbiness:* $B = -1$.)

Letting

$$\begin{aligned} H(t) &= h \circ \mathbf{r}(t) && (13)\\ &= 1 - e^{-f \circ \mathbf{r}(t)} - e^{-g \circ \mathbf{r}(t)} && (14)\\ &= 1 - e^{-F(t)} - e^{-G(t)} && (15) \end{aligned}$$

produces the function plotted in Figure 3.

The derivatives for our particular F, G and H are given as

$$\begin{aligned} F'(t) &= 2(r_x(t) - 1.2)r_x'(t) + 2r_y(t)r_y'(t) + 2r_z(t)r_z'(t) && (16)\\ G'(t) &= 2(r_x(t) + 1.2)r_x'(t) + 2r_y(t)r_y'(t) + 2r_z(t)r_z'(t) && (17)\\ H'(t) &= e^{-F(t)}F'(t) + e^{-G(t)}G'(t), && (18) \end{aligned}$$

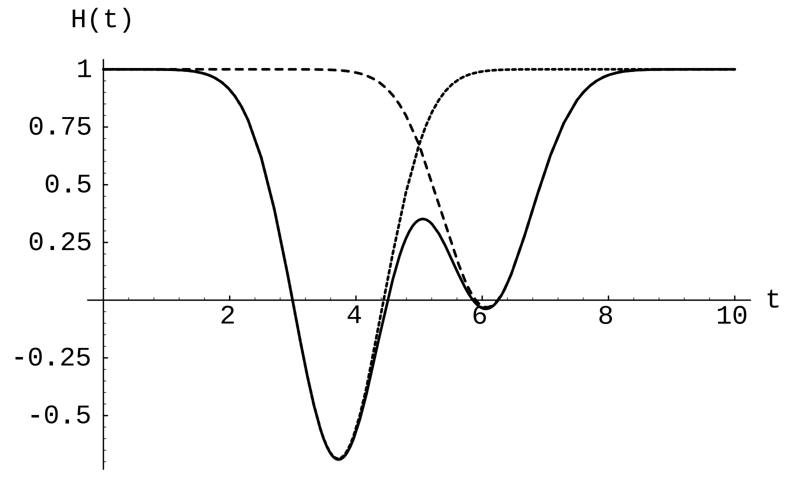

Figure 3: The graph of $H(t)$ for finding the intersection of a ray with blended spheres. The dashed curves represent the component functions $1 - e^{-F(t)}$ and $1 - e^{-G(t)}$.

where $\mathbf{r}(t) = (r_x(t), r_y(t), r_z(t))$. Alternatively [Mitchell, 1990], F', G' and H' are the directional derivatives

$$\begin{aligned} F'(t) &= \mathbf{x}_d \cdot \nabla f(\mathbf{r}(t)), & (19)\\ G'(t) &= \mathbf{x}_d \cdot \nabla g(\mathbf{r}(t)), & (20)\\ H'(t) &= \mathbf{x}_d \cdot \nabla h(\mathbf{r}(t)). & (21) \end{aligned}$$

Both Newton's method and *regula falsi* were used to find the roots of $H(t)$ in [Blinn, 1982]. Newton's method iteratively refines an initial guess at a root,

$$t_{i+1} = t_i - \frac{H(t_i)}{H'(t_i)}. \tag{22}$$

As demonstrated by Figure 4, Newton's method is unpredictable on general functions, and may converge to any of the roots, or may even diverge. Figure 5 shows which roots Newton's method converges to for each initial value t_0 in the complex plane (See [Peitgen *et al.*, 1984].)

Regula falsi is an improved binary search for any root. It assumes an initial interval $[t_0, t_1]$ such that $H(t_0)H(t_1) < 0$ — H has opposite signs at the endpoints of the interval. The *regula falsi* algorithm is described in Figure 6.

Unlike Newton's method, *regula falsi* does not explode, but like Newton's method, it does not necessarily converge onto the least positive root — the first ray intersection.

Papers since [Blinn, 1982] have solved $H(t)$ in two phases. First a *root isolation* phase slices up the domain into chunks containing a single root, then a *root refinement* phase hones in on the root to machine precision. The only serious flaw in [Blinn, 1982] is an ad hoc root isolation phase.

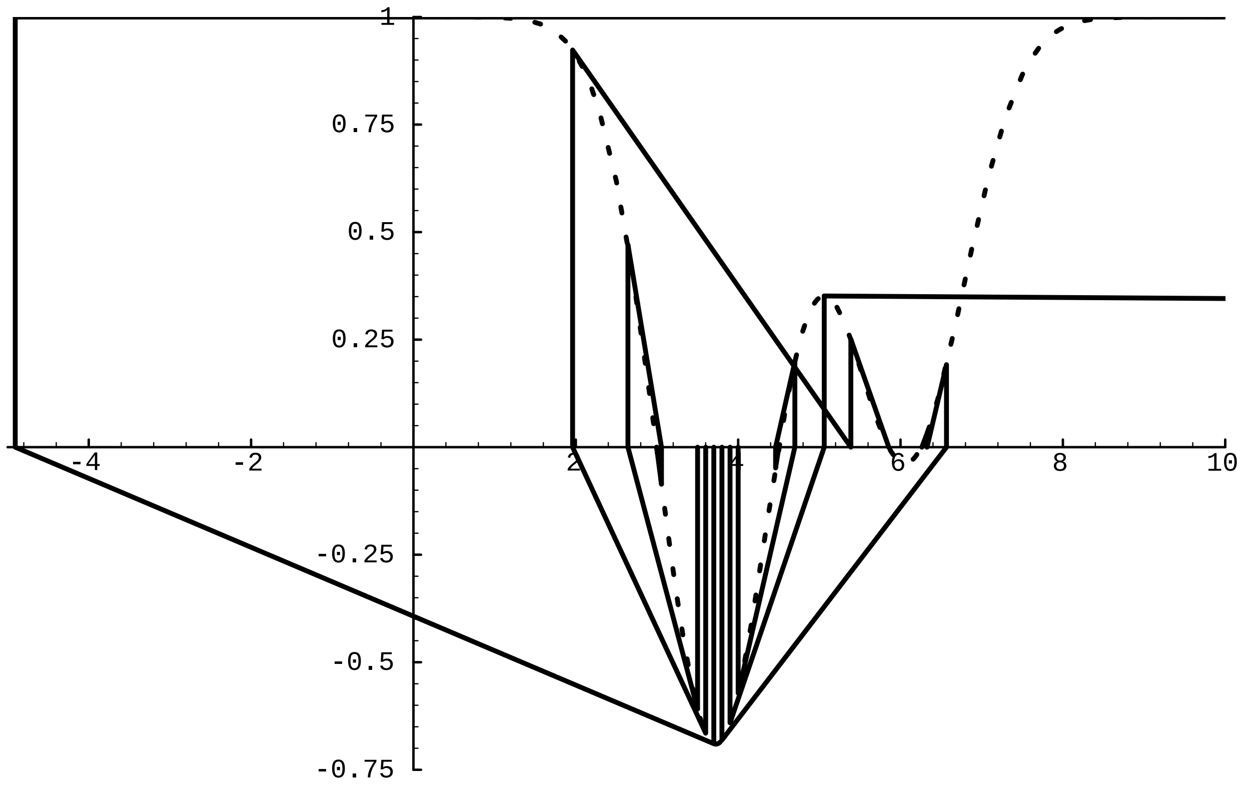

Figure 4: Newton's method for finding roots of H(t). At $t_0 = 3.5$ Newton's method converges on the first ray intersection. At $t_0 = 3.6$ the method converges to the third ray intersection. At $t_0 = 3.7$ the method shoots back to -5 where the slope is nearly zero, causing divergence. At $t_0 = 3.8$ the method converges on the fourth ray intersection. At $t_0 = 3.9$ the method is unlucky, finding a curve peak causing it to diverge. At $t_0 = 4.0$ the method converges to the second ray intersection.

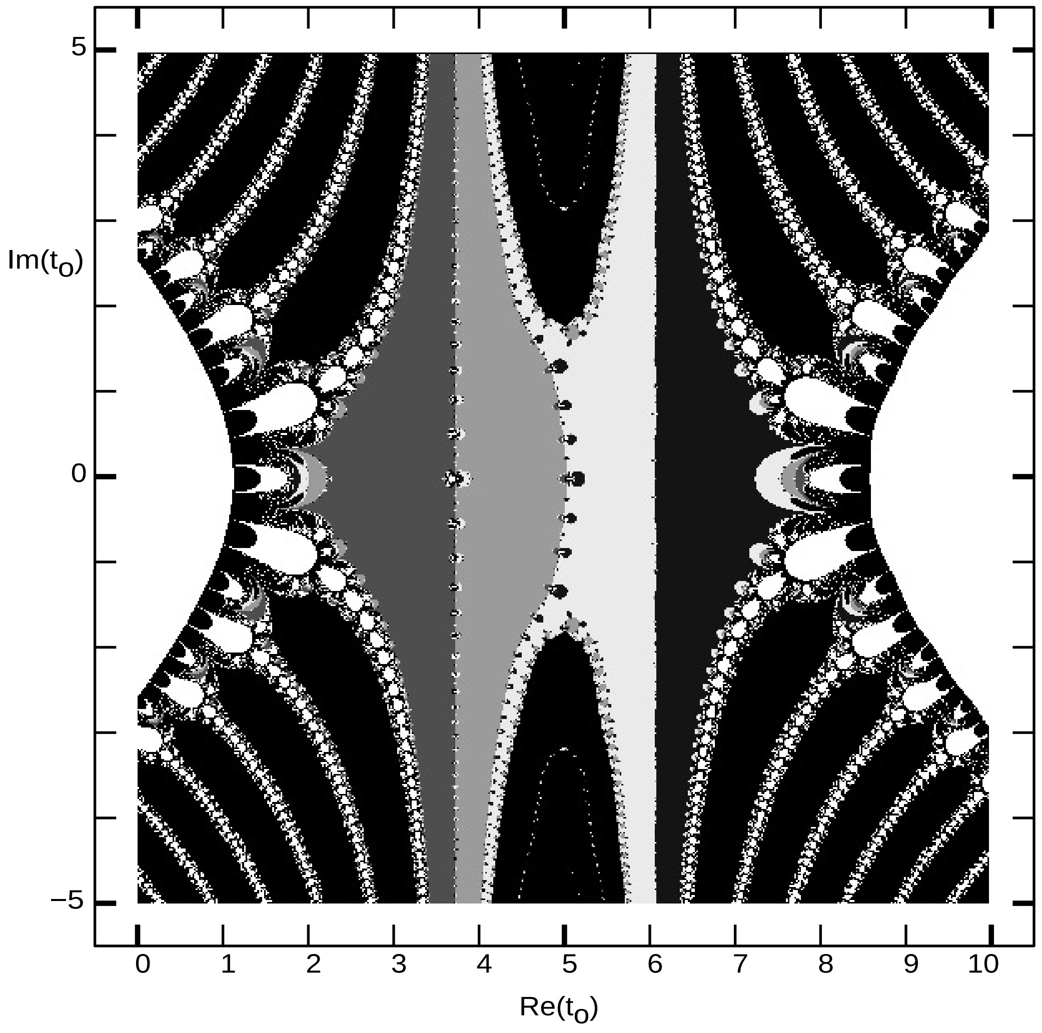

Figure 5: A map of the roots Newton's method converges to for initial points t_0 in the complex plane. Each shaded region indicates initial points that converge to a particular real root. The black regions converge to complex conjugate pairs, and the white regions diverge.

---

1. Let $t_m = (t_0 + t_1)/2$
2. If $H(t_m) = 0$ then return t_m.
3. If $H(t_0)H(t_m) < 0$ then repeat on $[t_0, t_m]$.
4. Otherwise, repeat on $[t_m, t_1]$.

Figure 6: The *regula falsi* algorithm.

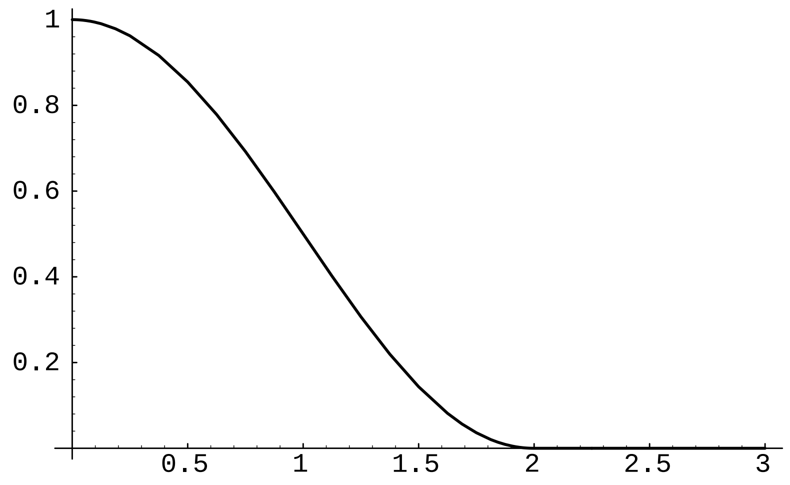

Figure 7: The function $C(r^2)$ plotted over r.

3 Polynomials

Root finding is much easier on nicer functions, like polynomials. Hence, there have been several polynomial approximations to the Gaussian distribution to make implicit surface rendering more efficient.

3.1 Metaballs

In [Nishimura *et al.*, 1985], the function $H(t)$ is approximated piecewise by quadratics. The individual components, the F and G, are each approximated by four quadratic curves. Their combination H is then segmented and each segment is represented by the quadratic resulting from the sum of the component quadratics. During ray intersection, the zeroes of each of the quadratic pieces are found and those falling within the appropriate domain denote ray intersections.

3.2 Soft Objects

In [Wyvill *et al.*, 1986], the Gaussian distribution e^{-r^2} is approximated by the polynomial function

$$C(r^2) = \begin{cases} -\frac{1}{144}(r^2)^3 + \frac{17}{144}(r^2)^2 - \frac{11}{18}r^2 + 1 & \text{if } r^2 < 4, \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

where r^2 is the squared distance to the center of a sphere. (Here the parameters are *magic* $= 1/2$ and $R = 2$.) This function is shown in Figure 7.

Under this model, the example blended spheres are represented by the implicit surface of the function

$$h(\mathbf{x}) = 1 - 2C \circ f(\mathbf{x}) - 2C \circ g(\mathbf{x}). \tag{24}$$

Letting $H(t) = h \circ \mathbf{r}(t)$ gives us the graph shown in Figure 8.

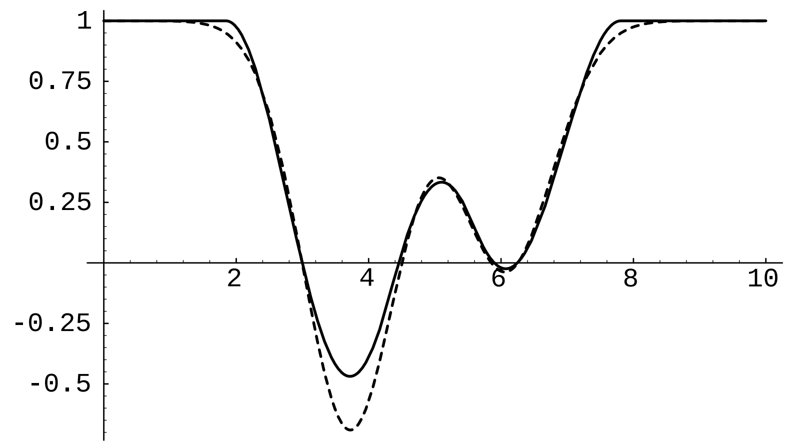

Figure 8: The polynomial $H(t)$ graph for soft objects (solid) compared to Blinn's exponential graph (dashed).

3.3 Algebraic Surfaces

In general, a polynomial function $f(x, y, z)$ implicitly describes an algebraic surface. The quadrics, including ellipsoids, cylinders and cones, are algebraic surfaces. So is the torus.

Algebraic blends of algebraic surfaces are described in [Hoffman & Hopcroft, 1985]. This method produces a blending surface which must be intergrated, via CSG, into the original model.

3.4 Polynomial Root Solving

The implicit function of an algebraic surface, when composed with the ray equation, produces a polynomial in t.

The quadratic equation finds all roots for degree two polynomials. Similar symbolic methods exist for finding roots of degree three and four polynomials. Finding the roots of degree five and greater polynomials requires numerical methods in general.

In [Hanrahan, 1983], DesCartes' rule of signs isolated the roots of a polynomial, and *regula falsi* refined them. The algorithm mapped $[t_0, t_1] \mapsto \mathbf{R}$, producing a new polynomial. If number of sign changes between neighboring coefficients in this polynomial is one, then a single root is contained in the domain $[t_0, t_1]$. No sign changes implies no root. If there are more than one sign changes, then the interval is subdivided and the process is repeated, first on the left interval, then on the right.

In [van Wijk, 1984], a method for ray tracing generalized cylinders, specifically surfaces constructed by sweepiing a sphere, was developed by specifying the swept surface implicitly by a polynomial function. This function was then solved by Sturm's method.

Several other methods exist for finding all roots of a polynomial, such as the methods of: Laguerre (reported in [Wyvill & Trotman, 1990]); and Madsen-Reid, Rolle, Budan (all mentioned in [Mitchell, 1990]). These methods can be found in most numerical analysis textbooks (for example, [Press *et al.*, 1988]).

Polynomial methods that return all roots (positive, negative, complex conjugate pairs) of the ray equation are usually the least efficient. Only the positive real roots correspond to useable ray intersections, and often only the smallest of these is desired.

4 Interval Analysis

Interval analysis originated in [Moore, 1966] as a method for solving general numerical analysis problems, including tracking numerical error. In [Mudur & Koparkar, 1984; Snyder, 1992] it was used for general geometry processing. In [Toth, 1985] it was used to find the intersection of a ray with a parametric surface. And finally, in [Mitchell, 1990], it was used to find the intersection of a ray with an implicit surface.

An interval $[a, b]$ is an ordered pair $a \leq b$ representing the range of numbers $\{x : a \leq x \leq b\}$. Arithmetic operations on intervals are defined to return an interval guaranteed to contain all possible solutions over the given domain. Hence, the standard operations on intervals are defined

$$\begin{aligned}
[a,b]+[c,d] &= [a+b, c+d], & (25)\\
[a,b]-[c,d] &= [a-d, b-c], & (26)\\
[a,b]\times[c,d] &= [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)], & (27)\\
[a,b]/[c,d] &= [a,b]\times[1/d,1/c] \;\text{ if } 0 \notin [c,d]. & (28)
\end{aligned}$$

Squaring is represented by self-multiplication, and exponentiation is defined

$$e^{[a,b]} = [e^a, e^b]. \tag{29}$$

Using these rules, an interval form of H can be constructed.

Interval analyis isolates the roots of $H(t)$ by the algorithm in Figure 9, and is shown in action in Figure 10. Once a root is isolated within a monotonic segment of the function, then a combination of Newton's method and regula falsi can quickly refine the root to machine precision. If the function ever becomes tangent to the x-axis, then the algorithm will subdivide around this intersection up to machine precision.

5 Lipschitz Methods

Let f be a function defining an implicit surface. Then f has the Lipschitz property if and only if there exists some positive constant λ such that

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq \lambda ||\mathbf{x} - \mathbf{y}||. \tag{30}$$

The smallest λ that satisfies (30) is called the *Lipschitz constant* of f, denoted Lip$f$.

If f is continuous, then the Lipschitz constant is the maximum slope of the function. This maximum occurs at the global minimum of f', at a zero of f''. Hence, using Lipschitz constants for finding the root of f can involve finding the roots of f''.

1. Let $[s_0, s_1] = H[t_0, t_1]$.
2. If $0 \notin [s_0, s_1]$ then no root in $[t_0, t_1]$.
3. Otherwise, let $[r_0, r_1] = H'[t_0, t_1]$.
4. If $0 \notin [r_0, r_1]$ then H is monotonic over $[t_0, t_1]$.
5. Otherwise, divide and conquer, repeating the process first on $[t_0, (t_0 + t_1)/2]$ then on $[(t_0 + t_1)/2, t_1]$.

Figure 9: The interval analysis method for finding ray intersections.

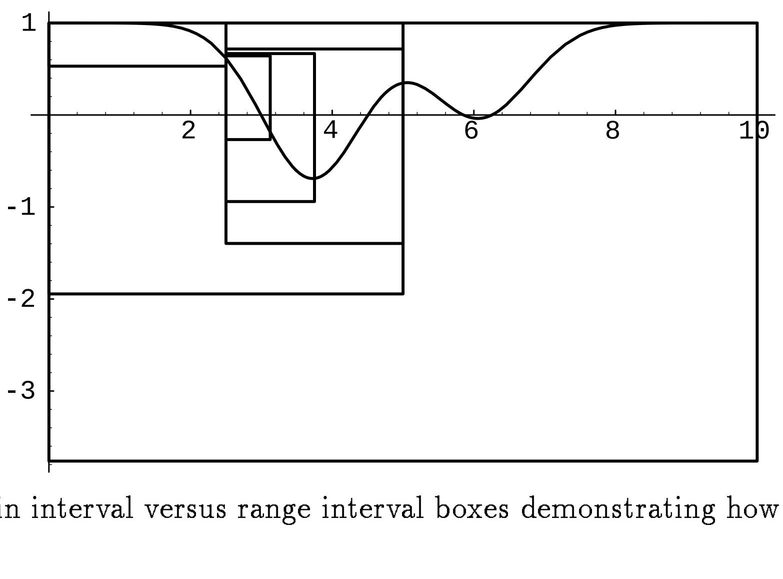

Figure 10: Domain interval versus range interval boxes demonstrating how interval analysis isolates roots.

1. Let $t_m = (t_1 + t_0)/2$ and $r = (t_1 - t_0)/2$.
2. Let $G = \mathrm{Lip} H'(t)$ (restricted to the domain $[t_0, t_1]$.
3. If $|H'(t_m)| > Gr$ then H is monotonic over $[t_0, t_1]$.
   (a) If $H[t_1]H[t_2] < 0$ then a single root is isolated.
   (b) Otherwise, there is no root in $[t_0, t_1]$.
4. Otherwise, divide and conquer, repeating first on $[t_0, t_m]$, then on $[t_m, t_1]$.

Figure 11: The LG-method for determining ray intersection

Lipschitz methods are most efficient when the Lipschitz constant is as small as possible. Restricting the domain of a function localizes the Lipschitz constant, eliminating the influence of steeper but irrelevant sections of the function.

5.1 LG-Surfaces

In [Kalra & Barr, 1989], Lipschitz constants of the implicit function and its directional derivative were used for root isolation. The constants

$$L = \mathrm{Lip} h(\mathbf{x}) \tag{31}$$
$$G = \mathrm{Lip} H'(t) \tag{32}$$

represented localized Lipschitz constants of the functions limited to a particular domain. The constant L bounds the rate h can change across space whereas G bounds the rate H' can change along the ray. The LG-surface ray intersection method is given in Figure 11.

5.2 Sphere Tracing

Sphere tracing was first developed in [Hart *et al.*, 1989] as a method for ray tracing deterministic fractals given an estimate of distance. It converges on the first intersection of a ray with an implicit surface by marching along the ray in positive steps guaranteed small enough not to penetrate the surface. When we come within machine precision of the surface, we have the first ray intersection. The general sphere tracing algorithm, from [Hart, 1993] is given in Figure 12.

```
1. Define $D(t) = H(t)/\mathrm{Lip}H(t)$.
2. For $t = t_0$ to $t_1$ step $D(t)$
    (a) If $H(t) < \epsilon$ then return $t$.
3. return no intersection.
```

Figure 12: The sphere tracing algorithm.

6 Spatial Partitioning

Root isolation methods almost always benefit from spatial partitioning preprocessing steps. This step culls sections of space known not to contain the implicit surface, and provides tighter initial ray intervals for root isolation. For Lipschitz methods, tighter initial ray intervals mean smaller domains, and usually smaller Lipschitz constants.

6.1 Bounding Volumes

The polynomial approximations to Gaussian distributions fall to zero within a finite radius of their peak. Hence, their contribution outside a sphere of this radius is zero, and need not be considered during root isolation [Wyvill & Trotman, 1990]. Such bounding spheres were also used for the Gaussian distributions under the assumption that small contributions were insignificant [Blinn, 1982].

6.2 Surface Tracking

In [Bloomenthal, 1988], a tracking method was developed where an implicit surface was approximated by boxes using a 3-D seed fill algorithm. See [Norton, 1982] for an early example of this method for very unwieldy implicit surfaces.

6.3 Octrees

Also in [Bloomenthal, 1988], an octree method was developed. This method subdivides octree boxes where one vertex is inside, and one vertex is outside, the implicit surface. The main flaw in this method is that it can miss fine details on the implicit surface.

The first half of [Kalra & Barr, 1989] overcomes this problem by subdividing octree boxes based on the Lipschitz condition. This partitioning algorithm is shown here in Figure 13.

The main point of this algorithm is that the value of $f(\mathbf{x})$ cannot differ from $f(\mathbf{x}_D)$ by any greater than Lr from $f(\mathbf{x})$ for all $\mathbf{x} \in D$.

1. Let domain D be an octree element (a box).
2. If one corner of D is inside, and another corner is outside, then accept D.
3. Otherwise, let $L = \mathrm{Lip} f$ over domain D.
4. Let $\mathbf{x}_D$ be the point at the center of domain D and let r be half the diameter of D (r is the distance from $\mathbf{x}_D$ to the farthest point from it in D).
5. If $|f(\mathbf{x_D})| > Lr$ then D does not intersect the implicit surface.
6. Otherwise, divide and conquer, repeating the process on each of the eight sub-domains of D.

Figure 13: Creating an octree using the Lipschitz condition.

7 CSG and Continuity

Constructive solid geometry operations are typically represented in ray-traced models through the use of a Roth diagram [Roth, 1982]. Roth diagrams require finding all intersections of a ray with the component objects, and resolve CSG operations at the rendering level, by sorting and merging these intersections

An alternative representation of CSG for implicit surfaces uses minimum and maximum operations [Ricci, 1974]. These operations resolve CSG operations at the modeling level, by producing a function defining the implicit surface resulting from the CSG operations.

The use of minimum and maximum functions produce discontinuities in the first derivative. Both interval analysis and the LG-surface methods require continuous derivatives. Hence, they must find all ray intersections and build Roth diagrams for CSG model rendering. Sphere tracing does not require continuous derivatives, allowing the use of the minimum and maximum functions for CSG.

8 Surface Normals

Ideally, the surface normal at point $\mathbf{x}$ of an implicit surface defined by function f is represented by the gradient of f,

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x}(\mathbf{x}), \frac{\partial f}{\partial y}(\mathbf{x}), \frac{\partial f}{\partial z}(\mathbf{x})\right]. \tag{33}$$

If f is not differentable, or if finding the partial derivatives is too hard, the gradient of f is easily and sufficiently approximated by

$$\nabla f(\mathbf{x}) \approx (f(x+\epsilon, y, z) - f(\mathbf{x}), f(x, y+\epsilon, z) - f(\mathbf{x}), f(x, y, z+\epsilon) - f(\mathbf{x})) \tag{34}$$

for small $\epsilon > 0$.

9 Deformations

A deformation $\mathbf{D} : \mathbf{R}^3 \rightarrow \mathbf{R}^3$ changes the shape of an object. The standard linear deformations include rotations, shears, and both uniform and non-uniform scales. In [Barr, 1984], non-linear deformations — tapers, twists and bends — are described.

Applying the deformation $\mathbf{D}$ to the implicit surface defined by a function f produces the implicit surface defined by $f \circ D^{-1}$. In other words, the space is inversely-deformed, and the implicit function classifies points in this inversely-deformed space.

The ray-intersection equation now becomes

$$F(t) = f \circ D^{-1} \circ \mathbf{r}(t). \tag{35}$$

If D is affine (a linear deformation plus a translation, commonly specified by a homogeneous 4×4 transformation matrix), then (35) can be computed by altering the ray coefficients. Let $\mathbf{x}_o = [x_o, y_o, z_o, 1]^T$ be a homogeneous column vector specifying the ray's anchor, and let $\mathbf{x}_d = [x_d, y_d, z_d, 0]^T$ specify the ray's direction. Let M be a homogeneous 4×4 matrix representing the affine deformation D. Then the deformed ray equation becomes

$$D^{-1}(\mathbf{r}(t)) = M^{-1}\mathbf{x}_o + tM^{-1}\mathbf{x}_d. \tag{36}$$

If D is non-affine, then it is best to treat $f \circ D^{-1}$ as the function of the implicit surface.

Hence, ray intersection requires storage of D^{-1}, not D. This is also convenient since D^{-1} is needed to properly deform the surface normal at the intersection [Barr, 1984].

10 Acknowledgements

I'd like to thank Jules Bloomenthal and Brian Wyvill for the opportunity to present the topic of ray tracing implicit surfaces. Conversations with Don Mitchell were invaluable in the preparation of this material. The illustrations and examples were made possible through the facilities of the Imaging Research Laboratory, WSU.

References

[Barr, 1984] Barr, A. H. Global and local deformations of solid primitives. *Computer Graphics*, 18(3):21–30, July 1984.

[Blinn, 1982] Blinn, J. F. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[Bloomenthal, 1988] Bloomenthal, J. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, Nov. 1988.

[Hanrahan, 1983] Hanrahan, P. Ray tracing algebraic surfaces. *Computer Graphics*, 17(3):83–90, 1983.

[Hart *et al.*, 1989] Hart, J. C., Sandin, D. J., and Kauffman, L. H. Ray tracing deterministic 3-D fractals. *Computer Graphics*, 23(3):289–296, 1989.

[Hart, 1993] Hart, J. C. Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces. Manuscript, Jan. 1993.

[Hoffman & Hopcroft, 1985] Hoffman, C. and Hopcroft, J. Automatic surface generation in computer aided design. *Visual Computer*, 1:92–100, 1985.

[Kalra & Barr, 1989] Kalra, D. and Barr, A. H. Guaranteed ray intersections with implicit surfaces. *Computer Graphics*, 23(3):297–306, July 1989.

[Mitchell, 1990] Mitchell, D. P. Robust ray intersection with interval arithmetic. In *Proceedings of* Graphics Interface '90, pages 68–74. Morgan Kauffman, 1990.

[Moore, 1966] Moore, R. E. *Interval Analysis.* Prentice Hall, 1966.

[Mudur & Koparkar, 1984] Mudur, S. P. and Koparkar, P. A. Interval methods for processing geomteric objects. *IEEE Computer Graphics and Applications*, 4(2):7–17, Feb. 1984.

[Nishimura *et al.*, 1985] Nishimura, H., Hirai, M., Kawai, T., Kawata, T., Shirakawa, I., and Omura, K. Object modeling by distribution function and a method of image generation. In *Proceedings of* Electronics Communication Conference '85, pages 718–725, 1985. (Japanese).

[Norton, 1982] Norton, A. Generation and rendering of geometric fractals in 3-D. *Computer Graphics*, 16(3):61–67, 1982.

[Peitgen *et al.*, 1984] Peitgen, H.-O., Saupe, D., and v. Haeseler, F. Cayley's problem and Julia sets. *Mathematical Intelligencer*, 6(2):11–20, 1984.

[Press *et al.*, 1988] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. *Numerical Recipes in C.* Cambridge University Press, 1988.

[Ricci, 1974] Ricci, A. A constructive geometry for computer graphics. *Computer Journal*, 16(2):157–160, May 1974.

[Roth, 1982] Roth, S. D. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18(2):109–144, February 1982.

[Snyder, 1992] Snyder, J. M. Interval analysis for computer graphics. *Computer Graphics*, 26(2):121–130, July 1992.

[Toth, 1985] Toth, D. L. On ray tracing parametric surfaces. *Computer Graphics*, 19(3):171–179, July 1985.

[van Wijk, 1984] van Wijk, J. Ray tracing objects defined by sweeping a sphere. In *Proceedings of* Eurographics '84, pages 73–82. Elsevier, 1984.

[Woodwark, 1987] Woodwark, J. R. Blends in geometric modelling. In Martin, R. R., editor, *The Mathematics of Surfaces II*, pages 255–297. Clarendon Press, 1987.

[Wyvill & Trotman, 1990] Wyvill, G. and Trotman, A. Ray tracing soft objects. In *Proceedings of* Computer Graphics International '90. Springer Verlag, 1990.

[Wyvill *et al.*, 1986] Wyvill, G., McPheeters, C., and Wyvill, B. Data structure for soft objects. *Visual Computer*, 2(4):227–234, 1986.