Illumination from Curved Reflectors

Don Mitchell †
Pat Hanrahan ‡

† AT&T Bell Laboratories
‡ † Princeton University

Abstract

A technique is presented to compute the reflected illumination from curved mirror surfaces onto other surfaces. In accordance with Fermat's principle, this is equivalent to finding extremal paths from the light source to the visible surface via the mirrors. Once pathways of illumination are found, irradiance is computed from the Gaussian curvature of the geometrical wavefront. Techniques from optics, differential geometry and interval analysis are applied to solve these problems.

CR Categories and Subject Descriptions: I.3.3 [**Computer Graphics**]: Picture/Image Generation; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Algorithms

Additional Keywords and Phrases: Caustics, Differential Geometry, Geometrical Optics, Global Illumination, Interval Arithmetic, Ray Tracing, Wavefronts

1. Introduction

Ray tracing provides a straightforward means for synthesizing realistic images on the computer. A scene is first modeled, usually by a collection of implicit or parametric surfaces. For each point in an image, a visual ray is traced from the eye into the scene. The visible (i.e., closest) surface intersection is found by geometrical and numerical methods, and the radiance (watts meter^{-2} steradian^{-1}) at that visible point is calculated according some shading model.

Whitted's shading model extends the notion of visibility by simulating reflecting and refracting surfaces [Whitted80]. A visual ray that encounters a reflective surface is bounced off the surface and continues in the direction of reflection (see Figure 1). Eventually, it will encounter a non-reflecting surface, and the shading calculation is performed.

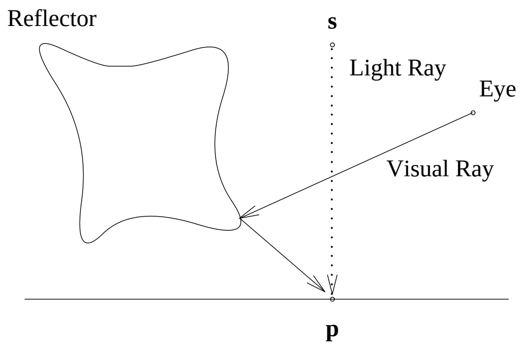

Figure 1. Reflected Visual Rays

This model simulates the effect of seeing the scene reflected in a mirror or refracted through glass, but it does not extend the notion of illumination by the light. Once a visual ray arrives at a point **p**, the shading calculation is limited to the direct component of illumination—just the effect of light traveling directly from the light at **s**. Shading occurs in two steps, first the irradiance (watts meter^{-2}) at **p** is computed, and next the reflected radiance in the direction of the visual ray is determined, based on the reflectance of the surface.

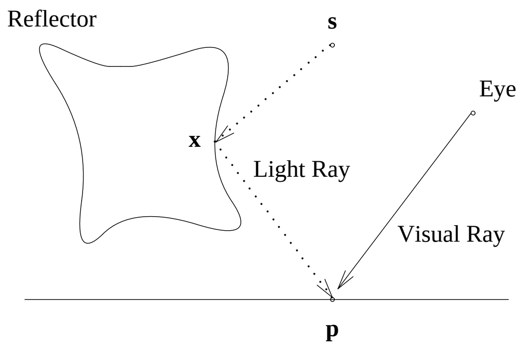

Figure 2. Reflected Illumination

Figure 2 illustrates a much more difficult task; to illuminate a visible point with reflected light. There are two problems involved. The first is finding the proper direction (or directions) in which to cast light rays, so that they arrive at the visible point **p**. The second problem is to compute the proper irradiance, given that the light may reflect off a curved surface and converge or diverge.

Two approaches have been used previously to simulate specularly reflected and refracted illumination. If the reflectors and refractors are polyhedral, illumination can be computed by *backward beam tracing* [Shinya87, Shinya89, Watt90]. For each face of a reflector visible from the light source, the subtended solid angle is traced outward until it reaches a non-specular surface, where radiant power is deposited.

For curved specular surfaces, *backward ray tracing* (also called light-ray or illumination-ray tracing) provides an approximate solution [Arvo86, Heckbert90, Shirley90]. In this method, rays are cast stochastically from the light source, like individual photons, reflecting through the scene until striking a non-specular surface. Shirley has produced some of the most impressive photorealistic images to date with this method (e.g., Figure 7 in [Shirley90]).

The primary difficulties with backward ray tracing involve sampling and estimating the distribution of radiant power (irradiance). Illumination rays arrive at their final destinations in a nonuniform pattern; and if they gave us samples of irradiance, we would have some difficulty with nonuniform interpolation. However, the problem is worse, because illumination rays just give samples of radiant power. We must estimate the *distribution* of radiant power from photon locations and density. This problem has been discussed by Heckbert [Heckbert90]. This is a troublesome problem which has not been completely solved. The method we propose will compute irradiance values at locations of our choosing, and thus avoids both of these problems.

2. Geometrical Optics

The shading models of image synthesis are based on the principles of *geometrical optics*, where wave-like behavior of light is assumed to occur only on an invisibly small scale. In this scheme, light emitted from some point in some direction travels along a curve or "ray" $C(t)$ through space. We define the *optical path length* from one point on a ray to another as the geometric path length weighted by the refractive index of the media:

$$S(t) = \int_C \eta \, dt$$

A surface of constant S (relative to some source point) is called a geometrical *wavefront*, which is always perpendicular to the light rays passing through it [Born80].

In homogeneous media, light rays travel in straight lines; and if there are regions of constant refractive index separated by smooth boundaries, then rays will travel along piecewise straight paths—reflecting or refracting at boundaries. In a medium with smoothly varying refractive index, rays would be curved. In all of these settings, *Fermat's Principle* is obeyed, which stipulates that light travels along paths of stationary optical length. That is, the optical path length is a local maximum or local minimum with respect to any small variation in the path.

Given a stationary path, the important question is how much light propagates along it. In optics, *intensity, I,* is defined to be the radiant power per wavefront area at any point along a ray. The irradiance of a surface is then $E = I \cos\theta$, where θ is the angle between the ray and the surface normal. In the case of direct illumination from a point source, rays emanate radially, and the wavefronts are spherical. At a distance d, the wavefront intensity is simply $I = P/(4\pi d^2)$, where P is the total power of the source. However, once the ray has reflected off a curved surface, the shape of the wavefront is no longer as simple.

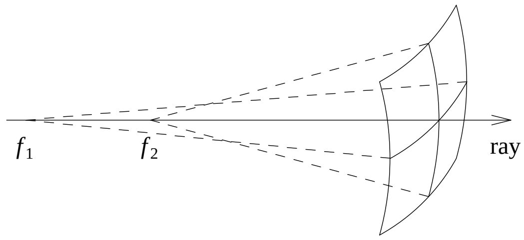

Figure 3. Wavefront and Principle Radii

Figure 3 illustrates the general situation in the neighborhood of a point on a rectilinear ray. A plane containing the ray will intersect the wavefront on a curve with some radius of curvature. From elementary differential geometry, we know there will be some orientation of the plane giving a curve with maximum radius of curvature r_1, and another orientation will give a minimum radius r_2. Furthermore, the planes associated with these *principle radii of curvature* will always be orthogonal. The circles of curvature are centered at points f_1 and f_2, on the ray. Let dA be an element of area on this wavefront. All the rays passing through dA will intersect some subsequent wavefront in an area dA'. Let $d\theta_1$ and $d\theta_2$ be the elements of angle subtended at the centers of curvature by these areas. By conservation of energy, we know the following *intensity law* must hold true:

$$\frac{I'}{I} = \frac{dA}{dA'} = \frac{r_1 r_2 \, d\theta_1 \, d\theta_2}{r_1' r_2' \, d\theta_1 \, d\theta_2} = \frac{r_1 r_2}{r_1' r_2'}$$

This shows the important fact that the intensity along the ray is proportional to the Gaussian curvature of the wavefront $1/(r_1 r_2)$.

3. Fermat Paths

A first step in computing the reflected illumination, as pictured in Figure 2, would be to find the stationary paths from the point **s** to **p**. In general, this would be a difficult problem of variational calculus; but in the case we are considering, it reduces to a simple optimization problem.

Figure 2 pictures a "one-bounce" path from the light **s** to **p** via a reflection at **x**. The total optical path length is a simple function of **x**.

$$d(\mathbf{x}) = \sqrt{(\mathbf{s} - \mathbf{x})^2} + \sqrt{(\mathbf{p} - \mathbf{x})^2}$$

If the mirror surface is defined implicitly by $g(\mathbf{x}) = 0$, then the constrained optimization of $d(\mathbf{x})$ subject to the contraint that all points lie on g can be accomplished by the method of *Lagrange multipliers.* This will give a system of four equations in four variables.

$$\nabla d(\mathbf{x}) + \lambda \nabla g(\mathbf{x})) = 0$$
$$g(\mathbf{x}) = 0$$

The solutions of these non-linear equations will yield paths of locally extremal length.

Recall from analytic geometry that the loci of points whose sum of distances to two points is constant is an ellipse. Varying the total distance yields a a family of confocal ellipsoids whose foci are **s** and **p**. Recall also that the gradient of an implicit function evaluated on a surface is in the direction of the normal to the surface. Thus, the system of equations produced by the method of Lagrange multipliers have the simple geometric interpretation that the extremal points must not only lie on the implicit surface g, but also that the ellipse and the surface must be tangent at those points (see Figure 4). Therefore, we seek those ellipses in the family of confocal ellipses that are tangent to the surface.

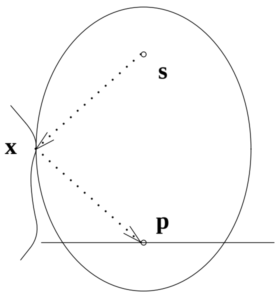

Figure 4. Osculating Ellipsoid

If the light is at infinity, we can use a simpler formula for optical path length.

$$d(\mathbf{x}) = \sqrt{(\mathbf{s} - \mathbf{x})^2} + (\mathbf{s} \cdot \mathbf{x})$$

where in this case **s** is a unit vector in the direction of the light source. The surfaces of varying d define a family of confocal paraboloids about the visible point **p**. The extrema paths are where these parabolods are tangent to the surface.

The case of one-bounce refraction is similar to reflection. If the light and the visible point are on different sides of a smooth boundary between two transparent media, the optical path length becomes:

$$d(\mathbf{x}) = \eta_1 \sqrt{(\mathbf{s} - \mathbf{x})^2} + \eta_2 \sqrt{(\mathbf{p} - \mathbf{x})^2}$$

where η_1 and η_2 are the respective refractive indices. Surfaces of constant d are confocal *Cartesian ovals.* A Cartesian oval has the appearance of an egg-shaped ellipse and is a quartic surface. The mathematics of these surfaces are discussed in [Stravroudis72].

If the curved mirror is defined by a parametric surface $\mathbf{x}(u,v)$, then the tangency conditions are equivalent to stating that the parametric derivatives of the surface are perpendicular to the normal of the ellipse.

$$\mathbf{x}_u(u,v) \cdot \nabla d(u,v) = 0$$
$$\mathbf{x}_v(u,v) \cdot \nabla d(u,v) = 0$$

where $\mathbf{x}_u$ and $\mathbf{x}_v$ are the derivatives in the u and v directions, respectively, and $d(u,v)$ is the distance function evaluated at points on the surface. This leads to a system of two equations in two unknowns. In this paper, we will focus on the problem of the implicitly defined reflecting surface. Such surfaces are are harder because they require a higher dimensional system of equations. However, they are typically easier to ray trace.

The more general case of paths with N bounces can be formulated as a system of $4N$ equations. $\mathbf{x}_0 = \mathbf{s}$ and $\mathbf{x}_{N+1} = \mathbf{p}$; however, only $\mathbf{x}_1$ through $\mathbf{x}_N$ are variables in the optimization problem. The total optical path length for multiple reflections is then:

$$d(\mathbf{x}_0, \cdots \mathbf{x}_{N+1}) = \sum_{i=0}^{N} \eta_i \sqrt{(\mathbf{x}_i - \mathbf{x}_{i+1})^2}$$

The implicit mirror surfaces are denoted by $g_i(\mathbf{x}_i) = 0$, and the so the system of equations to be solved are (for i from 1 to N):

$$\nabla_i d(\mathbf{x}_0, \cdots \mathbf{x}_{N+1}) + \lambda_i \nabla_i g_i(\mathbf{x}_i) = 0$$
$$g_i(\mathbf{x}_i) = 0$$

The gradient equations represent the condition that an ellipsoid with foci at $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$ must kiss the mirror surface $g_i(\mathbf{x}_i) = 0$. Notice that the $\nabla_i d$ contains only three terms, and hence the system of $4N$ equations is sparse.

There exist many special purpose methods for solving the above systems of equations for particular classes of surfaces. For example, finding the ellipse tangent to a plane or to a sphere is very easy. In Section 5, however, we present numerical methods based on interval techniques that work for general implicit surfaces.

Once a path is determined, it is still necessary to check that no intervening surface blocks light traveling along it. This is easily done with the ray tracer shadow test.

4. Wavefront Tracing

Once a path from the light source to the receiver has been determined, the intensity of the incoming ray must be computed. As was discussed in Section 2, the intensity at a point along a ray path is proportional to the Gaussian curvature of the wavefront associated with that point. This intensity law suggests an algorithm where we keep track of the Gaussian curvature as the ray interacts with surfaces on its way to the receiver. The implementation requires (i) tools to analyze the differential geometry of wavefronts and surfaces, and (ii) functions that transform the wavefront as it is transferred through homogeneous media and as it reflects and refracts from a surface. Methods for performing (i) are derived from classical differential geometry (For example [Struik61]); methods for performing (ii) are derived from classical geometrical optics (For example [Stavroudis72]). More details (particularly the derivations) of the results used this section are available in the quoted references.

The local properties of a surface can be derived by computing its derivatives in different directions. A unit tangent in a direction $d\mathbf{x}$ is equal to

$$\mathbf{t} = \frac{d\mathbf{x}}{ds}$$

where $ds^2 = d\mathbf{x} \cdot d\mathbf{x}$ is the differential arc length. Differentiating **t** again yields

$$\frac{d\mathbf{t}}{ds} = \kappa_n \mathbf{n} + \kappa_g (\mathbf{n} \times \mathbf{t})$$

which simply states that $d\mathbf{t}/ds$ is perpendicular to $\mathbf{t}$. The normal curvature κ_n is the component of curvature in the direction $\mathbf{n}$.

$$\kappa = \mathbf{n}\cdot\frac{d\mathbf{t}}{ds} = -\frac{d\mathbf{x}}{ds}\cdot\frac{d\mathbf{n}}{ds} = -\frac{d\mathbf{x}\cdot d\mathbf{n}}{d\mathbf{x}\cdot d\mathbf{x}}$$

where we use the fact that $\mathbf{t}\cdot\mathbf{n} = 0$ and hence $\mathbf{t}'\cdot\mathbf{n} = -\mathbf{t}\cdot\mathbf{n}'$.

The curvature is defined for all directions tangent to the surface. The radius of curvature, which is equal to the reciprocal of curvature, is the radius of the osculating circle attached to a curve created by cutting the surface with a normal plane, that is, a plane containing $\mathbf{n}$ and $\mathbf{t}$. Note that if a plane cuts the surface in a convex curve, the curvature will be *negative,* whereas if the curve is concave, the curvature will be *positive.* A classic result in differential geometry is that the curvature attains a minimum and a maximum value along two perpendicular directions called the lines of curvature. These extrema, or principal curvatures, are denoted by κ_1 and κ_2. The Gaussian curvature K equals $\kappa_1\kappa_2$. If the Gaussian curvature is positive, then κ_1 and κ_2 have the same sign, and hence the surface is locally strictly convex or concave. However, if the Gaussian curvature is negative, then locally the surface is saddle-shaped, being convex along one line of curvature and concave along another.

Given two principal directions $\mathbf{u}_1$ and $\mathbf{u}_2$, the curvature in two other perpendicular directions

$$\begin{aligned}\mathbf{u} &= \cos\theta\,\mathbf{u}_1 - \sin\theta\,\mathbf{u}_2\\ \mathbf{v} &= \sin\theta\,\mathbf{u}_1 + \cos\theta\,\mathbf{u}_2\end{aligned}$$

is given by Euler's Formula (which is easily derived from the definition of curvature).

$$\begin{aligned}\kappa_u &= \kappa_1\cos^2\theta + \kappa_2\sin^2\theta\\ \kappa_v &= \kappa_1\sin^2\theta + \kappa_2\cos^2\theta\\ \kappa_{uv} &= (\kappa_1 - \kappa_2)\cos\theta\sin\theta\end{aligned}$$

where κ_u and κ_v are the curvatures in the directions $\mathbf{u}$ and $\mathbf{v}$, respectively. These equations can be inverted to find the principal curvatures, given the curvatures in any two other orthogonal directions.

$$\begin{aligned}\tan 2\theta &= 2\kappa_{uv}/(\kappa_v - \kappa_u)\\ \kappa_1 &= \kappa_u\cos^2\theta + 2\kappa_{uv}\cos\theta\sin\theta + \kappa_v\sin^2\theta\\ \kappa_2 &= \kappa_u\sin^2\theta - 2\kappa_{uv}\cos\theta\sin\theta + \kappa_u\cos^2\theta\end{aligned}$$

This last formula will be used to derive the principal curvatures after a wavefront interacts with a surface.

Formulas for curvature are usually stated in terms of derivatives of parametric surfaces (see e.g. [Struik61]). In our implementation we wish to compute the curvature of an implicit surface, so we derive the curvature formula in this case. (Curiously, in our scan of the literature, we were not able to find a reference to these formulae.) Recall the formula for curvature,

$$\kappa = -\frac{d\mathbf{x}\cdot d\mathbf{n}}{d\mathbf{x}\cdot d\mathbf{x}}$$

This requires the evaluation of $d\mathbf{n}$, the derivative of the unit normal vector in the direction $d\mathbf{x}$. For an implicit surface $f(x,y,z)$,

$$\mathbf{n} = \frac{\mathbf{g}}{(\mathbf{g}\cdot\mathbf{g})^{1/2}}$$

where

$$\mathbf{g} = \nabla f$$

and

$$\mathbf{g}\cdot\mathbf{g} = (f_x^2 + f_y^2 + f_z^2)$$

From this we can calculate

$$d\mathbf{n} = \frac{d\mathbf{g}}{(\mathbf{g}\cdot\mathbf{g})^{1/2}} - \frac{\mathbf{g}(\mathbf{g}\cdot d\mathbf{g})}{(\mathbf{g}\cdot\mathbf{g})^{3/2}} = \frac{((\mathbf{g}\cdot\mathbf{g}) - (\mathbf{g}\mathbf{g}^T))\,d\mathbf{g}}{(\mathbf{g}\cdot\mathbf{g})^{3/2}}$$

where

$$\mathbf{g}\mathbf{g}^T = \begin{bmatrix} f_x f_x & f_x f_y & f_x f_z\\ f_y f_x & f_y f_y & f_y f_z\\ f_z f_x & f_z f_y & f_z f_z\end{bmatrix}$$

and the derivative of $\mathbf{g}$ is

$$d\mathbf{g} = H d\mathbf{x}$$

where H is the Hessian and equals

$$H = \begin{bmatrix} f_{xx} & f_{xy} & f_{xz}\\ f_{yx} & f_{yy} & f_{yz}\\ f_{zx} & f_{zy} & f_{zz}\end{bmatrix}$$

Thus, we arrive at a formula for the curvature that involves the 1st and 2nd partial derivatives of the surface. In Section 5.1 we discuss a method for efficiently computing these derivatives at a point using automatic differentiation. In Plate 1, colors code the Gaussian curvature of a quartic surface computed using these techniques. Similar illustrations have been produced by [Forrest79] and [Dill81] for parametric surfaces.

Plate 1. Gaussian Curvature

We now return to the task of tracing a wavefront along a ray path.

The above characterization of the local surface geometry in terms of principal curvature leads to a nice representation of the wavefront at a point:

typedef struct {
Point3 **u**, **v**, **n**;
float κ_u, κ_v;
} Wavefront;

The initial wavefront for a point light source is spherical; this implies that both radii of curvature are equal to the radius of the sphere and that there are no distinguished principal directions (that is, any directions may be used). The initial wavefront for a distant light source is a plane; this implies that both radii of curvature are 0, and once again there are no unique principal directions.

The wavefront is now traced through the system. This involves three operations:

- Transfer
- Reflection
- Refraction

The equations describing the evolution of the wavefront for each of these situations were originally derived in 1906 by Gullstrand [Gullstrand06], and more recently using modern notation by Kneisly [Kneisly64] and Stavroudis [Stavroudis72].

The equations of transfer are simplest:

$$\kappa_u{}' = \frac{\kappa_u}{1 - d\kappa_u}$$
$$\kappa_v{}' = \frac{\kappa_v}{1 - d\kappa_v}$$

where d is the distance the wavefront moves. Noting that $\kappa = 1/r$, where r is the radius of curvature, these equations also can be written in the form:

$$\frac{1}{r_u{}'} = \frac{1}{r_u - d}$$
$$\frac{1}{r_v{}'} = \frac{1}{r_v - d}$$

which simply states that the radius of curvature of the wavefront changes by an amount equal to the distance travelled. Remember that for a converging wavefront the radius of curvature is positive. Inspecting the above equations, we note that when a converging wavefront moves a distance equal to original radius of curvature, the denominator goes to 0, and hence the intensity goes to infinity. These positions of extremely high intensity are the caustics of the wavefront.

The equations for refraction and reflection are more complicated and involve the following three steps:

1) Recall the equations giving the directions of the reflected and refracted ray:

$$\mathbf{n}^{(r)} = \mathbf{n}^{(i)} + 2\cos i \ \mathbf{n}^{(s)}$$
$$\mathbf{n}^{(t)} = \eta \ \mathbf{n}^{(i)} + \gamma \ \mathbf{n}^{(s)}$$

In these equations $\mathbf{n}^{(i)}$ and $\mathbf{n}^{(s)}$ are the normals to the incident wavefront and surface, respectively; $\mathbf{n}^{(r)}$ and $\mathbf{n}^{(t)}$ are the normals to the reflected and transmitted (refracted) wavefronts. η is the ratio of the indices of refraction in the two media, η_1/η_2, and $\gamma = \eta \cos i + \cos t$ (i is the angle of incidence and t is the angle of refraction) [Stavroudis72].

2) The direction $\mathbf{u} = \mathbf{n}^{(i)} \times \mathbf{n}^{(s)}$ is tangent both to the incident wavefront and to the surface (since it is perpendicular to both normals). The curvatures of the incident wavefront in the direction **u** can be computed by rotating the principal curvatures using the angle between **u** and the line of curvatures and Euler's Formula. The curvatures of the surface in this direction can be computed using the curvature tensor of the surface.

3) The curvatures of the new wavefronts can be computed by taking the directional derivatives of $\mathbf{n}^{(r)}$ and $\mathbf{n}^{(t)}$ in the direction **u**. These derivatives can be computed directly from the formulae for the reflected and refracted vectors and the directional derivatives of the normals on the incident wavefront and the surface. This calculation can be found on pp. 149-157 of [Stavroudis72]; only the results are stated here.

For reflection:

$$\begin{aligned}
\kappa_u^{(r)} &= \kappa_u^{(i)} + 2\cos i \, \kappa_u^{(s)} \\
\kappa_{uv}^{(r)} &= -\kappa_{uv}^{(i)} - 2\kappa_{uv}^{(s)} \\
\kappa_v^{(r)} &= \kappa_v^{(i)} + (2/\cos i)\, \kappa_v^{(s)}
\end{aligned}$$

For refraction:

$$\begin{aligned}
\kappa_u^{(t)} &= \eta\, \kappa_u^{(i)} + \gamma \kappa_u^{(s)} \\
\kappa_{uv}^{(t)} &= \eta\, \kappa_{uv}^{(i)} + \gamma(\cos i/\cos t)\, \kappa_{uv}^{(s)} \\
\kappa_v^{(t)} &= \eta\, \kappa_v^{(i)} + \gamma(\cos i/\cos t)^2\, \kappa_v^{(s)}
\end{aligned}$$

Remember that the curvature of a plane is 0. Therefore, the curvatures of an outgoing wavefront reflected from a planar surface will be the same as the incoming wavefront (the fact the κ_{uv} switches sign is a result of the change in orientation of the coordinate system due to the reflection). This is as expected, since a perfectly reflected wave does not change its shape. Note also that a planar wavefront incident onto a reflecting surface essentially inherits the curvature of the surface. Thus if the surface is convex, the reflected wavefront will be diverging; whereas if the surface is concave, the wavefront will be converging, eventually forming a caustic.

Once we know the curvatures of the outgoing wavefront in the **u** direction, we can compute the directions of principal curvatures using Euler's Formula, converting to our canonical wavefront representation. This process is then repeated for the next surface that the wavefront is incident upon.

The outgoing intensity of reflected and refracted light should also be multiplied by the corresponding Fresnel coefficients to account for the changes in the magnitudes of reflected vs. refracted light as a function of the angle of incidence.

5. Numerical Techniques

Solving systems of nonlinear equations like the ones in section 3 can be a difficult numerical problem; and in graphics, there is a desire to include a diverse selection of implicit surfaces $g(\mathbf{x}) = 0$. The combination of two interesting techniques make this practical: *automatic differentiation* and *interval arithmetic.*

5.1. Automatic Differentiation

Imagine that we want to evaluate a function $f(x_0)$ at some point and also the derivative $f_x(x_0)$. We could symbolically differentiate *f*, but this can result in a large expression that is computationally expensive to evaluate. We could compute a finite-difference approximation to the derivative, but this yields poor numerical accuracy.

An alternative is to evaluate the pair of numbers $f(x_0)$ and $f_x(x_0)$ concurrently [Rall81]. For constant $f = c$, this pair will be $(c, 0)$, and for $f = x$, the value/derivative pair is simply $(x_0, 1)$. Starting from there, a formula for *f* can be evaluated by performing operations on these pairs of numbers. The pairs are combined according to familiar rules of differentiation, such as the following rules for multiplication and square root:

$$(g(x_0), g_x(x_0)) * (f(x_0), f_x(x_0))$$
$$\rightarrow (g(x_0) f(x_0), g(x_0) f_x(x_0) + f(x_0) g_x(x_0))$$

$$\sqrt{(f(x_0), f_x(x_0))}$$
$$\rightarrow (\sqrt{f(x_0)}, \frac{(f_x)(x_0)}{2\sqrt{f(x_0)}})$$

This method can be easily extended to evaluate partial derivatives with respect to different variables. It also can be extended to simulataneously compute higher order derivatives.

5.2. Interval Arithmetic

An alternative number system can be defined, based on *interval numbers.* A number of powerful numerical techniques are based on interval arithmetic [Moore79], and some of these techniques have been applied to problems in computer graphics [Toth85,Mitchell90]. An interval number corresponds to a range of real values and can be represented by a pair of numbers $[a, b]$, the lower and upper bounds of the interval. Given interval numbers *X* and *Y*, we would like to compute interval values of expressions such as $X + Y$ or function values $f(X)$. Ideally, these values should be exact bounds, the range of lowest to highest values of $f(x)$ for all $x \in X$. Often it is not possible to find the exact bound, but it may still be very useful to know an interval value guaranteed to contain the exact bound.

It is straightforward to define interval extensions of basic arithmetic operations:

$$[a,b] + [c,d] = [a + c, b + d] \tag{1a}$$

$$[a,b] - [c,d] = [a - d, b - c] \tag{1b}$$

$$[a,b] * [c,d] = [\min(ac,ad,bc,bd), \max(ac,ad,bc,bd)] \tag{1c}$$

and if $0 \notin [c,d]$

$$[a,b] / [c,d] = [a,b] * [1/d, 1/c] \tag{1d}$$

Given a rational function $r(x)$, the *natural interval extension* can be defined by evaluating the expression $r(X)$ for an interval argument $X = [x_0, x_1]$ and using the interval-arithmetic operations in (1). The resulting interval value of $r(X)$ is guaranteed to contain the exact bound of the real-valued $r(x)$ over the interval *X*. The tightness of the bound may depend on how $r(X)$ is expressed. For example, interval arithmetic is *subdistributive:* $X(Y + Z) \subseteq XY + XZ$. One of the most important properties of the natural interval extension is *inclusion monotonicity:*

$$X' \subset X, \quad \text{implies} \quad r(X') \subseteq r(X)$$

This means that as the interval *X* becomes more narrow, the interval $r(X)$ will converge to its real restriction.

5.3. Solving Nonlinear Systems

A robust and general method of solving nonlinear systems combines automatic differentiation with interval arithmetic. Suppose we are attempting to solve a system of *N* equations in *N* unknowns, $\mathbf{f}(\mathbf{x}) = 0$. Ordinary Newton's method does not provide a reliable way to locate every solution within a given region, but interval extensions have been developed which do [Moore79]. The most straightforward interval Newton's method is the sequence beginning with some interval vector $\mathbf{X}_0$:

$$\mathbf{X}_{k+1} = N(\mathbf{X}_k) \cap \mathbf{X}_k$$

where

$$N(\mathbf{X}) = m(\mathbf{X}) - [F'(\mathbf{X})]^{-1} f(m(\mathbf{X}))$$

Here, $m(\mathbf{X})$ is an ordinary vector made up of midpoint values of the $\mathbf{X}$ components, and $F'(\mathbf{X})^{-1}$ is the interval Jacobian of the system of equations at $\mathbf{X}$. The intersection $N(\mathbf{X}_k) \cap \mathbf{X}_k$ constrains the new interval to be within the original and is a common precaution against effects of finite-precision machine arithmetic.

A problem with Newton-step operator $N(\mathbf{X})$ is that it involves the inversion of an interval matrix. Krawczyk developed an alternative Newton-like method using an operator which is cheaper to compute. This operator only requires inversion of an ordinary (non-interval) matrix:

$$K(\mathbf{X}) = m(\mathbf{X}) - [m(F'(\mathbf{X}))]^{-1} f(m(\mathbf{X})) + \{I - [m(F'(\mathbf{X}))]^{-1} F'(\mathbf{X})\}(\mathbf{X} - m(\mathbf{X}))$$

Moore and others have proven a number of theorems about the convergence of these Newton-like methods which are used to construct a sound method for isolating and refining solution estimates [Moore79,Rall81].

Theorem 1 (NonExistance I) *If* $0 \notin \mathbf{F}(\mathbf{X})$, *there is no solution of* $\mathbf{f}(\mathbf{x}) = 0$ *in X.*

Theorem 2 (NonExistance II) *If* $K(\mathbf{X}) \cap \mathbf{X} = \emptyset$, *there are no roots in* $\mathbf{X}$.

Theorem 3 (Convergence). *If* $K(\mathbf{X}) \subseteq \mathbf{X}$ *and* $\|I - [m(F'(\mathbf{X}))]^{-1} F'(\mathbf{X})\| < 1$, *there is a unique root in* $\mathbf{X}$ *and a number of iterative methods will converge to it.*

The algorithm for finding all solutions is then quite simple. Given an interval X, these three theorems are automatically tested. The interval is discarded if it is known to have no solutions within it. Some Newton-like iterative method is applied if Theorem 3 guarantees convergence. Otherwise, the interval is subdivided and the procedure applied recursively. Subdivision must terminate when the limits of machine-arithmetic are reached (i.e., when the width of X is less than some ε_x).

In our problem of reflected illumination, the systems given in Section 3 must be solved. Since the equations involve ∇g and ∇d, the interval Newton method will require values of the second derivatives of g and d. The process of finding solutions must begin with an initial interval value for the variables x, y, z, λ. The first three dimensions are initialized to a box tightly enclosing the implicit surface $g(\mathbf{x}) = 0$.

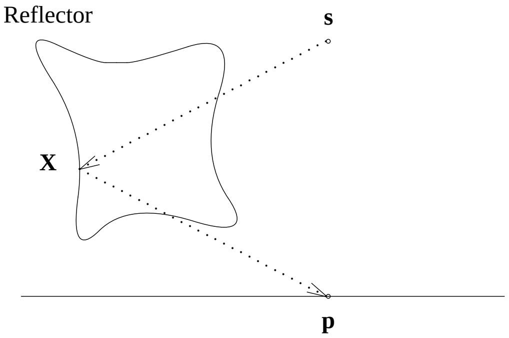

Figure 5. Back-Facing Reflection Points

As Figure 5 shows, we wish to exclude points on the mirror where the osculating ellipsoid touches a back-facing surface, since the body of the mirror itself would block the light's paths. These points are eliminated automatically by choosing the initial λ interval to be $[0, \lambda_{max}]$. This will only allow points where the mirror's normal points in the opposite direction of the normal on the osculating ellipsoid. A nice property of the interval root-finding methods is that they do not expend resources searching outside the initial interval, so no time is wasted on back-facing reflection points.

Its important to note that more than one stationary path may exist from a given **p** to the light source. Figure 6 shows the paths at a series of points in a two-dimensional scene. Rays from the light (in the upper right) to the surface are omitted to avoid cluttering the image. Regions containing one path and a region containing three paths are separated by a caustic surface. In the regions containing three stationary paths, notice that the middle pathway is a local maximum. A common misconception is that Fermat's principle requires paths to be of minimum length.

6. Implementing Multiple Extensions of a Function

Our ray tracer can accept an expression for an implicit surface at run time, and then construct and solve the system of equations described above. To accomplish this, expressions $g(\mathbf{x})$ are represented by sequences of codes for a push-down stack machine. For example, the function for a unit sphere,

$$g(\mathbf{x}) = x^2 + y^2 + z^2 - 1 = 0,$$

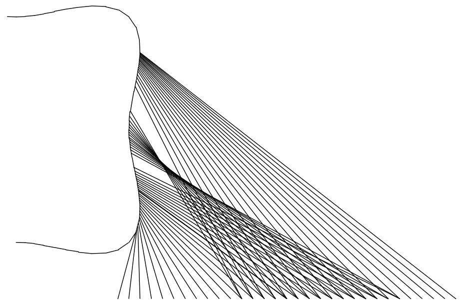

Figure 6. Examples of Stationary Paths of Illumination

is defined as:

```
Bytecode sphere[] = {
        LDX, SQR, LDY, SQR, LDZ, SQR,
        ADD, ADD,
        NUM, 1.0, SUB,
        RET,
};
```

The `LDX` instruction pushes the value of x onto the stack. `ADD` pops two values and pushes the sum, etc. A slightly more interesting surface, shaped like a rounded cube with the faces dented inward, is given by this quartic formula,

$$g(\mathbf{x}) = x^4 + y^4 + z^4 - x^2 - y^2 - z^2 = 0,$$

is defined as:

```
Bytecode cuboid[] = {
        LDX,SQR,SQR,
        LDY,SQR,SQR,
        LDZ,SQR,SQR,
        ADD,ADD,
        LDX,SQR,LDY,SQR,LDZ,SQR,ADD,ADD,
        SUB,
        RET,
}
```

These stack-machine codes are evaluated by a collection of six stack-machine interpreters in the ray tracer. Each of these interpreters takes a code sequence and argument **x**.

The first group of interpreters operates on real numbers. The simplest interpreter evaluates the code with ordinary floating-point arithmetic and returns the scalar value $g(\mathbf{x})$. This stack machine is used primarily during root refinement by ordinary Newton's method (when the interval methods have proven convergence within an interval). A second stack machine used automatic differentiation to compute $g(\mathbf{x})$ and a corresponding value of ∇g. This is used primarily to find surface normal vectors for the shading calculation. A third machine computes the Hessian, and is used in wavefront-curvature calculations.

The second group of interpreters operates on intervals. A first machine in this group computes an interval extension of g and an interval extension of the directional derivative $\partial g / \partial t$ along a ray. This stack machine is used by the ray/surface intersection routines as described in [Mitchell90]. A second machine computes interval extensions of g and its gradient ∇g, and a third machine computes these as well as an interval extension of the Hessian matrix for *g*. These two machines are used to compute the Krawczyk operator (8) during the root isolation phase.

7. Results

A basic ray tracer augmented with the reflection-illumination techniques described above was implemented in 3400 lines of C code. Two 512×512 images were rendered of a reflective copper-colored cuboid defined by:

$$g(\mathbf{x}) = x^4 + y^4 + z^4 - x^2 - y^2 - z^2 = 0$$

This surface is a good test of our method, because it has regions of concave, convex, and saddle-shaped curvature. In Plate 2, the wavefront intensity is not used, so the brightness of the reflection is simply a function of how many virtual light sources illuminate each visible point. Plate 3 is the same image, using the intensity laws developed in Section 4. Note that some reflections seen in Plate 2 have vanished because they make in insignificant contribution to the intensity. Interesting structure in the reflection includes the bright focal point below the concave side and the bright edges of the caustic structure—a singularity in intensity that occurs at the boundaries between regions having different numbers of virtual lights. Plates 4a though 4c (as well as Figure 6) further demonstrate that the table top is divided into distinct regions according to how many virtual lights (i.e, stationary paths to the light source) are present.

Plate 2. Illumination from Virtual Lights

Plate 3. Illumination with Intensity Law

Figure 7 shows a *spot diagram* of this system. This is the pattern of dots formed by casting light rays toward the reflector and recording where they hit the table. In this case, the light was not at infinity, so the shape of the caustic is slightly different, but the topology is essentially the same. Not only does this help verify the correctness of the geometry and intensity calculation, but it is an excellent demonstration of why backward ray tracing has difficulty deriving a smooth and correct irradiance value from a nonuniform pattern of radiant-power samples.

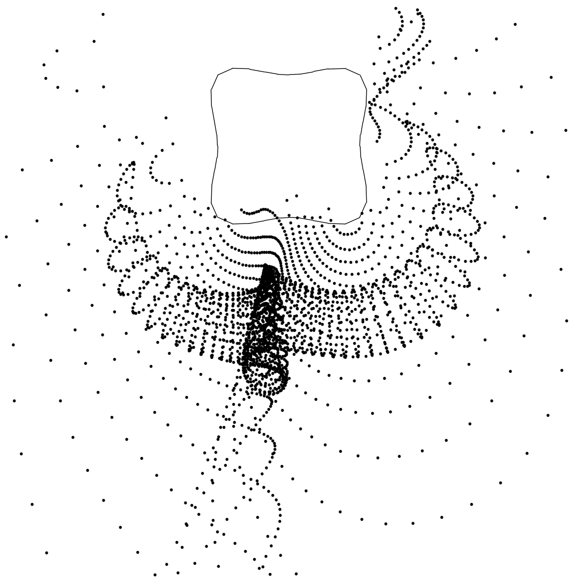

Figure 7. Spot Diagram of Rays Reflected off Cuboid

Plate 3 was computed on an 8-processor Silicon Graphics IRIS (model 480) in about 8 hours.

8. Discussion

Our algorithm is based on two key ideas. First, Fermat's Principle—that light travels along extremal paths—allows us to formulate the path calculation as a multidimensional non-linear optimization problem, that we can solve robustly using interval techniques. Second, the intensity law states that the incident irradiance at a point is proportional to the Gaussian curvature of the wavefront at that point. Gaussian curvature is computing by keeping track of the local geometry of the wavefront as it propogates along a Fermat path. The combination of the two techniques allows us to calculate directly the illumination via virtual lights at any point in the scene.

The intensity law has been used in the two-pass rendering algorithms developed by Shinya and Takahashi [Shinya87,Shinya89] and by Watt [Watt90]. What is novel about our approach is that we avoid the second pass, and hence the problem of resampling the light rays by the eye rays. We also formulate the intensity calculation in a way suitable for ray tracing programs. Shinya and Takahashi's formulation requires the notion of paraxial rays, and Watt's formulation only works for polygonal beams. We also discuss how to compute curvatures for general implicit functions.

Profiling our program, we find that the majority of the time is spent solving for path extrema. The time involved is significant, but we take solace in a observation by Turner Whitted that his first ray tracer spent the majority of its time in the line-surface intersection calculation. We believe that cost of solving for ellipse-surface tangency has roughly the same complexity as line-surface intersection, and, hence, can be significantly sped up.

An important application of this technique that we have yet to explore is the calculation of extended form factors [Sparrow62]. A form factor between two surface elements is defined to be the percentage of light leaving one element that impinges directly on the other. An extended form factor includes light traveling via intermediate specular surfaces. Fermat path tracing finds all the paths between the two surface elements explicitly. Previous techniques compute extended form factors by distributing rays over the hemisphere above a surface [Wallace87,Sillion89]. Since this involves sampling the hemisphere, it may miss important modes of light transport. Also, the wavefront intensity correction must be performed if the path involves interactions with curved surfaces. To the authors' knowledge, this has not been done in existing systems.

Finally, we have discussed the local geometry of wavefronts, but have ignored their global geometry. This distinction roughly corresponds to that of sampling visibility at a point vs. over an area. The key feature of wavefronts are their singularities, or caustics, where the intensity goes to infinity and bright burning spots are formed. Although caustics at first seem to be a problem (since they cause a potentially nasty divide by zero), they can also be used to advantage. Caustics separate ''ray-space'' into topologically uniform regions---for example, where there are one vs. three virtual light sources. Within topologically uniform regions we can compute paths incrementally using Newton's method, since there is no need to go through a root isolation phase, since we aready know a neighboring path. This observation has the potential for speeding up the program enormously. Also, the shapes of the caustics themselves are very interesting. There is a one-to-one correspondence between the types of caustics and the types of singularities, or catastrophes, produced by a function and its gradient map [Berry80]. Producing mathematical visualizations of such functions would be extremely interesting to mathematical physicists.

9. References

[Arvo86] Arvo, J., Backward ray tracing. *Developments in Ray Tracing* SIGGRAPH Course Notes, 12 (1986).

[Berry80] Berry, M. V., Upstill, C., Catastrophe Optics: Morphologies of Caustics and their Diffraction Patterns, *Progress in Optics* XVIII, 259-346, 1980.

[Born80] Born, M. and Wolf, E., *Principles of Optics.* Pergamon, 1980.

[Dill81] Dill, John C., An Application of Color Graphics to the Display of Surface Curvature *Computer Graphics* 15(3), 153-161, 1981.

[Forrest79] Forrest, A. R., On the Rendering of Surfaces. *Computer Graphics* 13(2), 253-259, 1979.

[Gullstrand06] Gullstrand, A. Die reelle optische Abbildung. *Sv. Vetensk. Handl.* 41, 1-119, 1906.

[Heckbert90] Heckbert, P. Adaptive Radiosity Textures for Bidrectional Ray Tracing. *Computer Graphics,* 24, 4 (August 1990), 145-154.

[Kneisly64] Kneisly, J. A. III, Local curvature of wavefronts in an optical system. *J. Opt. Soc. Amer.* 54, 229-235, 1964.

[Mitchell90] Mitchell, D. P. Robust ray intersection with interval arithmetic. *Graphics Interface 90,* (May 1990) 68-74.

[Moore79] Moore, R. E., *Methods and applications of interval analysis.* SIAM, 1979.

[Rall81] Rall, Louis B., *Automatic Differentiation: Techniques and Applications.* Springer-Verlag, 1981.

[Shinya87] Shinya, M., Takahashi, T, Naito, S. Principles and applicaions of pencil tracing. *Computer Graphics,* 21, 4 (July 1987), 45-54.

[Shinya89] Shinya, M., Saito, T., Takahashi, T. Rendering techniques for transparent objects. *Proc. Graphics Interface '89,* (1989) 173-182.

[Shirley90] Shirley, P. A ray tracing method for illumination calculation in diffuse-specular scenes. *Proc. Graphics Interface '90,* (1990) 205-212.

[Sillion89] Sillion, F., Puech, C., A General Two-Pass Method Integrating Specular and Diffuse Reflection. *Computer Graphics* 23(3), 335-344, 1989.

[Sparrow62] Sparrow, E. J., Eckert, E. R. G., Jonsson, V. K., An Enclosure Theory for Radiative Exchange Between Specularly and Diffusely Reflecting Surfaces. *Journal of Heat Transfer* 84C(4), 1962

[Stavroudis72] Stavroudis, O. N. *The Optics of Rays, Wavefronts, and Caustics,* Academic, 1972.

[Struik61] Struik, Dirk J., *Lectures on Classical Differential Geometry* 2nd. Edition, Dover Publications, New York, 1961.

[Toth85] Toth, D. L. On ray tracing parametric surfaces.

Computer Grahpics, 19, 3 (July 1985), 171-179.

[Wallace87] Wallace, J., Cohen, M. F., Greenberg, D. P., A Two-Pass Solution to the Rendering Equation: A Synthesis of Radiosity and Ray Tracing Methods. *Computer Graphics* 21(4), 311-320, 1987.

[Watt90] Watt, M. Light-water interaction using backward beam tracing. *Computer Graphics,* 24, 4 (August 1990) 377-385.

[Whitted80] Whitted, T. An improved illumination model for shaded display. *Comm. ACM,* 23, 6 (June 1980), 343-349.