Theory and Application of Specular Path Perturbation

MIN CHEN and JAMES ARVO
California Institute of Technology

In this paper we apply perturbation methods to the problem of computing specular reflections in curved surfaces. The key idea is to generate families of closely related optical paths by expanding a given path into a high-dimensional Taylor series. Our path perturbation method is based on closed-form expressions for linear and higher-order approximations of ray paths, which are derived using Fermat's Variation Principle and the Implicit Function Theorem (IFT). The perturbation formula presented here holds for general multiple-bounce reflection paths and provides a mathematical foundation for exploiting path coherence in ray tracing acceleration techniques and incremental rendering. To illustrate its use, we describe an algorithm for fast approximation of specular reflections on curved surfaces; the resulting images are highly accurate and nearly indistinguishable from ray traced images.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Display algorithms*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Raytracing*; G.1.1 [**Numerical Analysis**]: Interpolation—*Interpolation formulas*; G.1.2 [**Numerical Analysis**]: Approximation—*Elementary function approximation*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Implicit surfaces, optics, perturbation theory, specular reflection, Taylor series

1. INTRODUCTION

Two fundamental operations are involved in ray tracing: ray casting, which is used to follow optical paths through a simulated environment, and shading, which is applied at the points where rays intersect objects. Of the two operations, generating the optical paths is by far the most expensive, and typically the chief obstacle to interactive ray tracing. Consequently,

This work was supported in part by the US National Science Foundation Career Award (CCR9876332), the Army Research Office Young Investigator Program (DAAH04-96-100077), and the Alfred P. Sloan Foundation.
Authors' address: California Institute of Technology, 1200 E. California Blvd., Pasadena, CA 91125.

finding some means of quickly generating or reusing ray paths becomes a critical challenge for interactive ray tracing. To date, most research has focused on speeding the process of ray casting [Arvo and Kirk 1989]. The rationale for using path reuse instead is that updating a path can be far less expensive than recomputing it. Moreover, this approach leads to new algorithms that exploit coherence both within a single image and among similar images.

Many aspects of path reuse have been investigated for interactive rendering. Typically, a ray tree or equivalent data structure is used to retain all ray object intersections computed during the rendering of each pixel and then used in the generation of a nearby image. Cook [1984] and Séquin and Smyrl [1989] handled material changes in this way by retraversing the retained ray trees with modified parameters. Murakami and Hirota [1990] and Jevans [1992] handled changes in geometry by indexing each ray path with the list of voxels it traverses; any change to the scene affects some subset of the voxels, which in turn determines the potentially affected rays. Briere and Poulin [1996] employed a ray tree and a color tree to preserve the path information and shading expressions separately, and proposed a novel data structure to efficiently detect and recompute the exact portion of the image that has changed after an arbitrary manipulation of a scene. Although these techniques are very effective for geometrical and material changes, they are ineffective for viewpoint motion, where even a small eye movement may change the ray paths associated with most pixels. To compensate for the viewpoint movement, Badt [1988] and Adelson and Hodges [1995] reprojected the old intersections to the new view position by applying 3D warping, which exploits perspective coherence. This 3D warp amounts to an image space reprojection (2D warp) in stereoscopic ray tracing [Adelson and Hodges 1993]. Unfortunately, reprojection only works for the first level rays, and is only applicable to diffuse scenes. Chapman et al. [1990; 1991] computed "continuous intersection" information of rays with the scene through time by using the trajectory of the viewpoint through the scene; but their method is restricted to a polygonal scene model and predefined viewpoint paths.

The present work introduces perturbation theory into image synthesis and presents a new mathematical tool based on the Taylor expansion of a reflection path. The central contributions of the paper are closed-form expressions for the first- and second-order derivatives of the reflection points along a path, which we call *path Jacobians* and *path Hessians*, respectively. The analytical perturbation formula based on path Jacobians and path Hessians can be used to perturb a given path to obtain a family of closely related optical paths, properly accounting for multiple specular reflections. This technique leads to a new method for path reuse which can be applied in a number of contexts, including the computation of caustics and specular reflections [Chen 1999]. Previous work that explored the use of ray path perturbations includes pencil tracing [Shinya et al. 1987] and ray differentials [Igehy 1999], which we discuss further in the next section.

In Section 2 we give a general overview of our path perturbation method from the point of view of perturbation theory and path coherence, and introduce the concepts of path Jacobian and path Hessian. Section 3 reviews some preliminary principles and theorems from optics and calculus. Then, in Section 4, we use these basic tools to derive the path Jacobians for both single-bounce and multiple-bounce reflection paths. Section 5 extends path Jacobians to a higher order and derives the expressions for path Hessians using tensor differentiation. A perturbation formula combining path Jacobians and/or path Hessians is also presented. To illustrate the application of these tools, in Section 6 we describe a new approach to approximating specular reflections in arbitrary curved surfaces. The resulting images, which are nearly indistinguishable from ray traced images, not only verify the correctness of our analytical formula but also show that the path perturbation method leads to faster alternatives than ray tracing for simulating specular reflections. Finally, Section 7 describes other potential applications of the path perturbation formulas and future directions to extend our current work.

2. PATH PERTURBATION METHOD

2.1 Perturbation Theory

Finite difference and finite element methods are extremely popular numerical methods with applications in many branches of science and engineering. Both approaches operate by constructing discrete approximations to the original problem, which are then relatively straightforward to solve. Quite distinct from these methods, however, are *perturbation methods*, which have a very different emphasis. Given the solution to a mathematical problem, such methods ask how the solution is affected when the conditions of the problem are slightly altered. The systematic answer to this question forms the subject of *perturbation theory* [Lin and Segel 1988].

One of the most basic tools of perturbation analysis is the Taylor expansion. To answer the question posed above, we consider a solution to the given problem as a function of a perturbation quantity; the given solution then corresponds to the function value when the perturbation is zero. The slightly altered conditions of the problem can be expressed in terms of small values of the perturbation quantity. The solution under the changed conditions can then be approximated by expanding the parameterized function into a Taylor series. Much of the art of perturbation theory lies in finding the coefficients in the Taylor series, which depends on the application.

Analysis based on perturbation theory is approximate, since it relies on a truncated series expansion. However, the approach plays an important role in yielding solutions for equations that cannot be solved exactly. Perturbation methods have been applied to a wide range of disciplines [Cole 1968], including continuum mechanics [Dyke 1964], and heat transfer [Aziz and Na 1984]. In this paper we expand the list to include image synthesis.

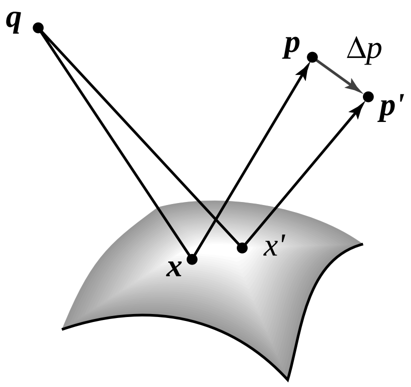

Fig. 1. Reflections tend to have a great degree of coherence; the corresponding reflection points **x** and **x′** will converge as **p** approaches **p′**.

2.2 Path Coherence

Let us consider what happens to a given optical path connecting points **p** and **q**, via a specular reflection in a curved surface when **p** moves slightly. Changes in the path can take one of two forms: the new path may hit an entirely different object, or the same object at a slightly different position. Strong coherence exists in the latter case.

For a scene containing specular objects, any given object point may be visible through a reflection path consisting of a sequence of one or more specular reflections. For smooth reflecting surfaces, it is likely that the reflection points along the path will vary continuously as a function of small changes to the object point or the vantage point, except at an object boundary. That is, reflections tend to have a great degree of coherence; as two objects grow nearer to each other, so will their reflections. As shown in Figure 1, as **p** approaches **p′**, the corresponding reflection points **x** and **x′** will converge. Therefore, the reflection paths corresponding to a neighborhood of the view point or a neighborhood of the object point are very similar. Although exceptions occur near object boundaries, coherence is the rule.

Path coherence arises in a number of guises in image synthesis. In image-based rendering, consider a collection of images of the same static scene with respect to nearly identical view positions. Even though each view position is associated with a different ray path from the same visible scene point, it is likely that nearly all these ray paths hit the same scene objects at slightly different positions. In particular, strong path coherence exists between stereoscopic image pairs. Thus, image warping can be used to transform a reference image to the desired image at the perturbed view point. As another example, consider the reflection of an object in a smooth mirror as seen from a fixed vantage point. As the object moves, the reflection also moves coherently, so the new reflection can be found by perturbing the old paths. In Section 6 we discuss the application of path perturbation in this context.

However, this type of coherence has proven much more difficult to exploit analytically. Although several mathematical formulations have been offered [Adelson and Hodges 1995], none apply to general ray traced paths. In this paper we apply perturbation theory to explore such coherence.

2.3 Path Perturbation

Our path perturbation approach is motivated by the natural connection between path coherence and perturbation theory. The key step is to formulate the problem of updating a reflection path as a perturbation problem. Specifically, the reflection point $\mathbf{x}$ of a given one-bounce reflection path connecting $\mathbf{q}$ and $\mathbf{p}$ can be considered as a function[1] of the two points, that is,

$$\phi : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3,$$

where $\phi(\mathbf{q}, \mathbf{p})$ returns the scene coordinates of the point $\mathbf{x}$ on the reflecting surface. By fixing one endpoint $\mathbf{q}$, the function $\phi(\mathbf{q}, \cdot)$ can be viewed as a mapping from a 3D point to its reflection

$$\Psi : \mathbb{R}^3 \to \mathbb{R}^3, \tag{1}$$

where $\Psi(\mathbf{y}) \equiv \phi(\mathbf{q}, \mathbf{y})$. We call Ψ the *path function* with respect to the point $\mathbf{p}$, since we are ultimately interested in sequences of reflection points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ forming an optical path from $\mathbf{p}$ to $\mathbf{q}$. For a small ϵ, we now consider $\Psi(\mathbf{p} + \Delta\mathbf{p})$ where $\|\Delta\mathbf{p}\| < \epsilon$, and obtain an asymptotic approximation to the new path function by means of a Taylor expansion.

The path function $\Psi : \mathbb{R}^3 \to \mathbb{R}^3$ relating a reflection point $\mathbf{x}$ to the perturbed point $\mathbf{p}$ is a vector-valued function taking a vector argument. The Taylor series of Ψ can be expressed in Cartesian tensor notation as

$$\begin{aligned}\Psi_i(\mathbf{p} + \Delta\mathbf{p}) = \Psi_i(\mathbf{p}) &+ \sum_j \Psi_{i,j}\epsilon_j \\ &+ \frac{1}{2}\sum_{jk} \Psi_{i,jk}\epsilon_j\epsilon_k \cdots \\ &+ \frac{1}{n!}\sum_{jk\ldots r} \Psi_{i,jk\ldots r}\epsilon_j\epsilon_k \ldots \epsilon_r + \cdots\end{aligned}$$

where $\Delta\mathbf{p} = (\epsilon_1, \epsilon_2, \epsilon_3)$ is the perturbation of $\mathbf{p}$, $\Psi = (\Psi_1, \Psi_2, \Psi_3)$, and

$$\Psi_{i,j} = \frac{\partial \Psi_i}{\partial p_j},$$

[1]More precisely, ϕ is a relation, since there may exist several reflection points for two given points, but locally it is a function.

$$\Psi_{i,jk} = \frac{\partial^2 \Psi_i}{\partial p_j \partial p_k},$$

$$\cdots$$

for $i, j, k = 1, 2, 3$. Here, p_j, p_k represent components of the independent variable $\mathbf{p}$, and all partial derivatives are evaluated at the given path through $\mathbf{p}$.

To obtain a second-order approximation of Ψ,we truncate the sequence after the first two dominant terms. Thus,

$$\Psi_i(\mathbf{p} + \Delta\mathbf{p}) = \Psi_i(\mathbf{p}) + \sum_j \Psi_{i,j}\epsilon_j + \frac{1}{2}\sum_{jk} \Psi_{i,jk}\epsilon_j\epsilon_k + O(\|\Delta\mathbf{p}\|^3).$$

Collecting the coefficients appearing in the three expansions of Ψ_1, Ψ_2, and Ψ_3 and putting them into familiar matrix forms, we obtain the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \Psi_{1,1} & \Psi_{1,2} & \Psi_{1,3} \\ \Psi_{2,1} & \Psi_{2,2} & \Psi_{2,3} \\ \Psi_{3,1} & \Psi_{3,2} & \Psi_{3,3} \end{bmatrix}, \tag{2}$$

and three Hessian matrices

$$\mathbf{H}^i = \begin{bmatrix} \Psi_{i,11} & \Psi_{i,12} & \Psi_{i,13} \\ \Psi_{i,21} & \Psi_{i,22} & \Psi_{i,23} \\ \Psi_{i,31} & \Psi_{i,32} & \Psi_{i,33} \end{bmatrix}, \tag{3}$$

for $i = 1, 2, 3,$ which constitutes a third-order tensor $\mathbf{H}$. In terms of $\mathbf{J}$ and $\mathbf{H}$, the second-order Taylor expansion of the path function Ψ about the given path through $\mathbf{p}$ can be expressed as

$$\Psi(\mathbf{p} + \Delta\mathbf{p}) = \Psi(\mathbf{p}) + \mathbf{J}\Delta\mathbf{p} + \frac{1}{2}\begin{bmatrix} \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^1\Delta\mathbf{p} \\ \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^2\Delta\mathbf{p} \\ \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^3\Delta\mathbf{p} \end{bmatrix} + O(\|\Delta\mathbf{p}\|^3), \tag{4}$$

which is the *second-order perturbation formula* for updating a given path through $\mathbf{p}$ to a new path associated with the nearby point $\mathbf{p} + \Delta\mathbf{p}$. Thus, $\mathbf{J}$ and $\mathbf{H}$ are actually the first- and second-order derivatives of the path function Ψ, that is,

$$\mathbf{J} = \frac{\partial \Psi(\mathbf{p})}{\partial \mathbf{p}}$$

and

$$\mathbf{H} = \frac{\partial^2 \Psi(\mathbf{p})}{\partial \mathbf{p}^2}.$$

We shall refer to the first derivative $\mathbf{J}$ as the *path Jacobian* and the second derivative $\mathbf{H}$ as the *path Hessian*. The path Jacobian and path Hessian provide, respectively, first- and second-order approximations to Ψ.

Similar algorithms that linearly approximate a perturbed ray path have been developed for use in ray tracers to calculate a bundle of rays at the cost of tracing a single ray. Pencil tracing [Shinya et al. 1987] employed paraxial ray theory [Born and Wolf 1965] to approximate the propagation of paraxial rays by a matrix, which linearizes each optical event (transfer, reflection, or refraction). The ray differential framework [Igehy 1999] computes a first-order Taylor approximation to a ray parameterized in terms of image space coordinates. By compositing a series of functions while propagating a ray, we can trace the value of its derivative with respect to the image plane using the chain rule. Both approaches essentially compute a Jacobian matrix with respect to view directions. Our most significant point of departure is that we parameterize a path in terms of the position of its end points and compute the path Jacobian as its first-order derivative with respect to the varying end point. Such parameterization and associated linear approximation allow us to freely perturb both ends of a ray traced path, leading to approximate analytical methods applicable in a variety of contexts, such as moving the view point, locating the reflection point of a varying scene point, etc. Furthermore, we also obtain a second-order approximation for a ray by computing path Hessians.

In order to apply the perturbation formula (4) to perturbed paths, we must compute $\mathbf{J}$ and $\mathbf{H}$ for any given path. To do this, we require tools from geometric optics and elementary classical analysis, which is briefly reviewed in the next section.

3. PRELIMINARIES

3.1 Fermat's Principle

In geometrical optics, the propagation of the light obeys *Fermat's principle*, also known as the principle of the *shortest optical path*. This principle asserts that the optical length of an actual light ray between any two points P_1 and P_2 is a local extremum among all paths between these points within a small neighborhood [Born and Wolf 1965, pp.128–129]. Here, the *optical length* from one point on a ray to another is defined as the geometric path length weighted by the refractive index of the medium, η:

$$\int_\gamma \eta ds, \tag{5}$$

where $\gamma(s)$ is the parametric path from P_1 to P_2 and s is arc length. Since light propagates with a velocity $v = c/\eta$ along the ray, we can write Eq. (5) in terms of time:

$$c\int_\gamma dt.$$

Consequently, Fermat's principle is also known as the *principle of least time*.

Fermat's principle is a fundamental one, underlying geometrical optics. It stipulates that light travels along paths of stationary optical length, which implies that the paths followed by photons traveling through any medium should be locally extremal in terms of both distance and time. This property suggests that determining an actual path (or *Fermat path*) between any two 3D points can be reduced to a problem of variational calculus. In a ray tracing setting where regions of constant refractive index are separated by smooth boundaries, rays will travel along piecewise straight paths, reflecting or refracting at boundaries. The optical length function d between two arbitrary points $\mathbf{p}$ and $\mathbf{q}$ is simply the sum of the segment lengths weighted by their corresponding refractive indices. That is,

$$d(\mathbf{x}_0, \ldots, \mathbf{x}_{N+1}) = \sum_{i=0}^{N} \eta_i \|\mathbf{x}_i - \mathbf{x}_{i+1}\|, \tag{6}$$

where $\mathbf{x}_0 = \mathbf{p}$ and $\mathbf{x}_{N+1} = \mathbf{q}$. For a reflection path in a uniform medium, Eq. (6) simplifies further, since the refractive indices are constant, and may be assumed to be one.

3.2 Lagrange Multiplier Theorem

The Fermat path problem represents a class of optimization problems with equality constraints, that is,

$$\text{minimize or maximize } f(\mathbf{x}) \text{ subject to } \mathbf{h}(\mathbf{x}) = \mathbf{0},$$

where $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, and $m \leq n$. Here, f is the *objective function* and $\mathbf{h}$ is the *constraint function*. The *Lagrange Multiplier Theorem* [Apostol 1969, p.315] provides a first-order necessary condition for the local extrema.

THEOREM 1 (LAGRANGE MULTIPLIER THEOREM). *Let $\mathbf{x}^*$ be a local extremal point of $f : \mathbb{R}^n \to \mathbb{R}$, subject to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$, and $\mathbf{x}^*$ is a regular point. Then there exists $\lambda^* \in \mathbb{R}^m$ such that*

$$\nabla f(\mathbf{x}^*) + \lambda^{*\mathbf{T}} D\mathbf{h}(\mathbf{x}^*) = \mathbf{0}^T, \tag{7}$$

where

$$D\mathbf{h}(\mathbf{x}^*) = \begin{bmatrix} \nabla h_1(\mathbf{x}^*) \\ \vdots \\ \nabla h_m(\mathbf{x}^*) \end{bmatrix}$$

is the Jacobian matrix of $\mathbf{h} = [h_1, \ldots, h_m]^T$ *at* $\mathbf{x}^*$.

We refer to the vector λ^* in the above theorem as the *Lagrange multiplier vector*. For a special case of only one constraint, where $n = 3$ and $m = 1$, the Lagrange condition (7) becomes

$$\nabla f(\mathbf{x}^*) + \lambda^* \nabla h(\mathbf{x}^*) = \mathbf{0}. \tag{8}$$

Here the scalar λ^* is referred to as the *Lagrange multiplier*.

3.3 The Implicit Function Theorem

By means of the Lagrange Multiplier Theorem, we can easily specify an implicit equation for the perturbed reflection problem; however, it is generally impossible to extract a closed-form solution from the resulting nonlinear equations. Fortunately, the *Implicit Function Theorem* (IFT) provides a method for explicitly computing the *derivative* of such an implicitly-defined function without finding the function; it is this tool that allows us to derive a closed-form expression for the path Jacobian.

THEOREM 2 (IMPLICIT FUNCTION THEOREM). *Let* $A \subset \mathbb{R}^n \times \mathbb{R}^m$ *be an open set and let* $F : A \to \mathbb{R}^m$ *be a function of class* C^p. *Suppose* $(\mathbf{x}_0, \mathbf{y}_0) \in A$ *such that* $F(\mathbf{x}_0, \mathbf{y}_0) = 0$ *and*

$$\det \begin{bmatrix} \frac{\partial F_1}{\partial y_1} & \cdots & \frac{\partial F_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial y_1} & \cdots & \frac{\partial F_m}{\partial y_m} \end{bmatrix} \neq 0, \tag{9}$$

where $F = (F_1, \ldots, F_m)$, *and the Jacobian matrix is evaluated at the point* $(\mathbf{x}_0, \mathbf{y}_0)$. *Then there exists an open neighborhood* $\mathbf{x}_0 \in U \subset \mathbb{R}^n$, *a neighborhood* $\mathbf{y}_0 \in V \subset \mathbb{R}^m$, *and a unique function* $f : U \to V$, *such that*

$$F(\mathbf{x}, f(\mathbf{x})) = 0 \tag{10}$$

for all $\mathbf{x} \in U$. *Furthermore,* $f \in C^p$.

See, for example, Marsden and Hoffman [1993, pp. 211–213] for a proof of the Implicit Function Theorem. By differentiating both sides of Eq. (10)

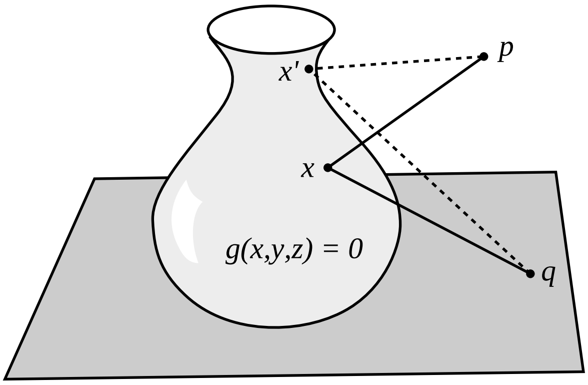

Fig. 2. Reflection paths obey Fermat's variational principle, stating that the length of the optical path connecting **p** and **q** is a local extremum. For any **p** and **q** there may be many such paths.

with respect to the independent variable $\mathbf{x} \in \mathbb{R}^n$, we obtain the following well-known corollary:

Corollary 1. The Jacobian matrix of the implicit function f in Theorem 2 is given by

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = -\begin{bmatrix} \frac{\partial F_1}{\partial y_1} & \cdots & \frac{\partial F_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial y_1} & \cdots & \frac{\partial F_m}{\partial y_m} \end{bmatrix}\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}. \tag{11}$$

Note that the inverse on the right-hand side of Eq. (11) is guaranteed to exist, since the determinant is necessarily nonzero. Thus, the Jacobian matrix of the implicit function f is a linear combination of derivatives of the known function F. This fact allows us to compute the Jacobian matrix without first determining the function f.

4. THE PATH JACOBIAN

In this section we shall first derive the expression for the path Jacobian for a single-bounce reflection path and then extend the result to a multiple-bounce case.

4.1 One-Bounce Path

Given a mirror surface and two points **p** and **q** in space, the problem of finding a point **x** on the mirror where a ray is reflected from **p** to **q** has a long history in optics. For a spherical mirror, this problem is known as *Alhazen's problem* [Neumann 1998]. Even for simple convex shapes, such as a sphere, it is difficult to find **x** analytically. For nonconvex surfaces, the problem is even more difficult, as there may be many such paths connecting two points, as shown in Figure 2. By computing the path Jacobian $\mathbf{J} = \partial x / \partial \mathbf{p}$, we can attack the problem of finding these reflection points in a completely different manner, by approximating the reflection $\mathbf{x}'$ of a point $\mathbf{p}'$ using a known reflection $\mathbf{x}$ of a nearby point $\mathbf{p}$. For example,

$$\mathbf{x}' = \mathbf{x} + \mathbf{J}(\mathbf{p}' - \mathbf{p}) \tag{12}$$

approximates the new reflection $\mathbf{x}'$ to first-order accuracy by perturbing the known reflection $\mathbf{x}$. We now demonstrate how the path Jacobian, which is the first derivative of path function Ψ, can be computed for an implicitly-defined reflecting surface using Fermat's principle and the Implicit Function Theorem.

Suppose $g(\mathbf{x}) = 0$ is the implicit definition of a reflecting surface $\mathbf{G}$ and $\mathbf{x}$ is a reflection point of a ray path connecting $\mathbf{p}$ and $\mathbf{q}$ in a homogeneous medium, as shown in Figure 2. By Fermat's principle, the path length assumes a local extremum. In addition, the reflection point $\mathbf{x}$ is required to lie on the surface $\mathbf{G}$. Therefore, we may recast the problem of computing $\mathbf{x}$ as a *constrained optimization problem*. Applying the method of Lagrange multipliers in the same fashion as demonstrated by Mitchell and Hanrahan [1992], we obtain

$$\begin{aligned} \nabla d(\mathbf{p}, \mathbf{x}, \mathbf{q}) + \lambda \nabla g(\mathbf{x}) &= \mathbf{0} \\ g(\mathbf{x}) &= 0, \end{aligned} \tag{13}$$

where λ is a Lagrange multiplier, and $d(\mathbf{p}, \mathbf{x}, \mathbf{q})$ is the length of the optical path from $\mathbf{p}$ to $\mathbf{q}$ via $\mathbf{x}$ in a homogeneous medium. Thus,

$$d(\mathbf{p}, \mathbf{x}, \mathbf{q}) = \|\mathbf{p} - \mathbf{x}\| + \|\mathbf{q} - \mathbf{x}\|.$$

By fixing one endpoint of the path $\mathbf{q}$, and allowing $\mathbf{p}$ to vary, we may rewrite Eq. (13) as an implicit equation that relates $\mathbf{p}$, $\mathbf{x}$, and λ:

$$F(\mathbf{p}, \mathbf{x}, \lambda) = \mathbf{0}, \tag{14}$$

where $F : \mathbb{R}^3 \times \mathbb{R}^4 \rightarrow \mathbb{R}^4$, and $\mathbf{p}$ and regarded as the independent variable. We refer to Eq. (14) as the *Fermat equation*. The explicit form of the Fermat equation can be derived by expanding the ∇ operator in Eq. (13), yielding

$$\begin{aligned} F_i(\mathbf{p}, \mathbf{x}, \lambda) &= -\frac{(p_i - x_i)}{\|\mathbf{p} - \mathbf{x}\|} - \frac{(q_i - x_i)}{\|\mathbf{q} - \mathbf{x}\|} + \lambda \frac{\partial g(\mathbf{x})}{\partial x_i} \\ F_4(\mathbf{p}, \mathbf{x}, \lambda) &= g(\mathbf{x}), \end{aligned} \tag{15}$$

where $i = 1, 2, 3$.

Unfortunately, it is generally impossible to solve these nonlinear equations for the reflection point $\mathbf{x}$ in a closed form, even for trivial functions $g(\mathbf{x})$ [Neumann 1998]. Mitchell and Hanrahan [1992] dealt with this problem by applying an iterative root-finding technique, known as the interval Newton method, to solve for $\mathbf{x}$. In contrast, the *derivative* of the reflection point can be computed directly, without knowing the explicit functional relationship between $\mathbf{x}$ and the end points $\mathbf{q}$ and $\mathbf{p}$, by means of the Implicit Function Theorem, as discussed in Section 3.3.

To show how the Implicit Function Theorem and its corollary can be applied to the path Jacobian computation, consider a ray reflected from a surface with implicit definition $g(\mathbf{x}) = 0$. Since this reflection path provides a solution $(\tilde{\mathbf{p}}, \tilde{\mathbf{x}}, \tilde{\lambda})$ to the Fermat equation (14), it follows from condition (9) in Theorem 2 that if

$$\det\left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{x}, \lambda)}\right] \neq 0$$

at $(\tilde{\mathbf{p}}, \tilde{\mathbf{x}}, \tilde{\lambda})$, then there exists a function $f : \mathbb{R}^3 \to \mathbb{R}^4$ such that

$$f(\mathbf{p}) = (\mathbf{x}, \lambda) \tag{16}$$

and

$$F(\mathbf{p}, f(\mathbf{p})) = \mathbf{0}$$

for all $\mathbf{p}$ sufficiently close to $\tilde{\mathbf{p}}$. That is, f solves the reflection problem in a neighborhood of the known reflection path. More importantly, from Corollary 1, we can solve for $\partial f / \partial \mathbf{p}$ in this neighborhood using Eq. (11), which yields

$$\left[\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}\right]_{4\times3} = -\left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{x}, \lambda)}\right]^{-1}_{4\times4}\left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{p})}\right]_{4\times3}. \tag{17}$$

For clarity, we shall frequently indicate the matrix dimensions with subscripts, as we have done above. Equation (16) shows that the path function $\Psi : \mathbf{p} \to \mathbf{x}$ is easily obtained from f by discarding its last component λ. By introducing an operator $\mathbf{sub} : \mathrm{Hom}(\mathbb{R}^3, \mathbb{R}^4) \to \mathrm{Hom}(\mathbb{R}^3, \mathbb{R}^3)$, which drops the last row of a 4×3 matrix, the 3×3 path Jacobian $\mathbf{J}$ can be expressed as

$$\boxed{\mathbf{J} = \left[\frac{\partial \Psi(\mathbf{p})}{\partial \mathbf{p}}\right]_{3\times3} = \mathbf{sub}\left(-\left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{x}, \lambda)}\right]^{-1}_{4\times4}\left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}}\right]_{4\times3}\right),} \tag{18}$$

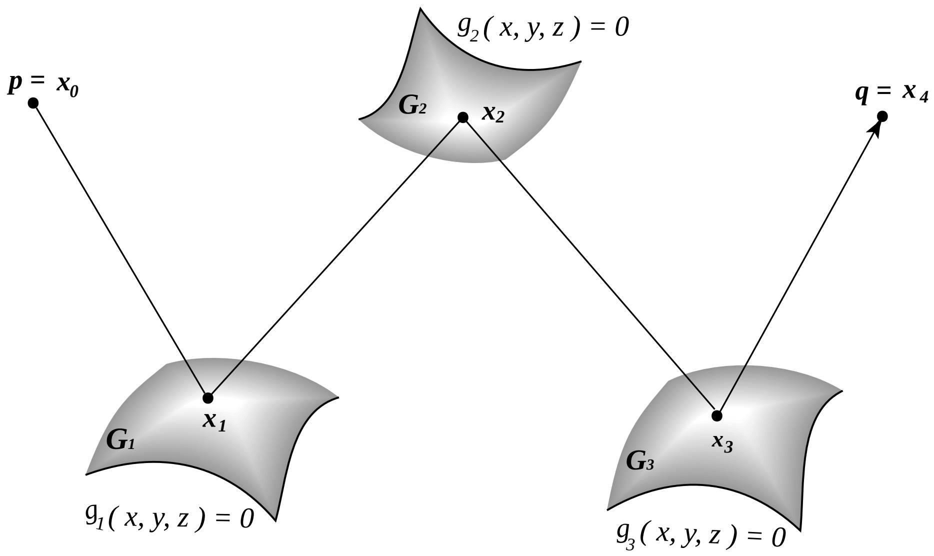

Fig. 3. A reflection path from $\mathbf{p}$ to $\mathbf{q}$ via three reflection points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ on three implicitly-defined surfaces $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$.

which characterizes the variation in $\mathbf{x}$ with respect to $\mathbf{p}$. Alternatively, we may simplify Eq. (18) in such a way that the path Jacobian $\mathbf{J}$ can be computed directly, without computing the inverse of a 4×4 matrix, as shown in Appendix A. Note that all quantities on the right of Eq. (18) can be obtained from the Fermat equation (15) and the implicit function g. We shall refer to the 4×3 matrix on the left of Eq. (17), before dropping the last row, as the *Fermat Jacobian*, and designate it by D.

4.2 N-Bounce Path

In this section we show how to compute path Jacobians for the more general case of N bounces. Given a path from a varying point $\mathbf{p}$ to a fixed point $\mathbf{q}$ via N reflecting surfaces, we order the reflection points from $\mathbf{p}$ to $\mathbf{q}$ as $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with $\mathbf{x}_0 = \mathbf{p}$ and $\mathbf{x}_{N+1} = \mathbf{q}$. The corresponding reflecting surfaces $\mathbf{G}_i$ and their implicit functions g_i are ordered accordingly, as shown in Figure 3, for a three-bounce path. We shall always consider the varying endpoint $\mathbf{p} = \mathbf{x}_0$ as the starting point and the fixed endpoint $\mathbf{q} = \mathbf{x}_{N+1}$ as the ending point of an N-bounce path. By viewing the position of each reflection point $\mathbf{x}_i$ as a function Ψ_i of the endpoint $\mathbf{p}$, we may define the path Jacobian $\mathbf{J}_i$ at $\mathbf{x}_i$ as the derivative of Ψ_i with respect to $\mathbf{p}$; this Jacobian characterizes how $\mathbf{x}_i$ changes with respect to perturbations in $\mathbf{p}$. Accordingly, the N reflection points in the path can be updated to the first order, using

$$\mathbf{x}'_i = \mathbf{x}_i + \mathbf{J}_i \Delta \mathbf{p} \tag{19}$$

for $i = 1, 2, \ldots, N$, where the N path Jacobian $\mathbf{J}_i$s can be computed either directly or recursively, as described in the following sections.

4.2.1 *Direct Computation of* $\mathbf{J}_i$. As a ray path between $\mathbf{p}$ and $\mathbf{q}$, Fermat's principle states that N reflection points are located in such a way that the optical length of the path $\mathbf{x}_0 - \mathbf{x}_1 - \cdots - \mathbf{x}_{N+1}$ is minimized or maximized. By analogy with the one-bounce case, we may apply the method of Lagrange multipliers to the entire path to obtain a Fermat equation satisfied by all the $\mathbf{x}_i$s; the Implicit Function Theorem can then be used to compute the N path Jacobians.

Consider a constraint vector $\mathbf{g} = [g_1, \ldots, g_N]^T$ formed by the N implicit functions. We apply the vector form (7) of the Lagrange condition to obtain a system of $4N$ equations satisfied by the N reflection points [Mitchell and Hanrahan 1992]:

$$\begin{aligned} \nabla_i\, d(\mathbf{x}_0, \ldots, \mathbf{x}_{N+1}) + \lambda_i\, \nabla_i g_i(\mathbf{x}_i) &= 0, \\ g_i(\mathbf{x}_i) &= 0, \end{aligned} \tag{20}$$

where $i = 1, \ldots, N$, and ∇_i is the gradient operator with respect to $\mathbf{x}_i$. Since each $\nabla_i d$ contains three points, Eq. (20) is equivalent to

$$\begin{aligned} Fg_i(\mathbf{p}, \mathbf{x}_1, \mathbf{x}_2, \lambda_1) &= 0, \\ &\vdots \\ Fg_N(\mathbf{x}_{N-1}, \mathbf{x}_N, \lambda_N) &= 0, \end{aligned} \tag{21}$$

where Fg_i is the Fermat equation for each one-bounce path segment $(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$, as in Eq. (15). We now consider Eq. (21) as an implicit function $F : \mathbb{R}^3 \times \mathbb{R}^{4N} \rightarrow \mathbb{R}^{4N}$, with $\mathbf{p}$ as its independent variable:

$$F(\mathbf{p}, \mathbf{x}_1, \lambda_1, \ldots, \mathbf{x}_N, \lambda_N) = 0. \tag{22}$$

The Implicit Function Theorem and its corollary can then be applied to Eq. (22), provided that

$$\det\left(\frac{\partial F(\mathbf{p}, \mathbf{x}_1, \lambda_1, \ldots, \mathbf{x}_N, \lambda_N)}{\partial(\mathbf{x}_1, \lambda_1, \ldots, \mathbf{x}_N, \lambda_N)}\right) \neq 0$$

at the given path. This condition ensures that the reflecting path does not include any points that are exactly on a boundary, and the rays of the path are nowhere tangent to a surface. In this case, there exists a function

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}^4 \times \cdots \times \mathbb{R}^4, \tag{23}$$

that maps $\mathbf{p}$ to N pairs of reflection points and Lagrange multipliers, $(\mathbf{x}_i, \lambda_i)$, which holds for all points $\mathbf{p}'$ within a small neighborhood of $\mathbf{p}$. From Corollary 1, the derivative of this function is given by

$$\left[\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}\right]_{4N\times 3}$$

$$= -\left[\frac{\partial F(\mathbf{p}, \mathbf{x_1}, \lambda_1, \ldots, \mathbf{x_N}, \ldots, \lambda_\mathbf{N})}{\partial(\mathbf{x_1}, \lambda_1, \ldots, \mathbf{x}_N, \lambda_N)}\right]^{-1}_{4N\times 4N}\left[\frac{\partial F(\mathbf{p}, \mathbf{x_1}, \lambda_1, \ldots, \mathbf{x}_N, \lambda_N)}{\partial \mathbf{p}}\right]_{4N\times 3.} \tag{24}$$

According to Eq. (23), the N path Jacobians $\mathbf{J}_1, \mathbf{J}_2, \ldots, \mathbf{J}_N$ can be formed from the $4N \times 3$ matrix on the left of Eq. (24) by dropping the rows relating to the Lagrange multiplier vector.

Unfortunately, to compute path Jacobians using Eq. (24) we must invert a $4N \times 4N$ matrix, which grows quadratically with the number of bounces. This computation makes the direct approach impractical. Alternatively, we may take each bounce separately and approach the problem of computing path Jacobians for an N-bounce path by recursively applying formula (18) for a one-bounce path. In the following section we describe such a recursive procedure, which significantly reduces the computation for long reflecting paths.

4.2.2 *Recursive Computation of* $\mathbf{J}_i$. By applying the chain rule to the definition of $\mathbf{J}_i$, it follows that we may express $\mathbf{J}_i$ as a product of i Jacobian matrices. That is,

$$\mathbf{J}_i = \frac{\partial \mathbf{x}_i}{\partial \mathbf{p}} = \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \cdot \frac{\partial \mathbf{x}_{i-1}}{\partial \mathbf{x}_{i-2}} \cdots \frac{\partial \mathbf{x}_i}{\partial \mathbf{p}} \tag{25}$$

for $i = 1, 2, \ldots, N$. Note that each factor on the right-hand side of Eq. (25) is a 3×3 Jacobian matrix, which denotes the derivative of the position of the ith reflection point with respect to its previous point $\mathbf{x}_{i-1}$. Let us define

$$\mathbf{J}_i^* \equiv \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \tag{26}$$

at each reflection point $\mathbf{x}_i$ ($i = 1, 2, \ldots, N$). Then Eq. (25) can be written as

$$\mathbf{J}_i = \mathbf{J}_i^* \cdot \mathbf{J}_{i-1}^* \cdots \mathbf{J}_1^*,$$

or, equivalently, by the recurrence relation

$$\mathbf{J}_i = \mathbf{J}_i^* \cdot \mathbf{J}_{i-1}, \tag{27}$$

where $\mathbf{J}_1 = \mathbf{J}_1^*$. We refer to the Jacobian matrix defined in Eq. (26) as *reflection Jacobian*. Using Eq. (27), we have transformed the problem of computing N path Jacobians $\mathbf{J}_i$ to an equivalent problem of computing N reflection Jacobians $\mathbf{J}_i^*$. We now describe how the reflection Jacobians can be computed.

Observe that for the last bounce $(\mathbf{x}_{N-1}, \mathbf{x}_N, \mathbf{q})$, where $\mathbf{q}$ is the fixed endpoint, the problem reduces to a simple one-bounce path. As discussed in Section 4.1, we have

$$Fg_N(\mathbf{x}_{N-1}, \mathbf{x}_N, \lambda_N) = \mathbf{0}, \tag{28}$$

where $\mathbf{x}_{N-1}$ is considered as the independent variable. The Implicit Function Theorem implies that if

$$\det\left(\frac{\partial Fg_N(\mathbf{x}_{N-1}, \mathbf{x}_N, \lambda_N)}{\partial(\mathbf{x}_N, \lambda_N)}\right) \neq 0,$$

there exists a function $f_N : \mathbf{x}_{N-1} \to (\mathbf{x}_N, \lambda_N)$. The function f_N can be decomposed into two components: $f_N(\mathbf{x}) = (f_{N1}(\mathbf{x}), f_{N2}(\mathbf{x}))$ where $f_{N1} : \mathbb{R}^3 \to \mathbb{R}^3$ and $f_{N2} : \mathbb{R}^3 \to \mathbb{R}$ map $\mathbf{x}_{N-1}$ to $\mathbf{x}_N$ and λ_N, respectively. With the existence of f_N, the last reflection Jacobian $\mathbf{J}_N^*$ has been derived in Eq. (18). As a segment of an N-bounce path, the ith bounce still observes Fermat's principle. Applying Lagrange multipliers to $(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$, where $i \neq N$, we obtain a similar Fermat equation:

$$Fg_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) = \mathbf{0}, \tag{29}$$

which differs from Eq. (14) by having two varying end points $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i+1}$. By processing the N bounces from the ending point $\mathbf{q}$ to the starting point $\mathbf{p}$, the functions f_is mapping $\mathbf{x}_{i-1}$ to $(\mathbf{x}_i, \lambda_i)$ $(i = N, \ldots, 1)$ can be implicitly determined in sequence from the Implicit Function Theorem. Thus when we reach the ith bounce, the function $f_{i+1} : \mathbb{R}^3 \to \mathbb{R}^4$ mapping $\mathbf{x}_i$ to $(\mathbf{x}_{i+1}, \lambda_{i+1})$ has been constructed implicitly from the $(i + 1)$-th bounce. By decomposing f_{i+1} into $f_{(i+1)1} : \mathbf{x}_i \to \mathbf{x}_{i+1}$ and $f_{(i+1)2} : \mathbf{x}_i \to \lambda_{i+1}$, we can express $\mathbf{x}_{i+1}$ in Eq. (29) in terms of $f_{(i+1)1}$, obtaining

$$Fg_i(\mathbf{x}_{i-1}, \mathbf{x}_i, f_{(i+1)1}(\mathbf{x}_i), \lambda_i) = \mathbf{0}, \tag{30}$$

which can be treated as an implicit equation of three variables $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, and λ_i, with $\mathbf{x}_{i-1}$ as the independent variable. That is,

$$H_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \lambda_i) = \mathbf{0} \tag{31}$$

where

$$H_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \lambda_i) = Fg_i(\mathbf{x}_{i-1}, \mathbf{x}_i, f_{(i+1)1}(\mathbf{x}_i), \lambda_i). \tag{32}$$

Applying the Implicit Function Theorem to Eq. (31), the ith explicit function $f_i : \mathbf{x}_{i-1} \rightarrow (\mathbf{x}_i, \lambda_i)$ exists under the condition that

$$\det\left(\frac{\partial H_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \lambda_i)}{\partial(\mathbf{x}_i, \lambda_i)}\right) = \det\left(\frac{\partial F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, f_{(i+1)1}(\mathbf{x}_i), \lambda_i)}{\partial(\mathbf{x}_i, \lambda_i)}\right) \neq 0. \tag{32}$$

Note that the existence of the function f_i makes the reflection Jacobian $\mathbf{J}_i^*$ in Eq. (26) well-defined. In terms of f_i, Eq. (30) can be reformulated as

$$Fg_i(\mathbf{x}_{i-1}, f_{i1}(\mathbf{x}_{i-1}), f_{(i+1)1}(f_{i1}(\mathbf{x}_{i-1})), f_{i2}(\mathbf{x}_{i-1})) = \mathbf{0}. \tag{33}$$

Differentiating both sides of Eq. (33) with respect to $\mathbf{x}_{i-1}$, we get

$$\frac{\partial F_{gi}}{\partial \mathbf{x}_{i-1}} + \frac{\partial F_{gi}}{\partial \mathbf{x}_i} \cdot \frac{\partial f_{i1}}{\partial \mathbf{x}_{i-1}} + \frac{\partial F_{gi}}{\partial \mathbf{x}_{i+1}} \cdot \frac{\partial f_{(i+1)1}}{\partial \mathbf{x}_i} \cdot \frac{\partial f_{i1}}{\partial \mathbf{x}_{i-1}} + \frac{\partial F_{gi}}{\partial \lambda_i} \cdot \frac{\partial f_{i2}}{\partial \mathbf{x}_{i-1}} = \mathbf{0}. \tag{34}$$

Rearranging the terms in Eq. 34 and substituting $\mathbf{J}_{i+1}^*$ computed previously for $\partial f_{(i+1)1} / \partial \mathbf{x}_i$, we obtain

$$\frac{\partial F_{gi}}{\partial \mathbf{x}_{i-1}} = -\left[\frac{\partial F_{gi}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^* + \frac{\partial F_{gi}}{\partial \mathbf{x}_i} \; \frac{\partial F_{gi}}{\partial \lambda_i}\right] \begin{bmatrix} \dfrac{\partial f_{i1}(\mathbf{x}_{i-1})}{\partial \mathbf{x}_{i-1}} \\ \dfrac{\partial f_{i2}(\mathbf{x}_{i-1})}{\partial \mathbf{x}_{i-1}} \end{bmatrix}.$$

By inverting the matrix, we solve for $\partial f_i / \partial \mathbf{x}_{i-1}$:

$$\left[\frac{\partial f_i}{\partial \mathbf{x}_{i-1}}\right]_{4\times 3} = -\left[\frac{\partial F_{gi}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^* + \frac{\partial F_{gi}}{\partial \mathbf{x}_i} \; \frac{\partial F_{gi}}{\partial \lambda_i}\right]^{-1} \left[\frac{\partial F_{gi}}{\partial \mathbf{x}_{i-1}}\right], \tag{35}$$

where the existence of the inverse is guaranteed by the condition in Eq. (32). Introducing an operator **aug** : Hom$(\mathbb{R}^3, \mathbb{R}^4) \rightarrow$ Hom$(\mathbb{R}^4, \mathbb{R}^4)$, which expands a 4×3 matrix by appending a zero column, we may rewrite Eq. (35) as

$$\left[\frac{\partial f_i}{\partial \mathbf{x}_{i-1}}\right]_{4\times 3} = -\left[\frac{\partial F_{gi}}{\partial(\mathbf{x}_i, \lambda_i),} + \mathbf{aug}\left(\frac{\partial F_{gi}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^*\right)\right]^{-1} \left[\frac{\partial F_{gi}}{\partial \mathbf{x}_{i-1}}\right]. \tag{36}$$

Finally, taking the submatrix of the left-hand side of Eq. 36, we have derived a recurrence relation for the ith reflection Jacobian $\mathbf{J}_i^*$:

$$\boxed{\mathbf{J}_i^* = \mathbf{sub}(-[A_i + \mathbf{aug}(B_i \cdot \mathbf{J}_{i+1}^*)]^{-1} T_i),} \tag{37}$$

where $i = N - 1, \ldots, 1$ and

$$A_i = \left[\frac{\partial F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i)}{\partial(\mathbf{x}_i, \lambda_i)}\right]_{4\times 4}$$

$$B_i = \left[\frac{\partial F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i)}{\partial(\mathbf{x}_{i+1})}\right]_{4\times 3}$$

$$T_i = \left[\frac{\partial F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i)}{\partial(\mathbf{x}_{i-1})}\right]_{4\times 3}. \tag{38}$$

Equation (37) suggests that the reflection Jacobians $\mathbf{J}_i^*$ for an N-bounce path are computed *backward*, from the fixed point $\mathbf{q}$ towards the perturbed point $\mathbf{p}$. The starting Jacobian $\mathbf{J}_N^*$ for the last reflection point $\mathbf{x}_N$, which is calculated first, can be viewed as a special case of this recurrence relation where $B_N = 0$ and $\mathbf{J}_{N+1}^* = \mathbf{0}$.

In order to use these reflection Jacobians $\mathbf{J}_i^*$ to perturb a given N-bounce path, a naive approach is to compute each path Jacobian $\mathbf{J}_i$ ($i = 1, 2, \ldots, N$) from the $\mathbf{J}_i^*$s with Eq. (25), and then update the corresponding reflection point $\mathbf{x}_i$ to first-order accuracy using Eq. (19). However, motivated by the observation that the term ultimately required for linear perturbation in Eq. (19) is $\mathbf{J}_i\Delta\mathbf{p}$, we may optimize the naive algorithm by perturbing the N reflection points incrementally in a prescribed order. Let $\Delta\mathbf{x}_i = \mathbf{J}_i\Delta\mathbf{p}$ ($i = 1, 2, \ldots, N$), which denotes the perturbation of each reflection point in this linear approximation. Then the perturbation formula (19) becomes

$$\mathbf{x}_i' = \mathbf{x}_i + \Delta\mathbf{x}_i. \tag{39}$$

Expressing $\Delta\mathbf{x}_i$ in terms of $\mathbf{J}_i^*$s using Eq. (27), we have

$$\Delta\mathbf{x}_i = \mathbf{J}_i^* \cdot \mathbf{J}_{i-1}\Delta\mathbf{p} = \mathbf{J}_i^*\Delta\mathbf{x}_{i-1}$$

for $i = 1, 2, \ldots, N$. Consequently, Eq. (39) can be expressed as

$$\mathbf{x}_i' = \mathbf{x}_i + \mathbf{J}_i^*\Delta\mathbf{x}_{i-1}, \tag{40}$$

where $i = 1, 2, \ldots, N$, $\Delta\mathbf{x}_i = \mathbf{x}_i' - \mathbf{x}_i$ and $\Delta\mathbf{x}_0 = \Delta\mathbf{p}$. We can interpret Eq. (40) as the first-order Taylor approximation of f_{i1} around its previous point $\mathbf{x}_{i-1}$, which allows us to perturb the reflection points in an N-bounce path incrementally, from the starting point $\mathbf{p}$ to the ending point $\mathbf{q}$.

It can easily be verified that the direct approach and the recursive approach yield the same answer for path Jacobians in a multiple-bounce path [Chen 1999]. However, compared to the direct method, the recursive method is more efficient and easier to implement, as it involves only 4×4 matrices.

5. THE PATH HESSIAN

As a linear approximation of the path function $\Psi : \mathbb{R}^3 \to \mathbb{R}^3$, the path Jacobian matrix $\mathbf{J}$ maps from tangent space to tangent space, which is equivalent to locally fitting the function Ψ with a linear form. To obtain better accuracy, we may compute an additional term in Eq. (4), the path Hessian $\mathbf{H}$, and fit the function Ψ locally with a quadratic form. As a second derivative of the path function, the path Hessian computation involves matrix differentiation and yields a tensor of the third order, which we can represent as a collection of three 3×3 matrices $\mathbf{H^1}$, $\mathbf{H^2}$, and $\mathbf{H^3}$, as shown in Eq. (3).

We first direct our attention to a one-bounce path $\mathbf{p} - \mathbf{x} - \mathbf{q}$. Using both the path Jacobian $\mathbf{J}$ and the path Hessian $\mathbf{H}$, we can perturb the reflection point $\mathbf{x}$ to the second order, using

$$\mathbf{x}' = \mathbf{x} + \mathbf{J}\Delta\mathbf{p} + \frac{1}{2}\begin{bmatrix} \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^1\Delta\mathbf{p} \\ \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^2\Delta\mathbf{p} \\ \Delta\mathbf{p}^{\mathrm{T}}\mathbf{H}^3\Delta\mathbf{p} \end{bmatrix}, \tag{41}$$

when $\mathbf{p}$ is moved to a nearby point $\mathbf{p} + \Delta\mathbf{p}$. Let $f : \mathbf{p} \to (\mathbf{x}, \lambda)$ denote the function implicitly defined by the Fermat equation (14) in the one-bounce case. Then the path function $\Psi = (\Psi_1, \Psi_2, \Psi_3)$ corresponds to the first three components of f, that is,

$$\Psi_i = f_i \tag{42}$$

for $i = 1, 2, 3$. According to Eq. (3), all the second partial derivatives $f_{i,jk}$ $(i, j, k = 1, 2, 3)$ must be evaluated for the path-Hessian $\mathbf{H}$. Observe that the first-order approximation in Eq. (18) yields

$$T(\mathbf{p}, \mathbf{x}, \lambda) = -\Gamma(\mathbf{p}, \mathbf{x}, \lambda)D(\mathbf{p}), \tag{43}$$

where T, Γ, and D are given by

$$T(\mathbf{p}, \mathbf{x}, \lambda) = \left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}}\right]_{4\times3}$$

$$\Gamma(\mathbf{p}, \mathbf{x}, \lambda) = \left[\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)}\right]_{4\times4}$$

$$D(\mathbf{p}) = \left[\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}\right]_{4\times3.}$$

To compute the derivatives of matrices T, Γ, and D in Eq. (43), it is convenient to write them in Cartesian tensor form. Thus, Eq. (43) becomes

$$T_{ij}(\mathbf{p}, \mathbf{x}, \lambda) = -\sum_{k=1}^{4} \Gamma_{ik}(\mathbf{p}, \mathbf{x}, \lambda) D_{kj}(\mathbf{p}),$$

where $i = 1, 2, 3, 4$ and $j = 1, 2, 3$. Replacing $(\mathbf{x}, \lambda)$ with the function f, we obtain

$$T_{ij}(\mathbf{p}, f(\mathbf{p})) = -\sum_{k=1}^{4} \Gamma_{ik}(\mathbf{p}, f(\mathbf{p})) D_{kj}(\mathbf{p}). \tag{44}$$

Differentiating both sides of Eq. (44) with respect to $\mathbf{p}$, we have

$$(\nabla T)_{mij} = -\sum_{k=1}^{4} ((\nabla \Gamma)_{mik} D_{kj}(\mathbf{p}) + \Gamma_{ik} (\nabla D)_{mkj}), \tag{45}$$

where ∇ denotes the gradient operator, which in this context is applied to matrices [Segel and Handelman 1977, p.62]. That is, the gradients of matrices T, Γ, and D are defined by

$$(\nabla T)_{mij} = T_{ij,\,m} = \frac{\partial T_{ij}(\mathbf{p}, f(\mathbf{p}))}{\partial p_m}$$

$$(\nabla \Gamma)_{mik} = \Gamma_{ik,\,m} = \frac{\partial T_{ik}(\mathbf{p}, f(\mathbf{p}))}{\partial p_m}$$

$$(\nabla D)_{mkj} = D_{kj,\,m} = \frac{\partial D_{kj}(\mathbf{p})}{\partial p_m}, \tag{46}$$

for $i, k = 1, 2, 3, 4$ and $j, m = 1, 2, 3$.

As shown in Figure 4, the gradient of a matrix T is a third-order array; the three ordered subscripts can be interpreted as "layer," "row," "column," respectively. The row and column (the second and third indices) correspond to the row and column in the matrix T, while the layer (the first index) corresponds to each coordinate of the differential variables. As a convention, we always use m to index the layer and i, j, k, l to index the row or column. For example, the component $(\nabla\ T)_{mij}$ in Eq. (46) denotes the partial derivative of T_{ij} with respect to p_m (the mth coordinate of $\mathbf{p}$). Each layer of ∇T constitutes a matrix $\{t_{ij}\}$ with $t_{ij} = (\nabla T)_{mij}$ for a fixed m, which we designate by $\nabla_m T$; Nevertheless, ∇T itself can be thought of as a generalized matrix[2] $\{s_{ij}\}$ with $s_{ij} = \nabla(T_{ij})$. That is, each entry is a gradient vector of its corresponding entry in T.

[2]Strictly speaking, it is not a matrix since its elements are gradient vectors rather than scalars.

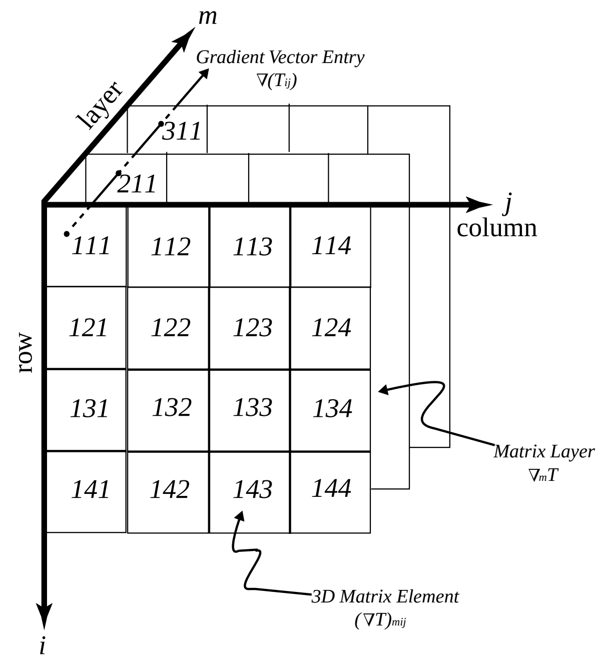

Fig. 4. The gradient of a matrix T, ∇T, is a third-order array with "layer," "row," and "column".

In terms of the mth layers of ∇T, $\nabla \Gamma$, and ∇D, the summation on the right-hand side of Eq. (45) can be interpreted as two matrix products:

$$\nabla_m T = -\nabla_m \Gamma \cdot D - \Gamma \cdot \nabla_m D.$$

Solving for each layer of ∇D, we obtain

$$\boxed{\nabla_m D = -\Gamma^{-1}(\nabla_m T + \nabla_m \Gamma \cdot D),} \tag{47}$$

for $m = 1, 2, 3$.

As the Fermat Jacobian, which we defined in Section 4, D has a matrix form

$$D = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \\ f_{4,1} & f_{4,2} & f_{4,3} \end{bmatrix}.$$

The mth layer of ∇D is obtained by differentiating each component of D with respect to the mth coordinate of $\mathbf{p}$,

$$\nabla_m D = \begin{bmatrix} f_{1,\,1m} & f_{1,\,2m} & f_{1,\,3m} \\ f_{2,\,1m} & f_{2,\,2m} & f_{2,\,3m} \\ f_{3,\,1m} & f_{3,\,2m} & f_{3,\,3m} \\ f_{4,\,1m} & f_{4,\,2m} & f_{4,\,3m} \end{bmatrix}.$$

It follows from Eq. (42) that the gradient of the path Jacobian **J** can be found from ∇D by dropping the last row for each layer:

$$\nabla_m \mathbf{J} = \mathbf{sub}(\nabla_m D) = \begin{bmatrix} f_{1,\,1m} & f_{1,\,2m} & f_{1,\,3m} \\ f_{2,\,1m} & f_{2,\,2m} & f_{2,\,3m} \\ f_{3,\,1m} & f_{3,\,2m} & f_{3,\,3m} \end{bmatrix}. \tag{48}$$

Alternatively, we may write each component of $\nabla \mathbf{J}$ as

$$(\nabla \mathbf{J})_{mij} = f_{i,jm}, \tag{49}$$

where $i, j, m = 1, 2, 3$. From the definition of path Hessians in Eq. (3) and Eq. (42), we have

$$\mathbf{H}_{ijm} = f_{i,jm}, \tag{50}$$

where $i, j, m = 1, 2, 3$. Notice that we combine three Hessian matrices into a third-order Hessian tensor. Combining Eqs. (48), (49), and (50), we obtain the path Hessian **H** by reorganizing ∇D:

$$\boxed{\mathbf{H}_{ijm} = (\nabla D)_{mij},} \tag{51}$$

where $i, j, m = 1, 2, 3$.

Viewing the matrix gradient as a generalized matrix with vector entries, we compute ∇T and $\nabla \Gamma$ in Eq. (47) by evaluating the gradient of each matrix component:

$$\begin{aligned} \nabla(T_{ij}) &= \frac{\partial T_{ij}(\mathbf{p}, f(\mathbf{p}))}{\partial \mathbf{p}} \\ &= \frac{\partial T_{ij}(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}} + \frac{\partial T_{ij}(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)} \cdot D \end{aligned} \tag{52}$$

$$\begin{aligned} \nabla(\Gamma_{ik}) &= \frac{\partial \Gamma_{ik}(\mathbf{p}, f(\mathbf{p}))}{\partial \mathbf{p}} \\ &= \frac{\partial \Gamma_{ik}(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}} + \frac{\partial \Gamma_{ik}(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)} \cdot D, \end{aligned} \tag{53}$$

which follows from the chain rule. In Eqs. (52) and (53), the first term on the right-hand side is a gradient vector and the second term is a product of a gradient vector with a 4×3 Jacobian matrix; thus, both right-hand sides consist of a sum of two row vectors.

Expressing the matrix components T_{ij} and Γ_{ik} in terms of the Fermat equation F, we have

$$T_{ij}(\mathbf{p}, \mathbf{x}, \lambda) = \left(\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}}\right)_{ij} = \frac{\partial F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j}$$

$$\Gamma_{ik}(\mathbf{p}, \mathbf{x}, \lambda) = \left(\frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)}\right)_{ik} = \frac{\partial F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k},$$

where $x_4 = \lambda$. It follows that the computation of $\nabla(T_{ij})$ and $\nabla(\Gamma_{ik})$ in Eqs. (52) and (53) depends on the second partial derivatives of the four explicit equations F_i ($i = 1, 2, 3, 4$) shown in Eq. (15). That is,

$$\nabla(T_{ij}) = \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j \partial \mathbf{p}} + \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j \partial (\mathbf{x}, \lambda)} \cdot D$$

$$\nabla(\Gamma_{ik}) = \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k \partial \mathbf{p}} + \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k \partial (\mathbf{x}, \lambda)} \cdot D \qquad (54)$$

where $i, k = 1, 2, 3, 4$ and $j = 1, 2, 3$.

The Hessians for multiple bounces can also be computed iteratively from the ending point $\mathbf{q}$ to the starting point $\mathbf{p}$ in a similar fashion as the Jacobians; for details, see Appendix B. Consequently, for an N-bounce path, a second-order perturbation formula similar to Eq. (40) can be applied from the starting point $\mathbf{p}$ to the ending point $\mathbf{q}$ as

$$x_i' = \mathbf{x}_i + \mathbf{J}_i^* \Delta \mathbf{x}_{i-1} + \frac{1}{2}(\Delta \mathbf{x}_{i-1})^T \mathbf{H}_i^* \Delta \mathbf{x}_{i-1}, \qquad (55)$$

where $i = 1, 2, 3, \ldots, N$, $\Delta \mathbf{x}_{i-1} = \mathbf{x}'_{i-1} - \mathbf{x}_{i-1}$, and $\Delta \mathbf{x}_0 = \Delta \mathbf{p}$. By viewing Eq. (55) as a second-order Taylor expansion of the function $f_{i1} : \mathbf{x}_{i-1} \to \mathbf{x}_i$ around $\mathbf{x}_{i-1}$, the third-order tensor $\mathbf{H}^*_i$ is an extension of the reflection Jacobian $\mathbf{J}_i^*$ to second order, defined by $\partial^2 \mathbf{x}_i / \partial \mathbf{x}_{i-1}^2$. Note that the tensor product $(\Delta \mathbf{x}_{i-1})^T \mathbf{H}_i^* \Delta \mathbf{x}_{i-1}$ in Eq (55) must be expanded, as we did in Eq. (41).

6. APPLICATION TO FAST SPECULAR REFLECTION

In this section we briefly describe a practical application of the machinery for path perturbation that we have developed in the previous sections. A more detailed description of this application can be found in our companion paper [Chen and Arvo 2000]. We show how the formulas (12), (40), (41), and (55) that we derived for perturbing specular reflections can be used to quickly approximate specular reflections in arbitrary curved surfaces. By exploiting path coherence from frame to frame, path perturbation provides a faster alternative than ray tracing for computing dynamic specular reflection effects. Our perturbation-based approach for approximating specular reflections is applicable to scenes consisting only of diffuse and specular surfaces; each surface must be tessellated into polygons; and each specular surface must be equipped with a corresponding implicit equation and a ray intersection procedure. We now summarize the major steps of the algorithm.

Using standard ray tracing, a sparse set of rays with respect to a given vantage point is traced through an environment consisting of only the static specular surfaces found in the original scene. The resulting ray paths are stored in a hierarchical data structure that enables fast searching. Then, based on these precomputed reflection paths, the perturbation formulas are employed to interpolate new reflection paths reaching each vertex of each diffuse object found in the original scene. Thus, the reflection of each object vertex in each reflector is approximated by perturbing the nearest known reflection path. Multiple-bounce specular reflections are handled by perturbing nearest multiple-bounce reflection paths. Depending on the local curvature of the curved reflecting surface, the reflection position of each vertex can be approximated from nearby reflection paths to first-order accuracy using Eqs. (12) and (40) or to second-order accuracy using Eqs. (41) and (55). In general, the linear approximation suffices for nearly flat surfaces, while the quadratic approximation is required for more curved surfaces.

Using reflection paths associated with object vertices, the algorithm approximates reflections at the object level rather than the pixel level. For each reflected object, its image in a curved reflector is rendered by constructing the associated *virtual object*. From the reflection point obtained for each object vertex, a virtual vertex is placed at a distance from the eye that is equal to the original optical path length from the eye to the vertex. Then the virtual object corresponding to each reflection is created by connecting the virtual vertices. The virtual object thus constructed is positioned behind the reflective surface with respect to the view point. Finally, the entire scene, consisting of both real and virtual objects, is rendered in a single pass through a standard graphics pipeline, where reflectivity is simulated using alpha-blended transparency to “merge” specular reflections onto real objects. By preserving the optical path length of virtual objects, hidden surface removal and relative visibility are correctly handled by z-buffering. This virtual object method is similar to the approach described by Ofek and Rappoport [1998]. The details of our approach, based on path perturbation, are given by Chen [1999] and Chen and Arvo [2000].

One complication that arises with this approach is that the shading of diffuse virtual objects must be computed with respect to the original

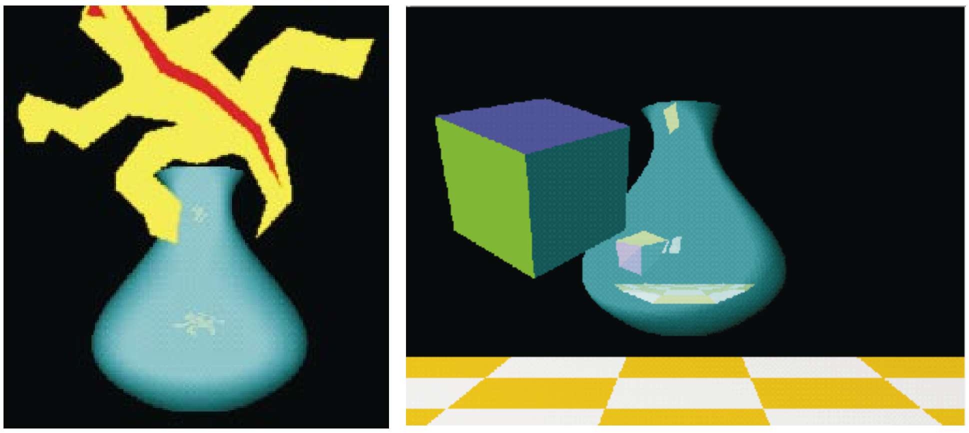

Fig. 5. One-bounce reflection imges generated by the perturbation method for a polygon (left) and a solid object (right). The visibility in the right image is handled correctly by z-buffering. The results are nearly identical to the ray traced image, yet the perturbed images can be computed very rapidly (approximately 0.1 seconds per update) as the lizard-shaped polygon or the cube is moved interactively.

positions of the objects, not their virtual coordinates. This is because the orientation of the virtual objects with respect to the light sources will in general be quite different from that of the actual object. Consequently, the shading must be precomputed when the virtual vertices are constructed, and not shaded by the graphics pipeline.

We have implemented our perturbation algorithm using Open Inventor on a SGI Indigo2. Virtual vertices were shaded according to the Phong model, so that the result would be consistent with the shading provided by the graphics hardware. One reflecting surface we chose is a vase model defined by an implicit function. This surface is a good test of our method because it has regions of mixed convexity. At some locations, an object will generate two reflection images on the vase, one near the top and the other near the bottom, as shown in Figure 5. The vase is tessellated with 20000 triangles for hardware rendering and equipped with a bounding slab hierarchy to speed up the ray-surface intersection. Thus the triangles are used both for hardware rendering and for ray/object intersection.

Figure 5 shows the first-level reflection images of a lizard-shaped polygon (left) and a solid cube (right) generated by our perturbation method. The diffuse lizard and cube both generate two reflection images on the vase. The bottom reflection is computed by linear perturbation using single-bounce path Jacobian, while the single-bounce path Hessian is used for the top reflections. The reflections computed by our perturbation method are nearly indistinguishable from the ray traced images. However, the perturbed images require only 0.1 seconds to update as the polygon or cube is moved interactively, while the ray traced images require 41 seconds per frame (excluding parsing and bounding slab creation) using PovRay.

By perturbing the sampled multiple-bounce reflection paths using recursive formula (37) for reflection Jacobians and/or (61) for reflection Hessians,

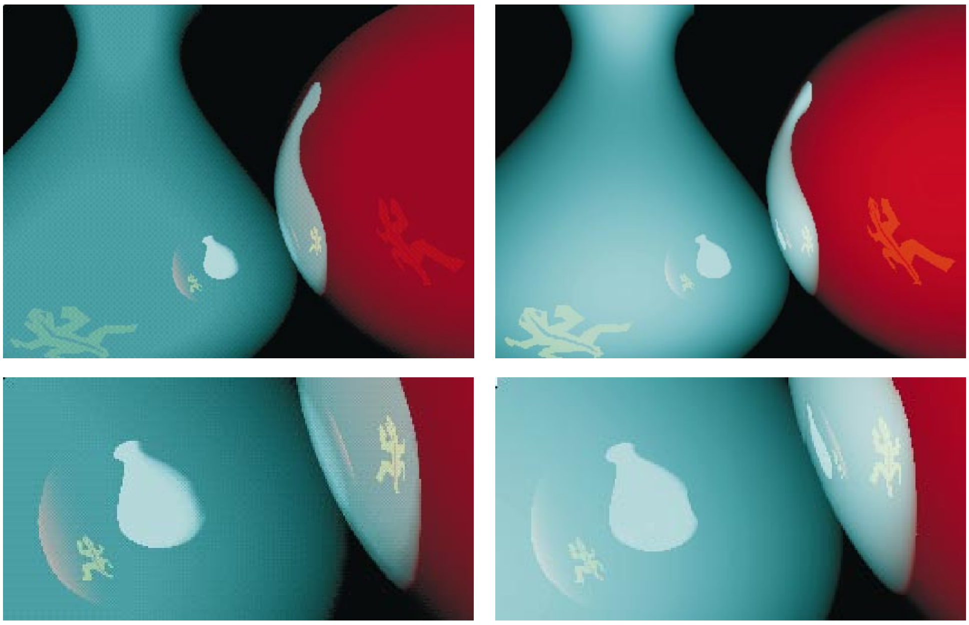

Fig. 6. Side-by-side comparison of multiple-bounce reflection images generated by the perturbation method (left) and ray tracing (right). Perturbed images on the left compute the reflections up to two levels; ray traced images on the right are generated with a maximum depth of 5. The closeup view shows they are indistinguishable, yet perturbed images can be updated in less than 1 second while moving the lizard interactively.

multiple-bounce specular reflections can also be handled using perturbation methods almost as easily as one-bounce reflections. To illustrate its use, we have rendered a scene consisting of two reflectors—a vase and a sphere—and a diffuse lizard-shaped polygon. Figure 6 shows a side-by-side comparison of the multiple-bounce reflection images of the scene generated by our perturbation method and ray tracing (rendered by PovRay). The top image is the full view of the scene, and the bottom shows a closeup of the second-level reflection near the bottom. On the left, all the first-level and second-level reflections of the vase, the sphere, and the lizard are generated by linear perturbation of specular-only paths using multiple-bounce path Jacobians. The reflections up to the second-level are nearly identical in the left and right images. The slight difference in shading between the two images is due to minor differences in the Phong shading algorithms incorporated in PovRay and Open Inventor. The higher-level (> 2) reflections shown in the ray traced image on the right can be generated similarly by perturbing the reflection paths with more bounces.

We compared the performance of the two methods for this multiple-bounce reflection scene at an image resolution of 640×480 by excluding sampling time from the perturbation method and excluding parsing and creation of the bounding slab tree from the ray tracing method. For the perturbation method, 80×60 rays were cast to sample the surface tessellations consisting of 1250 triangles. The perturbation method required 9.6

seconds to render the initial image on the left, while the ray traced image required 67 seconds. Since our perturbation method makes use of path coherence from frame to frame, it is very efficient in updating the scene during interaction. The image on the left can be updated within 0.7 seconds while moving the lizard polygon interactively. For ray tracing, the update rate is the same as rendering a new image from scratch, thus it still takes 67 seconds.

Our approach is most effective for scenes in which the view point and reflecting surfaces are static and only the diffuse objects are dynamic. However, the very same machinery that we developed for path Jacobians and path Hessians still applies to the case of moving viewpoints and reflectors, as long as we resample the precached rays. The resulting algorithm is still much faster than conventional ray tracing, but the speedup is less dramatic due to the resampling. We also note that our path Jacobian analysis fails when the nonsingularity condition (9) required for the Implicit Function Theorem is violated. However, this degenerate case can only occur when a ray is tangent to a reflecting surface, which we may simply regard as a nonreflecting path. A more fundamental limitation of our approach is that the path perturbation theory assumes the reflection point $\mathbf{x}$ varies continuously as a function of the position of the varying endpoint $\mathbf{p}$, as discussed in Section 2.2. When this condition is violated, such as near occlusion changes, the algorithm will produce incorrect results.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced the concepts *path Jacobian* and *path Hessian* to describe the linear and quadratic perturbations of a specular path, and presented a closed-form perturbation formula for specular reflections. This formula is expressed as a Taylor expansion and is based on closed-form expressions for the path Jacobian and path Hessian, which are derived from Fermat's principle, the Implicit Function Theorem, and tensor differentiation. Our method works for any implicitly-defined reflecting surface and multiple-bounce specular reflection paths, and provides a new mathematical tool for image synthesis. An algorithm that makes use of the perturbation formula has been demonstrated for rapid approximation of specular reflections in arbitrary curved surfaces. Our test results demonstrate the high accuracy and dramatic performance improvements that can be achieved by path perturbation.

A natural extension to this technique also accommodates refraction. Since Fermat's principle places no restriction on reflections or refractions, the tools used in deriving path Jacobians and path Hessians for a reflection path also apply to refraction; the only difference is the use of the refractive index. Consequently, nearly the same algorithm described in Section 6 could be applied to simulate lens effects.

To quantitatively estimate the error arising from the path Jacobian and/or path Hessian approximations, our path perturbation theory should be enhanced with rigorous error analysis, as in the pencil tracing approach proposed by Shinya et al. [1987]. Another interesting topic is to extend specular path perturbation to smooth parametric surfaces without requiring an implicit representation. This could be accomplished using a locally-defined height field over the tangent plane of the surface to obtain an implicit function in the neighborhood of a surface point.

Perturbation methods of this nature can also find many other potential applications in image synthesis. For example, the idea of path perturbation can be used to speed up the computation of caustics in the approach proposed by Mitchell and Hanrahan [Chen 1999]. Instead of computing the reflecting path for every pixel on the floor using costly interval analysis and automatic differentiation, we could sparsely sample it and interpolate the remaining paths using perturbation. Another direct application would be image-based rendering. By projecting the path Jacobian into the image plane, we can introduce and compute an *image Jacobian*, which approximates how pixels move with respect to changes in the viewer. Then, an image differentiation approach could be applied to image warping where specular effects are prominent, as in the work of Lischinski and Rappoport [1998]. Finally, the benefit of analytical path perturbation over random perturbation may provide a new low-variance mutation strategy in the context of metropolitan light transport [Veach and Guibas 1997].

APPENDIX

A. SIMPLIFICATION OF EQUATION (18)

Observing the special structures of the Fermat equation (15) and the resulting two matrices on the right-hand side of Eq. (18), we can actually compute the path Jacobian $\mathbf{J}$ directly, without introducing the operator **sub**. Let the vector $\mathbf{h} = \partial g(\mathbf{x})/\partial \mathbf{x}$, and we observe that:

(1) The matrix $\partial F(\mathbf{p}, \mathbf{x}, \lambda)/\partial(\mathbf{x}, \lambda)$ is symmetric, and can be expressed as the block matrix

$$\begin{bmatrix} M_{3\times 3} & \mathbf{h}_{3\times 1} \\ \mathbf{h}^{\mathrm{T}}_{1\times 3} & 0 \end{bmatrix}, \tag{56}$$

where M is a 3×3 symmetric matrix.

(2) The 4×3 matrix $\partial F(\mathbf{p}, \mathbf{x}, \lambda)/\partial \mathbf{p}$ has a form

$$\begin{bmatrix} B_{3\times 3} \\ 0_{1\times 3} \end{bmatrix}.$$

It follows that the inverse of matrix (56) is also of the form

$$\begin{bmatrix} A_{3\times 3} & \mathbf{v}_{3\times 1} \\ \mathbf{v}^{\mathrm{T}}_{1\times 3} & d \end{bmatrix},$$

where A is symmetric and satisfies the following equation

$$\begin{bmatrix} M_{3\times3} & \mathbf{h}_{3\times1} \\ \mathbf{h}^{\mathrm{T}}_{1\times3} & 0 \end{bmatrix} \begin{bmatrix} A_{3\times3} & \mathbf{v}_{3\times1} \\ \mathbf{v}^{\mathrm{T}}_{1\times3} & d \end{bmatrix} = \begin{bmatrix} MA + \mathbf{h}\mathbf{v}^{\mathrm{T}} & M\mathbf{v} + d\mathbf{h} \\ \mathbf{h}^{\mathrm{T}}A & \mathbf{h}^{\mathrm{T}}\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3\times3} & 0 \\ 0 & 1 \end{bmatrix}, \tag{57}$$

where the scalar d, the vector $\mathbf{v}$, and the 3×3 symmetric submatrix A are to be determined. Here $\mathbf{I}$ denotes the 3×3 identity matrix. Solving the linear system (Eq. (57)) for A, we obtain

$$A = M^{-1}\left(\mathbf{I} - \frac{\mathbf{h}\mathbf{h}^{\mathrm{T}}M^{-1}}{\mathbf{h}^{\mathrm{T}}M^{-1}\mathbf{h}}\right). \tag{58}$$

By Eq. (18), the path Jacobian $\mathbf{J}$ we are interested in is composed of the first three rows of

$$\begin{bmatrix} A_{3\times3} & \mathbf{v}_{3\times1} \\ \mathbf{v}^{\mathrm{T}}_{1\times3} & d \end{bmatrix} \begin{bmatrix} B_{3\times3} \\ \mathbf{0}_{1\times3} \end{bmatrix}.$$

Thus, we may evaluate it directly as the product of two 3×3 matrices,

$$\mathbf{J} = -A \times B. \tag{59}$$

Equation (59) provides a more efficient way to evaluate $\mathbf{J}$ in practice, since the matrix to be inverted is 3×3 instead of the 4×4 matrix in Eq. (18). Using the same technique, we can derive an analogous formula for $\mathbf{J}_i$ ($i = 1, 2, \ldots, N$) that simplifies Eq. (37) in the case of multiple bounces.

B. THE HESSIANS FOR N-BOUNCE PATHS

In accordance with Eq. (40) in the first-order approximation, we may approach the problem of perturbing an N-bounce path to second-order accuracy by incrementally updating the reflection points to the second order in a prescribed order using Eq. (55), which involves third-order tensors $\mathbf{H}^*_i$s as quadratic extensions of reflection Jacobian $\mathbf{J}^*_i$s. Expressing the position of $\mathbf{x}_i$ as a function f_{i1} of its previous point $\mathbf{x}_{i-1}$, which was shown to exist in Section 4.2, $\mathbf{H}^*_i$ at the ith reflection point is defined as the second-order derivative of f_{i1} with respect to $\mathbf{x}_{i-1}$. For consistency, we call $\mathbf{H}^*_i$s *reflection Hessians*. In this Appendix, we show that the recurrence relation for reflection Jacobian $\mathbf{J}^*_i$s in an N-bounce path can be extended to the second order, yielding a corresponding recurrence relation for reflection Hessian $\mathbf{H}^*_i$s.

While deriving path Jacobians for an N-bounce path, we have shown that there exists an implicitly-defined function $f_i : \mathbf{x}_{i-1} \rightarrow (\mathbf{x}_i, \lambda_i)$ associated with each reflection point $\mathbf{x}_i$, and it consists of two components: $f_{i1} : \mathbf{x}_{i-1}$

$\rightarrow \mathbf{x}_i$ and $f_{i2} : \mathbf{x}_{i-1} \rightarrow \lambda_i$. Analogous to (43), the first-order approximation for an N-bounce path in the recursive formula (37) yields

$$T_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) = -\Gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) D_i(\mathbf{x}_{i-1}), \tag{60}$$

where T_i, Γ_i, and D_i are defined as

$$T_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) = \left[\frac{\partial F_{gi}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i)}{\partial \mathbf{x}_{i-1}}\right]_{4\times 3}$$

$$\Gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) = [A_i + \mathbf{aug}(B_i \cdot \mathbf{J}^*_{i+1})]_{4\times 4}$$

$$D_i(\mathbf{x}_{i-1}) = \left[\frac{\partial f_i(\mathbf{x}_{i-1})}{\partial \mathbf{x}_{i-1}}\right]_{4\times 3}.$$

A_i and B_i are defined in Eq. (38). Since Eq. (60) has the same form as Eq. (43), following the same derivation will result in a similar formula for the gradient of Fermat Jacobian D_i:

$$\boxed{\nabla_m D_i = -\Gamma_i^{-1}(\nabla_m T_i + \nabla_m \Gamma_i \cdot D_i),} \tag{61}$$

for $m = 1, 2, 3$ and $i = 1, \ldots, N$ in an N-bounce path. Note that ∇ is the gradient operator with respect to the previous point $\mathbf{x}_{i-1}$. Furthermore, the relation between the function f_i and its component f_{i1} suggests that Eq. (51) shown in the one-bounce case still holds for the conversion between the reflection Hessian $\mathbf{H}^*_i$ and the gradient of the corresponding Fermat Jacobian, ∇D_i.

Due to the existence of the functions $f_i : \mathbf{x}_{i-1} \rightarrow (\mathbf{x}_i, \lambda_i)$ and $f_{i+1} : \mathbf{x}_i \rightarrow (\mathbf{x}_{i+1}, \lambda_{i+1})$, the matrix variables T_i, Γ_i can be considered as functions of $\mathbf{x}_{i-1}$. That is,

$$(T_i)_{jk}(\mathbf{x}_{i-1}, f_{i1}(\mathbf{x}_{i-1}), f_{(i+1)1}(f_{i1}(\mathbf{x}_{i-1})), f_{i2}(\mathbf{x}_{i-1})) \tag{62}$$

$$(\Gamma_i)_{jl}(\mathbf{x}_{i-1}, f_{i1}(\mathbf{x}_{i-1}), f_{(i+1)1}(f_{i1}(\mathbf{x}_{i-1})), f_{i2}(\mathbf{x}_{i-1})), \tag{63}$$

for $j, l = 1, 2, 3, 4$ and $k = 1, 2, 3$. The gradient ∇T_i can be computed by differentiating each element $(T_i)_{jk}$ with respect to the independent variable $\mathbf{x}_{i-1}$,

$$\nabla((T_i)_{jk}) = \frac{\partial (T_i)_{jk}(\mathbf{x}_{i-1}, f_{i1}(\mathbf{x}_{i-1}), f_{(i+1)1}(f_{i1}(\mathbf{x}_{i-1})), f_{i2}(\mathbf{x}_{i-1}))}{\partial \mathbf{x}_{i-1}}$$

$$= \frac{\partial (T_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial (T_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \frac{\partial f_{(i+1)1}}{\partial \mathbf{x}_i} \cdot \frac{\partial f_{i1}}{\partial \mathbf{x}_{i-1}} + \frac{\partial (T_i)_{jk}}{\partial (\mathbf{x}_i, \lambda_i)} \cdot \frac{\partial f_i}{\partial \mathbf{x}_{i-1}}$$

$$= \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}^*_{i+1} \cdot \mathbf{J}^*_i + \frac{\partial(T_i)_{jk}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot D_i. \tag{64}$$

Thus, each element of ∇T_i can be obtained from the component of the gradient vector on the left of Eq. (64):

$$(\nabla T_i)_{mjk} = \left(\frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}^*_{i+1} \cdot \mathbf{J}^*_i + \frac{\partial(T_i)_{jk}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot D_i \right)_m .$$

In computing $\nabla \Gamma_i$, we must consider the two cases:

$$(\Gamma_i)_{jl} = \begin{cases} (A_i)_{jl} + \sum_{k=1}^{3} (B_i)_{jk} (\mathbf{J}^*_{i+1})_{kl} & l = 1, 2, 3 \\ (A_i)_{jl} & l = 4 \end{cases} \tag{65}$$

When $l = 1, 2, 3$, we differentiate the first equation in (65) with respect to $\mathbf{x}_{i-1}$ and express the result in terms of the gradient of matrices:

$$(\nabla \Gamma_i)_{mjl} = (\nabla A_i)_{mjl} + \sum_{k=1}^{3} ((\nabla B_i)_{mjk} (\mathbf{J}^*_{i+1})_{kl} + (B_i)_{jk} (\nabla C_i)_{mkl}). \tag{66}$$

Since ∇ above is the gradient operator with respect to $\mathbf{x}_{i-1}$, another symbol ∇C_i is introduced for the gradient of $\mathbf{J}^*_{i+1}$ with respect to $\mathbf{x}_{i-1}$, which is different from $\nabla \mathbf{J}^*_{i+1}$. That is,

$$\nabla((\mathbf{J}^*_{i+1})_{kl}) = \frac{\partial(\mathbf{J}^*_{i+1})_{kl}}{\partial \mathbf{x}_i} \tag{67}$$

$$\nabla((C_i)_{kl}) = \frac{\partial(\mathbf{J}^*_{i+1})_{kl}}{\partial \mathbf{x}_{i-1}} = \nabla((\mathbf{J}^*_{i+1})_{kl}) \cdot \mathbf{J}^*_i. \tag{68}$$

In Eq. (67), we interpret $\mathbf{J}^*_{i+1} = \partial f_{(i+1)1}(\mathbf{x}_i) / \partial \mathbf{x}_i$ as a function of $\mathbf{x}_i$ and take its derivative with respect to $\mathbf{x}_i$. However, with $f_i : \mathbf{x}_{i-1} \rightarrow (\mathbf{x}_i, \lambda_i)$, we can also consider $\mathbf{J}^*_{i+1}$ as a function dependent on $\mathbf{x}_{i-1}$ and calculate its derivative with respect to $\mathbf{x}_{i-1}$ in Eq. (68). $\nabla \mathbf{J}^*_{i+1}$ and ∇C_i are related by the identity (Eq. 68) derived from the chain rule. Writing Eq. (66) in terms of the mth layers of $\nabla \Gamma_i$, ∇A_i, ∇B_i and ∇C_i, we obtain a matrix form

$$\nabla_m \Gamma_i = \nabla_m A_i + \nabla_m B_i \cdot \mathbf{J}^*_{i+1} + B_i \cdot \nabla_m C_i \tag{69}$$

for $m = 1, 2, 3$. By considering A_i, B_i as functions of $\mathbf{x}_{i-1}$, similar to Eq. (62), we can compute ∇A_i, ∇B_i in the same way as ∇T_i. Thus,

$$\nabla((A_i)_{jr}) = \frac{\partial(A_i)_{jr}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(A_i)_{jr}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}^*_{i+1} \cdot \mathbf{J}^*_i + \frac{\partial(A_i)_{jr}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot D_i$$

$$\nabla((B_i)_{jk}) = \frac{\partial(B_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(B_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}^*_{i+1} \cdot \mathbf{J}^*_i + \frac{\partial(B_i)_{jk}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot D_i$$

$$\nabla((C_i)_{kl}) = \nabla((\mathbf{J}^*_{i+1})_{kl}) \cdot \mathbf{J}^*_i, \tag{70}$$

where j, $r = 1, 2, 3, 4$ and k, $l = 1, 2, 3$. Combining Eq. (69) with the case of $l = 4$, we can write each component of $\nabla\Gamma_i$ as

$$(\nabla\Gamma_i)_{mjl} = \begin{cases} (\nabla_m A_i + \nabla_m B_i \cdot \mathbf{J}^*_{i+1} + B_i \cdot \nabla_m C_i)_{jl} & l = 1, 2, 3 \\ (\nabla A_i)_{mjl} & l = 4 \end{cases}$$

for $m = 1, 2, 3$ and $j = 1, 2, 3, 4$. Note that all partial derivatives on the right-hand side of Eqs. (64) and (70) can be evaluated from the second partial derivatives of the Fermat equation Fg_i, just as we did in Eq. (54).

Finally, the reflection Hessian $\mathbf{H}^*_i$ is obtained by reorganizing ∇D_i using Eq. (51). As can be seen from Eq. (70), the reflection Hessian at a reflection point $\mathbf{x}_i$ is not only dependent on its reflection Jacobian $\mathbf{J}^*_i$, but also dependent on the reflection Jacobian $\mathbf{J}^*_{i+1}$, and the reflection Hessian (implied in $\nabla\mathbf{J}^*_{i+1}$) computed for the following point $\mathbf{x}_{i+1}$. This dependence suggests that for the second-order approximation of a multiple-bounce path, we must compute the reflection Jacobian $\mathbf{J}^*_i$ followed by the reflection Hessian $\mathbf{H}^*_i$ for each reflection point, starting from the ending point $\mathbf{q}$ and recursively propagating to the starting point $\mathbf{p}$.

ACKNOWLEDGMENTS

The authors wish to thank Anil Hirani and Al Barr for many valuable discussions, Don Mitchell and Pat Hanrahan for their patience in answering our questions, and Mark Meyer and the anonymous reviewers for their helpful comments.

REFERENCES

ADELSON, S. J. AND HODGES, L. F. 1995. Generating exact ray-traced animation frames by reprojection. *IEEE Comput. Graph. Appl. 15*, 3, 43–52.

ADELSON, S. J. AND HODGES, L. 1993. Stereoscopic ray tracing. *Visual Comput. 10*, 3, 127–144.

APOSTOL, T. M. 1969. *Calculus II: Multi-Variable Calculus and Linear Algebra, with Applications to Differential Equations and Probability*. John Wiley and Sons, Inc., New York, NY.

ARVO, J. AND KIRK, D. 1989. A survey of ray tracing acceleration techniques. In *An Introduction to Ray Tracing*, A. S. Glassner, Ed. Academic Press Ltd., London, UK, 201–262.

AZIZ, A. AND NA, T. Y. 1984. *Perturbation Methods in Heat Transfer*. Hemisphere Publishing Corp., New York, NY.

BADT, S. JR. 1988. Two algorithms for taking advantage of temporal coherence in ray tracing. *Visual Comput. 4*, 3 (Sept.), 123–132.

BORN, M. AND WOLF, E. 1965. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. 3rd ed. Pergamon Press, Inc., Tarrytown, NY.

BRIÈRE, N. AND POULIN, P. 1996. Hierarchical view-dependent structures for interactive scene manipulation. In *Proceedings of the 23rd Annual Conference on Computer Graphics*

(SIGGRAPH '96, New Orleans, LA, Aug. 4–9), J. Fujii, Chair. Annual conference series. ACM Press, New York, NY, 83–90.

CHAPMAN, J., CALVERT, T. W., AND DILL, J. 1990. Exploiting temporal coherence in ray tracing. In *Proceedings of the Conference on Graphics Interface* (Halifax, Nova Scotia, May 14–18), S. MacKay and E. M. Kidd, Eds. Canadian Information Processing Society, Toronto, Canada, 196–204.

CHAPMAN, J., CALVERT, T. W., AND DILL, J. 1991. Spatio-temporal coherence in ray tracing. In *Proceedings of the Conference on Graphics Interface '91* (Calgary, Alberta, June 3–7), W. A. Davis and B. Wyvill, Chairs. Morgan Kaufmann Publishers Inc., San Francisco, CA, 101–108.

CHEN, M. 1999. Perturbation methods for image synthesis. Master's Thesis. California Institute of Technology, Pasadena, CA. ftp://ftp.cs.caltech.edu/tr/cs-tr-99-05.ps.Z.

CHEN, M. AND ARVO, J. 2000. Perturbation methods for interactive specular reflections. *IEEE Trans. Visual. Comput. Graph. 6*, 3 (July-Sept.), 253–264.

COLE, J. D. 1968. *Perturbation Methods in Applied Mathmatics*. Blaisdell Publishing Co., London, UK.

COOK, R. L. 1998. Shade trees. In *Proceeding of the 25th SIGGRAPH Conference Celebration on Seminal Graphics: Pioneering Effects that Shaped the Field* (SIGGRAPH '98), R. Wolfe, Ed. ACM Press, New York, NY, 137–145.

DYKE, M. V. 1964. *Perturbation Methods in Fluid Mechanics*. Academic Press, Inc., New York, NY.

IGEHY, H. 1999. Tracing ray differentials. In *Proceedings of the Conference on Computer Graphics* (SIGGRAPH 99, Aug.). ACM Press, New York, NY, 179–186.

JEVANS, D. A. 1992. Object space temporal coherence for ray tracing. In *Proceedings of the Conference on Graphics Interface '92* (Vancouver, BC, Canada, May 11–15), K. S. Booth and A. Fournier, Eds. Morgan Kaufmann Publishers Inc., San Francisco, CA, 176–183.

LIN, C. C. AND SEGEL, L. A. 1988. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. SIAM, Philadelphia, PA.

LISCHINSKI, D. AND RAPPOPORT, A. 1998. Image-based rendering for non-diffuse synthetic scenes. In *Proceedings of the 9th Eurographics Workshop on Rendering* (Vienna, Austria, June). Springer-Verlag, New York, NY, 301–314.

MARSDEN, J. E. AND HOFFMAN, M. J. 1993. *Elementary Classical Analysis*. W. H. Freeman and Co., New York, NY.

MITCHELL, D. AND HANRAHAN, P. 1992. Illumination from curved reflectors. *SIGGRAPH Comput. Graph. 26*, 2 (July), 283–291.

MURAKAMI, K. AND HIROTA, K. 1990. Incremental ray tracing. In *Conference Proceedings of the Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics* (Rennes, France, June). 15–29.

NEUMANN, P. M. 1998. Reflections on reflection in a spherical mirror. *Am. Math. Monthly 105*, 6 (June-July), 523–528.

OFEK, E. AND RAPPOPORT, A. 1998. Interactive reflections on curved objects. In *Proceedings of the 25th Annual Conference on Computer Graphics* (SIGGRAPH '98, Orlando, FL, July 19–24), S. Cunningham, W. Bransford, and M. F. Cohen, Chairs. ACM Press, New York, NY, 333–342.

SEGEL, L. A. AND HANDELMAN, G. H. 1977. *Mathematics Applied to Continuum Mechanics*. Macmillan Publishing Co., Inc., Indianapolis, IN.

SÉQUIN, C. H. AND SMYRL, E. K. 1989. Parameterized ray-tracing. *SIGGRAPH Comput. Graph. 23*, 3 (July), 307–314.

SHINYA, M., TAKAHASHI, T., AND NAITO, S. 1987. Principles and applications of pencil tracing. *SIGGRAPH Comput. Graph. 21*, 4 (July), 45–54.

VEACH, E. AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (SIGGRAPH '97, Los Angeles, CA, Aug. 3–8), G. S. Owen, T. Whitted, and B. Mones-Hattal, Chairs. ACM Press/Addison-Wesley Publ. Co., New York, NY, 65–76.

Received: June 1999; revised: November 2000; accepted: January 2001