

# Theory and Application of Specular Path Perturbation

Min Chen James Arvo

California Institute of Technology

January 30, 2001

## Abstract

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. 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 of high accuracy and nearly indistinguishable from ray traced images.

**Keywords:** perturbation theory, implicit surfaces, optics, ray tracing, specular reflection

# 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, 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 up the process of ray casting [4]. The rationale for considering 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, then used in the generation of a nearby image. Cook [13], and Séquin and Smyrl [26] handled material changes in this way by re-traversing the retained ray trees with modified parameters. Murakami and Hirota [22], and Jevans [16] 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 [7] 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 [17], and Adelson and Hodges [1] 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 [2]. Unfortunately, reprojection only works for the first level rays and is applicable only to diffuse scenes. Chapman et al. [8, 9] 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 [10]. Previous work that explored the use of ray path perturbations includes pencil tracing [27] and ray differentials [15], 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* [18].

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

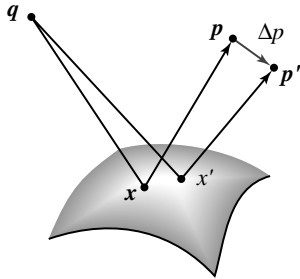


Figure 1: *Reflections tend to have a great degree of coherence, the corresponding reflection points  $\mathbf{x}$  and  $\mathbf{x}'$  will converge as  $\mathbf{p}$  approaches  $\mathbf{p}'$ .*

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 [12] including continuum mechanics [14] and heat transfer [5]. In this paper, we expand the list to include image synthesis.

## 2.2 Path Coherence

Let us consider what happens to a given optical path connecting points  $\mathbf{p}$  and  $\mathbf{q}$ , via a specular reflection in a curved surface, when  $\mathbf{p}$  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  $\mathbf{p}$  approaches  $\mathbf{p}'$ , the corresponding reflection points  $\mathbf{x}$  and  $\mathbf{x}'$  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 [1], 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<sup>1</sup> of the two points, that is,

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

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:

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

where  $\Psi(\mathbf{y}) \equiv \phi(\mathbf{q}, \mathbf{y})$ . We call  $\Psi$  the *path function* with respect to the point  $\mathbf{p}$  since we are ultimately interested in sequences of reflection points  $\mathbf{x}_1, \dots, \mathbf{x}_N$  forming an optical path from  $\mathbf{p}$  to  $\mathbf{q}$ . For a small  $\epsilon$ , 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  $\Psi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  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  $\Psi$  can be expressed in Cartesian tensor notation as

$$\Psi_i(\mathbf{p} + \Delta\mathbf{p}) = \Psi_i(\mathbf{p}) + \sum_j \Psi_{i,j} \epsilon_j$$

---

<sup>1</sup>More precisely,  $\phi$  is a relation since there may exist several reflection points for two given points, but locally it is a function.

$$\begin{aligned}
& + \frac{1}{2} \sum_{jk} \Psi_{i,jk} \epsilon_j \epsilon_k \cdots \\
& + \frac{1}{n!} \sum_{jk\dots r} \Psi_{i,jk\dots r} \epsilon_j \epsilon_k \cdots \epsilon_r + \cdots
\end{aligned}$$

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

$$\begin{aligned}
\Psi_{i,j} &= \frac{\partial \Psi_i}{\partial p_j}, \\
\Psi_{i,jk} &= \frac{\partial^2 \Psi_i}{\partial p_j \partial p_k}, \\
&\dots
\end{aligned}$$

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  $\Psi$ , 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  $\Psi_1, \Psi_2$  and  $\Psi_3$  and putting them into familiar matrix forms, we obtain the Jacobian matrix

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

and three Hessian matrices

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

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  $\Psi$  about the given path through  $\mathbf{p}$  can be expressed as:

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

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  $\Psi$ , that is,

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

and

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

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 first- and second-order approximations to  $\Psi$ , respectively.

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 [27] employed paraxial ray theory [6] to approximate the propagation of paraxial rays by a matrix, which linearizes each optical event (transfer, reflection, or refraction). The ray differential framework [15] 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 perturb both ends of a ray traced path freely, 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 will be 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 [6, 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,  $\eta$ :

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

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 equation (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 principle 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,

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

where  $\mathbf{x}_0 = \mathbf{p}$  and  $\mathbf{x}_{N+1} = \mathbf{q}$ . For a reflection path in a uniform medium, equation (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,

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

where  $\mathbf{x} \in \mathbb{R}^n$ ,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , and  $m \leq n$ . Here,  $f$  is the *objective function* and  $\mathbf{h}$  is the *constraint function*. The *Lagrange Multiplier Theorem* [3, pp. 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  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , subject to  $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ , where  $\mathbf{h} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $m \leq n$ , and  $\mathbf{x}^*$  is a regular point. Then, there exists  $\lambda^* \in \mathbb{R}^m$  such that

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

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  $\mathbf{h} = [h_1, \dots, h_m]^\top$  at  $\mathbf{x}^*$ .

We refer to the vector  $\lambda^*$  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

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

Here the scalar  $\lambda^*$  is referred as the *Lagrange multiplier*.

### 3.3 The Implicit Function Theorem

By means of 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 non-linear 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  $F : A \rightarrow \mathbb{R}^m$  be a function of class  $C^p$ . Suppose  $(\mathbf{x}_0, \mathbf{y}_0) \in A$  such that  $F(\mathbf{x}_0, \mathbf{y}_0) = \mathbf{0}$  and

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

where  $F = (F_1, \dots, F_m)$ , 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  $f : U \rightarrow V$  such that

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

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

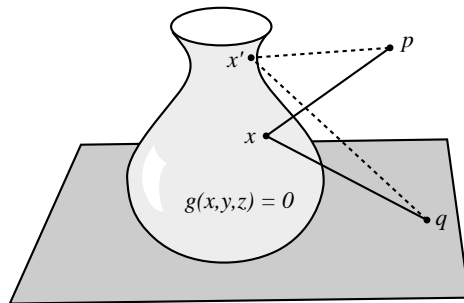


Figure 2: Reflection paths obey Fermat’s variational principle, stating that the length of the optical path connecting  $\mathbf{p}$  and  $\mathbf{q}$  is a local extremum. For any  $\mathbf{p}$  and  $\mathbf{q}$  there may be many such paths.

See, for example, Marsden and Hoffman [20, pp.211–213] for a proof of the Implicit Function Theorem. By differentiating both sides of equation (10) 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*

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = - \begin{bmatrix} \frac{\partial F_1}{\partial y_1} & \dots & \frac{\partial F_1}{\partial y_m} \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial y_1} & \dots & \frac{\partial F_m}{\partial y_m} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \dots & \frac{\partial F_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial F_m}{\partial x_1} & \dots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}. \quad (11)$$

Note that the inverse on the right hand side of equation (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  $\mathbf{p}$  and  $\mathbf{q}$  in space, the problem of finding a point  $\mathbf{x}$  on the mirror where a ray is reflected from  $\mathbf{p}$  to  $\mathbf{q}$  has a long history in optics. For a spherical mirror, this problem is known as *Alhazen’s problem* [23]. Even for simple

convex shapes, such as a sphere, it is difficult to find  $\mathbf{x}$  analytically. For non-convex surfaces the problem is even more difficult, as there be many such paths connecting two points, as shown in Figure 2. By computing the path Jacobian  $\mathbf{J} = \partial\mathbf{x}/\partial\mathbf{p}$  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,

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

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  $\Psi$ , 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, as in the same fashion demonstrated by Mitchell and Hanrahan [21], we obtain

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

where  $\lambda$  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 equation (13) as an implicit equation that relates  $\mathbf{p}$ ,  $\mathbf{x}$  and  $\lambda$ :

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

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

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

where  $i = 1, 2, 3$ .

Unfortunately, it is generally impossible to solve these non-linear equations for the reflection point  $\mathbf{x}$  in a closed form, even for trivial functions  $g(\mathbf{x})$  [23]. Mitchell and Hanrahan [21] 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 the 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  $f : \mathbb{R}^3 \rightarrow \mathbb{R}^4$  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 equation (11), which yields

$$\left[ \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} \right]_{4 \times 3} = - \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{x}, \lambda)} \right]_{4 \times 4}^{-1} \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial(\mathbf{p})} \right]_{4 \times 3}. \tag{17}$$

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

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

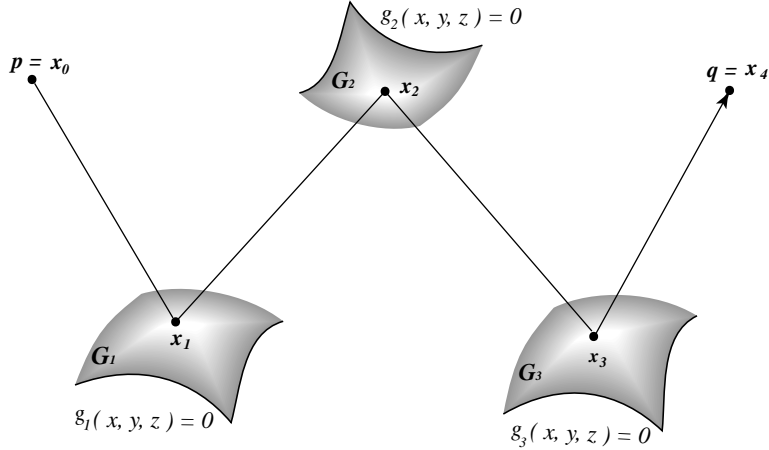


Figure 3: A reflection path from  $\mathbf{p}$  to  $\mathbf{q}$  via three reflection points  $\mathbf{x}_1$ ,  $\mathbf{x}_2$  and  $\mathbf{x}_3$  on three implicitly-defined surfaces  $\mathbf{G}_1$ ,  $\mathbf{G}_2$  and  $\mathbf{G}_3$ .

which characterizes the variation in  $\mathbf{x}$  with respect to  $\mathbf{p}$ . Alternatively, we may simplify equation (18) in such a way that the path Jacobian  $\mathbf{J}$  can be computed directly, without computing the inverse of a  $4 \times 4$  matrix, as shown in Appendix A. Note that all quantities on the right of equation (18) can be obtained from the Fermat equation (15) and the implicit function  $g$ . We shall refer to the  $4 \times 3$  matrix on the left of equation (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  $\mathbf{x}_1, \dots, \mathbf{x}_N$ , 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  $\Psi_i$  of the endpoint  $\mathbf{p}$ , we may define the path Jacobian  $\mathbf{J}_i$  at  $\mathbf{x}_i$  as the derivative of  $\Psi_i$  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

$$\mathbf{x}'_i = \mathbf{x}_i + \mathbf{J}_i \Delta \mathbf{p} \quad (19)$$

for  $i = 1, 2, \dots, N$ , where the  $N$  path Jacobians  $\mathbf{J}_i$ 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 the  $N$  reflection points are located in such a way that the optical length of the path  $\mathbf{x}_0 - \mathbf{x}_1 - \dots - \mathbf{x}_{N+1}$  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  $\mathbf{g} = [g_1, \dots, g_N]^T$  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 [21]:

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

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

$$\begin{aligned} F_{g_1}(\mathbf{p}, \mathbf{x}_1, \mathbf{x}_2, \lambda_1) &= 0, \\ &\vdots \\ F_{g_N}(\mathbf{x}_{N-1}, \mathbf{x}_N, \lambda_N) &= 0, \end{aligned} \quad (21)$$

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

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

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

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

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

$$f : \mathbb{R}^3 \rightarrow \mathbb{R}^4 \times \dots \times \mathbb{R}^4, \quad (23)$$

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

$$\left[ \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} \right]_{4N \times 3} = - \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}_1, \lambda_1, \dots, \mathbf{x}_N, \lambda_N)}{\partial (\mathbf{x}_1, \lambda_1, \dots, \mathbf{x}_N, \lambda_N)} \right]_{4N \times 4N}^{-1} \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}_1, \lambda_1, \dots, \mathbf{x}_N, \lambda_N)}{\partial \mathbf{p}} \right]_{4N \times 3}. \quad (24)$$

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

Unfortunately, to compute path Jacobians using equation (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,

$$\mathbf{J}_i = \frac{\partial \mathbf{x}_i}{\partial \mathbf{p}} = \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} \cdot \frac{\partial \mathbf{x}_{i-1}}{\partial \mathbf{x}_{i-2}} \cdots \frac{\partial \mathbf{x}_1}{\partial \mathbf{p}} \quad (25)$$

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

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

at each reflection point  $\mathbf{x}_i$  ( $i = 1, 2, \dots, N$ ). Then equation (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

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

where  $\mathbf{J}_1 = \mathbf{J}_1^*$ . We refer to the Jacobian matrix defined in equation (26) as *reflection Jacobian*. Using equation (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

$$F_{g_N}(\mathbf{x}_{N-1}, \mathbf{x}_N, \lambda_N) = \mathbf{0}, \quad (28)$$

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

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

there exists a function  $f_N : \mathbf{x}_{N-1} \rightarrow (\mathbf{x}_N, \lambda_N)$ . The function  $f_N$  can be decomposed into two components:  $f_N(\mathbf{x}) = (f_{N1}(\mathbf{x}), f_{N2}(\mathbf{x}))$  where  $f_{N1} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$  and  $f_{N2} : \mathbb{R}^3 \rightarrow \mathbb{R}$  map  $\mathbf{x}_{N-1}$  to  $\mathbf{x}_N$  and  $\lambda_N$  respectively. With the existence of  $f_N$ , the last reflection Jacobian  $\mathbf{J}_N^*$  has been derived in equation (18). As a segment of an  $N$ -bounce path, the  $i$ th 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:

$$F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) = \mathbf{0}, \quad (29)$$

which differs from equation (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  $f_i$ s mapping  $\mathbf{x}_{i-1}$  to  $(\mathbf{x}_i, \lambda_i)$  ( $i = N, \dots, 1$ ) can be implicitly determined from the Implicit Function Theorem in sequence. Thus when we reach the  $i$ th bounce, the function  $f_{i+1} : \mathbb{R}^3 \rightarrow \mathbb{R}^4$  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  $f_{(i+1)1} : \mathbf{x}_i \rightarrow \mathbf{x}_{i+1}$  and  $f_{(i+1)2} : \mathbf{x}_i \rightarrow \lambda_{i+1}$ , we can express  $\mathbf{x}_{i+1}$  in equation (29) in terms of  $f_{(i+1)1}$ , obtaining

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

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

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

where

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

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

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

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

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

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

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

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

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

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

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

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

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

Finally, taking the submatrix of the left hand side of equation (36), we have derived a recurrence relation for the  $i$ th reflection Jacobian  $\mathbf{J}_i^*$ :

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

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

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

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

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  $\mathbf{J}_i$  ( $i = 1, 2, \dots, N$ ) from the  $\mathbf{J}_i^*$ s with equation (25), and then update the corresponding reflection point  $\mathbf{x}_i$  to first order accuracy using equation (19). However, motivated by the observation that the term ultimately required for linear perturbation in equation (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  $\Delta \mathbf{x}_i = \mathbf{J}_i \Delta \mathbf{p}$  ( $i = 1, 2, \dots, N$ ), 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 equation (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  $i = 1, 2, \dots, N$ . Consequently, equation (39) can be expressed as

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

where  $i = 1, 2, \dots, N$ ,  $\Delta \mathbf{x}_i = \mathbf{x}'_i - \mathbf{x}_i$  and  $\Delta \mathbf{x}_0 = \Delta \mathbf{p}$ . We can interpret equation (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  $\mathbf{p}$  until the ending point  $\mathbf{q}$ .

It can be easily verified that the direct approach and the recursive approach yield the same answer for path Jacobians in a multiple bounce path [10]. However, compared to the direct method, the recursive method is more efficient and easier to implement, as it involves only  $4 \times 4$  matrices.

## 5 The Path Hessian

As a linear approximation of the path function  $\Psi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ , the path Jacobian matrix  $\mathbf{J}$  maps from tangent space to tangent space, which is equivalent to locally fitting the function  $\Psi$  with a linear form. To obtain better accuracy, we may compute an additional term in equation (4), the path Hessian  $\mathbf{H}$ , and fit the function  $\Psi$  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  $3 \times 3$  matrices  $\mathbf{H}^1$ ,  $\mathbf{H}^2$  and  $\mathbf{H}^3$ , as shown in equation (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

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

when  $\mathbf{p}$  is moved to a nearby point  $\mathbf{p} + \Delta\mathbf{p}$ . Let  $f : \mathbf{p} \rightarrow (\mathbf{x}, \lambda)$  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

$$\Psi_i = f_i \quad (42)$$

for  $i = 1, 2, 3$ . According to equation (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 equation (18) yields:

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

where  $T$ ,  $\Gamma$  and  $D$  are given by

$$\begin{aligned} T(\mathbf{p}, \mathbf{x}, \lambda) &= \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}} \right]_{4 \times 3} \\ \Gamma(\mathbf{p}, \mathbf{x}, \lambda) &= \left[ \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)} \right]_{4 \times 4} \\ D(\mathbf{p}) &= \left[ \frac{\partial f(\mathbf{p})}{\partial \mathbf{p}} \right]_{4 \times 3}. \end{aligned}$$

To compute the derivatives of matrices  $T$ ,  $\Gamma$  and  $D$  in equation (43), it is convenient to write them in Cartesian tensor form. Thus, equation (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

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

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

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

where  $\nabla$  denotes the gradient operator, which in this context is applied to matrices [25, pp.62]. That is, the gradients of matrices  $T$ ,  $\Gamma$  and  $D$  are defined by

$$\begin{aligned} (\nabla T)_{mij} &= T_{ij,m} = \frac{\partial T_{ij}(\mathbf{p}, f(\mathbf{p}))}{\partial p_m} \\ (\nabla \Gamma)_{mik} &= \Gamma_{ik,m} = \frac{\partial \Gamma_{ik}(\mathbf{p}, f(\mathbf{p}))}{\partial p_m} \\ (\nabla D)_{mkj} &= D_{kj,m} = \frac{\partial D_{kj}(\mathbf{p})}{\partial p_m}, \end{aligned} \quad (46)$$

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

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 variable. 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 equation (46) denotes the partial derivative of  $T_{ij}$  with respect to  $p_m$  (the  $m$ th coordinate of  $\mathbf{p}$ ). Each layer of  $\nabla T$  constitutes a matrix  $\{t_{ij}\}$  with  $t_{ij} = (\nabla T)_{mij}$  for a fixed  $m$ , which we designate by  $\nabla_m T$ ; Nevertheless,  $\nabla T$  itself can be thought of as a generalized matrix<sup>2</sup>  $\{s_{ij}\}$  with  $s_{ij} = \nabla(T_{ij})$ . That is, each entry is a gradient vector of its corresponding entry in  $T$ .

In terms of the  $m$ th layers of  $\nabla T$ ,  $\nabla \Gamma$  and  $\nabla D$ , the summation on the right hand side of equation (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  $\nabla D$ , we obtain

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

---

<sup>2</sup>Strictly speaking, it is not a matrix since its elements are gradient vectors rather than scalars.

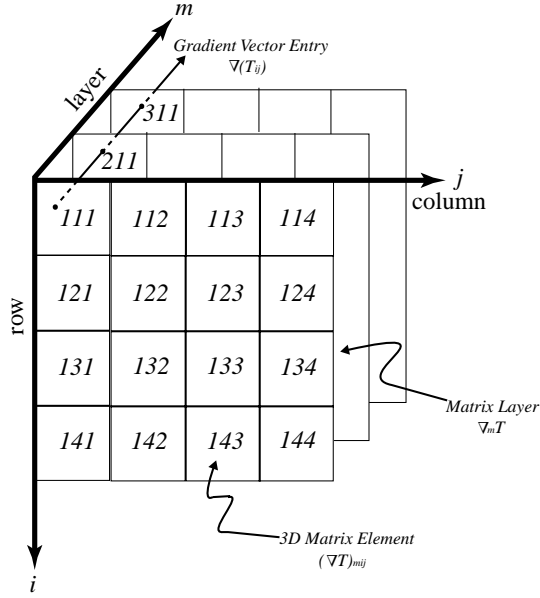


Figure 4: The gradient of a matrix  $T$ ,  $\nabla T$ , is a third-order array with “layer”, “row” and “column”.

for  $m = 1, 2, 3$ .

As the Fermat Jacobian 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  $m$ th layer of  $\nabla D$  is obtained by differentiating each component of  $D$  with respect to the  $m$ th 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 equation (42) that the gradient of the path Jacobian  $\mathbf{J}$  can be found from  $\nabla D$  by dropping the last row for each layer:

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

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

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

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

$$\mathbf{H}_{ijm} = f_{i,jm}, \quad (50)$$

where  $i, j, m = 1, 2, 3$ . Notice that we combine three Hessian matrices into a third order Hessian tensor. Combining equations (48), (49) and (50), we obtain the path Hessian  $\mathbf{H}$  by reorganizing  $\nabla D$ :

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

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

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

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

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

which follows from the chain rule. In equations (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  $4 \times 3$  Jacobian matrix; thus, both right hand sides consists of a sum of two row vectors.

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

$$\begin{aligned} T_{ij}(\mathbf{p}, \mathbf{x}, \lambda) &= \left( \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial \mathbf{p}} \right)_{ij} = \frac{\partial F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j} \\ \Gamma_{ik}(\mathbf{p}, \mathbf{x}, \lambda) &= \left( \frac{\partial F(\mathbf{p}, \mathbf{x}, \lambda)}{\partial (\mathbf{x}, \lambda)} \right)_{ik} = \frac{\partial F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k}, \end{aligned}$$

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

$$\begin{aligned}\nabla(T_{ij}) &= \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j \partial \mathbf{p}} + \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial p_j \partial (\mathbf{x}, \lambda)} \cdot D \\ \nabla(\Gamma_{ik}) &= \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k \partial \mathbf{p}} + \frac{\partial^2 F_i(\mathbf{p}, \mathbf{x}, \lambda)}{\partial x_k \partial (\mathbf{x}, \lambda)} \cdot D\end{aligned}\tag{54}$$

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 equation (40) can be applied from the starting point  $\mathbf{p}$  to the ending point  $\mathbf{q}$  as:

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

where  $i = 1, 2, 3, \dots, N$ ,  $\Delta \mathbf{x}_{i-1} = \mathbf{x}'_{i-1} - \mathbf{x}_{i-1}$ , and  $\Delta \mathbf{x}_0 = \Delta \mathbf{p}$ . By viewing equation (55) as a second-order Taylor expansion of the function  $f_{i1} : \mathbf{x}_{i-1} \rightarrow \mathbf{x}_i$  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 equation (55) must be expanded, as we did in equation (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 [11]. We show how the formulae (12), (40) and (41), (55) that we have 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 pre-computed reflection paths, the perturbation formulae 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 equations (12) and (40) or to second-order accuracy using equations (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 [24]. The details of our approach, based on path perturbation, are given by Chen [10] and Chen and Arvo [11].

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

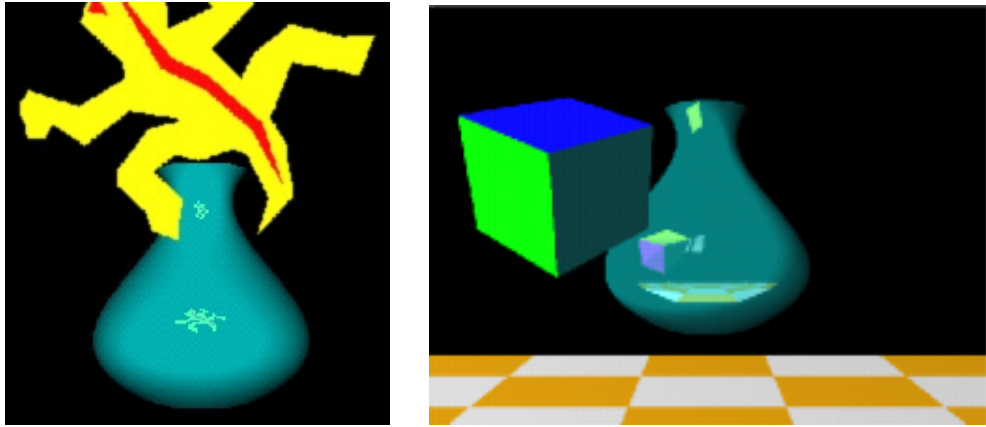


Figure 5: *One-bounce reflection images generated by the perturbation method for a polygon (left) and a solid object (right). The visibility in the right image is correctly handled by z-buffering. The results are nearly identical to the ray traced image, yet the perturbed images can be computed very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

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 the recursive formula (37) for reflection Jacobians and/or (61) for reflection Hessians, 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

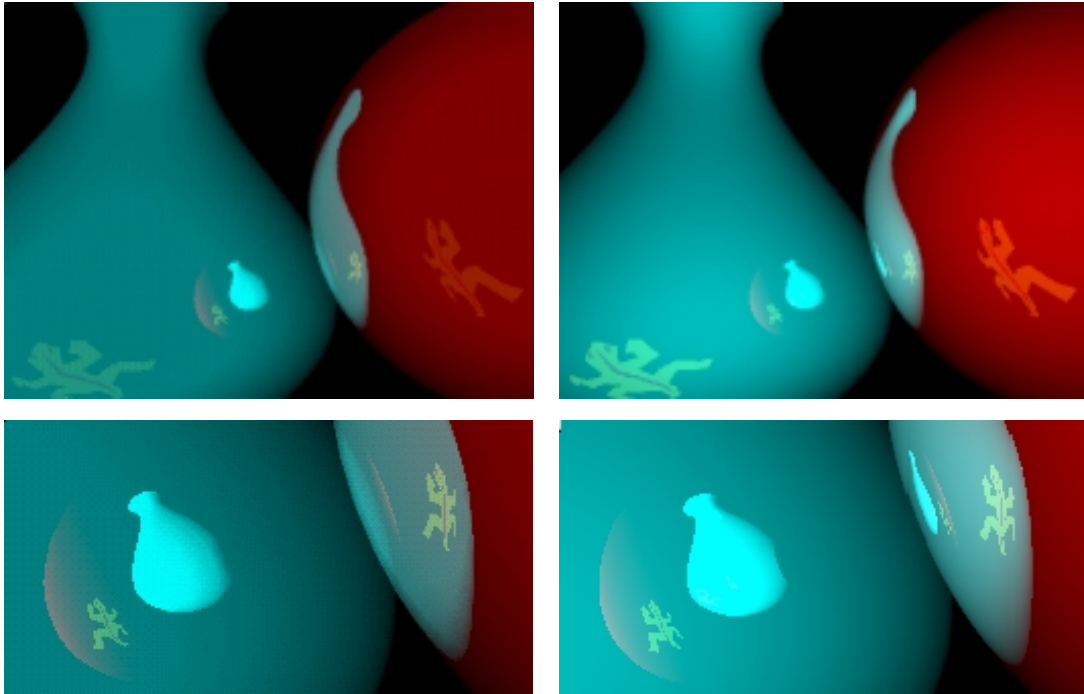


Figure 6: *Side-by-side comparison of multiple-bounce reflection images generated by the perturbation method (left) and ray tracing (right). Perturbed images in the left compute the reflections up to two levels, ray traced images in the right are generated with maximum depth of 5. The closeup view shows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

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  $640 \times 480$ , 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,  $80 \times 60$  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 have developed for path Jacobians and path Hessians still applies to the case of moving viewpoints and reflectors, so long as we resample the pre-cached 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 non-reflecting 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 of 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 improvement that can be achieved by path perturbation.

A natural extension to this technique is to also accommodate 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 done in the pencil tracing approach proposed by Shinya et. al [27]. 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 [10]. 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 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 [19]. Finally, the benefit of analytical path perturbation over random perturbation may provide a new low-variance mutation strategy in the context of metropolis light transport [28].

## 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. This work was supported in part by US National Science Foundation Career Award (CCR9876332), the Army Research Office Young Investigator Program (DAAH04-96-100077), and the Alfred P. Sloan Foundation.

## 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 equation (18), we can actually compute the path Jacobian  $\mathbf{J}$  directly without introducing the operator  $\mathbf{sub}$ . Let the vector  $\mathbf{h} = \partial g(\mathbf{x})/\partial \mathbf{x}$  and observe that

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

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

where  $M$  is a  $3 \times 3$  symmetric matrix.

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

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

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

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

where  $A$  is symmetric and satisfies the following equation

$$\begin{bmatrix} M_{3 \times 3} & \mathbf{h}_{3 \times 1} \\ \mathbf{h}_{1 \times 3}^T & 0 \end{bmatrix} \begin{bmatrix} A_{3 \times 3} & \mathbf{v}_{3 \times 1} \\ \mathbf{v}_{1 \times 3}^T & d \end{bmatrix} = \begin{bmatrix} MA + \mathbf{h}\mathbf{v}^T & M\mathbf{v} + d\mathbf{h} \\ \mathbf{h}^T A & \mathbf{h}^T \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & 0 \\ 0 & 1 \end{bmatrix}, \quad (57)$$

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

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

By equation (18), the path Jacobian  $\mathbf{J}$  we are interested in is composed of the first three row of  $\begin{bmatrix} A_{3 \times 3} & \mathbf{v}_{3 \times 1} \\ \mathbf{v}_{1 \times 3}^T & d \end{bmatrix} \begin{bmatrix} B_{3 \times 3} \\ \mathbf{0}_{1 \times 3} \end{bmatrix}$ . Thus, we may directly evaluate it as the product of two  $3 \times 3$  matrices,

$$\mathbf{J} = -A \times B. \quad (59)$$

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

## Appendix B: The Hessians for $N$ -Bounce Paths

In accordance with equation (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 equation (55), which involves third order tensors  $\mathbf{H}_i^*$ s as quadratic extensions of reflection Jacobians  $\mathbf{J}_i^*$ s. Expressing the position of  $\mathbf{x}_i$  as a function  $f_{i1}$  of its previous point  $\mathbf{x}_{i-1}$ , which has been shown to exist in section 4.2,  $\mathbf{H}_i^*$  at the  $i$ th 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 Jacobians  $\mathbf{J}_i^*$ s in an  $N$ -bounce path can be extended to the second order, yielding a corresponding recurrence relation for reflection Hessians  $\mathbf{H}_i^*$ 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:

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

where  $T_i$ ,  $\Gamma_i$  and  $D_i$  are defined as

$$\begin{aligned} T_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) &= \left[ \frac{\partial F_{g_i}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i)}{\partial \mathbf{x}_{i-1}} \right]_{4 \times 3} \\ \Gamma_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}, \lambda_i) &= \left[ A_i + \mathbf{aug}(B_i \cdot \mathbf{J}_{i+1}^*) \right]_{4 \times 4} \\ D_i(\mathbf{x}_{i-1}) &= \left[ \frac{\partial f_i(\mathbf{x}_{i-1})}{\partial \mathbf{x}_{i-1}} \right]_{4 \times 3}. \end{aligned}$$

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

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

for  $m = 1, 2, 3$  and  $i = 1, \dots, N$  in an  $N$ -bounce path. Note that the gradient operator  $\nabla$  is 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 equation (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,  $\nabla D_i$ .

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  $T_i, \Gamma_i$  can be considered as functions of  $\mathbf{x}_{i-1}$ . That is,

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

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

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

$$\begin{aligned} \nabla((T_i)_{jk}) &= \frac{\partial(T_i)_{jk}(\mathbf{x}_{i-1}, f_{i1}(\mathbf{x}_{i-1}), f_{(i+1)1}(f_{i1}(\mathbf{x}_{i-1})), f_{i2}(\mathbf{x}_{i-1}))}{\partial \mathbf{x}_{i-1}} \\ &= \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \frac{\partial f_{(i+1)1}}{\partial \mathbf{x}_i} \cdot \frac{\partial f_{i1}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(T_i)_{jk}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot \frac{\partial f_i}{\partial \mathbf{x}_{i-1}} \\ &= \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial(T_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^* \cdot \mathbf{J}_i^* + \frac{\partial(T_i)_{jk}}{\partial(\mathbf{x}_i, \lambda_i)} \cdot D_i. \end{aligned} \quad (64)$$

Thus, each element of  $\nabla T_i$  can be obtained from the component of the gradient vector on the left of equation (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:

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

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:

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

Since the gradient operator  $\nabla$  above is with respect to  $\mathbf{x}_{i-1}$ , another symbol  $\nabla C_i$  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,

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

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

In equation (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 equation (68).  $\nabla \mathbf{J}_{i+1}^*$  and  $\nabla C_i$  are related by the identity (68) derived from the chain rule. Writing equation (66) in terms of the  $m$ th layers of  $\nabla \Gamma_i$ ,  $\nabla A_i$ ,  $\nabla B_i$  and  $\nabla C_i$ , we obtain a matrix form

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

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

$$\begin{aligned} \nabla ((A_i)_{jr}) &= \frac{\partial (A_i)_{jr}}{\partial \mathbf{x}_{i-1}} + \frac{\partial (A_i)_{jr}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^* \cdot \mathbf{J}_i^* + \frac{\partial (A_i)_{jr}}{\partial (\mathbf{x}_i, \lambda_i)} \cdot D_i \\ \nabla ((B_i)_{jk}) &= \frac{\partial (B_i)_{jk}}{\partial \mathbf{x}_{i-1}} + \frac{\partial (B_i)_{jk}}{\partial \mathbf{x}_{i+1}} \cdot \mathbf{J}_{i+1}^* \cdot \mathbf{J}_i^* + \frac{\partial (B_i)_{jk}}{\partial (\mathbf{x}_i, \lambda_i)} \cdot D_i \\ \nabla ((C_i)_{kl}) &= \nabla ((\mathbf{J}_{i+1}^*)_{kl}) \cdot \mathbf{J}_i^* \end{aligned} \quad (70)$$

where  $j, r = 1, 2, 3, 4$  and  $k, l = 1, 2, 3$ . Combining equation (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 equations (64) and (70) can be evaluated from the second partial derivatives of the Fermat equation  $F_{g_i}$ , just as we did in equation (54).

Finally, the reflection Hessian  $\mathbf{H}_i^*$  is obtained by reorganizing  $\nabla D_i$  using equation (51). As seen from equation (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  $\mathbf{p}$ .

## References

- [1] ADELSON, S. J., AND F.HODGES, L. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications* 15 (1995), 43–52.
- [2] ADELSON, S. J., AND HODGES, L. Stereoscopic ray tracing. *The Visual Computer* 10, 3 (1993), 127–144.
- [3] APOSTOL, T. M. *Calculus II: Multi-Variable Calculus and Linear Algebra, with Applications to Differential Equations and Probability*. John Wiley & Sons, New York, 1969.
- [4] ARVO, J., AND KIRK, D. A survey of ray tracing acceleration techniques. In *An Introduction to Ray Tracing*, A. S. Glassner, Ed. Academic Press, New York, 1989, ch. 6.
- [5] AZIZ, A., AND NA, T. Y. *Perturbation Methods in Heat Transfer*. Hemisphere Publishing Corporation, New York, 1984.
- [6] BORN, M., AND WOLF, E. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, third ed. Pergamon Press, New York, 1965.
- [7] BRIERE, N., AND POULIN, P. Hierarchical view-dependent structures for interactive scene manipulation. In *Computer Graphics Proceedings* (Aug. 1996), Annual Conference Series, ACM SIGGRAPH, pp. 83–90.
- [8] CHAPMAN, J., CALVERT, T. W., AND DILL, J. Exploiting temporal coherence in ray tracing. In *Proceedings of Graphics Interface '90* (May 1990), pp. 196–204.
- [9] CHAPMAN, J., CALVERT, T. W., AND DILL, J. Spatio-temporal coherence in ray tracing. In *Proceedings of Graphics Interface '91* (June 1991), pp. 101–108.
- [10] CHEN, M. Perturbation methods for image synthesis. Master's thesis, California Institute of Technology, May 1999.
- [11] CHEN, M., AND ARVO, J. Perturbation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics* 6, 3 (July–September 2000), 253–264.
- [12] COLE, J. D. *Perturbation Methods in Applied Mathematics*. Blaisdell, Waltham, Mass., 1968.

- [13] COOK, R. L. Shade trees. *Computer Graphics* 18, 3 (July 1984), 137–145.
- [14] DYKE, M. V. *Perturbation Methods in Fluid Mechanics*. Academic Press, New York, 1964.
- [15] IGEHY, H. Tracing ray differentials. In *Computer Graphics Proceedings* (Aug. 1999), Annual Conference Series, ACM SIGGRAPH, pp. 179–186.
- [16] JEVANS, D. A. Object space temporal coherence for ray tracing. In *Proceedings of Graphics Interface '92* (May 1992), pp. 176–183.
- [17] JR., S. B. Two algorithms taking advantage of temporal coherence in ray tracing. *The Visual Computer* 4, 3 (1988), 123–132.
- [18] LIN, C. C., AND SEGEL, L. A. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. Society for Industrial and Applied Mathematics, Philadelphia, 1988.
- [19] LISCHINSKI, D., AND RAPPOPORT, A. Image-based rendering for non-diffuse synthetic scenes. In *Proceedings of the Ninth Eurographics Workshop on Rendering* (Vienna, Austria, June 1998).
- [20] MARSDEN, J. E., AND HOFFMAN, M. J. *Elementary Classical Analysis*. W. H. Freeman, New York, 1993.
- [21] MITCHELL, D., AND HANRAHAN, P. Illumination from curved reflectors. *Computer Graphics* 26, 2 (July 1992), 283–291.
- [22] MURAKAMI, K., AND HIROTA, K. Incremental ray tracing. In *Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, Conference Proceedings* (Rennes, France, June 1990), pp. 15–29.
- [23] NEUMANN, P. M. Reflections on reflection in a spherical mirror. *The American Mathematical Monthly* 105, 6 (June–July 1998), 523–528.
- [24] OFEK, E., AND RAPPOPORT, A. Interactive reflections on curved objects. In *Computer Graphics Proceedings* (July 1998), Annual Conference Series, ACM SIGGRAPH, pp. 333–342.
- [25] SEGEL, L. A., AND HANDELMAN, G. H. *Mathematics Applied to Continuum Mechanics*. Macmillan Publishing Company, New York, 1977.
- [26] SÉQUIN, C. H., AND SMYRL, E. K. Parameterized ray tracing. *Computer Graphics* 23, 3 (July 1989), 307–314.

- [27] SHINYA, M., TAKAHASHI, T., AND NAITO, S. Principles and applications of pencil tracing. *Computer Graphics* 21, 4 (July 1987), 45–54.
- [28] VEACH, E., AND GUIBAS, L. J. Metropolis light transport. In *Computer Graphics Proceedings* (Aug. 1997), Annual Conference Series, ACM SIGGRAPH, pp. 65–76.