B-Splines for Physically-Based Rendering

Michael M. Stark
William Martin
Elaine Cohen
Tom Lyche
Richard F. Riesenfeld

UUCS-02-005

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

January 9, 2002

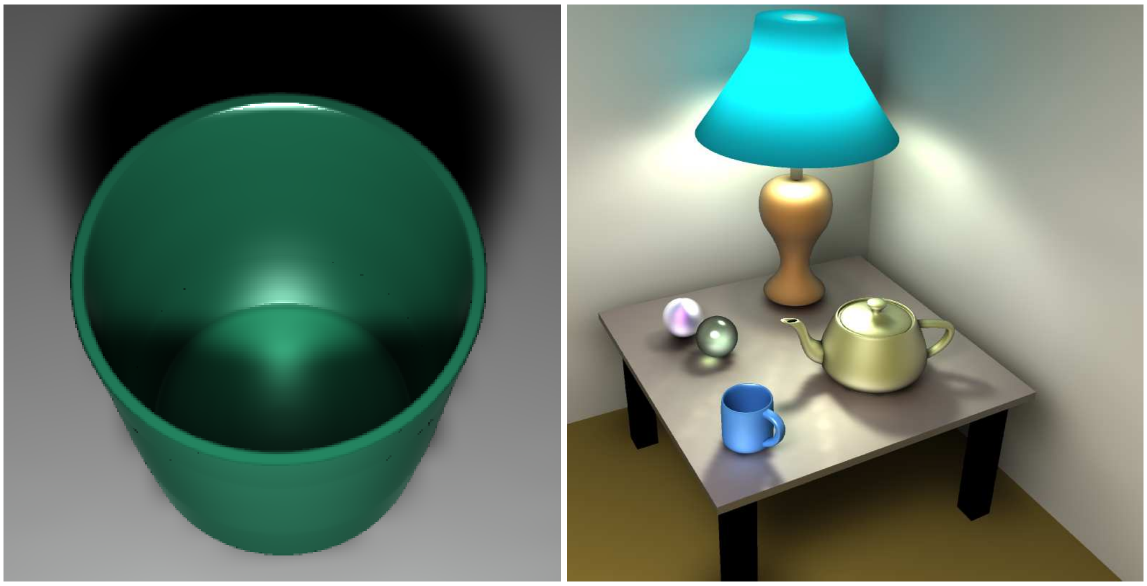
Abstract

Although B-spline curves and surfaces have enjoyed a long established place in the graphics community as constructive modeling tools, the use of B-spline approximation techniques has received relatively little attention in rendering. In this work we explore the use of 4D and 5D tensor product B-spline functions to represent surface radiance, and establish that, when appropriately applied, they can be used effectively for static scenes with diffuse to moderately specular elements. Once computed, the surface radiance representation is view independent, can be evaluated quickly, and is equally suited for incorporation into ray tracing or scan-line rendering algorithms. Furthermore, we use B-spline approximation techniques to solve the problem of global illumination for general parametric surfaces with a wide range of reflectance and transmission properties. We conclude that addressing functional approximation aspects offers a fertile research ground relative to the already impressive gains that splines have made in other fields.

B-Splines for Physically-Based Rendering

Michael M. Stark William Martin Elaine Cohen Tom Lyche Richard F. Riesenfeld

School of Computing
University of Utah



Abstract

Although B-spline curves and surfaces have enjoyed a long established place in the graphics community as constructive modeling tools, the use of B-spline approximation techniques has received relatively little attention in rendering. In this work we explore the use of 4D and 5D tensor product B-spline functions to represent surface radiance, and establish that, when appropriately applied, they can be used effectively for static scenes with diffuse to moderately specular elements. Once computed, the surface radiance representation is view independent, can be evaluated quickly, and is equally suited for incorporation into ray tracing or scan-line rendering algorithms. Furthermore, we use B-spline approximation techniques to solve the problem of global illumination for general parametric surfaces with a wide range of reflectance and transmission properties. We conclude that addressing functional approximation aspects offers a fertile research ground relative to the already impressive gains that splines have made in other fields.

CR Categories: I.3.3 [Computing Methodologies]: Computer Graphics—Picture/Image Generation; I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

Keywords: Curves & Surfaces, Illumination, Numerical Analysis, Ray Tracing, Rendering

1 Introduction

A fundamental quantity in radiometry and therefore in physically-based rendering, radiance can intuitively be thought of as radiant power carried along a line. Spectral radiance depends on both spatial position and direction as well as wavelength. More formally, in an arbitrary environment either synthetic or real, radiance is given by the *plenoptic function* of spatial position, direction, time, and wavelength and therefore represents the appearance of an environment from all possible vantage points for all time. To have access to the plenoptic function, therefore, is to make rendering a simple matter. The problem of representing and storing a good approximation to the plenoptic function for subsequent evaluation is the problem of image-based rendering. Computing the plenoptic function itself is more closely related to the problem of global illumination.

Computer graphics environments typically consist of a collection of surfaces with specified reflection and transmission properties. Any emitting surface in the environment has an intrinsic radiance emission and directly illuminates the rest of the environment. A surface can also self-illuminate if it is not convex. All surfaces can contribute indirect illumination by reflected radiance. Global illumination is the problem of computing final surface radiance according to light transport theory. If the region between surfaces in graphics environments does not attenuate or scatter light, then the radiance leaving a surface remains constant along a line. With this simplification representing the plenoptic function reduces to representing all surface radiance functions.

One of our goals here is to illustrate the advantage of good approximation methods by using tensor product B-splines to represent surface radiance functions directly. While it is well understood that this method can work for a wide variety of situations when enough knots are introduced in each dimension, the approach must be carefully managed to avoid an explosion in the storage requirements. Therefore, in what follows, it is imperative to find good representations of surface radiance by exploiting the intrinsically powerful properties of B-spline approximation. In pursuing these goals we favor visual appearance above numerical accuracy.

Using a B-spline representation for radiance in a scene modeled with B-spline surface geometry yields a unification leading to a more tractable global illumination problem.

1.1 Previous Work

Central to this paper is the concept of representing radiance to capture surface appearance. This is a core part of realistic image synthesis, and area where there has been a long and impressive history of research in this area. Determining surface radiance entails solving the global illumination problem.

Computing global illumination (GI) involves the solution of a system of integral equations in the surface radiance functions. One of the earliest methods for GI is the radiosity method [Cohen and Wallace 1993; Sillion and Puech 1994] in which constant radiance is assumed on a finite collection of surface patches. The basic radiosity method has been improved in a variety of ways and it has been extended to more general types of light transport (*e.g.*, [Rushmeier and Torrance 1990; Christensen et al. 1997]).

Many of numerous global illumination methods that have been developed in recent years are currently being used in practice. Statistical ray tracing methods such as Monte Carlo integration, path tracing[Shirley 2000], and Metropolis sampling [Veach and Guibas 1997] solve the GI problem by simulating the path light takes through the environment. More recently, the photon map technique has been successfully applied both for radiance representation and global illumination [Jensen 2001].

Much investigation has into representing the plenoptic function, general radiance, and the *light field* has been done in recent years, emphasizing both efficiency of representation and interactive evaluation. [Peter and Straßer 2001; Gortler et al. 1996; Levoy and Hanrahan 1996; Peter and Straßer 2001; Christensen et al. 1994]

1.2 Overview

In this research we use tensor product B-splines to represent surface radiance. Section 2 contains mathematical background, including a review of tensor product B-spline functions and their approximation properties. Using something akin to an image-based approach, Section 3 develops the approximation method for existing surface radiance. Section 4 then exploits the B-spline representation to solve the radiance integrals and develops a global illumination algorithm in the spirit of the gathering approach used in radiosity. Section 5 discusses implementation issues, including how the tensor product B-splines are incorporated into the renderer, and performance observations. In Section 6, we discuss future directions indicated by this research.

2 Mathematical Problem Formulation

In this section we present a mathematical formulation of the problem of representing and computing surface radiance. Additionally we recall some relevant properties of B-splines.

2.1 The Environment

We assume the environment $\mathcal{E}$ consists of a collection of surfaces $\mathcal{E} = \{S_1, \ldots, S_p\}$ in 3-space, each S_i having a regular parametric representation $s_i(u, v)$, where the parameter domain is a connected subset of $[0, 1] \times [0, 1]$. At each surface point $s(u, v)$ of some $S \in \mathcal{E}$

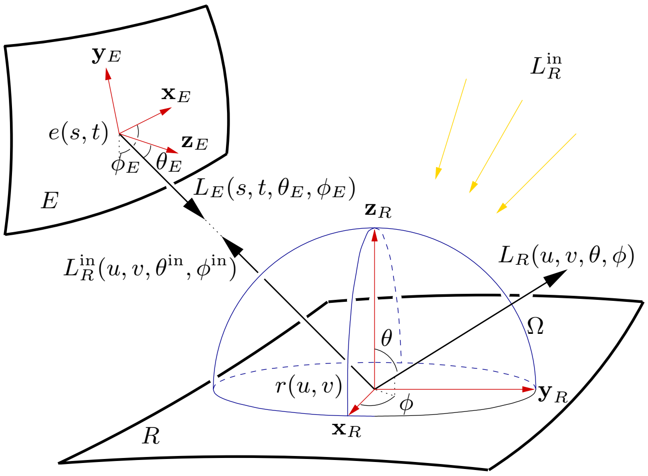


Figure 1: *Geometry for surface radiance and light transport. The outgoing surface radiance along a ray L_R is a function of both surface position as well as the angle the ray makes with the local coordinate axes. L_R can be computed by integrating incoming radiance over the hemisphere Ω above the surface, or from a surface integral over all emitting surfaces E.*

there is a local coordinate system, or local frame, given by

$$\mathbf{x}_S = \frac{\mathbf{s}_u}{\|\mathbf{s}_u\|} \tag{1}$$

$$\mathbf{z}_S = \frac{\mathbf{s}_u \times \mathbf{s}_v}{\|\mathbf{s}_u \times \mathbf{s}_v\|} \tag{2}$$

$$\mathbf{y}_S = \mathbf{z}_S \times \mathbf{x}_S, \tag{3}$$

where $\| \ \|$ denotes Euclidean distance. Note that the local coordinate axes are functions of the parameters u and v and that the vector $\mathbf{z}_S$ is the unit surface normal at parameter values (u, v). The local frame coordinates of a vector $\mathbf{d}$ at the surface point $s(u, v)$ are given by $(\mathbf{d}\cdot\mathbf{x}_S, \mathbf{d}\cdot\mathbf{y}_S, \mathbf{d}\cdot\mathbf{z}_S)$. The local frame also induces a coordinate system in (θ, ϕ) for the local unit hemisphere above the surface. These spherical coordinates are used to specify a direction on the surface. If $\mathbf{r}$ is a point in space different from the surface point $s(u, v)$, then the spherical coordinates of the direction $\mathbf{d} = \mathbf{r} - s(u, v)$ are given by

$$\theta = \arccos \frac{\mathbf{d}\cdot\mathbf{z}_S}{\|\mathbf{d}\|} \tag{4}$$

$$\phi = \arctan(\mathbf{d}\cdot\mathbf{y}_S, \mathbf{d}\cdot\mathbf{x}_S) \tag{5}$$

where the function $\arctan(a, b)$ denotes the two dimensional arctan function. Note the spherical coordinates are dependent on the local frame, so the same vector may have different spherical coordinates at different positions on the surface.

2.1.1 Radiance

A fundamental radiometric quantity, *radiance*, measures radiant power per unit projected area per unit solid angle (in units of W/m^{2}sr.) Radiance depends on spatial position and direction, and, intuitively, can be thought of the radiant power carried along a line.

Following [Arvo 1995], the radiance at the point $x \in \mathbb{R}^3$ in the direction ω is often denoted by $L(x, \omega)$. If nothing blocks the line segment between two points x and x', then the radiance leaving x in the direction of $x' - x$ is the radiance coming to x' in the opposite direction. That is, radiance is constant along a line provided there is no participating (attenuating) medium.

General graphics environments consist of a collection of surfaces perhaps with some *participating media*, such as smoke or fog, that attenuate radiance in the space between surfaces. Each point on every surface may have an associated radiance exitance, although computing this overall involves energy transfer between surfaces and is the problem of global illumination. The radiance exiting from each surface is used for the final rendering of a scene. A ray tracer, for example, renders by tracing ray from an vantage point, or through a virtual camera system, and then reports the radiance at each ray-surface intersection. Similarly, a scan-line renderer might decompose the surface into micro-facet polygons and use the radiance value in the direction of the eye point for a color.

In general, each surface S in a scene $\mathcal{E}$ has an outgoing spectral radiance $L_S(u,v,\theta,\phi,\lambda)$. This surface radiance is a function both of surface position (u,v), direction (θ,ϕ) in the local frame and wavelength λ. This value gives the radiant spectrum at render time, when the surface point at (u,v) is viewed from direction (θ,ϕ). Our goal is to approximate L_S with a tensor product B-spline.

2.1.2 Radiance Integrals

In many cases radiance in a scene $\mathcal{E}$ can be computed by a method based on the global or hemispherical integral formulation. Each surface S in the environment has an intrinsic emissive radiance function $L_S^{\mathrm{emit}}(u,v,\theta,\phi,\lambda)$, and a Bi-directional Reflectance Distribution Function (BRDF) $\rho_S(u,v,\theta,\phi,\theta^{\mathrm{in}},\phi^{\mathrm{in}},\lambda)$. Except for a few light sources in the scene the emission is zero in most cases. The light sources illuminate the other surfaces of the scene by direct energy transfer. Subsequently surfaces become indirectly illuminated by reflection from other surfaces not initially light sources. The total amount of outgoing radiance L_S is determined by integrating the incoming radiance L_S^{in} against the BRDF ρ over the hemisphere and add the result to the emitted radiance,

$$
\begin{aligned}
L_S(u,v,\theta,\phi,\lambda) &= L_S^{\mathrm{emit}}(u,v,\theta,\phi,\lambda) \\
&\int_\Omega L_S^{\mathrm{in}}(u,v,\theta,\phi,\lambda)\rho(u,v,\theta,\phi,\theta^{\mathrm{in}},\phi^{\mathrm{in}},\lambda)\cos\theta^{\mathrm{in}}\,d\omega
\end{aligned} \tag{6}
$$

where $d\omega = \sin\theta^{\mathrm{in}} d\theta^{\mathrm{in}} d\phi^{\mathrm{in}}$. Note that the outgoing spectral radiance $L_S(u,v,\theta,\phi,\lambda)$ is a function of surface position (u,v), direction (θ,ϕ) in the local frame and wavelength λ. This value yields the radiant spectrum at render time, when the surface point at (u,v) is viewed from direction (θ,ϕ). In general S can be illuminated by radiance from all the other surfaces in the scene $\mathcal{E}$ including itself.

When we write down equation (6) for each $S \in \mathcal{E}$ we obtain a system of coupled integral equations which defines the *global* or *hemispherical* integral formulation of L_S. To solve this system of integral equations is to solve the global illumination problem. We need to compute the final, or steady-state surface radiance after reflected light is sufficiently attenuated by reflection. In pursuit of this goal we approximate L_S with a tensor product B-spline.

To start we first consider the case where the scene consists of only two objects R and E, and reformulate the contribution to the receiving surface R from the emitting surface E. Suppose E is parameterized by $e(s,t)$, with derivatives and local frame coordinates as above, and that we have an outgoing emission function $L_E(s,t,\theta,\phi,\lambda)$. A change of variables can be applied to the integral of (6), to obtain the surface integral

$$
\begin{aligned}
&L_R(u,v,\theta,\phi,\lambda) = \\
&\quad + \int_E \mathrm{vis}(r,e)\frac{\cos\theta_R^{\mathrm{in}}\cos\theta_E}{d^2} L_E(s,t,\theta_E,\phi_E,\lambda) \\
&\qquad \rho(u,v,\theta,\phi,\theta_R^{\mathrm{in}},\phi_R^{\mathrm{in}},\lambda)\,dE,
\end{aligned} \tag{7}
$$

where $\mathrm{vis}(r,e) = 1$ if the point $e(s,t)$ is visible from $r(u,v)$ and 0 otherwise. The surface integral can be expressed directly in terms of the surface function e over

$$
\begin{aligned}
L_R \;=\; & L_R^{\mathrm{emit}} + \int_0^1\int_0^1 \chi_e \mathrm{vis}(r,e)\frac{(e-r)\cdot\mathbf{z}_R\,(r-e)\cdot\mathbf{z}_E}{\|r-e\|^4}\,\cdot \\
& L_E\rho\|\mathbf{r}_s\times\mathbf{r}_t\|\,ds dt
\end{aligned} \tag{8}
$$

where χ_e denotes the characteristic function of the parameter domain of $e(s,t)$: $\chi_e(s,t)$ is 1 if (s,t) is in the parameter domain of $e(s,t)$, and 0 otherwise.

2.2 A Review of B-Spline Approximation

B-spline curves and surfaces have been well established in the graphics community since they were introduced by Riesenfeld as modeling primitives [Riesenfeld 1973]. However, B-splines were studied by Schoenberg many years earlier as a technique for approximating functions. Outside of graphics they have found a solid place in numerical analysis as a powerful approximation technique, and have numerous applications to industrial problems.

Formally we define a spline-space $S_{d,\boldsymbol{\tau}}$ of splines of degree d with knots $\boldsymbol{\tau}$ as

$$
S_{d,\boldsymbol{\tau}} = \mathrm{span}\{B_0,\ldots,B_{n-1}\}, \tag{9}
$$

where $B_i(x) = B_{i;d,\boldsymbol{\tau}}(x)$ is the ith B-spline of degree d on the *knot vector* $\boldsymbol{\tau} = (\tau_0,\ldots,\tau_{n+d})$ of non-decreasing real values. Thus every spline can be written as a linear combination of B-splines. The B-spline B_i is a nonnegative piecewise polynomial, is zero outside the interval $[\tau_i,\tau_{i+d+1}]$ and $\sum_i B_i(x) = 1$ for all $x \in (\tau_d,\tau_n)$. Moreover B_i has $d-m$ continuous derivatives at a knot $x = \tau_j$ which occurs m-times among $\tau_i,\ldots,\tau_{i+d+1}$. If $f(x) = \sum_{i=0}^{n-1} a_i B_{i;d,\boldsymbol{\tau}}(x)$ is a spline of degree d and $x \in [\tau_I,\tau_{I+1}]$ then $f(x)$ is a sum of only $d+1$ terms

$$
f(x) = \sum_{i=I-d}^{I} a_i B_i(x) \tag{10}
$$

Because splines are piecewise polynomials they are easily differentiated and integrated and there are stable and efficient algorithms for computing with them. We will make use of the formula for integrating a spline of degree d on $\boldsymbol{\tau}$

$$
\int_{\tau_1}^{\tau_{n+d}} \sum_{i=0}^{n-1} a_i B_{;i,(}(x)\,dx = \sum_{i=0}^{n-1} a_i \frac{\tau_{i+d+1}-\tau_i}{d+1}. \tag{11}
$$

2.2.1 B-Spline Approximation

To approximate a function f entails two tasks. We need to choose a suitable spline space $S_{d,\boldsymbol{\tau}}$ and then compute the B-spline coefficients a_i of a spline $g \in S_{d,\boldsymbol{\tau}}$ so that $g = \sum_{i=0}^{n-1} a_i B_i$ is a good approximation to f. Existing methods for computing spline approximations can be divided into two classes, local and global methods. A method is local if the value of the approximation $g(x)$ at a point x depends only on values of f in a neighborhood of x and global otherwise. A global method generally requires solving a n-by-n linear system of equations for the unknown B-spline coefficients. Examples of global methods are cubic spline interpolation and least squares. In a local method the coefficients are given explicitly. For example in Schoenberg's variation diminishing spline approximation method the ith B-spline coefficient is given by $a_i = f(t_i^*)$, with the evaluation node t_i^* an average of contiguous knot values

$$
t_i^* = \frac{\tau_{i+1}+\cdots+\tau_{i+d}}{d}. \tag{12}
$$

The Schoenberg method belongs to a class of methods known as *quasi-interpolants* ([Lee et al. 2000]). We obtain an example of a quadratic quasi-interpolant by choosing the B-spline coefficient of the ith quadratic B-spline $B_{i,2}$ as

$$a_i = -\frac{1}{2}f(\tau_{i+1}) + 2f(t_i^*) - \frac{1}{2}f(\tau_{i+2}),$$

where $t_i^* = (\tau_{i+1}+\tau_{i+2})/2$. This method has approximation order $O(h^3)$, while the Schoenberg method is $O(h^2)$ for all degrees d. On the other hand the words "Variation Diminishing" alludes to the fact that the Schoenberg method has desirable shape preserving properties. For an $O(h^4)$ cubic quasi-interpolant see [Lee et al. 2000].

2.2.2 Tensor Product B-spline Functions

There are several natural ways of generalizing univariate B-spline functions to multivariate functions. Since speed is an issue in this the paper tensor product form of B-splines is a natural choice.

Tensor product spline functions are defined on a rectangular grid using a knot vector in each space dimension. In the two-dimensional case there are two knot vectors $\boldsymbol{\tau}_x$ and $\boldsymbol{\tau}_y$, two degrees d_x and d_y, and two dimensions m and n. The coefficients a_{ij} can be stored in a $m \times n$ matrix, and the tensor product spline function is written as the double summation

$$f(x,y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} a_{ij} B_{j;d_y,\boldsymbol{\tau}_y}(y) B_{i;d_x,\boldsymbol{\tau}_x}(x). \tag{13}$$

Notice the terms have one basis function in each variable, and all product pairs are represented. Also notice the degrees and knot vectors are independent in each dimension, so, for example, the tensor product of a quadratic having 100 knots with a cubic having only 4 is perfectly valid.

Higher-dimensional tensor product B-spline functions follow naturally. The coefficient matrix becomes a general tensor. A k-dimensional tensor product spline function is the k-fold sum

$$f(x_1,\ldots,x_k) = \sum_{i_1=0}^{n_1-1}\cdots\sum_{i_k=0}^{n_k-1} a_{i_1\cdots i_k} B_{i_k}(x_k)\cdots B_{i_1}(x_1). \tag{14}$$

where the coefficients form a k^{th}order tensor. Notice the knot vector and degree subscripts on the basis functions have been omitted for brevity. In general we will use an even simpler notation, where the explicit variable dependence and the summation limits are omitted. For a four-dimensional tensor product B-spline, a function of particular interest to this paper, we will write simply

$$\sum_{ijkl} a_{ijkl} B_l B_k B_j B_i. \tag{15}$$

2.2.3 Evaluation of Tensor Product B-Spline Functions

Evaluating a tensor product spline is fast because for a spline of degree d only $d+1$ B-splines are nonzero at any value. For example, a two-dimensional cubic spline has only four nonzero basis functions in each variable, and the summation in this case can be written as the matrix product

$$[B_{I-3}\cdots B_I]\begin{bmatrix} a_{I-3,J-3} & \cdots & a_{I-3,J} \\ \vdots & & \vdots \\ a_{I,J-3} & \cdots & a_{I,J}\end{bmatrix}\begin{bmatrix} B_{J-3} \\ \vdots \\ B_J\end{bmatrix} \tag{16}$$

where I and J are determined by the knot intervals as in (10). This matrix product is amenable to hardware vectorization.

Higher-order evaluations require a more involved formulation, but the idea is the same. It is worth noting that if fast computation

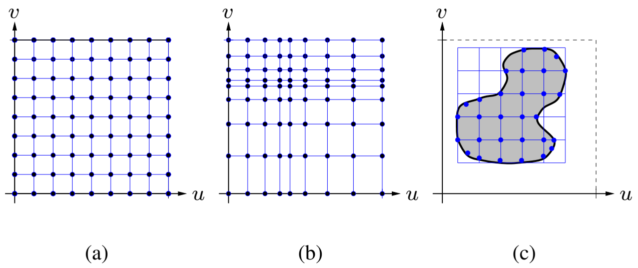


Figure 2: *Tensor product knots and knot lines. (a) Uniform knot spacing. (b) Nonuniform spacing. The knots can be chosen independently in each dimension, but not arbitrarily over the domain. (c) The domain is inherently rectangular. In the case of a non-rectangular domain, the knots are chosen around the domain as tightly as possible, and evaluation points can be moved inside the domain.*

of 1D and 2D B-splines is available (*e.g.*, in hardware) a simple re-association of the tensor product summations shows how 3D and 4D splines can be quickly evaluated using well known univariate spline algorithms:

$$\sum_{ijk} a_{ijk} B_k B_j B_i = \sum_i \left(\sum_{jk} a_{ijk} B_k B_j\right) B_i \tag{17}$$

$$\sum_{ijkl} a_{ijkl} B_l B_k B_j B_i = \sum_{ij} \left(\sum_{kl} a_{ijkl} B_l B_k\right) B_j B_I \tag{18}$$

and of course higher-dimensional extensions follow.

3 B-splines for Radiance

This section is concerned with the approximation of the spectral radiance on the surface assuming that the actual value of L_S is available at each point. As $L_S(u,v,\theta,\phi,\lambda)$ is a five-variate function, a five-dimensional tensor product B-spline is indicated. We need to construct an appropriate spline space and approximation method.

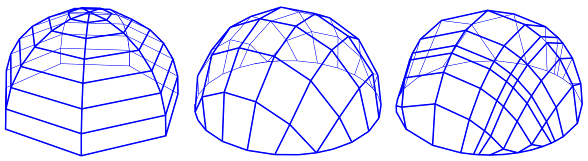
3.1 Lambertian Surfaces

To develop the B-spline approximation technique, we first assume that the surface is Lambertian, that is, the outgoing radiance is independent of the direction. If the wavelength dependence is also ignored, the surface exitant radiance reduces to a function $L(u,v)$ of only two variables. To construct an approximation $\tilde{L}(u,v)$ to L involves choosing the degrees and the knot vectors in u and v, and applying an approximation method (Schoenberg or quasi-interpolation).

For knot vectors we can for example use uniform knots with multiple knots at each end:

$$[\underbrace{a,\ldots,a}_{d+1}, a+h, a+2h, \ldots, b-h, \underbrace{b,\ldots,b}_{d+1}] \tag{19}$$

where d is the degree, $m \geq 0$ is the number of internal knots, $h = (b-a)/(m+1)$ and the domain of the parameter (u or v) is the closed interval $[a,b]$. Normally the domain $[a,b]$ is the unit interval $[0,1]$, but for practical reasons the radiance function might not be defined on the edge of a surface patch so we use instead



(a) (b) (c)

Figure 3: *Knots on the hemisphere, for the angular parameters. (a) The standard θ, ϕ spherical parameterization results in poor knot spacing. (b) Our trimmed square-to-sphere mapping is much more uniform. (c) Nonuniform knot spacing can be used with our mapping, to better approximate a highlight, for example.*

$[\varepsilon, 1-\varepsilon]$ for some small ε, this having the effect of pulling evaluation points away from the edges. Technically this makes parameter values less than ε or greater than $1-\varepsilon$ outside the domain of B-spline function, but in practice this causes little difficulty due to the continuous dependence of a spline as a function of its knots.

The knot vectors need not be uniform. If there is extra detail in on one part of the surface, for example, the knots can be clustered there. However, since we are using tensor products knot lines extend through the entire domain.

3.2 View-Dependent Radiance

The spherical coordinate parameterization for the hemisphere given by (θ, ϕ) above is a candidate for the spline parameters, as the domain is rectangular: $0 \le \theta < \pi/2$, $-\pi \le \phi < \pi$. However, there are two difficulties. First, we need periodicity in the ϕ-variable. This can be handled by a simple periodic extension of the corresponding knot vector so that the resulting spline function becomes periodic in ϕ.

The more serious difficulty, however, is that the standard spherical parameterization is highly nonuniform: knot values in ϕ get pinched together near the pole. See Figure 3.

To solve this problem we use a different set of local parameters for the spline approximation. For any point in the tangent plane at a surface point $s(u, v)$ on a surface S we let (α, β) be the coordinates of that point with respect to the coordinate system defined by $(\mathbf{x}_S, \mathbf{y}_S)$ given by (1),(3).

What is needed then is a smooth, low distortion mapping of the unit square to the unit hemisphere. Such mappings do exist, but we also need something fast to evaluate because the inverse mapping, must be computed at every radiance evaluation at render time. Instead we use a mapping from the unit disc to the unit hemisphere, and apply the trick of pushing evaluation nodes inside the disc that we use for trimming curves.

If $r = \sqrt{\alpha^2 + \beta^2}$, then the rectangular coordinates (in the local frame) of the corresponding point on the hemisphere are given by

$$z = 1 - r^2 \tag{20}$$

$$x = \frac{\sqrt{1-z^2}}{r}\alpha \tag{21}$$

$$y = \frac{\sqrt{1-z^2}}{r}\beta. \tag{22}$$

Here a point (α, β) in the unit disc is lifted to the paraboloid $z = 1 - r^2$ and then moved out horizontally to the hemisphere. Notice the mapping is undefined outside the unit disc. To avoid this, we "pull in" node values outside the disc: if $r > R$ then α and β are scaled by R/r. Thus every point in the plane outside the unit disc is moved radially to the disc of radius R. Normally R would be set to 1, but radiances are properly zero (otherwise undefined) at normal angle $\pi/2$, and this could cause unwanted darkening at grazing angles—the opposite of the Fresnel effect exhibited by many reflective materials. If the maximum normal angle θ is to be $\pi/2 - \delta$, then $R \approx \sqrt{1-\delta}$. We denote by $\theta = \theta(\alpha, \beta)$ and $\phi = \phi(\alpha, \beta)$ the local spherical coordinates of the point (x, y, z) given by (20)-(22).

3.3 Approximation

Given the mappings $\theta(\alpha, \beta)$ and $\phi(\alpha, \beta)$ in the previous subsection we can approximate the radiance $L(u, v, \theta, \phi, \lambda)$ for a given surface by a 5-dimensional tensor product spline in the variables $(u, v, \alpha, \beta, \lambda)$. Thus

$$L(u, v, \theta(\alpha, \beta), \phi(\alpha, \beta), \lambda) \approx \tilde{L}(u, v, \alpha, \beta, \lambda),$$

where

$$\tilde{L}(u, v, \alpha, \beta, \lambda) = \sum_{i,j,k,l,m} a_{ijklm} B_m B_l B_k B_j B_i. \tag{23}$$

For the variables u and v we use the knot vector given by (19), while for α and β we use

$$[\underbrace{-R, \ldots, -R}_{d+1}, -R+h, -R+2h, \ldots, R-h, \underbrace{R, \ldots, R}_{d+1}], \tag{24}$$

where $h = 2R/(m+1)$. Finally, for the wavelength parameter λ we can use a uniform knot vector, say from 400nm to 800nm.

The coefficients are computed from the Schoenberg method, the surface radiance evaluated at the Schoenberg node values

$$a_{ijklm} = L(u_i^*, v_j^*, \theta(\alpha_k^*, \beta_l^*), \phi(\alpha_k^*, \beta_l^*), \lambda_m^*) \tag{25}$$

We note that they could also be constructed using a quasi-interpolation method.

3.3.1 Wavelength Dependence

For many rendering applications, including our renderer, wavelength-dependent radiance is represented using a trichromatic representation with three real values. This can be done with a 5D B-spline function, where the degree in λ is 0, and there are 4 knot values. It is probably faster, though, to use a tri-color value (RGB) for each B-spline coefficient, and reduce to a 4D tensor product function. In the remainder of this paper we frequently omit the wavelength dependence and use 4-dimensional splines for surface radiance.

4 Global Illumination

Now we consider solving the global (hemispherical) integral equation in the form (8). This is a coupled system of integral equations, one equation for each surface in the scene. We turn to the evaluation of the integrals in terms of tensor product B-splines. Since the integrands can be evaluated quickly at any sample point the (hemispherical) integral can be evaluated using standard methods. The impetus for storing the incremental radiances is to facilitate this computation.

Consider first the hemispherical integral formulation. For the radiance approximation, the B-spline coefficient is computed from

$$a_{ijkl} = L_R^{\text{emit}}(u_i^*, v_j^*, \theta_{kl}^*, \phi_{kl}^*) + \int_\Omega L_R^{\text{in}}(u_i^*, v_j^*, \theta_{kl}^*, \phi_{kl}^*)\rho(u_i^*, v_j^*, \theta_{kl}^*, \phi_{kl}^*, \theta^{\text{in}}, \phi^{\text{in}}) \cos\theta^{\text{in}} d\omega \tag{26}$$

where the * variables indicate the evaluation nodes (θ^*_{kl} and ϕ^*_{kl} are functions of α^*_k and β^*_l). For simplicity we shall use θ and ϕ in this section, and note in practice they are functions of the spline parameters α and β. The incoming radiance L^{in} is evaluated, in our implementation, using the same ray-tracing method described above for rendering. Given this, and the BRDF ρ, the integral can be evaluated using any standard numerical method that uses point sampling. For example, the integral can be computed using an importance sampling method based on the values of the BRDF.

Our approach to evaluating the radiance integrals is to perform a B-spline approximation to the integrand, and evaluate this exactly using existing techniques. The integrand for the hemispherical integral is six-variate, or seven-variate if the wavelength is included. The approximation therefore has the form

$$\int_\Omega L_R^{\text{in}} \rho \cos\theta^{\text{in}}\, d\omega \tag{27}$$

$$= \int_0^{2\pi}\int_0^{\pi/2} L_R^{\text{in}} \rho \cos\theta^{\text{in}} \sin\theta^{\text{in}}\, d\theta^{\text{in}}\, d\phi^{\text{in}} \tag{28}$$

$$= \int_0^{2\pi}\int_0^{\pi/2} f(u,v,\theta,\phi,\theta^{\text{in}},\phi^{\text{in}})\, d\theta^{\text{in}}\, d\phi^{\text{in}} \tag{29}$$

$$= \int_0^{2\pi}\int_0^{\pi/2} \sum_{ijklpq} f(u^*_i, v^*_j, \theta^*_{kl}, \phi^*_{kl}, \theta^{\text{in}*}_p, \phi^{\text{in}*}_q) B_q B_p B_l B_k B_j B_i\, d\theta^{\text{in}}\, d\phi^{\text{in}} \tag{30}$$

where the B-spline coefficients a_{ijklpq} are the integrand, evaluated at node values u^*_i, v^*_j, θ^*_{kl}, ϕ^*_{kl}, $\theta^{\text{in}*}_p$, and $\phi^{\text{in}*}_q$. The choice of the first four is dependent on the knot vectors for th B-spline radiance representation on R and is therefore fixed, but the latter two come from knot vectors that can be arbitrarily chosen, and could even be different for each integral evaluation. But given the knot vectors σ and τ for θ^{in} and ϕ^{in}, respectively, the B-spline approximation to the integral can be evaluated exactly as

$$\int_0^{2\pi}\int_0^{\pi/2} \sum_{ijklpq} a_{ijklpq} B_q B_p B_l B_k B_j B_i\, d\theta^{\text{in}}\, d\phi^{\text{in}} \tag{31}$$

$$= \sum_{ijklpq} a_{ijklpq} \frac{\tau_{q+d_{\phi^{\text{in}}}+1} - \tau_q}{d_{\phi^{\text{in}}}+1} \frac{\sigma_{p+d_{\theta^{\text{in}}}+1} - \sigma_p}{d_{\theta^{\text{in}}}+1} B_l B_k B_j B_i \tag{32}$$

from which the B-spline coefficients for $\tilde{L}_R$ may be taken.

The surface integral formulation can be approximated in the same way.

$$\int_0^1\int_0^1 \chi_e \text{vis}(r,e) \frac{(e-r)\cdot\mathbf{z}_R\,(r-e)\cdot\mathbf{z}_E}{\|r-e\|^4} L_E \rho \|\mathbf{r}_s \times \mathbf{r}_t\|\, dsdt \tag{33}$$

$$= \int_0^1\int_0^1 f(u,v,\theta,\phi,s,t)\, dsdt \tag{34}$$

$$= \sum_{ijkl}\left(\sum_{pq} f(u^*_i, v^*_j, \theta^*_{kl}, \phi^*_{kl}, s^*_p, t^*_q) \frac{\tau_{q+d_t+1} - \tau_q}{d_t+1} \frac{\sigma_{p+d_s+1} - \sigma_p}{d_s+1}\right) B_l B_k B_j B_i \tag{35}$$

and thus gives the explicit formula for the B-spline coefficients for the outgoing radiance $\tilde{L}_R$ reflected off of R from E. The total actual outgoing radiance B-spline on R is the sum of these coefficients, from all other surfaces, including R itself if it is non-convex.

4.1 Generalized Form Factors

Although this last equation gives a direct formula for each B-spline coefficient for the exitant radiance reflected off R from E, it is in terms of the integrand evaluated at node points. It would be more natural, and closer to the classical radiosity method, to have the coefficients in terms of the outgoing radiance on E directly. Theoretically this is not difficult.

Suppose that the knot vectors used for representation as well as integration are the same, and fixed for the entire environment at the outset. Then the terms in the integrand, except for L_E are all constant. Furthermore, for a given sample point of L_E, the B-spline basis functions are constant, and thus the sampled values of L_E are linear functions of the coefficients of the coefficients. Consequently, the coefficients for L_R are all linear functions of the coefficients of L_E. In other words, the tensor of coefficients T_R for the reflected radiance from E off of R is a linear function of the tensor T_E of exitant radiance from E. Writing this function as

$$T_R = \mathcal{F}_{RE}(T_E) \tag{36}$$

the total gathered radiance tensor becomes

$$T_R = \sum_S \mathcal{F}_{RS}(T_S) \tag{37}$$

and this is directly analogous to the classical radiosity method.

4.2 Gathering

The solution process corresponds to a gathering approach similar to that used in classical radiosity. Figure 4 shows the data structures. A `Bspline4D` object contains the knot vectors as well as the tensor or coefficients, and a `surface_ radiance` object stores three `Bspline4D` objects: one for the cumulative gathered radiance, one for an incremental radiance from the most recent gather, and one temporary value for the gathered radiance. Algorithm 1 outlines the method.

Algorithm 1 Gathering Algorithm

// Direct Lighting Phase
for each surface S **do**
  S.L ← 0
  for each emitting surface E **do**
    gather from E to S, store the radiance in S.L
  end for
  add the gathered radiance to the cumulative radiance: S.Le ← S.L
end for

// Indirect Lighting Phase
while each .L is too large **do**
  for each surface S **do**
    gather globally from the scene to S.Ltmp
    // the emission is taken from the .L functions on other surfaces
  end for
  for each surface S **do**
    add the gathered radiance S.Ltmp to S.Le
    replace the incremental radiance S.L with S.Ltmp
  end for
end while

// Final Gather Phase
for each surface S **do**
  refine S.L as necessary
  gather globally from the scene to S.L
  // the emission is taken from the .Le functions on other surfaces
end for

The first iteration is the "direct lighting" stage, where surfaces are only illuminated by emissive surfaces, *i.e*, light sources. For this we use the surface integral formulation because it is far more efficient, as only a relatively small solid angle on the hemisphere is producing radiance.

```
struct Bspline4D {
  radiance_tensor4D   C            // B-spline coefficients
  real                knots[4][]   // knot vectors
  int                 degree[4]    // B-spline degrees
}

struct surface_radiance {
  Bspline4D   Le     // cumulative exitant radiance
  Bspline4D   L      // incremental exitant radiance
  Bspline4D   Ltmp   // temporary exitant radiance
}
```

Figure 4: *Data structures for tensor product B-splines and surface radiances.*

After the direct lighting computation, the L field for each surface contains the radiance due to direct illumination. From these we gather to each surface, using the hemispherical integral, from the entire environment, placing the result in the Ltmp field. This radiance is the indirect lighting caused by a single reflection. We add this to the cumulative radiance Le, and store this into the radiance L field and repeat. Notice that the L fields contain the indirect lighting after exactly n reflections, while the cumulative radiance Le is not used for any of the indirect lighting.

It might seem more natural to gather from the cumulative radiance Le and indeed this would require only two B-spline functions for each surface for the global illumination computation. One reason for the extra temporary radiance is that we can usually get away with a much coarser approximation to the surface radiances during the global illumination computation.

The final gather is something like a radiosity reconstruction. A finer approximation is usually required, and the integral evaluations may need to be done with more knots. The gathering can be done using the surface integral for brightly emitting objects (like the original light sources) and the hemispherical integral can be used for the rest—surfaces from which the surface-to-surface value is used are treated as non-emissive by the hemispherical integral.

4.3 Transmission

The integrals in 6 and 7 govern the reflected transport, if a surface also has a bi-directional transmission distribution function (BTDF). To handle transmission, the radiance function must be extended to directions below the local surface tangent plane, *i.e.*, having normal angle $\theta > \pi/2$. We do this by adding a second radiance function L'_R for the transmitted radiance.

5 Implementation

We have implemented the 4D B-spline techniques presented here in the context of a standard ray tracer for both rendering and global illumination computation. Our scenes consist of triangles, parallelograms, Bézier patches, and trimmed NURBS surfaces. For the global illumination computation, ray tracing is required for evaluating the visibility function in the surface integral formulation, and for evaluating the L^{in} value in the hemispherical integral. Therefore our ray tracer must be able to compute ray intersections with these surfaces and the corresponding parameter values as well as surface derivatives to compute the local frame.

Regardless of how a surface radiance B-spline is computed (or captured) it can be treated as a general surface shader. The shader, shown in pseudo-code in Figure 6, is dependent on view direction and surface position. Virtually any rendering system that can render the scene surfaces can render the scene using the B-spline radiances. We incorporated the B-spline shader into GL-based renderer. OpenGL does not currently support hardware B-spline function evaluation, nor does it support surface texturing from an arbitrary function, so we divide each surface into micro-faceted polygons and render each using smooth shading. The vertex colors are computed from our B-spline surface function, with the eye point corresponding to the center of projection. Figure 7 shows a simple scene so rendered.

5.1 Representation Issues

B-spline approximation functions are known to converge, so it is clear that the methods presented in this paper will work—provided enough knots are chosen. But a memory explosion could easily result. For example, if a modest 100 coefficients are required in each dimension, then one hundred million control points would be required, pushing the limits of a modern workstation. A multi-resolution approach could certainly cut down on the number of coefficients but we are investigating the tensor product approach as an interactive technique. We are interested, therefore, in finding good representations of surface radiances by exploiting the intrinsic properties of B-spline functions. Our goals emphasize visual realism more than numeric accuracy.

The primary drawback of tensor product B-splines is their inability to represent high frequency detail. In graphics, these occur along shadow edges, at points of contact between surfaces, and in near-mirror reflections. Only in the latter situation does the B-spline approach method truly break down; tensor product B-splines tend work well for glossy and moderately specular surfaces. Problems with shadow discontinuities and points of contact can be largely handled by re-meshing the surfaces as necessary. Figure 8 shows an example of this.

The radiance from a purely diffuse surface is view independent and could therefore be represented by a 2D spline. We propose a simple decomposition

$$L(u, v, \theta, \phi) = D(u, v)F(\theta, \phi) + S(u, v, \theta, \phi). \tag{38}$$

Here D is a diffuse component (*i.e*, irradiance) and is scaled by a directional dependent term F that captures, for example, the Fresnel behavior of increasing specularity at grazing angles. A low specular surface with much surface detail will benefit greatly from this decomposition.

In general we use quadratic (degree 2) splines in all four variables primarily because quadratic splines tend to represent the general shape of the approximated function better than higher-order splines, and they are faster to evaluate. Theoretically the second-order discontinuities that exist in quadratic splines might be noticeable, but we have not found this to be a problem. In some cases it appears that using cubic splines in the spatial dimensions improves the appearance along shadow discontinuities.

We have found that using 16 coefficients in each of the four dimensions is sufficient to represent radiance for a typical surface patch—note this amounts to as many coefficients as there are pixels in a 256×256 texture map image. Glossy surfaces with a small diffuse component tend to require fewer spatial knots and more angular knots, while diffuse surfaces require more spatial knots. The number of spatial knots required also depends on the physical or apparent size of the surface patch. A micro-facet, for example, might need only one spatial coefficient. Surface that exhibit near mirror reflection, however, can require many knots in both directions.

5.2 Performance

The most important performance consideration is the evaluation of the 4D tensor product B-spline functions. Our evaluation is done in software. Figure 5 shows a graph of some benchmarks run on a (single) MIPS R12K 400 MHz processor. The general upward trend

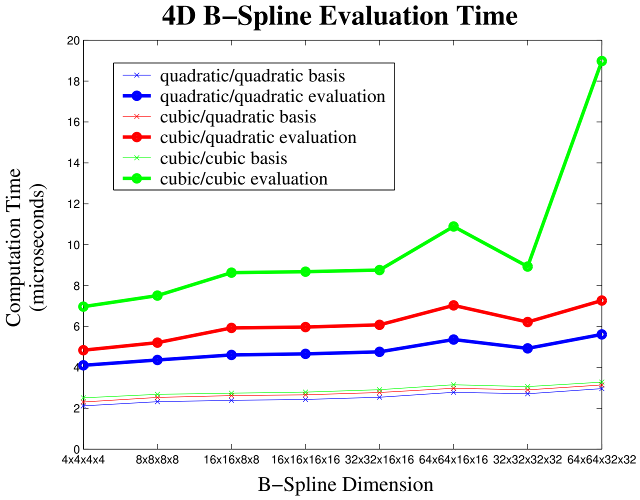


Figure 5: *Benchmarks for 4D tensor product B-spline evaluation (MIPS R12K, 400MHz, single CPU). The total computation time per evaluation, including the hemispherical mapping, is shown (thick lines), along with the time of computing the B-spline basis functions (thin lines). Three different degree combinations are shown, with degree in u and v first. The horizontal labels indicate the number of knots in each parameter.*

of evaluation time with larger dimension is primarily due to degrading cache coherence. Quadratic 4D splines of modest dimensions can generally be evaluated at rates of better than 200K.

In our ray tracer, the cost of finding the surface intersections generally outweighs the cost evaluating the surface B-splines, particularly when there are many NURBS surfaces in the scene. For the GL implementation this is not the case. In both implementation though, the B-spline evaluation time dominates the sphere mapping and local frame coordinate conversion. The simple scene illustrated in Figure 7 runs at roughly 10 frames per second on a fast PC. The polygon and sphere are each divided into 128^2 micro-triangles.

The global illumination computation remains far from interactive. A scene with only a few surfaces and modest sampling (a coarse solution uses 7×7 knots points for the surface integration and 10×10 points on the hemispherical integration) and only one or two indirect gathers takes from a few seconds to a few minutes. More complicated scenes, finer knot sampling, and more iterations can push the computation time up to several hours on a single processor.

5.3 Results

Figures 7–11 and the "teaser" images show some results using our method. Except for the GL scene, all images are ray traced using the B-spline shader. The global illumination was computed with from 1 to 4 indirect gathers. The surfaces are polygons, Beziér patches and trimmed NURBS. We generally used the Ashikhmin-Shirley BRDF [Ashikhmin and Shirley 2000], because it is energy conserving and exhibits the Fresnel behavior of increased specularity at glancing angles. The specular exponents were limited to be under 100.

6 Conclusion

In this paper we have demonstrated that we can effectively approximate radiance using a suitably crafted high dimensional tensor

`radiance` BSPLINE_SHADE (`surface` S, `real` u, `real` v, `point` *eye*)
  // Returns the B-spline radiance function at u, v, as viewed from eye.
{
  $r \leftarrow S(u, v)$
  $\hat{\mathbf{d}} \leftarrow (eye - r)/\|eye - r\|$
  compute the surface frame $\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S$ at u, v.
  compute $\hat{\mathbf{w}}$, the direction of $\hat{\mathbf{d}}$ in the local frame at u,v
  map $\hat{\mathbf{w}}$ to spline parameters α, β
  return $S.L(u, v, \alpha, \beta)$
}

`radiance` BSPLINE_SHADE (`surface` S, `point` x, `unit vector` $\hat{\mathbf{d}}$)
  // Returns the B-spline radiance function at x, situated on the surface S
  // as viewed from the direction $\hat{\mathbf{d}}$
{
  compute the u, v parameter values of x on the surface S
  compute the surface frame $\mathbf{x}_S, \mathbf{y}_S, \mathbf{z}_S$ at u, v.
  compute $\hat{\mathbf{w}}$, the direction of $\hat{\mathbf{d}}$ in the local frame at u,v
  map $\hat{\mathbf{w}}$ to spline parameters α, β
  return $S.L(u, v, \alpha, \beta)$
}

Figure 6: *Functions to render a surface point using a B-spline radiance function. The first is more suited to scan-line rendering, the second, to ray tracing.*

product spline. Furthermore we can take advantage of the unification accorded by representing both geometry and physical behavior by the same functional form. As the examples illustrate the representation captures a wide variety of phenomena in a single form. It is also amenable to fast evaluation and can deal with self-illumination of free-form surfaces.

The standard tensor product B-splines are fast to evaluate, but they may not be the most efficient representation. A multiresolution approach, using spline wavelets for example, is likely to reduce the storage requirements. This could be particularly useful for the high frequency detail the present method misses.

In this paper we have taken a rather direct approach to the approximation and integral evaluation. Our images all use uniform knot spacing both for representation and for integration. Nonuniform knots could improve this. Knot clustering, for example, could improve the representation on a specular surface, or a surface with non-uniformly distributed detail.

Our approach to evaluating the illumination integrals is very straightforward. We have not attempted to speed the computation because we view it as a pre-process. For example, the hemispherical integral could be importance sampled according to the BRDF, and similarly the knot distribution for the surface integral evaluation can be adjusted to capture a BRDF spike. Hierarchical or clustering methods might also improve performance.

Finally, we note some natural extensions. Time varying radiance is a relatively simple extension, as is the representation of volumetric data, including scattering.

In summary we conclude that tensor product B-splines are a flexible representation for encoding view-dependent surface radiances. It is likely that more gains lie in this direction.

References

ARVO, J. 1995. *Analytic Methods for Simulated Light Transport*. PhD thesis, Yale University.

ASHIKHMIN, M., AND SHIRLEY, P. S. 2000. An anisotropic phong brdf model. 25–32.

de Boor, C., and Fix, G. 1973. Spline approximation by quasi-interpolants. *J. Approximation Theory 8*, 19–45.

Chen, M., and Arvo, J. 2000. Pertubation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics 6*, 3 (July-Sept.), 253–264.

Christensen, P. H., Stollnitz, E. J., Salesin, D. H., and DeRose, T. D. 1994. Wavelet radiance. In *Fifth Eurographics Workshop on Rendering*, 287–302.

Christensen, P. H., Lischinski, D., Stollnitz, E. J., and Salesin, D. H. 1997. Clustering for glossy global illumination. 3–33. ISSN 0730-0301.

Cohen, M. F., and Wallace, J. R. 1993. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Cambridge, MA.

Cohen, E., Riesenfeld, R. F., and Elber, G. 2001. *Geometric Modeling with Splines: An Introduction*. A K Peters, Natick, MA.

Fournier, A. 1995. Separating reflection functions for linear radiosity. *Eurographics Rendering Workshop 1995* (June), 296–305. Held in Dublin, Ireland.

Glassner, A. S. 1995. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco.

Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. 1993. Wavelet radiosity. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 221–230. ISBN 0-201-58889-7.

Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. 1996. The lumigraph. In *SIGGRAPH 96 Conference Proceedings*, Addison Wesley, H. Rushmeier, Ed., Annual Conference Series, ACM SIGGRAPH, 43–54. held in New Orleans, Louisiana, 04-09 August 1996.

Granier, X., Drettakis, G., and Walter, B. 2000. Fast Global Illumination Including Specular Effects. In *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, Springer Wien, New York, NY, B. Peroche and H. Rushmeier, Eds., 47–58.

Greger, G., Shirley, P., Hubbard, P. M., and Greenberg, D. P. 1998. The irradiance volume. *IEEE Computer Graphics & Applications 18*, 2 (March-April), 32–43. ISSN 0272-1716.

Heidrich, W., and Seidel, H.-P. 1999. Realistic, hardware-accelerated shading and lighting. *Proceedings of SIGGRAPH 99* (August), 171–178. ISBN 0-20148-560-5. Held in Los Angeles, California.

Heidrich, W. 2001. Interactive display of global illumination solutions for non-diffuse environments - a survey. 225–244. ISSN 1067-7055.

Jensen, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. A K Peters, Natick, MA.

Kautz, J., and McCool, M. D. 1999. Interactive rendering with arbitrary brdfs using separable approximations. *Eurographics Rendering Workshop 1999* (June). Held in Granada, Spain.

Lee, B.-G., Lyche, T., and Mørken, K. 2000. Some examples of quasi-interpolants constructed from local spline projectors. In *Mathematical Methods in CAGD: Oslo 2000*, Vanderbilt University Press, Nashville, TN, T. Lyche and L. L. Schumaker, Eds., 243–252.

Levoy, M., and Hanrahan, P. M. 1996. Light field rendering. In *Proceedings of SIGGRAPH 96*, ACM SIGGRAPH / Addison Wesley, New Orleans, Louisiana, Computer Graphics Proceedings, Annual Conference Series, 31–42. ISBN 0-201-94800-1.

Lischinski, D., Tampieri, F., and Greenberg, D. P. 1993. Combining hierarchical radiosity and discontinuity meshing. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 199–208. ISBN 0-201-58889-7.

Lyche, T., and Schumaker, L. L. 1975. Local spline approximation methods. *J. Approximation Theory 15*, 294–325.

McCool, M. D., Ang, J., and Ahmad, A. 2001. Homomorphic factorization of brdfs for high-performance rendering. *Proceedings of SIGGRAPH 2001* (August), 171–178. ISBN 1-58113-292-1.

McCool, M. D. 1995. Analytic antialiasing with prism splines. In *Proceedings of SIGGRAPH 95*, ACM SIGGRAPH / Addison Wesley, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 429–436. ISBN 0-201-84776-0.

Miller, G. S., and Hoffman, C. R. 1984. Illumination and reflection maps: Simulated objects in simulated and real environments. *SIGGRAPH '84 Advanced Computer Graphics Animation seminar notes* (July).

Mitchell, D. P., and Hanrahan, P. 1992. Illumination from curved reflectors. In *Computer Graphics (SIGGRAPH '92 Proceedings)*, E. E. Catmull, Ed., vol. 26, 283–291.

Peter, I., and Strasser, W. 2001. The wavelet stream: Interactive multi resolution light field rendering. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, Eurographics, 127–138. ISBN 3-211-83709-4.

Ramamoorthi, R., and Hanrahan, P. 2001. An efficient representation for irradiance environment maps. *Proceedings of SIGGRAPH 2001* (August), 497–500. ISBN 1-58113-292-1.

Redner, R. A., Lee, M. E., and Uselton, S. P. 1995. Smooth B-Spline illumination maps for bidirectional ray tracing. *ACM Transactions on Graphics 14*, 4 (Oct.), 337–362. ISSN 0730-0301.

Riesenfeld, R. F. 1973. *Applications of B-spline Approximation to Geometric Problems of Computer-Aided Design*. PhD thesis, Syracuse University.

Rushmeier, H. E., and Torrance, K. E. 1990. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Transactions on Graphics 9*, 1 (Jan.), 1–27.

Shirley, P., and Chiu, K. 1997. A low distortion map between disk and square. *Journal of Graphics Tools 2*, 3, 45–52. ISSN 1086-7651.

Shirley, P. 1990. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, 205–212.

Shirley, P. 2000. *Realistic Ray Tracing*. A K Peters, Ltd.

Sillion, F. X., and Puech, C. 1989. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, J. Lane, Ed., vol. 23, 335–344.

Sillion, F., and Puech, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco.

Sillion, F. X., Arvo, J. R., Westin, S. H., and Greenberg, D. P. 1991. A global illumination solution for general reflectance distributions. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, T. W. Sederberg, Ed., vol. 25, 187–196.

Stark, M. M., Cohen, E., Lyche, T., and Riesenfeld, R. F. August 1999. Computing exact shadow irradiance using splines. *Proceedings of SIGGRAPH 99*, 155–164. ISBN 0-20148-560-5. Held in Los Angeles, California.

Veach, E., and Guibas, L. J. 1997. Metropolis light transport. In *Proceedings of SIGGRAPH 97*, ACM SIGGRAPH / Addison Wesley, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 65–76. ISBN 0-89791-896-7.

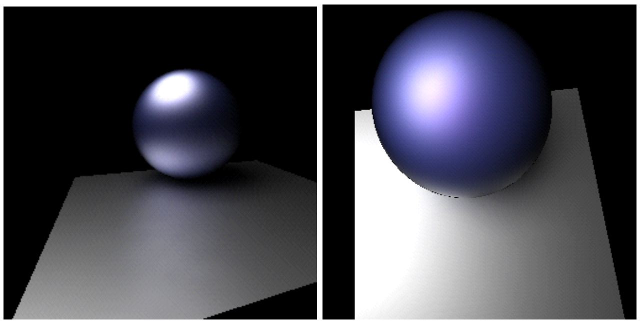


Figure 7: *OpenGL renderings of a simple scene (global illumination computed offline) using a B-spline shader. Each object is rendered with* 128^2 *micropolygons.*

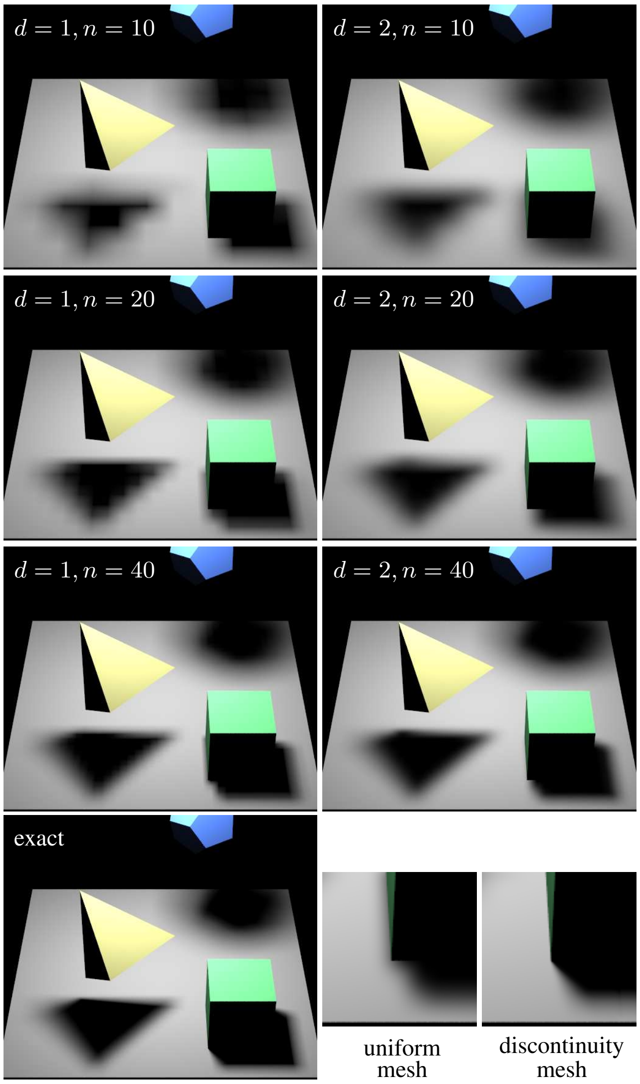


Figure 8: *2D B-splines for a diffuse surface with varying degree and knot density. Note the degree 1 case reduces to linear interpolation. The tensor product nature is most noticable on shadow edges diagonal to the knot lines. Discontinuity meshing helps with this, here, only one division was necessary for a good approximation at the point of contact.*

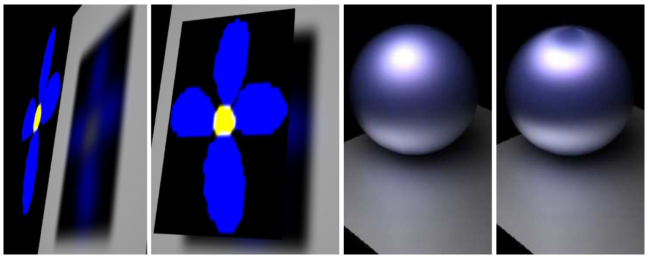


Figure 9: *A rectangle with a phong-like transmission function casting light onto a plane (left), an isotropic and anisotropic sphere (right).*

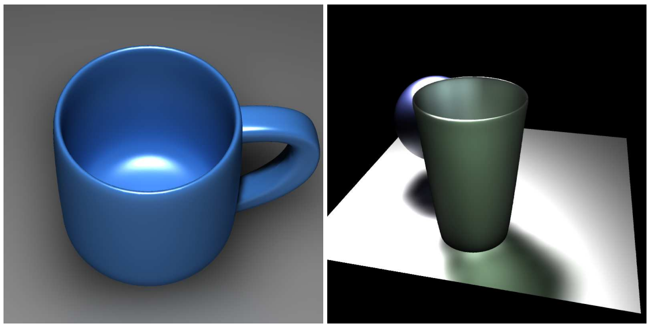


Figure 10: *A mug with much indirect lighting (5 indirect iterations) and a translucent cup. The mug consists of only two surfaces, (one for the handle, one for the cup) with 10 knots in each angular parameter and 41 in the spatial parameters. The cup is a single surface, with 17 knots in each parameter.*

Figure 11: *An anisotropic sphere and a transparent sphere, illuminated by a source in the shape of a light-bulb. The left sphere has 7 knots in each parameter; the right, 21. Notice there are sufficient knots (25 in each dimension) on the table to reproduce the caustic, but not the shadows.*