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.
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.


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 [Veuch 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 Sträßer 2001; Gortler et al. 1996; Levoy and Hanrahan 1996; Peter and Sträßer 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 \( S \in \mathcal{E} \) there is a local coordinate system, or local frame, given by

\[
\begin{align*}
  x_s &= \frac{s_u}{\|s_u\|} \\
  z_s &= \frac{s_u \times s_v}{\|s_u \times s_v\|} \\
  y_s &= z_s \times x_s,
\end{align*}
\]

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

\[
\begin{align*}
  \theta &= \arccos \frac{\mathbf{d} \cdot \mathbf{z}_s}{\|\mathbf{d}\|} \\
  \phi &= \arctan \frac{\mathbf{d} \cdot \mathbf{y}_s}{\mathbf{d} \cdot \mathbf{x}_s}
\end{align*}
\]

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 \( W/m^2sr \)). 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 \( \omega \) 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 \( x' - x \) 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 extant, 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 a 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 $E$ has an outgoing spectral radiance $L_S(u,v,\theta,\phi,\lambda)$. This surface radiance is a function both of position $(u,v)$, direction $(\theta,\phi)$ in the local frame and wavelength $\lambda$. This value gives the radiant spectrum at render time, when the surface point at $(u,v)$ is viewed from direction $(\theta,\phi)$. 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 $E$ 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 $L_{\text{in}}^S(u,v,\theta,\phi,\lambda)$, and a Bi-directional Reflectance Distribution Function (BRDF) $\rho_S(u,v,\theta,\phi,\theta^\text{in},\phi^\text{in},\lambda)$. Except for a few light sources in the scene the emission is zero in most cases. The light sources illuminate 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 $L_{\text{in}}^S$ against the BRDF $\rho$ over the hemisphere and add the result to the emitted radiance,$n-1$

$$L_S(u,v,\theta,\phi,\lambda) = L_{\text{in}}^S(u,v,\theta,\phi,\lambda) \int_{[0,\pi]} L_{\text{in}}^S(u,v,\theta,\phi,\theta^\text{in},\phi^\text{in},\lambda) \rho(u,v,\theta,\phi,\theta^\text{in},\phi^\text{in},\lambda) \cos\theta^\text{in} d\omega$$  

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

When we write down equation (6) for each $S \in E$ 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 $E_E(s,t,\theta,\phi,\lambda)$. A change of variables can be applied to the integral of (6) to obtain the surface integral

$$L_R(u,v,\theta,\phi,\lambda) = + \int_{E_E(s,t,\theta_E,\phi_E,\lambda)} \rho(u,v,\theta,\phi,\theta^\text{in},\phi^\text{in},\lambda) dE,$$

where $\text{vis}(r,e) = 1$ 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 $\lambda$

$$L_R = \int_{r_0}^{r_1} \int_{t_0}^{t_1} \text{vis}(r,e) \frac{(e - r) \cdot z_E}{\|r - e\|^2} d\tau dt$$

where $\chi_\text{vis}$ denotes the characteristic function of the parameter domain of $e(s,t)$. $\chi_\text{vis}(s,t)$ 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 $S_d,\tau$ of splines of degree $d$ with knots $\tau$ as

$$S_d,\tau = \text{span}\{B_i, 0 \leq i \leq n-d-1\},$$

where $B_i(x) = B_{i,d}(x(x))$ is the $i$th B-spline of degree $d$ on the knot vector $\tau = (\tau_0, \ldots, \tau_{n+d})$ 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, that is zero outside the interval $[\tau_i, \tau_{i+d+1}]$ and $\sum B_i(x) = 1$ for all $x \in [\tau_i, \tau_{i+d+1}]$. Moreover, $B_i$ has $d-m$ continuous derivatives at a knot $x = \tau_i$ which occurs $m$-times among $\tau_i, \ldots, \tau_{i+d+1}$. If $f(x) = \sum_{i=0}^{n-d-1} a_i B_i(x)$ 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

$$f(x) = \sum_{i=0}^{d} a_i B_i(x)$$

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 $\tau$.

$$\int_{\tau_0}^{\tau_{n+d}} \sum_{i=0}^{n-d-1} a_i B_i(x) dx = \sum_{i=0}^{n-d-1} a_i \frac{\tau_{i+d+1} - \tau_i}{d+1}.$$

### 2.2.1 B-Spline Approximation

To approximate a function $f$ entails two tasks. We need to choose a suitable spline space $S_d,\tau$ and then compute the B-spline coefficients $a_i$ of a spline $g \in S_d,\tau$ so that $g = \sum_{i=0}^{n-d-1} a_i B_i$ 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 $i$th B-spline coefficient is given by $a_i = f(t_i^*)$, with the evaluation node $t_i^*$ an average of contiguous knot values

$$t_i^* = \frac{t_{i+1} + \cdots + t_{i+d}}{d}.$$
of the ith quadratic B-spline \( B_{i,2} \) as
\[
a_i = \frac{1}{2} f(t_{i+1}) + 2f(t_i) - \frac{1}{2} f(t_{i+2}),
\]
where \( t_i = (t_{i+1} + t_{i+2})/2 \). 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 \( O(h^3) \) 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 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 \( \tau_x \) and \( \tau_y \), 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
\[
f(x,y) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} a_{ij} B_{i,d_x}(y) B_{j,d_y}(x).
\]

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

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
\[
\sum_{i,j,k,l} a_{ijkl} B_{i}B_{j}B_{k}B_{l}.
\]

### 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
\[
[B_{j-3} \cdots B_{j}] \begin{bmatrix} a_{1,-3,j-3} & \cdots & a_{1,-3,j} \\ \vdots & \ddots & \vdots \\ a_{1,j-3} & \cdots & a_{1,j} \end{bmatrix} = \begin{bmatrix} B_{j-3} \\ \vdots \\ B_{j} \end{bmatrix}
\]

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 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:
\[
\sum_{i,j,k} a_{ijk} B_iB_jB_k = \sum_i \left( \sum_{j,k} a_{ijk} B_kB_j \right) B_i
\]

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 radiance reduces to a function \( L(u,v) \) of only two variables. To construct an approximation \( L(u,v) \) 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:
\[
[a, a + h, a + 2h, \ldots, b - h, b, \ldots, b] \quad \text{for } d \geq 1
\]

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 \( (u \text{ or } v) \) 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
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 $\theta$ is to be $\pi/2 - \delta$, then $\Omega \approx \sqrt{1 - \delta}$. 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

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

where

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

For the variables $u$ and $v$ we use the knot vector given by (19), while for $\alpha$ and $\beta$ we use

$$[-R, \ldots -R, -R + h, -R + 2h, \ldots, R - h, R, \ldots, R],$$

where $h = 2R/(m + 1)$. Finally, for the wavelength parameter $\lambda$ 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

$$a_{ijklm} = L(u^*, v^*, \theta(\alpha^*, \beta^*), \phi(\alpha^*, \beta^*), \lambda^*)$$

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 $\lambda$ 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

$$a_{ijkl} = L^m_{R}(u^*, v^*, \theta_{kl}, \phi_{kl})$$

$$+ \int_{\Omega} L^m_{R}(u^*, v^*, \theta_{kl}, \phi_{kl}) \rho(u, v, \theta_{kl}, \phi_{kl}) \cos \beta \cos \theta \omega$$

where

$$L^m_{R}(u^*, v^*, \theta_{kl}, \phi_{kl})$$

is the local radiance at the point $(u^*, v^*, \theta_{kl}, \phi_{kl})$.
where the * variables indicate the evaluation nodes (\( \theta_{1*}^n \) and \( \phi_{1*}^n \) are functions of \( \alpha_1^* \) and \( \beta_1^* \)). For simplicity we shall use \( \theta \) and \( \phi \) in this section, and note in practice they are functions of the spline parameters \( \alpha \) and \( \beta \). The incoming radiance \( L^i \) is evaluated, in our implementation, using the same ray-tracing method described above for rendering. Given this, and the BRDF, \( \rho \), the integral can be evaluated using any standard numerical method that uses point sampling. For example, the integral can be computed using an important sampling method based on the values of the BRDF.

Our approach to evaluating the radiances 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

\[
\int_{\Omega} L^i_{R} \rho \cos \theta^m \, d\omega = \int_{0}^{2\pi} \int_{0}^{\pi/2} L^i_{R} \rho \cos \theta^m \sin \theta^m \, d\theta^m \, d\phi^m
\]

\[
= \int_{0}^{2\pi} \int_{0}^{\pi/2} f(u, v, \theta, \phi, \theta^m, \phi^m) \, d\theta^m \, d\phi^m
\]

\[
= \int_{0}^{2\pi} \int_{0}^{\pi/2} \sum_{ijklpq} f(u_i, v_j, \theta_{i*}, \phi_{j*}, \theta_{p*}, \phi_{q*}) \, d\theta^m \, d\phi^m
\]

where the B-spline coefficients \( a_{ijklpq} \) are the integrand, evaluated at node values \( u_i, v_j, \theta_{i*}, \phi_{j*}, \theta_{p*}, \phi_{q*} \), respectively, the B-spline approximation to the integral can be evaluated exactly as

\[
\int_{0}^{2\pi} \int_{0}^{\pi/2} \sum_{ijklpq} a_{ijklpq} B_q B_p B_{k} B_{j} B_{i} \, d\theta^m \, d\phi^m
\]

\[
= \sum_{ijklpq} a_{ijklpq} \frac{\tau_q + d_{gm} + 1 - \tau_q}{\sigma_p + d_{gm} + 1 - \sigma_p} B_q B_p B_{k} B_{j} B_{i}
\]

from which the B-spline coefficients for \( \tilde{L}_{R} \) may be taken.

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

\[
\int_{0}^{1} \int_{0}^{1} \chi_{vis}(r, e) \, \frac{(e - r) \cdot z_E (r - e) \cdot z_E}{\|r - e\|^4} \, L_E \rho \|r_a \times r_e\| \, ds \, dt
\]

\[
= \int_{0}^{1} \int_{0}^{1} f(u, v, \theta, \phi, s, t) \, ds \, dt
\]

\[
= \sum_{ijklpq} \sum_{ijklpq} f(u_i, v_j, \theta_{i*}, \phi_{j*}, \theta_{p*}, \phi_{q*}) \frac{\tau_q + d_{gm} + 1 - \tau_q}{\sigma_p + d_{gm} + 1 - \sigma_p} \frac{d_{el} + 1}{d_s + 1}
\]

\[
B_i B_j B_k B_l
\]

and thus gives the explicit formula for the B-spline coefficients for the outgoing radiance \( \tilde{L}_{R} \) reflected off \( R \) 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 nodal 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

\[
T_{R} = \mathcal{F}_{RE}(T_{E})
\]

the total gathered radiance tensor becomes

\[
T_{R} = \sum_{S} \mathcal{F}_{RS}(T_{S})
\]

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 \( \text{B spline4D object} \) contains the knot vectors as well as the tensor or coefficients, and a surface radiance object stores three \( \text{B spline4D objects} \): 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: \( S.L := S.L \)

end for

// Indirect Lighting Phase

while each \( L \) is too large do

for each surface \( S \) do

// the emission is taken from the \( L \) functions on other surfaces

gather globally from the scene to \( S \).\( L \)

end for

// the emission is taken from the \( L \) functions on other surfaces

gather globally from the scene to \( S \).\( L \)tmp

// the emission is taken from the \( L \) functions on other surfaces

replace the incremental radiance \( S \).\( L \)tmp with \( S \).\( L \)

end for

// Final Gather Phase

for each surface \( S \) do

// refine \( S \).\( L \) as necessary

// the emission is taken from the \( L \) functions on other surfaces

gather globally from the scene to \( S \).\( L \)

// the emission is taken from the \( L \) functions on other surfaces

replace the incremental radiance \( S \).\( L \)tmp with \( S \).\( L \)

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.
radius is the indirect lighting caused by a single reflection. 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 $L^{th}$ 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 to 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

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

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 x 256 texture map image. Glossy surfaces with a small diffuse component tend to require fewer spatial knots and more angular, 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
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 1283 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, Bezier 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 specularly 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 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 multi-resolution 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 algorithms might 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


Figure 7: OpenGL renderings of a simple scene (global illumination computed offline) using a B-spline shader. Each object is rendered with 128² 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 noticeable 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.