Smooth B–Spline Illumination Maps for Bi–Directional Ray Tracing

Richard A. Redner
The University of Tulsa

Mark E. Lee
Amoco Production Research

Samuel P. Uselton
Computer Sciences Corporation

Abstract

In this paper we introduce B–Spline illumination maps and their generalizations and extensions for use in realistic image generation algorithms. The B–Spline lighting functions (i.e. illumination maps) are defined as weighted probability density functions. The lighting functions can be estimated from random data and may be used in bidirectional distributed ray tracing programs as well as radiosity oriented algorithms. The use of these lighting functions in a bidirectional ray tracing system which can handle dispersion as well as the focusing of light through lenses is presented.

1 Introduction. The use of distributed light sources in computer graphics has increased in recent years in an effort to generate more realistic computer generated images. Area light sources were incorporated into ray tracing algorithms by Cook [Cook84] who coined the phrase *distributed ray tracing.* In this context, a distributed source is treated as a probability density function so that area light sources can be statistically sampled [Cook84, Lee85]. In the field of bidirectional ray tracing, internal lighting functions over the surfaces of objects are created from data and these distributed sources are then sampled [Heck90] [Chen91].

We begin this paper by introducing the nonparametric probability density estimation problem and present several common nonparametric density estimators. We then make the connection between nonparametric density estimation and computer graphics and discuss some of the computational difficulties which exist. The B–Spline density estimator and extensions of this idea are then presented to help solve some of these problems. We show how B–Spline lighting functions can be created from data, sampled and rapidly evaluated. Statistical theorems concerning the asymptotic properties of B–Spline lighting functions are also presented. Finally we show how we have used B–Spline lighting functions in a bidirectional ray tracing system capable of rendering images which show dispersion and the concentration of light by translucent objects.

2 Density estimation. A density function on the real line is a nonnegative function $p(x)$ which integrates to one. Given a random quantity X with probability density function $p(x)$, the probability that an observed value of X falls between two numbers a and b is

$$\text{Probability}(a \le X \le b) = \int_a^b p(x)\,dx.$$

Knowledge of the random variable X can be obtained by discovering its probability density function, and this function can be used to make decisions and to perform other computations.

In many cases, however, the density function is not known and must be estimated from data. Specifically, given independent identically distributed observations $X_1, \ldots, X_n$ with unknown density function $p(x)$ we wish to estimate $p(x)$ from this data. If the functional form of the density is known, but specific parameters are unknown, we have a *parametric* density estimation problem. The use of

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \text{and} \qquad \hat{\sigma}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \hat{\mu})^2$$

as estimators of the mean and variance of a normally distributed random variable illustrates this case. In other situations, the form of the density is not known, and so we must attempt to estimate a completely unknown density function based on the data alone. This is the *nonparametric* density estimation problem.

The most common nonparametric density estimate is based on the histogram. The histogram, normalized by the area under its graph, is a probability density function, and so this normalized histogram is a nonparametric density estimate. To compute the histogram from a set of data, one must select either the number of bins into which the data is sorted or equivalently choose the width of the individual bins. The final estimate is sensitive to this choice. However, as long as the width h approaches 0 and nh approaches $+\infty$ as the sample size increases, the normalized histogram density estimate will converge to the true underlying density at all points of continuity of p as the sample size n increases.

In Figure 1, we see examples of four histogram density estimates based on 50 random sample points within the interval $[0, 100]$. The true density function is the dashed curve in each of the graphs. Clearly two bins in the histogram is not enough (Figure 1a) and sixteen bins (Figure 1d) is too many. Better estimates can be obtained if the sample size is increased. In all cases, of course, the histogram density estimate is discontinuous at the boundaries of the bins.

A second class of density estimates are kernel density estimators which were introduced by Parzen [Parz62]. In this setting a kernel is a density function and is usually chosen to be symmetric about zero. Given data $X_1, \ldots, X_n$, a kernel function $K(x)$ and a positive number c, the kernel density estimate of $p(x)$ is

$$\hat{p}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{c} K\left(\frac{x - X_i}{c}\right)$$

which is simply the convolution of $\frac{1}{nc}K(x/c)$ with the data. One of the difficulties with using kernel density estimates is the fact that all of the data must be stored to evaluate the density function. There can also be edge effects at the endpoints of the intervals if the density does not approach zero at the end points. The estimated density is sensitive to the choice of the parameter c and small values of c yield chaotic estimates.

In Figure 2, we see examples of four kernel density estimates based on the same 50 data points used in Figure 1. The kernel function K for this example is a quadratic B-spline on the interval $[-3/2, 3/2]$. The density estimate is highly variable when we use $c = 10$ and edge effects at 0 and 100 are quite severe for all values of c used in these examples.

Another density estimator is based on orthogonal series. Given an interval $[a, b]$, we consider a set of functions $\{\phi_k(x)\}_{k=1}^{\infty}$ with the property that $\int_a^b \phi_j(x)\phi_k(x)\,dx = \delta_{jk}$, where $\delta_{jk} = 1$ if $j = k$ and is zero otherwise. We also suppose that the set of functions is complete in the sense that if $p(x)$ is square integrable over the interval $[a, b]$ (i.e. $\int_a^b p^2(x)\,dx < \infty$) then,

$$p(x) = \sum_{k=1}^{\infty} \alpha_k \phi_k(x)$$

where

$$\alpha_k = \int_a^b \phi_k(x) p(x)\,dx.$$

If $p(x)$ is a density function then, using a suitably chosen value of m, the orthogonal series density estimate based on the data $X_1, \ldots, X_n$ is

$$\hat{p}_n(x) = \sum_{k=1}^{m} \hat{\alpha}_k \phi_k(x)$$

where

$$\hat{\alpha}_k = \frac{1}{n} \sum_{i=1}^{n} \phi_k(X_i) \qquad \text{for} \qquad k = 1, \ldots, m.$$

This density estimate has the very desirable property of being unbiased, so that the expected value of each $\hat{\alpha}_k$ is α_k, i.e.

$$E(\hat{\alpha}_k) = \int_a^b \hat{\alpha}_k p(x)\,dx = \alpha_k$$

and therefore

$$E\left(\hat{p}_n(x)\right) = \sum_{k=1}^{m} \alpha_k \phi_k(x)$$

for every $x \in [a, b]$. So we see that $\hat{\alpha}$ is an unbiased estimator of α and $\hat{p}_n(x)$ is an unbiased estimator of $\sum_{k=1}^{m} \alpha_k \phi_k(x)$.

Unfortunately the orthogonal series estimator is not necessarily a density, since $\hat{p}_n$ may be negative at some points and almost never integrates to one. These problems can be overcome. However, it is not clear how to generate data from the estimated density.

There are many other nonparametric density estimators. In particular, variable kernel estimates have very nice properties, and nearest neighbor density estimates and penalized maximum likelihood density estimators are both very interesting. Many one dimensional density estimates can be easily extended to densities in higher dimensions, and the reader is referred to Silverman [Silv86], Thompson and Tapia [Thom90], and Scott [Scot92] for a more complete discussion of the nonparametric density estimation problem.

3 Computer graphics applications which use density estimates. While there are many techniques used in the generation of realistic computer generated images, two algorithms, ray tracing and radiosity, stand out and have received considerable attention. As was pointed out by Kajiya [Kaji86], both ray tracing and radiosity paradigms provide methods for solving a particular integral equation commonly referred to as the rendering equation. Many of the algorithms used (bidirectional ray tracing and all radiosity algorithms in particular) require the storage of lighting information across the surfaces in the scene. In all cases, this information can be thought of as a scaled probability density function. The efficient construction and further manipulation of these functions are fundamental problems.

In particular, in the radiosity method as described in Goral et al. [Gora84] (see also Hottel [Hott54] or Sparrow [Spar63]) lighting functions are piecewise constant and hence the lighting information is a weighted histogram. Surfaces are partitioned into grids and each cell is assigned a radiosity value by the light propagation algorithm. The radiosity algorithm was improved by Cohen and Greenberg in [Cohe85] by the introduction of the hemi–cube structure and was further extended to include specular reflection [Imme86].

In the paper by Wallace et al. [Wall87] a "two–pass solution" is described where the light propagation algorithm computes the diffuse lighting effects and the specular lighting effects are determined by a ray tracing procedure. This procedure was extended by Sillion and Puech [Sill89]. Again the intermediate lighting function is associated with a discrete grid structure. Finally the paper by Campbell and Fussell [Camp90] introduces adaptive mesh generation in order to improve the quality and efficiency of radiosity generated images. All of these methods are based on some sort of mesh generation to store and represent the intermediate lighting function. Although these algorithms are deterministic in design and not statistically based, the lighting representation is discrete and has the structural form of a two dimensional histogram.

The paper by Chen et al. [Chen91] represents one of the latest efforts to fully solve the rendering equation. This paper combines progressive refinement radiosity, light ray tracing and monte carlo path tracing to account for all of the energy which can pass from the light sources to the eye. In order to capture focussing effects, light rays from the light sources are generated and the intersection of these rays with surfaces is stored. Kernel density estimates are then used to construct a smooth lighting function over the surface. In order to insure that these lighting functions are smooth, the width of each kernel is adjusted so that the kernel covers at least m neighbors for some fixed constant m.

The ray tracing algorithms compute an estimate of the solution to the rendering equation by tracing rays. Rays from the eye are sent into the scene and at each object intersection, reflected and refracted rays are generated. A shadow ray to each of the light sources is also generated at each of these intersection points in order to compute the amount of light which would be transported along the ray back to the eye. If the light is an area light source then a random point on the light source is chosen according to the power distribution or power density of the source. Cook et al. [Cook84] introduced these distributed (i.e. area) lights to ray tracing. In most cases, an area light is assumed to be uniformly bright across its surface, but this restriction is not required and any power distribution can be used as long as there is an algorithm for generating a random point from this density function. The color or wavelength of a light source may also depend on the position on the light source. In this case the shadow ray should be chosen according to the power distribution of the light source as a function of wavelength and position. Thus, the power distribution is modeled by sampling.

Sampling of area light sources can produce penumbras and other interesting effects. The sampling idea has been applied to simulate a variety of other effects as well. Sampling the direction of reflected rays can be used to simulate rough surfaces. Sampling in time and space provides motion blur. The number of applications of the "distributed" approach shows that it is flexible and requires little customizing.

While distributed ray tracing is a very powerful method for generating images which exhibit specular effects, one of its weaknesses is the expense involved in capturing the effects of diffuse illumination. Arvo [Arvo86] suggested that ray tracing could be performed in the forward and backward direction in order to capture diffuse lighting effects. Heckbert [Heck90] described a bidirectional ray tracing system based on adaptive radiosity textures. Again we see the use of a simple but adaptive mesh to store the intermediate lighting information. In the same paper, he put forward the idea that a lighting function is a density and that histograms or kernel estimates could be used to estimate density functions. This was exploited in Chen et al. [Chen91] but it also opens the door for the use of other types of nonparametric density estimators in the latest radiosity algorithms and in bidirectional ray tracing algorithms.

In particular, our current computer graphics research effort in bidirectional ray tracing [Gehr92, Redn93] requires a probability density estimation scheme which can be implemented over a subset of $\Re^3$. In this application, density functions represent normalized diffuse lighting functions on surfaces. These are generated from light rays which have been emitted by the light source and propagated through the scene. The three dimensions include two dimensions for the surface and one dimension for color or wavelength. Ultimately the research will investigate directionally dependent lighting functions and the addition of the directional dependence of the lighting functions will add two more dimensions to the problem.

These are clearly very computationally demanding problems. It is well known that the nonparametric density estimation problem is exceedingly difficult in higher dimensions. If we are to overcome this problem, we must use a very large number of observations. In order to have a practical method we require a nonparametric density estimation scheme which requires a minimal amount of storage. Additionally, the nonparametric density estimate must be easily generated from the data and we must be able to rapidly evaluate the density function. Finally, we will want to be able to evaluate conditional densities [Hogg65] and need the density estimate to not only be continuous, but also smooth, in order to eliminate the possibility of mach banding effects.

Given these requirements we rule out the use of a kernel density estimate because all of the data must be stored to evaluate the density. Given a sample size n, storage is $O(n)$, and evaluation of the function at many points is most efficiently accomplished by first sorting the data which is an $O(n \log(n))$ operation. Evaluation of the estimated density function at each point is then $o(n)$. As mentioned previously, edge effects are also a serious problem.

The histogram density estimator is unacceptable because it does not generate a continuous density function. However, a spline can be fit to the histogram generating a continuous density function. This estimator is called a histospline [Bone71] and histosplines are appealing in one dimension because only a histogram of the data is required to generate the density estimate. Given that the number of bins is chosen so that the standard consistency results for the histogram density estimate are satisfied, storage is $o(n)$ and evaluation can be performed in constant time. The extension of the histospline to multiple dimensions would require fitting a multidimensional spline to the multidimensional histogram. This could be accomplished but the histospline can become negative and there may be edge effects. Another method, the penalized maximum likelihood estimate is also a spline but with knots at all of the sample points and so storage is $O(n)$. Furthermore the penalized maximum likelihood estimate is not easily extended to multiple dimensions [Thom90].

We will observe in the next section that there is a better way to estimate a density using splines. We will introduce a nonparametric density estimator based on partitions of unity and introduce the B–Spline density estimate which was designed to solve current graphics problems. The nonparametric B–Spline density estimate is a generalization of the histogram density estimate, and like the histogram, is related to orthogonal series estimators (see Schwartz [Schw67]). The B-Spline density estimate requires $o(n)$ storage and evaluation at each point can be performed in constant time. Since the density estimate is based on splines, conditional densities can be rapidly generated and the degree of smoothness is easily controlled.

4 The B–Spline density estimator. Let I denote a finite interval of real numbers. A partition of unity for I is a finite set of basis functions $\{B_k(x)\}$ defined on I with the property that each $B_k(x)$ is continuous and nonnegative and that for every x

in I,

$$\sum_k B_k(x) = 1.$$

For this application we will also assume that the numbers

$$b_k = \int_I B_k(x) > 0 \quad \text{for each } k.$$

Given such a partition of unity and data $X_1, \ldots, X_n$ with density function $p(x)$, define the nonparametric density estimate

$$\hat{p}_n(x) = \sum_k \frac{\alpha_k}{b_k} B_k(x), \tag{1}$$

where

$$\alpha_k = \frac{1}{n} \sum_{i=1}^{n} B_k(X_i). \tag{2}$$

Observe that $\hat{p}_n(x)$ is always a continuous density.

B–Spline basis functions are a particularly useful instance of a partition of unity. A spline of order K with knots $u_1 < u_2 < \ldots < u_m$ is any function $s(x)$ defined on $[u_1, u_m)$ which is a polynomial of degree $K-1$ on each interval $[u_i, u_{i+1})$ for $1 \leq i < m$ such that $s(x)$ is $K-2$ times differentiable over the interval $[u_1, u_m)$. If we restrict $s(x)$ to the interval $[u_K, u_{m-K+1})$, then there exist functions $\{B_k(x)\}_{k=1}^{m-K}$ and constants $\{C_k\}_{k=1}^{m-K}$ so that

$$s(x) = \sum_{k=1}^{m-K} C_k B_k(x). \tag{3}$$

and $\{B_k(x)\}_{k=1}^{m-K}$ form a partition of unity on the interval $[u_K, u_{m-K+1}]$. The basis functions $\{B_k(x)\}_{k=1}^{m-K}$ depend only on the knots. Any spline of order K on $[u_K, u_{m-K+1})$ can be written as Eq. 3. For a comprehensive discussion of splines as used in this research refer to Bartels et al. [Bart87].

Up to K consecutive knots can be coincident and such a point is referred to as a multiple knot. The degree of smoothness is however, decreased by the presence of multiple knots. Since we usually want our density estimates to be very smooth, the use of multiple knots is not recommended except perhaps at the end points of the interval. In this case the first $K-1$ are identified with the left hand end point of the interval, the last $K-1$ knots are identified with the right hand end point of the interval and the remaining knots are placed between the endpoints.

Given a set of knots and $X_1, \ldots, X_n$, the B–Spline density estimator is defined by Eq. 1 and Eq. 2 where the functions $\{B_k\}$ are the B–Spline basis functions determined by the knot sequence. Of course we require that as the sample size increases, the B–Spline density estimate should converge to the true density function. Throughout the rest of this section we assume that for each sample size, n, we have a partition of unity of (possibly nonuniform) B–Splines over the interval $[a, b]$ with $0 < h = \min_k |u_{k+K-1} - u_k|$ and $\bar{h} = \max_k |u_{k+K-1} - u_k|$.

Theorem 1. *Assume that there is a number C, independent of n, so that $1 \leq \bar{h}/h \leq C$ and that $\bar{h} \to 0$ and that $nh \to \infty$ as $n \to \infty$. If p is a bounded function over the interval I and if p is continuous at the point x in I then as $n \to \infty$, $E(\hat{p}_n(x) - p(x))^2 \to 0$ in which case we say that $\hat{p}_n(x)$ converges in mean square error (MSE) to $p(x)$.*

The function p is said to be Lipschitz continuous on I if there is a number C_0 so that $|p(x) - p(y)| \leq C_0|x - y|$ for all x and y in I.

Theorem 2. *Under the conditions of Theorem 1, and assuming that p is Lipschitz continuous in the interval I then $E\left(\int_I (\hat{p}_n(x) - p(x))^2 \, dx\right) \to 0$. In this case we say that $\hat{p}_n$ converges in Integrated Mean Square Error (IMSE) to p.*

The uniform B–Spline case has been extensively investigated in [Gehr90] and in [Gehr92]. Non uniform B–Splines and partitions of unity in a much more abstract setting were considered in [Redn93]. Proofs of all of the results given in this section can be found in these references.

In Figure 3, we see examples of quadratic B–Spline density estimates. These density estimates are based on the same random data used in the simulations in Section 2. We see that a B–spline density estimate with too few basis functions (for example m=4 for 50 data points) lack flexibility and that an estimate with too many basis function fluxuates wildly (for example m=15). However, the B–spline density estimate with m=6 or m=8 basis functions offer very good estimates of the underlying probability density function. Edge effects are kept to a minimum and the density estimate is smooth.

5 Multiple dimension nonparametric density estimates. The ideas of the last section are easily extended to multiple dimensions and in particular to surfaces of various topologies including parametrically defined surfaces. Density estimates over a rectangle $R \equiv [a, b] \times [c, d]$ in the plane can be constructed using tensor product splines. In this case let $(X_1, Y_1), \ldots (X_n, Y_n)$ be independent identically distributed random observations with density $p(x, y)$. Given a partition of unity in each of the x and y directions, we can form a partition of unity over the plane by taking products of basis functions in the x direction with basis functions in the y direction.

We define a two dimensional probability density estimate $\hat{p}_n(x, y)$ in the following way. Let

$$\hat{p}_n(x, y) = \sum_j \sum_k \frac{\alpha_{jk}}{b_{jk}} B_j(x) B_k(y)$$

where

$$\alpha_{jk} = \frac{1}{n} \sum_{i=1}^{n} B_j(X_i) B_k(Y_i) \tag{4}$$

$$b_{jk} = b_j b_k.$$

A density estimate over a region A in the plane can be obtained by using Eq. 4 with

$$b_{jk} = \int_A \int B_j(x) B_k(y) \, dx \, dy.$$

As the following theorem demonstrates, consistency results are also available for this case.

Let $\{(X_i, Y_i)\}_{i=1}^n$ be identically distributed random variables from some fixed but unknown density function p on A. Choose a sequence of knots in the x and y directions whose spacing depends on n and let $B_j(x)$ and $B_k(y)$ denote the corresponding B–spline basis functions. Let $\bar{h}(n) = \max_{jk} b_{jk}$ and $h(n) = \min_{jk} b_{jk}$. We assume that the knot spacing is chosen so that (1) the knot spacing in the x and y directions goes to zero as n goes to infinity, (2) there is a number C independent of n so that $\bar{h}(n)/h(n) \leq C$ and (3) that $n \cdot h(n) \to \infty$ as $n \to \infty$.

Theorem 3. *If (x, y) is in A and if p is a bounded continuous function over A, then $E(\hat{p}_n(x, y) - p(x, y))^2 \to 0$ as $n \to \infty$ in which case we say that $\hat{p}_n(x, y)$ converges in mean square error (MSE) to $p(x, y)$.*

The extension to higher dimensions is straightforward and consistency results can be found in the works of Gehringer and Redner in [Gehr92] and [Redn93].

As our final statistical result we show that density estimates over parametrically defined surfaces can be created in a way to preserve mean square error convergence of the estimated density to the true density.

Let A be a region in the plane and let $\mathbf{r} : A \to R^3$ be a smooth 1-1 function. Then $M \equiv \mathbf{r}(A) = \{X = \mathbf{r}(u, v) | (u, v) \in A\}$ is a parameterized surface. Since $\mathbf{r}$ is 1-1, we define the function $(u(X), v(X)) : M \to A$ as the inverse function of $\mathbf{r}$ and define a density estimate over M in terms of a density estimate on A. Given a sample of random points $\{X_i\}_{i=1}^n$ from a density function f on M, we define a density function on A by

$$\hat{p}_n(u, v) = \sum_j \sum_k \alpha_{jk} \frac{B_j(u) B_k(v)}{b_{jk}}$$

where

$$\alpha_{jk} = 1/n \sum_i B_j(u(X_i)) B_k(v(X_i)).$$

We then define a density function on M as

$$\hat{p}_n^M(X) = \hat{p}_n(u(X), v(X)) / ||\partial \mathbf{r}/\partial u \times \partial \mathbf{r}/\partial v|| \tag{5}$$

Theorem 4. *For each sample size n let $\{B_k(x) B_j(y)\}$ be a partition of unity on the set A in the plane and suppose that there are numbers a and b so that*

$$||\partial \mathbf{r}/\partial u|| \leq b \qquad ||\partial \mathbf{r}/\partial v|| \leq b$$

and

$$0 < a \leq ||\partial \mathbf{r}/\partial u \times \partial \mathbf{r}/\partial v|| \leq b$$

Under the conditions of Theorem 3 the density estimate converges in mean square error at each point of $\mathbf{r}(A)$. If $p(x)$ is Lipschitz continuous on M then the density estimate converges in integrated mean square error to the true density function.

The most useful way of thinking about this construction is that we have built a nonparametric density estimate over the region A in the plane and that this can be transformed using Eq. 5 into a density estimate over the surface M. From an operational point of view this is very satisfying since most of the computation takes place in the plane where the arithmetic and computation is simple. To generate a random point X on the surface from the density function $\hat{p}_n^M$ we generate a random point (u, v) in A using the density function $\hat{p}_n$ and define $X = \mathbf{r}(u, v)$. To evaluate the density $\hat{p}_n^M(X)$ we evaluate the inverse maps $u(X)$ and $v(X)$ and then evaluate $\hat{p}_n(u, v)$ using the surface Jacobian. This last step can be performed using a numerical estimate of the derivative.

6 Representation of the lighting function. Let M be a surface in R^3 and I be an interval of real numbers corresponding to the wavelengths of visible light. Let H be the hemisphere of directions above any point on the surface. We define a lighting function as a function

$$l : M \times I \times H \rightarrow R^+$$

on the non-negative real numbers.

The function l represents the lighting function specified by the particular graphics application. The function can represent intensity, as in earlier rendering systems based on local illumination models [Blin77,Cook82]. For our application we think of a lighting function as representing the energy per unit time per unit area per unit interval of wavelength per unit solid angle and thus l may be referred to as a spectral radiance function or as a power density function. The total power is defined by

$$P = \int_M \int_I \int_H l \cos(\theta)\ d\omega\, d\lambda\, dA$$

and thus $l \cos(\theta)/P$ is a density function in the sense of probability theory.

Before we continue further, we observe that lighting functions may be defined over other domains. In particular M may be a volume, a curve or a point. The color domain may be an interval larger than the interval of visible light or may be one of the usual 3 dimensional color spaces. As M changes, the set of admissible directions H may also be suitably modified. All of our comments in this paper are fairly easily extended to the multiplicity of lighting domains.

A special case of the lighting functions described above are those lighting functions which are independent of direction. Lighting functions representing diffusely reflected light are functions of only space and wavelength. In this case,

$$l : M \times I \rightarrow R^+$$

is a radiosity function representing the energy per unit time per unit area per unit interval of wavelength. The total power is defined by

$$P = \int_M \int_I l\, d\lambda\, dA$$

and therefore l/P is a density function in the sense of probability theory. The representation of radiosity functions in a ray tracing program based only on rgb values is even simpler. In this case one needs only a lighting function which varies over the surface for each of the r, g, and b bands.

B–Spline density estimators were created for the purpose of representing lighting functions over surfaces. We begin by describing a monochromatic B–Spline lighting function over a rectangle R in the $x-y$ plane. We suppose that the knot sequence has been chosen in each of the x and y directions and that the order of the spline has been selected. As previously mentioned, this completely specifies the B–Spline basis functions in the x and y directions. We use equally spaced knots in our graphics application. Given this arrangement, the lighting function can be defined as

$$l(x,y) = \sum_j \sum_k \alpha_{jk} B_j(x) B_k(y) / b_{jk}.$$

where

$$b_{jk} = \int_R \int B_j(x) B_k(y)\, dx\, dy.$$

Since the B–Spline basis functions are non-negative, $l(x,y)$ is a power density (i.e. radiosity) function if all the α_{jk}'s are non-negative. In this case the power of the light source is $P = \sum_j \sum_k \alpha_{jk}$.

We now construct the power density function estimate l, on the rectangle R, given a random sample of rays generated by the light sources in the scene. Each such random ray is composed of a triple of numbers (X_i, Y_i, P_i) which are the x and y locations of the intersection of the ray with the rectangle and the power of the ray. Of course the ray also has a direction, but this information is not used to reconstruct the estimate of a diffuse light source. Given n such rays the nonparametric estimate of the radiosity function is

$$\hat{l}_n(x,y) = \sum_j \sum_k \hat{\alpha}_{jk} \frac{B_j(x) B_k(y)}{b_{jk}}. \tag{6}$$

where

$$\hat{\alpha}_{jk} = 1/n \sum_{i=1}^{n} P_i B_j(X_i) B_k(Y_i).$$

If we are building a lighting function for a traditional ray tracing program based on r, g, and b values, then we use the power of each of the r, g, and b bands to construct three lighting functions.

If we are working with a system based on wavelength, then each sample from the lighting function is composed of four numbers, $(X_i, Y_i, P_i, \lambda_i)$ and the direction of the ray. The density estimate based on this data uses

$$\hat{\alpha}_{jkl} = 1/n \sum_i P_i B_j(X_i) B_k(Y_i) B_l(\lambda_i).$$

The total power of the light source is

$$P = \sum_j \sum_k \sum_l \hat{\alpha}_{jkl}.$$

These ideas can be extended to create a lighting function over a surface M by expressing M as a parametrically defined surface and by using the ideas presented in section 5. For a surface M which can not be easily defined parametrically, a partition of unity may be constructed directly on the surface M. The theory developed in Redner and Gehringer [Redn93] uses abstract metric spaces and is sufficiently general to adapt to almost any situation which might arise in a graphics application.

To conclude this section we observe that a B–Spline lighting function is an efficient structure for the representation of a lighting function over a surface. We see that such a function is easily constructed from random samples and that the required storage is determined by the number of basis functions and not the sample size. Evaluation of B–Splines is rapid and the manipulation and evaluation of B–Splines is well understood within the graphics community. By using quadratic or higher order splines, lighting functions are continuously differentiable and hence mach bands do not occur. The generation of random values with distribution given by a B–Spline lighting function is straightforward and is described in Appendix 1. Finally, important quantities like the power of the source and even conditional densities are also rapidly evaluated due to the structure and polynomial nature of the B–Spline lighting functions.

7 Some bidirectional ray tracing details. Bidirectional ray tracing is composed of two phases. The first phase is the light propagation step. In this step, rays are shot from the light sources. There are numerous strategies for performing this step and we will describe a method based on the delivery of random rays. This method may be preferred because it does not create classical aliasing artifacts but instead generates high frequency noise of (hopefully) low amplitude.

Given a number n and a wavelength dependent lighting function we describe a procedure for generating n random rays which properly sample the power distribution of the light source. We begin by computing the total power of the diffuse source, which is easily done if the source has the form given in equations 4 or 6. Remember that the radiosity function $\hat{l}_n(x, y, \lambda)/P$ is a density function in the sense of probability theory. For each number i from 1 to n, a random direction is determined and a random point from the density $\hat{l}_n/P$ is generated and is given power $2P\cos(\theta)/n$ (the unexpected factor of 2 is required since the average value of $\cos\theta$ over the hemisphere is only 1/2.) As well as the origin and the direction of the ray, a wavelength and a power value for the ray are kept as the ray propagates through the system.

When the ray strikes a diffusely reflecting surface or a surface which reflects at least part of its power diffusely, the fraction of the power which is reflected diffusely is deposited on the surface. These power values could be stored as histograms, but of course we use the B–Spline nonparametric density estimates to record the deposition of this power on the surface.

In order that most of the rays intersect the object or objects which have been targeted, a spherical bounding volume is created about the important objects. A cone is specified by a point source at one end and the bounding volume at the other. At this time we have implemented point sources as the primary sources of light and have used spherical bounding volumes. We only generate light rays which lie in this light cone. To further increase the sampling efficiency, stratification of the direction of light rays is used. The details of this computation are presented in Appendix 2. Stratification in wavelength sampling is also used to insure color balance and to improve efficiency. The adaptation of the cone to line and area light sources is straightforward but is slightly more expensive since the linear transformation given in the appendix must be recomputed every time a ray is generated.

The power associated with each ray depends on the number of rays, n, the aperture of the cone which is determined by an angle, ϕ_0, and the total power, P, of the source. The power associated with the ray is therefore given by

$$P_{ray} = \frac{P}{n} \, \frac{2\pi(1 - \cos(\phi_0))}{4\pi}$$

where the second factor is the fraction of the area of the sphere of radius one which lies inside the cone.

In order for this to be consistent with the traditional ray tracing portion of the program, the lighting computation in the ray tracer must account for the distance from the point on the surface to the light source. In particular, if distance is denoted by r, the factor of

$$\frac{1}{4\pi r^2}$$

must be included in the traditional ray tracing computation when a shadow ray is shot towards a light. The correctness of this approach is well known [Kaji90] but this factor is frequently ignored.

In the light propagation portion of the algorithm, two special classes of objects are recognized, objects at which light rays are targeted and objects upon which lighting functions will be built. For each target object and for each light source, rays are generated towards the target object.

Those rays which strike the objects are allowed to propagate through the scene. If they hit one of the objects upon which a lighting function is to be built, then the lighting function for that object is updated. Upon completion of the light propagation algorithm, the lighting functions are written to a small file. A traditional ray tracing program is applied to the original scene with the lighting functions added as internal lighting sources. Care must be taken so that no redundancy occurs in the accumulation of lighting information. In all of the pictures displayed in this paper, we accumulate direct lighting in the classical ray tracing portion of the program and so this information must be ignored in the light propagation phase. But it is also easy to alter this to accumulate direct lighting in the light propagation portion of the algorithm. In either case, since the forward process is independent of the view, the traditional ray tracing step can be performed from different points of view without rerunning the light propagation algorithm. Depending on the specifics of the implementation, dispersion, the focusing of lenses, color bleeding and indirect lighting can be demonstrated. These are effects which are not usually found in ray traced images, however, images of rainbows have been created by Musgrave [Musg89].

Image 1, is an assembly of the results from nine tests. Each test used the same scene, light source and viewpoint description. The only object visible is a rectangle made of a diffusely reflecting material viewed obliquely and lit from directly above by a single, D65 standard point source. Between the light and the rectangle, but out of view, is a transparent sphere. In each test a B–Spline illumination map was computed on the rectangle. The rows of images 1, 2 and 3 were created using splines of order 1, 2 and 3 (constant, linear and quadratic splines). The three columns correspond to images made with a 10 by 10, a 20 by 20 and a 30 by 30 grid of basis functions. Clearly piecewise constant splines are not adequate without interpolation and the linear structures and mach banding effects are quite clearly visible in the piecewise linear splines. The quadratic splines with a 10 by 10 grid show a surprising lack of color. This is caused by the low resolution of the red green and blue lighting functions and so the individual colors have been reconstituted into white. The final image in the lower right hand corner is a quadratic spline with a 30 by 30 grid. The image quality is good although there is some low level noise. The number of forward rays for each of the individual images was 10,000 and only one ray per pixel was used. Improved stratification of the light cone would allow us to greatly decrease the number of rays needed.

In Image 2 we have used the same geometry as in Image 1. The number of forward rays has been increased to 50,000 and the image was made using cubic splines with a 40 by 40 grid. This image is also antialiased using up to 50 rays per with a variance stopping criterion. In almost all cases, quadratic splines are adequate, but there can be slight improvement in image quality when higher order splines are used.

8 Image description. A bidirectional ray tracing program has been implemented at the University of Tulsa. Light propagation rays from light sources are wavelength dependent so that dispersion can be captured. In Image 3 we see five spheres floating over a horizontal plane. The index of refraction for the spheres is a non–constant function of wavelength. Therefore the focussed light which appears beneath each of the spheres contains color variations.

This image was created by using a 30 by 30 grid of quadratic B–Splines to represent the lighting function over the rectangle. In an ultra-conservative effort to insure image quality, 50,000 rays from each of 18 overlapping wavelength intervals was used. The final image was rendered with 512 by 512 pixels with a minimum of eight rays per pixel and a maximum of 20 rays per pixel. The exact number of rays for each pixel is determined dynamically based on the variance of the color values already computed [Lee85]. The maximum tree depth of the ray tracing stacks was set to five. Even with this exorbitant number of light rays generated in the light propagation portion of the algorithm, this portion of the algorithm was only about 5 percent of the total run time.

In Image 4 we show the effects of a prism on natural sunlight. Physical data from Wyszecki and Stiles [Wysz82] is used to describe a bright point light source at a considerable distance from the objects in the scene. A beautiful and natural looking spectrum appears on the wall in the background. In this example the light propagation algorithm ran in about one half the time of the traditional ray tracing program. There are many ways that this run time can be further reduced and these ideas are currently being explored. In Image 5 we see a blowup (magnification 8) of the spectrum on the wall. We observe that the spectrum is smooth with very few artifacts even though each pixel in this image has been replicated 64 times.

Image 6 is a more elaborate picture demonstrating that bidirectional ray tracing can be effectively used to create reasonably complex images involving dispersion and the focussing of light.

9 Conclusions and future work. We have presented work on the extension of bidirectional ray tracing to create images with prismatic effects. B–Spline density estimators have been introduced in this paper for use in the construction of realistic computer generated images. B–Spline density estimators have been shown to have the appropriate large sample properties and can be easily generated, evaluated and used in a monte-carlo sampling scheme. B–Spline density estimators have been incorporated into a bidirectional ray tracing program which can produce dispersion, the focussing of light by lenses and color bleeding.

Further areas of research include refinements to the ray propagation portion of the algorithm. Specifically, work needs to be done in making the ray propagation more efficient by the extension of the cone idea of section 7 to area light sources and by the improvement of stratification schemes.

B–Spline lighting functions currently represent only diffuse lighting functions over surface. The extension to directionally dependent distributed lighting functions needs to be investigated and the incorporation of directionally dependent distributed lights into bi–directional ray tracing needs to be pursued.

Acknowledgements The authors would like to thank Sun Microsystems, Inc. for their genererous equipment donations and the National Science Foundation for support under grant number CCR-8915693. The authors would also like to thank the referees for their many helpful comments.

References

[1] Arvo, James, "Backwards Ray Tracing", Developments in Ray Tracing, ACM SIGGRAPH 1986 Course Notes no. 12.

[2] Bartels, Richard, John Beatty, and Brian Barsky, *An Introduction to Splines for use in Computer Graphics & Geometric Modeling,* Morgan Kaufmann Publishers, 1987.

[3] Blinn, J.F. "Models of Light Reflection for Computer Synthesized Pictures", *Computer Graphics* 11,2 (July 1977), pp. 192-198.

[4] Boneva, Liliana I., David Kendall and Ivan Stefanov, "Spline Transformations: Three New Diagnostic Aids For The Statistical Data-Analysist", J. of the Roy. Stat. Soc., B, Vol. 33, No. 1 (1971).

[5] Campbell, A.T. III, and D.S. Fussell, "Adaptive Mesh Generation for Global Diffuse Illumination", *Computer Graphics* 24,4 (Aug. 1990), pp. 155-164.

[6] Chen, S. E., Rushmeier, H.E., G. Miller, and Douglass Turner, "A Progressive Multi-Pass Method for Global Illumination", *Computer Graphics* 25,4 (Aug. 1991), pp. 157-164.

[7] Cohen, M.F., and D.P. Greenberg, "The Hemi-cube: A Radiosity Solution for Complex Environments", *Computer Graphics* 19,3 (July 1985), pp. 31-40.

[8] Cook, R.L., and K.E. Torrance, "A Reflectance Model for Computer Graphics", *ACM Transactions on Graphics* 1,1 (Jan. 1982), pp. 7-24.

[9] Cook, R.L., T. Porter, and L. Carpenter, "Distributed Ray Tracing", *Computer Graphics* 18,3 (July 1984), pp. 137-145.

[10] Gehringer, Kevin, Nonparametric probability density estimation using normalized B–Splines, MS Thesis, The University of Tulsa, 1990.

[11] Gehringer, K.R. and R.A. Redner, "Nonparametric Density Estimation Using Tensor Product Splines", *Comm. in Stat.–Simula.* 21(3), (1992) pp. 849-878.

[12] Gerald, Curtis and Patrick Wheatley, *Applied Numerical Analysis,* Fourth Edition, Addison-Wesley 1989.

[13] Goral, C.M., K.E. Torrance, D.P. Greenberg and B. Battaile, "Modeling the Interaction of Light Between Diffuse Surfaces", *Computer Graphics* 18,3 (July 1984), pp. 213-222.

[14] Heckbert, Paul S., "Adaptive Radiosity Textures for Bidirectional Ray Tracing", *Computer Graphics* 24,4 (Aug. 1990), pp. 145-154.

[15] Hogg, Robert B. and Allen T. Craig *Introduction to mathematical statistics* 2d ed. New York, Macmillan, 1965.

[16] Immel, D.S., M.F. Cohen and D.P. Greenberg, "A Radiosity Method for Non-diffuse Environments", *Computer Graphics* 20,4 (Aug. 1986), pp. 133-142.

[17] Kajiya, James T., "The Rendering Equation", *Computer Graphics* 20,4 (Aug. 1986), pp. 143-150.

[18] Kajiya, James T., "Radiometry and Photometry for Computer Graphics", ACM SIGGRAPH Course Notes no. 24, (1990) pp. 3.1-3.30. .

[19] Lee, M.E., R.A. Redner and S.P. Uselton, "Statistically Optimized Sampling for Distributed Ray Tracing", *Computer Graphics* 19,3 (July 1985), pp. 61-67.

[20] Musgrave, F.K., "Prisms and Rainbows: a Dispersion Model for Computer Graphics",*Graphics Interface'89* (1989), pp. 227-234.

[21] Parzen, E., On Estimation of a Probability Density Function and Mode, *Annals of Mathematical Statistics*, Vol. 33, (1962) pp. 1065–1076.

[22] Redner R.A., and K.R. Gehringer, "Non-parametric Density Estimation Using Partitions of Unity", to be submitted.

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

[24] Silverman, B. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York (1986).

[25] Scott, David W., *Multivariate Density Estimation: Theory, Practice, and Visualization*, Wiley–Interscience, New York (1992).

[26] Thompson, J.R. and R. Tapia, *Nonparametric Function Estimation, Modelling and Simulation.* SIAM, Philadelphia, 1990.

[27] Wallace, J.R., M.F. Cohen and D.P. Greenberg, "A Two-pass Solution to the Rendering Equation: a Synthesis of Ray-tracing and Radiosity Methods", *Computer Graphics* 21,4 (July 1987), pp. 311-320.

[28] Wyszecki, G., and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* second edition, Wiley, New York, 1982.

Appendix 1.

The generation of random values from a B–Spline density. In this Appendix we describe how random values can be generated from a B–Spline lighting function based on a uniform knot spacing. We begin with the observation that, if $X_1, X_2, \ldots, X_K$ are independent uniformly distributed random values on the interval $[0,1]$, then the random value $Y = \sum_i X_i$ has as its density function the $K-th$ order B–Spline with knots at the integers $0, 1, \ldots, K$. This implies that, given a uniform knot spacing, a random value from the density function $B_k(x)/b_k$ with support on the interval $[u_k, u_{k+K}]$ is generated by computing

$$W = (u_{k+K} - u_k)Y/K - u_k.$$

Given a one dimensional density function of the form

$$\sum_k \alpha_k \frac{B_k(x)}{b_k}$$

with

$$\sum_k \alpha_k = 1,$$

a random value from this density is generated by first choosing a random class k with probability α_k. The final random value is generated by producing a random value from the density function $B_k(x)/b_k$ according to the procedure described in the first paragraph.

The extension to the generation of random rays from a lighting function of the form

$$l_n(x,y) = \sum_j \sum_k \alpha_{jk} \frac{B_j(x)B_k(y)}{b_{jk}}.$$

where

$$\alpha_{jk} = 1/n \sum_{i=1}^{n} P_i B_j(X_i) B_k(Y_i)$$

and

$$P = \sum_j \sum_k \alpha_{jk}$$

is straightforward. We normalize the lighting function by dividing by P, so that the new function is a probability density function. We choose a random pair of integers j and k with probability α_{jk}/P. Random values X and Y are then generated from the densities $B_j(x)$ and $B_k(y)$ respectively. The desired ray is the ray with origin (X,Y) with a direction randomly chosen over the hemisphere of directions and with power $\pi \cos(theta)P$.

Appendix 2.

The generation of rays within a cone Consider the mapping

$$D : [0, 2\pi] \times [-1, 1] \to \text{ the unit sphere.}$$

defined by

$$D(\theta, t) = (\sin(\theta)\sqrt{1-t^2}, \cos(\theta)\sqrt{1-t^2}, t).$$

This function is an onto mapping which is one to one on the interior of the rectangle. Since $||D_t \times D_\theta|| = 1$ this mapping is area preserving. This implies that if (θ, t) is uniform in $[0, 2\pi] \times [-1, 1]$ then $D(\theta, t)$ is a uniform random variable on the sphere.

A word of warning is in order. It is common to write the vector

$$(\sin(\theta)\sqrt{1-t^2}, \cos(\theta)\sqrt{1-t^2}, t)$$

as

$$(\sin(\theta)\sin(\phi), \cos(\theta)\sin(\phi), \cos(\phi)).$$

Points on the unit sphere can then be generated by generating θ and $t = \cos(\phi)$ uniformly on $[0, 2\pi]$ and $[-1, 1]$ respectively, with $\sin(\phi)$ defined as $\sqrt{1-\cos^2\phi}$. If, instead, you generate ϕ uniformly on $[0, 2\pi]$, you do not get a uniform distribution on the sphere. Instead you get a distribution which is too bright at both poles.

Now we consider the problem of generating random vectors whose angle with the positive z axis is less than or equal to some angle ϕ_0. If we let $z_0 = \cos(\phi_0)$ then the region $R_{\phi_0} = [0, 2\pi] \times [z_0, 1]$ is mapped onto the intersection of the sphere with the cone pointing in the direction of the positive z-axis with angle ϕ_0.

We can therefore generate direction vectors within this cone by generating vectors in the rectangle R_{ϕ_0} and applying the mapping D.

We make the interesting observation that since the area of the rectangle $[0, 2\pi] \times [z_0, 1]$ is $2\pi(1 - z_0)$ then the area of a 'cap' of the sphere, i.e. the set of all points on the sphere with z coordinate greater than or equal to some value z_0, must also be $2\pi(1-z_0)$. This is simply 2π times the thickness of the cap.

If the center of the cone of interest does not point in the direction of the positive z-axis, then the data generated by the above process must be rotated. If the direction of the cone is defined by a unit vector $U = (u_1, u_2, u_3)$ then we will want a rotation which maps the vector $(0, 0, 1)$ to U. This can be done in two stages, a rotation about the y-axis followed by a rotation about the z-axis. If we let $v_3 = \sqrt{1-u_3^2}$ then the matrices associated with these two rotations are

$$\begin{pmatrix} u_3 & 0 & v_3 \\ 0 & 1 & 0 \\ -v_3 & 0 & u_3 \end{pmatrix}$$

and

$$\frac{1}{v_3}\begin{pmatrix} u_1 & -u_2 & 0 \\ u_2 & u_1 & 0 \\ 0 & 0 & v_3 \end{pmatrix}$$

whose product (taken in the proper order of course) is

$$\frac{1}{v_3}\begin{pmatrix} u_1u_3 & -u_2 & u_1v_3 \\ u_2u_3 & u_1 & u_2v_3 \\ -v_3^2 & 0 & u_3v_3 \end{pmatrix}$$

This rotation can also be described through the use of a Householder transformation [Gera89].

If a single bounding sphere for the entire scene is constructed, then only one cone for each light source is required. If, for each light source, a collection of cones is used, then there may be overlap between the cones. When generating rays from the second cone, for example, one must test that the generated ray does not lie within the first cone. This condition can be tested by the evaluation of the dot product of the generated ray with the ray which forms the central axis of the first cone. If there is overlap then the contribution of this ray is set to zero.

Stratification of samples can be used to reduce the variance of estimates of lighting function parameters. In the generation of rays from the light source, stratification can be implemented in both the direction of the rays and their associated wavelengths. Stratification in the direction of the light rays can be accomplished by stratifying the rectangle $[0,1] \times [-\pi, \pi]$ and using the scheme described in this Appendix. A version of this has been implemented. Stratification in wavelength is also strongly recommended and is also implemented in our ray tracing system.

Image 1.

Image 2.

Image 3. Spheres with Dispersion

Image 4. Prism and a spectrum

Image 5. Magnified spectrum

Image 6. Spectra generated by a prism and a torus