View-Dependent Precomputed Light Transport using Non-Linear Gaussian Function Approximations

by

Paul Elijah Green

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

October 2005

Author ....
Department of Electrical Engineering and Computer Science
October 31, 2005

Certified by ....
Frédo Durand
Assistant Professor
r

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

View-Dependent Precomputed Light Transport using Non-Linear Gaussian Function Approximations

by

Paul Elijah Green

Submitted to the Department of Electrical Engineering and Computer Science
on October 31, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We propose a real-time method for rendering rigid objects with complex view-dependent effects under distant all-frequency lighting. Existing precomputed light transport approaches can render rich global illumination effects, but high-frequency view-dependent effects such as sharp highlights remain a challenge. We introduce a new representation of the light transport operator based on sums of Gaussians. The non-linear parameters of the representation allow for 1) arbitrary bandwidth because scale is encoded as a direct parameter; and 2) high-quality interpolation across view and mesh triangles because we interpolate the average direction of the incoming light, thereby preventing linear cross-fading artifacts. However, fitting the precomputed light transport data to this new representation requires solving a non-linear regression problem that is more involved than traditional linear and non-linear (truncation) approximation techniques. We present a new data fitting method based on optimization that includes energy terms aimed at enforcing good interpolation. We demonstrate that our method achieves high visual quality for a small storage cost and fast rendering time.

Thesis Supervisor: Frédo Durand
Title: Assistant Professor

Acknowledgments

I would like to thank Frédo Durand for advising this thesis. I would also like to thank Jan Kautz for mentoring me, and introducing me to the world of PRT. Eric Chan helped me out with his secret GPU voodoo magic. Eugene Hsu, for being an eccentric office mate, yet always willing to help. Jaakko Lehtinen provided hours of insightful conversation. I want to thank Robert Sumner for use of the best makefiles in the world. Jonathan Ragan-Kelley, Wojciech Matusik, Addy Ngan, Matthias Zwicker, Jiawen Chen, Robert Wang, and Sylvain Paris provided useful comments and reviews. Lastly, I would like to thank Jane Malcolm for her patience and encouragement. This work was supported in part by a Ford Foundation Predoctoral Fellowship.

Contents

List of Figures

Chapter 1

Introduction

The quality of real-time rendering can be dramatically improved by precomputing the interaction of an object with light. This however raises a formidable challenge, the storage and re-use of data over a six-dimensional domain: two dimensions for each of lighting (assumed to be at infinity), view direction, and surface position (See Fig. 1-1). Current methods address this issue with compromises between bandwidth (or sharpness) of the illumination effects, storage costs, and rendering time. In this thesis we focus on view-dependent effects, such as glossy reflections or refractions, since view-independent effects like shadows can be handled with known techniques [22, 28]. View-dependent effects are harder to address because they involve variation over the full 6D domain, and highlights can vary quickly over space and view direction, which may result in cross-fading artifacts when linear interpolation is used between views or mesh vertices. We introduce a method that affords arbitrary bandwidth for view-dependent effects with a compact storage and real-time performances, at the cost of a more involved preprocess.

Precomputed light transport techniques compute a pixel color at render-time based on a precomputed approximation of the light transport operator, which evaluates the contribution of all lighting directions for a given spatial location on a mesh and a given view direction. Typically, the transport operator is represented by linear-basis functions such as spherical harmonics or wavelets, and the number of coefficients

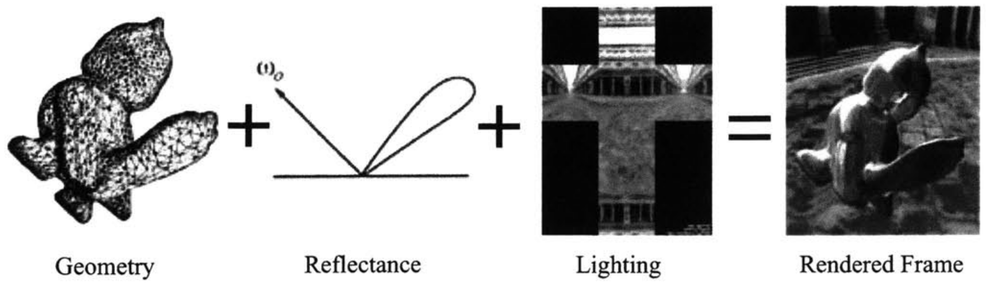


Figure 1-1: The six dimensional domain of light transport. Two dimensions for each of geometry, reflectance and incident lighting. Our technique precomputes a compact representation of this 6D domain that allows rendered frames to be produced at real-time frame rates. (Lightmaps ©Paul Debevec)

directly constrains the sharpness of the effects that can be handled. Compression techniques such as the non-linear truncation of linear basis [22] or the use of separable approximations [19, 33] improve on both storage and computation, but the number of coefficients required to handle high-frequency effects remains large and the rendering cost is directly proportional.

In our work, we address these issues and present a technique that can achieve high-quality view-dependent lighting effects at a small storage cost. We represent the transport operator using sums of Gaussian functions defined on the sphere of directions corresponding to the environment lighting (see cubemap shown in Fig. 1-1). The spherical Gaussians are parameterized by their color, mean direction, and variance. The non-linear effect of the variance parameter allows for arbitrary scaling of the Gaussian functions, thereby affording all-frequency effects. The non-linear effect of the mean direction parameter permits smooth interpolation across space (mesh triangles) and view direction; and prevents cross-fading artifacts, in the spirit of Phong interpolation. We typically use a small number of Gaussians, between one and three. Finally, the representation can be easily evaluated at run-time using a mip-mapped pre-filtered environment map where each level correspond to a Gaussian variance. However, the price to pay is a more involved preprocess, since non-linear function fitting is not as straightforward as linear projection or basis truncation. To address this challenge, we introduce an optimization technique that ensures good approximation

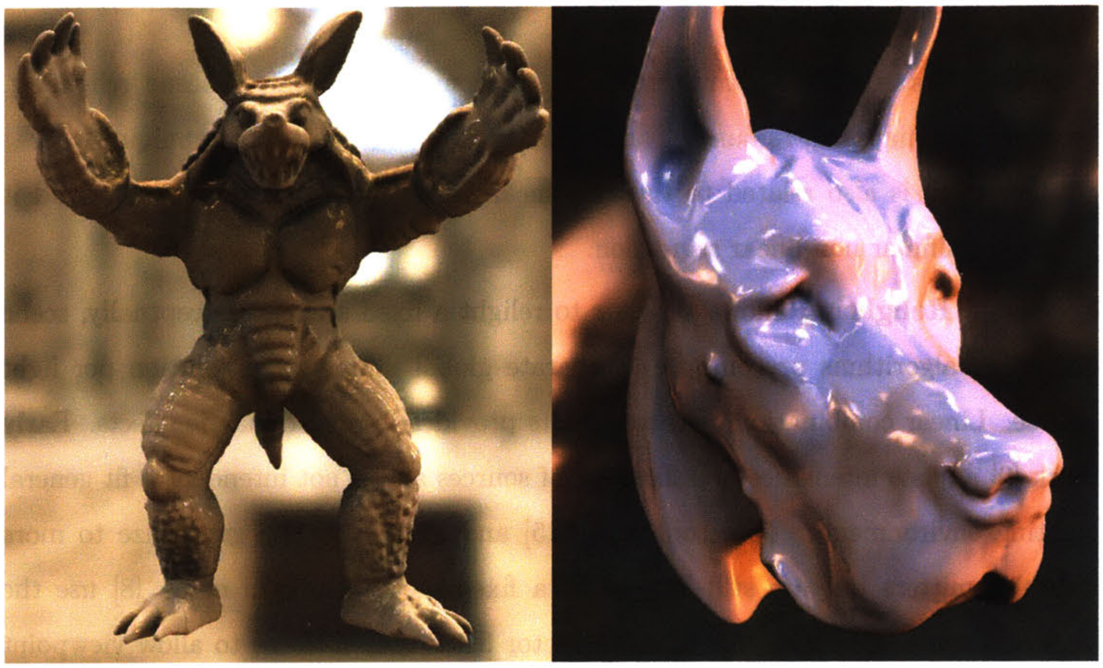


Figure 1-2: Rendering of the Armadillo model with $\sim$ 50K vertices and the Dane model with 15K vertices. Full rendering frame rate for Armadillo is $\sim$ 10Hz. Dane renders at $\sim$ 40Hz

as well as high-quality interpolation over view directions and mesh triangles. In the end, the per-pixel interpolation of the Gaussian direction and variance is critical to achieve high visual quality.

The three major contributions of this work are:

- We introduce a non-linear representation for light transport based on a sum of Gaussians that is compact and leads to a fast integration with the lighting at render time.

- Our representation affords high-quality interpolation across mesh location and view direction, which facilitates rendering from arbitrary viewpoints and requires only sparse sampling.

- We present an optimization approach to fitting Gaussian mixture models to pre-computed light transport data. In particular, we include energy terms to ensure good interpolation.

1.1 Related Work

Our work draws from a number of areas in computer graphics: scene relighting, environment mapping, radiance transfer methods, and acquisition techniques.

Our work is also related to the non-linear Lafortune BRDF model [17], and it requires a similar non-linear regression.

Scene Relighting allows the user to relight a fixed scene. Conceptually, scene relighting algorithms precompute a separate global illumination solution per light source. Linear combination of these results provides limited dynamic effects. Early work [9] adjusts intensities of a fixed set of sources and is not intended to fit general lighting environments. Nimeroff et al. [25] and Teo et al. [32] generalize to more flexible emitters but are still limited to a fixed view. Dobashi et al. [8] use the spherical harmonics basis and transfer vector fields over surfaces to allow viewpoint change but restrict lighting changes to the directional intensity distribution of an existing set of non-area light sources in diffuse scenes. Debevec et al. [6] relight faces using a directional light basis.

The concept of relighting has also been used for texture maps. Malzbender et al. [20] precompute a higher-dimensional texture, called a *polynomial texture map*, which allows real-time inter-reflection effects as well as shadowing. A similar approach using a steerable basis for directional lighting is used in [2].

None of these methods can deal with high-frequency reflections and offer only restricted view-dependent effects.

Environment Mapping. Blinn and Newell[3] proposed environment maps to approximate specular reflections. Later on, Greene [11, 10] observed that a preconvolved environment map could be used to simulate diffuse and glossy reflections. Since then, much research has been done in this area [12, 15, 27]. All these algorithms have in common that no self-shadowing or indirect lighting is taken into account.

Precomputed Radiance Transport Methods permit the relighting of an object with full spherical incident lighting. Sloan et al. [29] use the spherical harmonics basis to represent how an object casts shadows and interreflections onto itself, called *precomputed radiance transfer.* This work was limited to diffuse reflectors or to surfaces with Phong-like glossy reflections. Improvements have been presented to incorporate more complex BRDFs [16, 18], but the use of the spherical harmonics basis limits these approaches to low-frequency incident lighting (memory consumption is too high for high-frequency lighting). Even with compression [28] the lighting cannot be very high-frequency.

Ng et al. [22] proposed to use non-linear approximation (truncated) of wavelets as basis functions instead of spherical harmonics. With only about 100 coefficients, they were able to incorporate high-frequency lighting effects (for diffuse surfaces or static views only). The significant coefficients are selected based on the incident lighting, which prevents the use of highly glossy surfaces; in this case, coefficients would need to be selected based on the surface reflectance properties (transfer coefficients) and not based on the lighting, otherwise reflections will show only an approximated lighting environment. Arbitrary BRDFs can be incorporated using a method to evaluate triple product integrals [23] but with the limitation of direct lighting only. Arbitrary BRDFs and wavelets can be combined using separable BRDF approximations [19, 33]. Here the basic technique is limited to single fixed BRDFs only (precomputation is done on a per-BRDF basis; multiple BRDFs could be supported through running the precomputation for all BRDFs but at the cost of high memory consumption). Furthermore, high-frequency BRDFs can only be represented using prohibitively many terms in the separable approximation [19].

The closest technique to our work is the recent method that Sloan et al. developed to deal with local deformations [30]. They represent the local effect of light transport using a weighted sum of rotated zonal harmonics. This non-linear approximation to the transport is fitted using a greedy method combined with a local BFGS optimization step [26]. Interestingly, the main goal of their work was the ability to efficiently rotate the representation in order to handle geometric deformation, albeit in the domain of low frequency lighting and transport. In contrast, we focus on the representation of arbitrary-bandwidth view-dependent effects, which explains our choice of Gaussians functions. The arbitrary rotation of zonal harmonics is similar to the encoding of the mean direction of Gaussians in our work. Although this arbitrary rotation provides the possibility for similar interpolation gains, they do not take advantage of this in their work. Our approach also encodes scale as a non-linear parameter, while zonal harmonics represent different frequencies using a linear sum of basis functions. To further emphasize their restriction to low frequencies, we point out that their lighting-transport integration is computed in the spherical harmonic basis, which has been shown[22] to require large numbers of coefficents to represent all-frequency lighting content. Thus, fundamentally they are limited to the same class of low frequency transport functions as spherical harmonics. In fact, they require a mulit-lobe fit to accurately represent an equivalent low frequency spherical harmonic solution, and thus do not even gain compression efficiency over spherical harmonics.

The visual quality of all of the above techniques depends on the tessellation of the model; high frequency effects require a fine tessellation or cross-fading artifacts are visible. Our method lessens this requirement through non-linear interpolation of transport functions, in contrast to linear blending used by all previous PRT techniques.

Acquisition Methods deal with similar data, but instead of pre-computing transport information, they infer it from real scenes. These approaches are quite related to our work because a number of them use non-linear function approximation. We focus on such methods and omit techniques that mostly deal with the inverse problem. The initial environment matting technique [34] describes the observation in each pixel using only one 2D box function. This leads to a compact non-linear representation based on the box coordinates and weight. This algorithm works well for specular and refractive objects. Higher quality is achieved by replacing the box functions with oriented 2D Gaussian functions [4].

The above methods assume a fixed viewpoint, i.e. their light transport representations are tied to a 2D plane. A hybrid method [21] combines a forward method [6] for low-frequency components and high-quality environment matting [4] for high frequency reflections and refractions. Data is sampled from multiple viewpoints and combined with 3D geometry. Only one Gaussian is used for each type of allowed

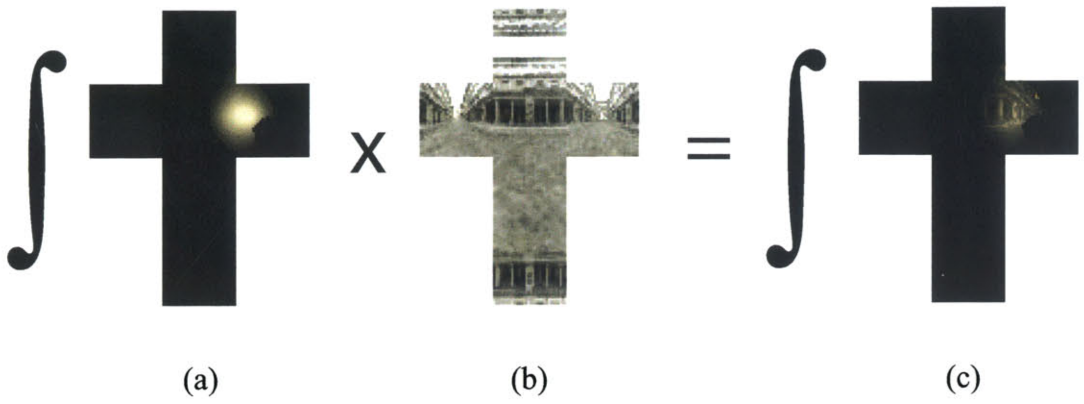


Figure 1-3: Exit radiance is computed by the integrating transport function(a) with incident lighting (b). (c) shows the pointwise multiplication of the transport function and the lighting.

transport (i.e., reflection and refraction), which is interpolated for in-between views. Rendering with this representation is done in software and does not achieve interactive rates.

1.2 Review of Light Transport

Under the assumption that our object is not emissive, it is common to view light transport as a linear integral operator [29, 22]. The light that arrives at our eye is a linear transformation of the distant incident lighting. This operator $\mathcal{T}$, which we call the light transport operator, encodes the effects of the material properties and light transport interactions between different patches of the objects surface. Thus we can describe the exit radiance $L_o(\mathbf{x}, \omega_o)$ at a point $\mathbf{x}$ (on the object) along the outgoing (viewing) direction ω_o as the result of applying our integral operator $\mathcal{T}$ on the (distant) incident lighting L_i:

$$L_o(\mathbf{x}, \omega_o) = (\mathcal{T} L_i)(\mathbf{x}, \omega_o) = \int_S T_{\mathbf{x},\omega_o}(\omega_i) L_i(\omega_i) \mathrm{d}\omega_i \tag{1.1}$$

where $T_{\mathbf{x},\omega_o}$ is the integration kernel of $\mathcal{T}$, also called a *transport function*. For fixed $\mathbf{x}$ and ω_o, $T_{\mathbf{x},\omega_o}$ is a 2-D function parameterized over the sphere S of incoming directions.

It describes, for each direction ω_i, the contribution of $L_i(\omega_i)$ to the total reflected radiance leaving $\mathbf{x}$ along ω_o. With

$$T_{\mathbf{x},\omega_o}(\omega_i) = f_r(\omega_i, \omega_o) V_{\mathbf{x}}(\omega_i)(\mathbf{n}_{\mathbf{x}} \cdot \omega_i), \tag{1.2}$$

where ω_i is the incoming direction, $\mathbf{n}_{\mathbf{x}}$ is the surface normal at $\mathbf{x}$, $V_{\mathbf{x}}$ is a binary visibility function, f_r is the bidirectional reflectance function (BRDF), and substituting for the integration kernel we can derive the familiar reflectance equation for direct lighting [14]:

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega f_r(\omega_i, \omega_o) V_{\mathbf{x}}(\omega_i)(\mathbf{n}_{\mathbf{x}} \cdot \omega_i)\ L_i(\omega_i) \mathrm{d}\omega_i \tag{1.3}$$

However, the integration kernel $T_{\mathbf{x},\omega_o}$ is not limited to direct illumination, and can describe many other complex transport effects such as interreflections, refraction, self-occlusion, caustics, subsurface scattering, and other indirect lighting effects.

Unfortunately, direct evaluation of $\mathcal{T} L_i$ (Equation 1.1) is infeasible (even assuming distant lighting, light transport is 6-D) and fundamentally all previous PRT work is concerned with approximating and compressing some slice of the light transport operator. This thesis describes new methods for computing and approximating the integration kernels $T_{\mathbf{x},\omega_o}$, as well as efficiently integrating Equation 1.1.

Chapter 2

Method

As previously noted the key issue common to all PRT technique is the representation of the light transport operator. There are several criteria used to judge the quality of any proposed representation: rendered visual quality, compactness, efficient integration/evaluation, angular and spatial frequency bandwidth.

Our new representation is based on Gaussian function approximations. We first introduce the representation for a given sample on a mesh and a given view direction. We next show how this representation affords high-quality interpolation between views and spatial locations. We discuss the challenges raised by the estimation of the model parameters and outline our optimization approach. Finally, we show how evaluation at render time reduces to simple look-ups on a pre-filtered environment map.

2.1 Light Transport Representation

We use a non-linear model [7] to represent the light transport integration kernel $T_{\mathbf{x},\omega_o}$ (see Sec. 1.2 for notation and review). In contrast to linear approximation, where all data is approximated by a projection onto a fix set of basis functions (e.g. 5th order spherical harmonics), the parameters of our model have a non-linear effect on the approximation. This is a significant difference with previous work based on non-linear approximation [22] that start from a linear basis (e.g. Haar wavelets), and truncate small coefficients. We approximate $T_{\mathbf{x},\omega_o}$ as a a sum of N weighted isotropic

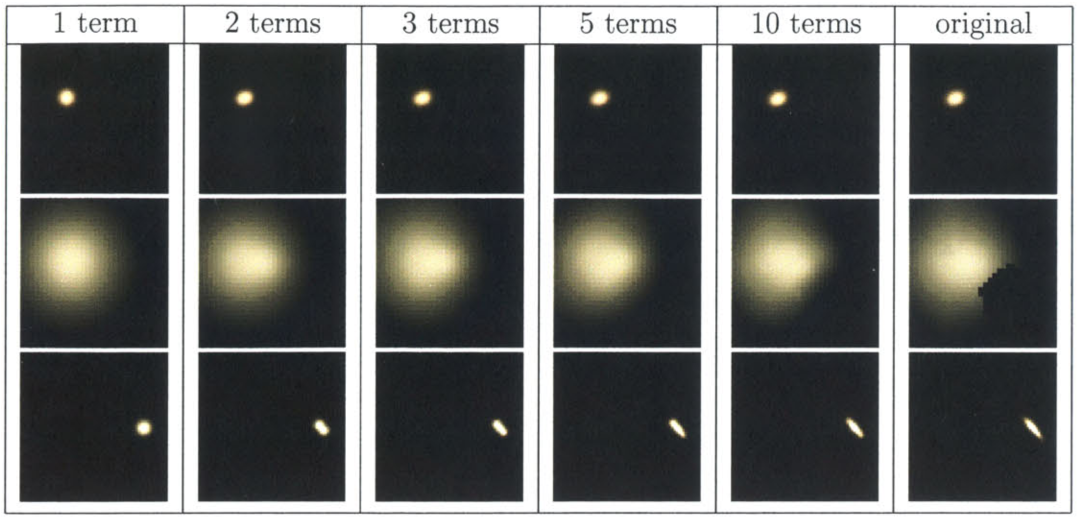


Figure 2-1: Multi-term approximations for three example transport functions. The number of terms in the approximation increases left to right in each column, from one to ten terms. The rightmost column shows the original transport data.

Gaussians:

$$\tilde{T}_{\mathbf{x},\omega_o}(\omega_i) = \sum_{k}^{N} w_k G(\omega_i, \mu_k, \sigma_k), \tag{2.1}$$

where G is a 2-D spherical Gaussian centered around μ_k with standard deviation σ_k and weight w_k .

Our choice of a non-linear model such as Gaussians is motivated by three main properties: Arbitrary bandwidth, good data approximation when using a small number of coefficients, and high visual quality of the interpolation between Gaussians. In return, they require non-linear regression for the estimation of the parameters, as described in Section 2.2. Non-linear models such as Gaussian functions have the potential to better fit data with a small number of coefficients. when their functional form is well-adapted to the problem at hand. In our case, we have chosen Gaussians because they are smooth, have a compact set of parameters, have arbitrary bandwidth, and approximate rendering data such as BRDFs well. The non-linear effect of the variance σ_k parameter allows for direct encoding of scale, thereby affording arbitrary bandwidth. The non-linear effect of the mean direction μ_k permits better interpolation and prevents cross-fading artifacts (see Fig. 2-2). However, when using

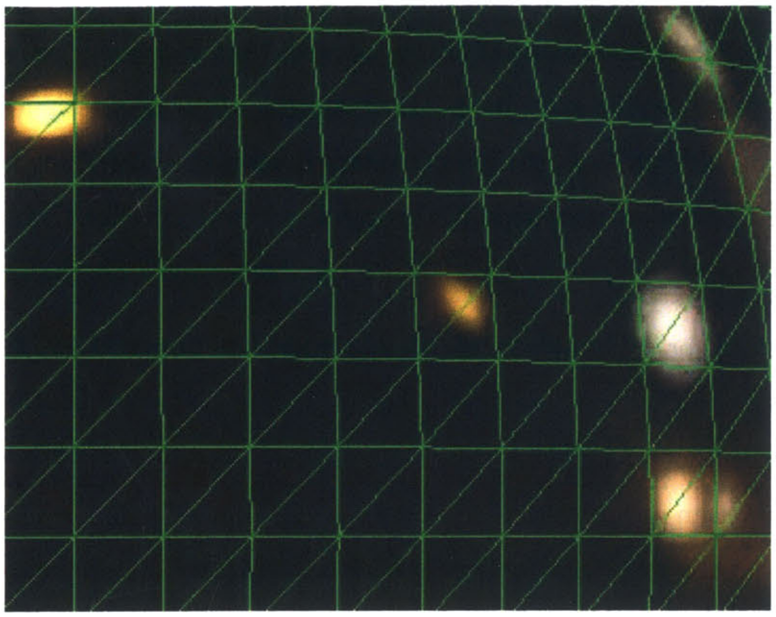


Figure 2-2: This close-up of a sphere shows that the highlights contain detail finer than the tessellation of the sphere would allow without our per-pixel interpolation of Gaussian parameters.

multiple Gaussians, good correspondence needs to be ensured for good interpolation. We discuss our solution below.

2.1.1 Sparsity & Interpolation

We have described how a Gaussian function approximation can represent the 2D light transport integration kernel $T_{\mathbf{x},\omega_o}$ at a given point $\mathbf{x}$ for a given view direction ω_o. Conveniently, it also enables a sparse representation in the remaining four dimensions for space and view direction. Key to this is the fact that interpolation of the Gaussian parameters closely matches the behavior of integration kernels. In particular, the non-linear parameters μ_k and σ_k preserve the qualitative shape of the kernel across interpolation which leads to a good visual reproduction of highlights. The quality gained compared to the linear interpolation of the highlights is similar to the difference between Gouraud and Phong shading.

In practice, we compute and approximate the transport functions $\tilde{T}_{\mathbf{x}_n,\mathbf{v}_j}$ only at vertices $\mathbf{x}_n$ and only for a set of 92 fixed viewing directions $V = \{\mathbf{v}_j\}$, obtained through subdivision of an icosahedron. Transport functions for other views or positions are not represented explicitly, instead we interpolate them from nearby samples.

2.1.2 Model Estimation

Our model requires non-linear parameter estimation for the variance, weight and mean of the Gaussians for each view direction and spatial location. When using multiple Gaussians, we must in addition ensure proper correspondence between neighboring views and spatial locations.

We introduce an optimization technique that minimize the L_2 error with the data, and that also includes energy terms that enforce correspondence between Gaussians fit for neighboring views and mesh vertices. Ideally, we would run this optimization globally over the whole mesh, but but for better scalability, we introduce a greedy approach that flood-fills over the mesh. In addition, we can separate different components of light transport such as refraction and reflection to better partition the problem. Our non-linear parameter estimation is described in Section 2.2.

Method Overview An overview of our method can be found in Figure 2-3. During the preprocessing step, we simulate light transport using any existing technique. We then fit our Gaussian model using a new optimization approach described in Section 2.2. We store Gaussian parameters for each vertex and view direction.

At runtime, rendering is straightforward. The approximated and interpolated transport function $\tilde{T}_{\mathbf{x},\mathbf{v}}$ is integrated against the lighting. This turns out to be a fast computation, since the integration of the individual Gaussians with the lighting can be precomputed and stored as a mip-mapped environment map and only a lookup is necessary at run-time. More details will follow in Section 2.3.

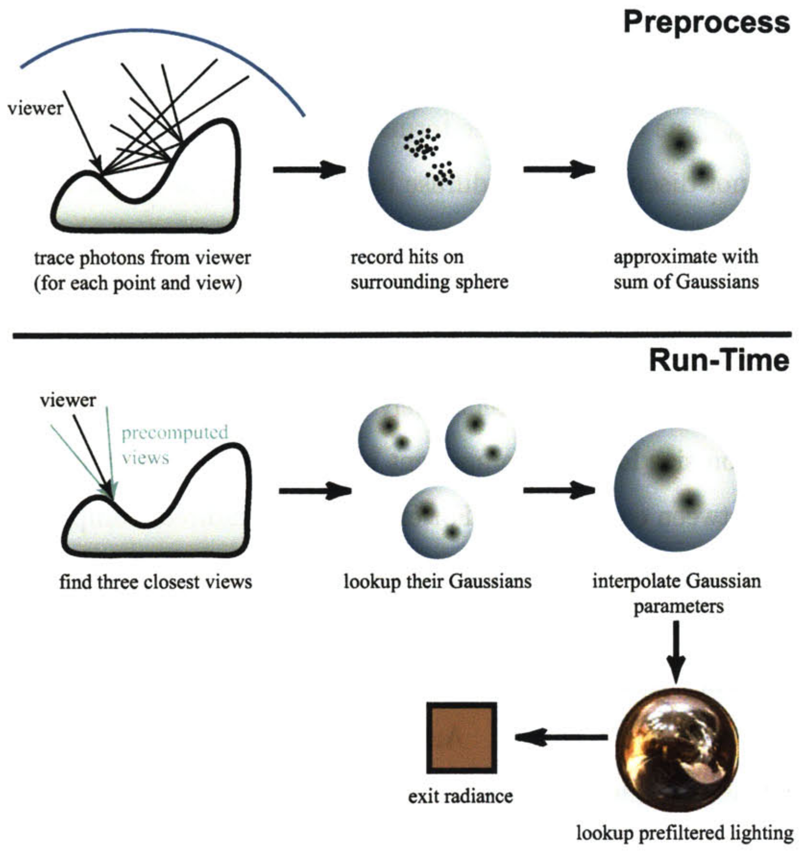


Figure 2-3: **Brief Overview**. Path tracing is one method we have used to estimate the transport function. We approximate a transport function for each view with a sum of Gaussian functions. At run-time we interpolate the parameters from the three closest views to the current view and perform a simple lookup in a Gaussian prefiltered environment map.

2.2 Precomputation

Our precomputation proceeds in two steps: we first compute light-transport data for all views and mesh vertices, and we next fit our Gaussian parameters using an optimization approach. The first step is not specific to our technique and any rendering approach could be used. Our contribution lies in the fitting of these data.

2.2.1 Light Transport Data

We compute $T_{\mathbf{x},\omega_o}$ for each view ω_o and for each vertex $\mathbf{x}$ of the model. In practice, we decompose $T_{\mathbf{x},\omega_o}$ as a sum of three terms: direct glossy, indirect glossy, and direct diffuse. We do not discuss indirect diffuse light transport as it is a view-independent effect and can be handled efficiently in the reduced 4D domain (vs. 6D for view-dependent effects). For high-quality data, we use an appropriate precomputation method for each component; a common approach in path tracing implementations [13]. Computing each component separately allows us to choose an appropriate cubemap discretization size for each component independently and without consequences or interaction between components. For example, the indirect components often contain only low frequencies, which allows us to use smaller cubemaps (e.g. 6x8x8) than the direct components, reducing storage costs and significantly accelerating precomputation time. The direct glossy and diffuse components are evaluated for each texel of cubemap in a fragment shader. An indirect glossy component is computed by BRDF importance sampled path tracing along each direction $\mathbf{v} \in V$ onto point $\mathbf{x}$ and recording the path until it is absorbed or reflected off to infinity where it is stored in corresponding cubemap texel. Each of these processes produces a set of tabulated transport operators $T_{\mathbf{x},\omega_o}$ sampled over the texels of a cubemap. These cubemaps are the input to the next stage of precomputation, and core of our technique: non-linear Gaussian parameter estimation through optimization.

2.2.2 Approximating Light Transport

The main goal of our non-linear optimization is to find the Gaussian parameters that best fit the light transport data. In addition, when using multiple Gaussians to encode the same component, we must ensure good correspondence across views and mesh vertices. In particular, note that from a data fitting perspective, any permutation of N Gaussians yields the same error. This is why we introduce additional energy terms that favor consistent correspondences.

In practice, we concurrently fit the Gaussians across all views $\omega_{o,v}$ for a given vertex

$\mathbf{x}$ using non-linear optimization [5] minimizing the following objective function:

$$\min_{\mu_k^{\omega_{o,v}}, \sigma_k^{\omega_{o,v}}} \sum_v \alpha D + \beta M + \gamma V + \delta S, \tag{2.2}$$

where $k = 1..N$ is one of the Gaussians, v is one of the 92 views, and μ_k and σ_k are the Gaussian parameters to be optimized. D is a data-fitting term, and the following terms favor correspondences for the mean and variance across views, and finally across spatial locations, as we describe below.

Data Term. The data term estimates the Gaussian parameters to approximate the given transport function. The N term approximation is defined as follows:

$$D := \sum_{\omega_i \in S} \| \left[\sum_{k=1}^{N} w_k G(\omega_i, \mu_k^{\omega_{o,v}}, \sigma_k^{\omega_{o,v}}) \right] - T_{\mathbf{x},\omega_{o,v}}(\omega_i) \|^2 \tag{2.3}$$

The data term D minimizes the L_2 error between the data T and our Gaussian function approximation $\sum_k^N G_k$.

Alignment of Means. As we want to interpolate Gaussians across views, we make sure that corresponding Gaussians in neighboring views have consistent means:

$$M := \sum_{k=1}^{N} \| \left[\frac{1}{R} \sum_{r(v)} \mu_k^{\omega_{o,r(v)}} \right] - \mu_k^{\omega_{o,v}} \|^2, \tag{2.4}$$

where $r(v)$ are the neighboring views of v. This equation says that the average mean of the Gaussians of neighboring views should be similar to the mean for view v, which encourages consistent means across views.

Alignment of Standard Deviation. This term is similar to the mean alignment term M above; it forces the standard deviation of the Gaussians in view v to be similar to the average standard deviation in neighboring views:

$$V := \sum_{k=1}^{N} \| \left[\frac{1}{R} \sum_{r(v)} \sigma_k^{\omega_{o,r(v)}} \right] - \sigma_k^{\omega_{o,v}} \|^2 \tag{2.5}$$

Both of the alignment terms M, V are crucial for smoothly interpolating parameters across views. In situations where visibility has caused views to have zero transport data (and consequently zero weights w_k), the alignment terms help interpolation by estimating μ_k and σ_k parameters that smoothly interpolate the neighboring parameters.

Spatial Term. The spatial term ensures that the Gaussians of the current vertex (for all views) have good corresponding Gaussians in (already processed) neighboring vertices. Good correspondences means that the Gaussians for a vertex $\mathbf{x}$ are arranged in a similar manner in neighboring vertices, as this will allow spatial interpolation of Gaussians. We force this correspondence by making sure the relative arrangement of Gaussians is consistent:

$$S := \sum_{k_1=1}^{N-1} \sum_{k_2=k_1}^{N} \sum_{s(\mathbf{x})} \max\left((-\langle \mu_{\mathbf{x},k_1}^{\omega_{o,v}} - \mu_{\mathbf{x},k_2}^{\omega_{o,v}}, \mu_{s(\mathbf{x}),k_1}^{\omega_{o,v}} - \mu_{s(\mathbf{x}),k_2}^{\omega_{o,v}} \rangle)^3, 0 \right) \tag{2.6}$$

where $\mathbf{x}$ is the current vertex and $s(\mathbf{x})$ are the neighboring vertices that have already been processed. A positive dot product between the difference vectors of Gaussian means μ in neighboring vertices says the Gaussians are arranged in a similar manner, whereas a negative product says that the Gaussians are flipped in some way.

Optimizing for the Gaussian parameters of all views and all vertices at the same time is impossible due to the very large number of variables. Instead, we only concurrently fit all parameters and all view at a single vertex. To make sure that spatial consistency is attained nonetheless, we proceed as follows. We start with a single vertex and optimize all its Gaussian parameters. After all its parameters have been optimized, we process a neighboring vertex using the above spatial consistency term. We continue processing neighboring vertices until all vertices have been processed (which essentially "flood-fills" the mesh).

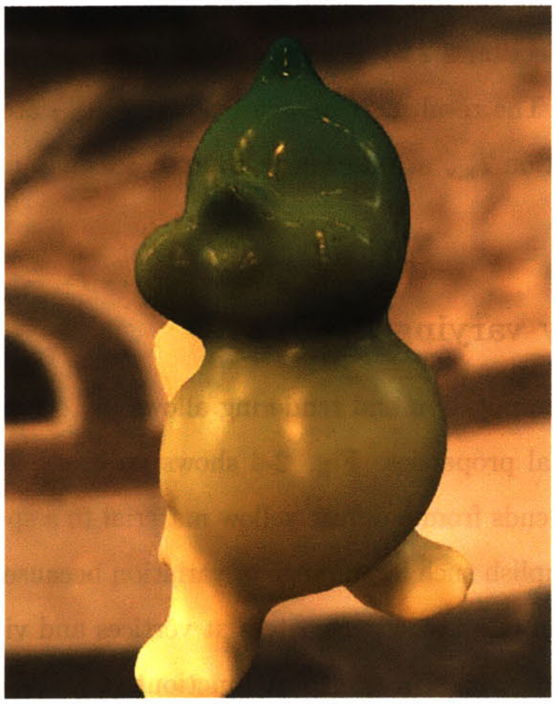


Figure 2-4: A bird model with spatially varying material properties. The material transforms from a diffuse yellow material at the feet to a highly specular green material at the head. The tessellation of the model is low (~12K vertices), nevertheless the reflections contain high frequencies due to our novel interpolation scheme.

2.3 Rendering

Rendering an object involves integrating the lighting $L_i(\omega_i)$ with the (interpolated) transport functions $\tilde{T}_{\mathbf{x},\mathbf{v}}(\omega_i)$, see Equation 1.1. This integration has to be repeated for every visible point $\mathbf{x}$.

$$L_o(\mathbf{x}, \omega_o) = \int_S \left(\sum_k^N w_k G(\omega_i, \mu_k, \sigma_k) \right) L_i(\omega_i) \mathrm{d}\omega_i \tag{2.7}$$

We use well-known techniques to evaluate this integral with simple look-ups to Gaussian pre-filtered environment maps [1].

The interpolation of the transport function is performed using barycentric interpolation of the Gaussian parameters. The non-linear effect of these parameters ensures high-quality rendering. At each of the three closest spatial samples $\mathbf{x}_{1,2,3}$ (triangle vertices), we lookup the Gaussians for the three closest directions $\mathbf{v}_{1,2,3}$. Corresponding Gaussians (see Section 2.2.2) at each vertex $\mathbf{x}_l$ are interpolated [31] using the barycentric weights. The resulting Gaussians at each $\mathbf{x}_{1,2,3}$ are then interpolated to yield the actual function $\tilde{T}_{\mathbf{x},\mathbf{v}}$ at position $\mathbf{x}$. Section 3.2 gives practical details or our implementation.

2.3.1 Spatially varying BRDFs

Our method of precomputation and rendering allows us to render objects with spatially varying material properties. Fig. 2-4 shows an example of the tweety model with a BRDF that blends from a diffuse yellow material to a specular green material. We are able to accomplish such large range of variation because our representation of the transport function (Gaussian parameters at vertices and views) does not rely on any factorization of the space of transport functions (e.g. PCA or clustered PCA). Instead, our parameters fully, and independently encode the transport for a given vertex and view, and thus can change arbitrarily over the mesh. In addition, our functions can have arbitrary bandwidth, consequently we are able to capture a full range of transport effects, from diffuse to highly specular, on the same model.

Chapter 3

Implementation and Evaluation

In this sections we describe our implementation and evaluate our results from a numerical, visual, and performance perspective.

3.1 Precomputation

As noted earlier, it is necessary to compute the transport function at each vertex and for a set of fixed view directions, specified in a global coordinate frame. In practice we used 92 viewing directions, with on average 46 directions falling in the upper hemisphere of any given vertex. The view directions are defined in a global coordinate frame to simplify angular interpolation.

We decompose the transport operator into a diffuse (view-independent) component and a glossy (view-dependent) component (see Fig 3-1). We represent the diffuse component using 5th order linear spherical harmonics, however any other suitable technique could easily be used (e.g. [22, 28]).

We optimize the objective function described in equation 2.2 and section 2.2.2. In practice we use the MATLAB optimization toolbox routine *fmincon*. As non-linear optimization is a compute intensive procedure, we used a cluster of 20 machines to distribute the work. Typical precomputation times ranged from one to three hours. Note that, in our implementation, transport functions are computed and

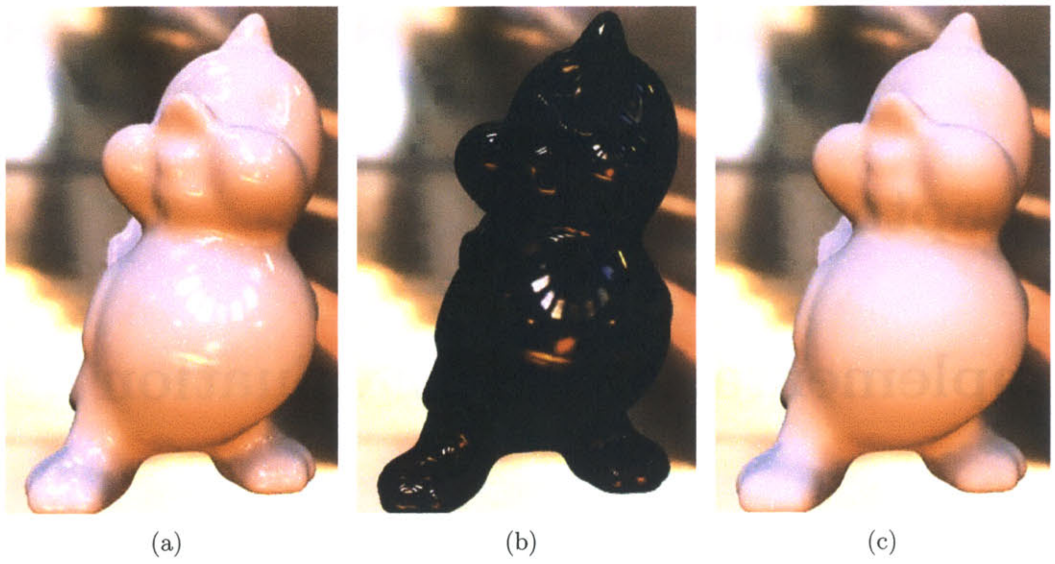


Figure 3-1: Decomposition of a (a) rendered frame into the (b) glossy component and the (c) diffuse component.

approximated at the vertices of a mesh. It is possible to do the same computation on a per-texel basis for texture mapped objects. This may be useful when one wants to disassociate the sampling of geometry and light transport.

Data-Fitting Optimization Details We now describe the details of the implementation of the non-linear optimization used to solve the data-fitting term(Eqn. 2.3). The task is to find the set of parameters w_k, μ_k, and σ_k that comprise the best N-Gaussian least-squares approximation of the transport data T. We repeat Eqn. 2.3, omitting all references to vertex ($\mathbf{x}$) or outgoing view ($\omega_{o,v}$):

$$D = \sum_{\omega_i \in S} \| \left[\sum_{k=1}^{N} w_k G(\omega_i, \mu_k, \sigma_k) \right] - T(\omega_i) \|^2 \tag{3.1}$$

Where G is a 2-d Gaussian function defined over the sphere S:

$$G(\omega, \mu, \sigma) = \frac{1}{2\pi\sigma^2} e^{(-[M_\mu(\omega) - M_\mu(\mu)]^T [M_\mu(\omega) - M_\mu(\mu)]/2\sigma^2)} \tag{3.2}$$

As mentioned in section 1.2, for a fixed vertex and outgoing direction, T is a 2-d spherical function which we have chosen to parameterize over the faces of a cubemap (see Fig. 2-1). A direct consequence is that we must define a function M_μ that maps the mean parameters μ_k from 3-d into the cubemap parameterization. In addition, M_μ defines a distance metric between arbitrary directions on the sphere. We defined M_μ to unwrap a cubemap onto the cartesian plane with the origin lying at the center of the face (of the cubemap) that contains point μ. Appendix A contains code for the map function M_μ.

Accelerating Data-Fitting. Eqn. 3.1 can be evaluated in $O(|S| \times N)$ time, where $|S|$ is the discretization size of the cubemap (e.g. 6x64x64) and N is the number of Gaussian functions used in the approximation. Unfortunately, non-linear optimization is an iterative procedure, and our particular optimization often required several hundred iterations to converge. Therefore we designed several input-sensitive algorithms to evaluate Eqn. 3.1. Both algorithms rely on factoring Eqn. 3.1.

Algorithm 1 Let $S_\emptyset = \{\omega_i \in S | T(\omega_i) = 0\}$ be the set of directions that have a zero transport function value and $S_\exists = \{\omega_i \in S | T(\omega_i) \neq 0\}$ be the set of directions with nonzero transport, s.t. $S = S_\emptyset + S_\exists$. For brevity, let $G_k(\omega_i) = w_k G(\omega_i, \mu_k, \sigma_k)$ and $F(\omega_i) = \sum_{k=1}^{N} G_k(\omega_i)$. Using this notation, we can separate the sum in Eqn. 3.1 into two subsums over $S_\emptyset$ and $S_\exists$ respectively:

$$\begin{aligned}
D &= \sum_{\omega_i \in S_\emptyset} \|F(\omega_i) - T(\omega_i)\|^2 + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 && (3.3)\\
&= \sum_{\omega_i \in S_\emptyset} \|F(\omega_i)\|^2 + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 && (3.4)\\
&= \sum_{\omega_i \in S} \|F(\omega_i)\|^2 - \sum_{\omega_i \in S_\exists} \|F(\omega_i)\|^2 + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 && (3.5)\\
&= \sum_{\omega_i \in S} \|F(\omega_i)\|^2 + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 - \|F(\omega_i)\|^2 && (3.6)
\end{aligned}$$

We have factored D into a summation over the entire domain S and a summation over the set of directions $S_\exists$ with nonzero transport function T.

$$\begin{aligned}
\sum_{\omega_i \in S} \|F(\omega_i)\|^2 &= \sum_{\omega_i \in S} \|\sum_{k=1}^{N} G_k(\omega_i)\|^2 && (3.7)\\
&= \sum_{\omega_i \in S} \sum_{k=1}^{N} \sum_{j=1}^{N} G_k(\omega_i) G_j(\omega_i) && (3.8)\\
&= \sum_{k=1}^{N} \sum_{j=1}^{N} \sum_{\omega_i \in S} G_k(\omega_i) G_j(\omega_i) && (3.9)\\
&= \sum_{k=1}^{N} \sum_{j=1}^{N} I_{kj} && (3.10)
\end{aligned}$$

Where I_{kj} is a closed-form solution of $\sum_{\omega_i \in S} G_k(\omega_i) G_j(\omega_i)$, computed by analytic integration (with appropriate weighting for the giving cubemap discretization size). Combining everything we arrive at this form of the data fitting term:

$$D = \sum_{k=1}^{N} \sum_{j=1}^{N} I_{kj} + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 - \|F(\omega_i)\|^2 \qquad (3.11)$$

which can be implemented in an $O(N^2 + N|S_\exists|)$ time procedure.

Algorithm 2 We designed and implemented a second algorithm to evaluate the data-fitting term (Eqn. 3.1). The idea of this algorithm is to again factor D,

$$D = \sum_{\omega_i \in S_\emptyset} \|F(\omega_i)\|^2 + \sum_{\omega_i \in S_\exists} \|F(\omega_i) - T(\omega_i)\|^2 \qquad (3.12)$$

and to directly evaluate the sum over $S_\emptyset$, however at a coarser discretization level than the input cubemap. The general idea is to build a quadtree subdivision of each face of the cubemap. A node of the quadtree is subdivided until the pixels of the cubemap covered by the node contain entirely zero or entirely nonzero values. The nonzero nodes comprise the set $S_\exists$, while the zero nodes make up the set $S_\emptyset$. Eqn. 3.12 is evaluated over the respective sets, with directions in $S_\emptyset$ being weighted by their relative area with respect to the original cubemap discretization size. Algorithm 1 was less numerically stable than Algorithm 2, and thus in practice Algorithm 2 was preferred, despite any bounds on running time.

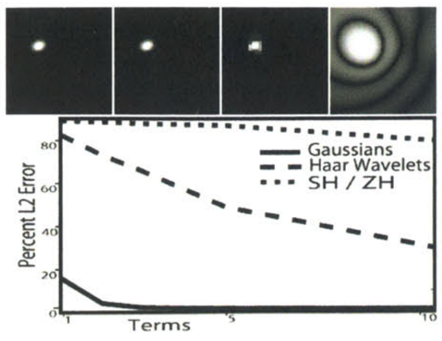
3.2 Rendering

```
// Vertex Program
void main(
   float4   position : POSITION,
   float4   mean_var : TEXCOORD0,
   float3   wt       : TEXCOORD1,

   uniform float    cubeMapSize,
   uniform float4x4 modelViewProj,
   uniform float3x3 lightMatrix,

   out   float4   oPos       : POSITION,
   out   float4   omean_var  : TEXCOORD0,
   out   float3   owt        : TEXCOORD1 )
{
   float3 m; float v;
   oPos = mul(modelViewProj,position);

   // rotate mean dir with lighting
   m = mul(lightMatrix, mean_var.xyz );

   // find the mip-map level
   v = log2(max(mean_var.w*cubeMapSize,1));

   owt = wt;
   //pack the mean and var into one texunit
   omean_var = float4(m.x,m.y,m.z,v);
}
```

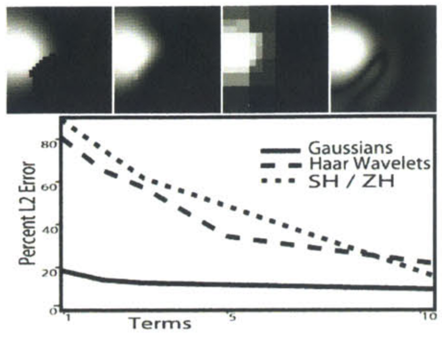
```
// Fragment Program
float3 main(
   float4   mean_var : TEXCOORD0,
   float3   wt       : TEXCOORD1,

   uniform samplerCUBE envmap )
     : COLOR
{
  // do a lod texture lookup
  float4 C = texCUBElod( envmap, mean_var );

  // return weighted result
  return (C.xyz * wt);
}
```

Figure 3-2: Vertex and fragment shader code.

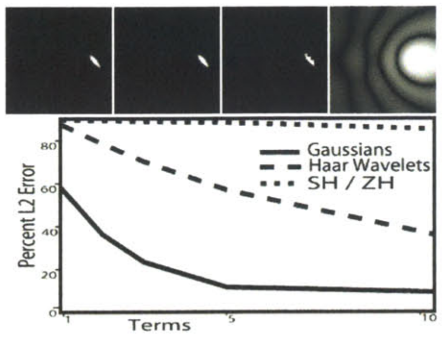
Rendering is simple, with the majority of the computation occurring in vertex and fragment shaders. In practice, we have implemented the interpolation across view in a vertex shader, and the interpolation across triangles in a pixel shader (Fig. 3-2 gives sample shader code). The integral of the interpolated Gaussians with the environment map is a simple mip-map look up to a pre-filtered environment cube. The per-pixel interpolation of the Gaussian direction and variance is critical to achieve high visual quality. The CPU is only used to locate the data for the three nearest views of each vertex and pack it into vertex buffer objects to be sent to the GPU. In fact, even this stage could be accomplished on the GPU by adding several texture look-ups to the vertex shaders. The main bottleneck of our implementation is the data transfer from the CPU to the GPU. Despite the unoptimized rendering code the glossy rendering code still achieves frame rates of greater than 60Hz for modest sized models (10-30K vertices) and the combined spherical harmonics and glossy component can be rendered at 30-40Hz.



(a)



(b)



(c)

Figure 3-3: Visual and error analysis of three typical transport functions and their approximations. Top row of each subfigure shows from left to right: un-approximated data, Gaussian Approximations using 10 Gaussian lobes, non-linear wavelet approximation using the 70 largest coefficients, and 10th order Spherical Harmonic approximation. Graphs shows percentage L^2 error as a function of number of Gaussians and the equivalent number of wavelet and spherical harmonic coefficients.

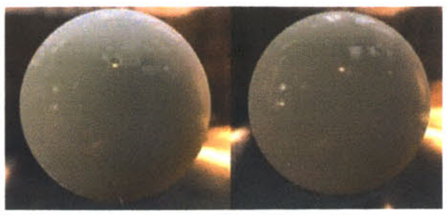
3.3 Error Analysis and Comparisons

We demonstrate the expressive power of our model by calculating the relative L^2 error of approximated transport functions using our Gaussian representations, non-linear Haar wavelet approximation[22] and spherical harmonics(SH). We choose not to compare directly with zonal harmonics(ZH)[30], as ZH are a subset of SH. One might argue that the non-linear fitting of ZH used in [30] may add more expressive power to ZH, however it should be noted that ZH are band-limited in the same way as SH. Thus, high frequencies require higher-order ZH, which is prohibitively expensive at rendering time.

Figure 3-3 shows relative L^2 error for three examples: A small symmetric glossy lobe, a large lobe with visibility, and a highly anisotropic, highly specular lobe. A fair comparison of representations as different as non-linear Gaussians and truncated Haar wavelet is difficult. We have chosen to take into account the number of coefficients used in the representation. In practice, our Gaussians are represented using seven floating point values: three for RGB weights, three for a mean, and one for variance (see equation 2.1). We therefore decided to compare a one-Gaussian approximation with seven-coefficient truncated Haar and spherical harmonic approximations. Note that this is far from an optimal encoding of Gaussians, but it was chosen for simplicity. In addition, Haar encoding might require more RGB coefficients. The situation for zonal harmonics is a bit more complicated as they have the ability for multi-lobe approximations. However, since zonal harmonics are bandlimited, we feel it is sufficient (in terms of accuracy of the approximation) to compare only with spherical harmonics. As such, the number of parameters was determined based on a standard SH representation. Despite this generous allotment of coefficients, it is clear from Fig. 3-3 that the Gaussian representation is superior both in terms of L^2 error and smoothness for such a small number of coefficients.

3.4 Visual quality

Next we demonstrate the visual accuracy and quality of our model. Fig.3-4 shows renderings of the *alum-bronze* and *acrylic-white* acquired materials[24] rendered using our technique and ray-traced. The BRDF model used in the precomputation stage was the same as that used to render the ray-traced images. Results are shown using a single Gaussian lobe, rendered at frame rates above 60Hz.


(a)

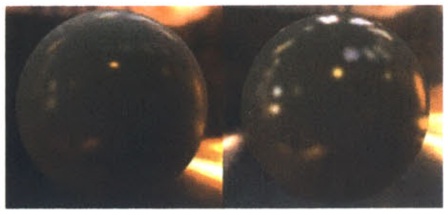

(b)

Figure 3-4: Comparison of our method(left) and ray-traced image(right) for two different BRDFs. Images were rendered using a single Gaussian for the glossy component and 5th order spherical harmonics for the diffuse component. Reference Images courtesy A. Ngan [24].

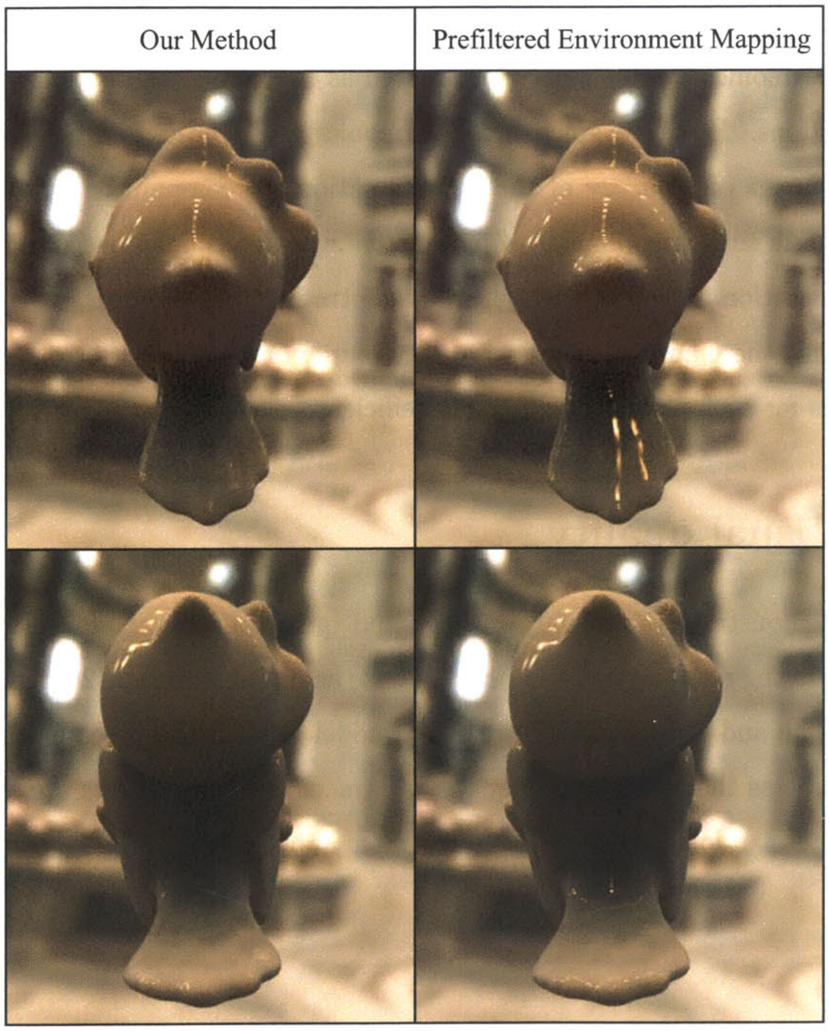


Figure 3-5: Comparison of our method and prefiltered environment mapping [12, 15, 27] for two different views of the tweety model. Environment mapping can produce strong highlights in shadowed regions because it ignores visibility, while our method correctly clamps shadowed highlights.

The most important aspect of any rendering technique is the quality of the final images. The non-linear effect of the Gaussian mean direction and variance allow for high-quality interpolation across view and triangle. The gain is similar to that of Phong interpolation, but we additionally encode visibility and indirect effects. Figure 2-2 demonstrates that fine highlight details are captured using a sparse sampling in both the view directions and mesh location.

In order to test the visual effect of including visibility in the precomputation stage of our method, we choose to compare images rendered with our method to images rendered by a prefilitered enviroment mapping [12, 15, 27] style technique that ignores visibility (shading is dependent on surface normal orientation only). Fig. 3-5 shows two sample frames. As is to be expected, removing visibility causes cases where highlights appear in shadowed regions of the object. In contrast, our method correctly removes the shadowed highlights.

In our experience, one Gaussian per transport component is usually enough to obtain high subjective quality. While the numerical gain is significant for the following Gaussians, the visual gain is not as high, probably partially because the visual complexity makes it harder to assess the complex reflection patterns. For practical purposes, we advocate the partition of light transport into multiple components when possible. The coherence terms in the optimization perform well, but we have sometimes observed Gaussian flipping, especially at the end of the mesh flood fill when neighboring vertices favor incompatible constraints. We believe that an additional relaxation step can improve results.

Chapter 4

Conclusions

This thesis has presented a real-time method for rendering static objects under distant all-frequency lighting, which captures view-dependent effects at a small storage cost. Our method is based on a non-linear representation of light transport based on Gaussian function approximations. This representation presents several advantages. It is compact and thus requires only modest storage and memory requirements. Our method can capture arbitrary bandwidth transport effects which allows us to render models with spatially varying material properties, from very diffuse to very specular. Unique to our representation is the ability for high-quality interpolation of spatial and angular content, drastically reducing the tessellation requirements of the input geometry. We have presented an optimization approach to fit the Gaussian parameters to light transport data. We have taken care to accurately model transport functions while simultaneously achieving good correspondences across view direction and mesh location. The rendering phase of our technique involves simple shaders and it achieves high quality and high performance.

In the future, we would like to extend our system to dynamic scenes based on a similar representation. Furthermore, we are interested in removing the restriction to distant lighting, which raises critical challenges to address the resulting high-dimensional space.

Chapter 5

Appendix A

Code for map function M_μ defined in Section 3.1.

```
typedef enum CUBEMAPFACE {X_POS=0, X_NEG, Y_POS, Y_NEG, Z_POS, Z_NEG} CUBEMAPFACE;
static float FACE_SIGN[] = {1.f,-1.f,1.f,-1.f,1.f,-1.f};
static unsigned int FACE_COORD[] = {0,0,1,1,2,2};


// Map function - maps a 3d point 'in' onto face 'face'
void Map( const Vec3f &in,
          Vec2f &out,
          const CUBEMAPFACE face )
{
  CUBEMAPFACE a = FindCubeFace(in);
  Vec3f temp = in/fabs(in[FACE_COORD[a]]);

  float v = FACE_SIGN[a]*FACE_SIGN[face];
  float t = v*temp[FACE_COORD[a]];
  temp[FACE_COORD[a]] = (-v*temp[FACE_COORD[face]]) + (2*FACE_SIGN[a]);
  temp[FACE_COORD[face]] = t;

  switch (face)
  {
    case X_POS:
    case X_NEG:
      out[0] = temp[2]; //z
      out[1] = temp[1]; //y
      break;
    case Y_POS:
    case Y_NEG:
      out[0] = temp[0]; //x
      out[1] = temp[2]; //z
      break;
    case Z_POS:
    case Z_NEG:
      out[0] = temp[0]; //x
      out[1] = temp[1]; //y
      break;
  }
}
```

```
// find the face of the cubemap that
// the 3d point 'v' project onto
CUBEMAPFACE FindCubeFace(const Vec3f & v)
{
  Vec3f t = v;
  t.normalize();
  Vec3f a = t.abs();
  float m = a.max();
  int i=0;
  for(;i<3;++i)
    if(a[i] == m)
      break;

  return (CUBEMAPFACE) ( 2*i + (t[i]<0?1:0) );
}
```

Bibliography

[1] M. Ashikhmin and At Ghosh. Simple blurry reflections with environment maps. *Journal of Graphics Tools*, 7(4):3–8, 2002.

[2] M. Ashikhmin and P. Shirley. Steerable illumination textures. *ACM Transactions on Graphics*, 21(1):1–19, January 2002.

[3] J. Blinn and M. Newell. Texture and reflection in computer generated images. *Communications of the ACM (SIGGRAPH '76 Proceedings)*, 19(10):542–547, October 1976.

[4] Y.-Y. Chuang, D. Zongker, J. Hindorff, B. Curless, D. Salesin, and R. Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 121–130, 2000.

[5] T. Coleman and Y. Li. An Interior, Trust Region Approach for Nonlinear Minimization Subject to Bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[6] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 145–156, July 2000.

[7] R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

[8] Y. Dobashi, K. Kaneda, H. Nakatani, and H. Yamashita. A Quick Rendering Method Using Basis Functions for Interactive Lighting Design. *Computer Graphics Forum*, 14(3):229–240, August 1995.

[9] J. Dorsey, F. Sillion, and D. Greenberg. Design and Simulation of Opera Lighting and Projection Effects. In *Proceedings SIGGRAPH*, pages 41–50, Las Vegas, Nevada, July 1991.

[10] N. Greene. Applications of World Projections. In *GI*, pages 108–114, May 1986.

[11] N. Greene. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics & Applications*, 6(11):21–29, November 1986.

[12] W. Heidrich and H.-P. Seidel. Realistic, hardware-accelerated shading and lighting. In *Computer Graphics*, SIGGRAPH 99 Proceedings, pages 171–178, Los Angeles, CA, August 1999.

[13] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping.* AK Peters, 2001. ISBN: 1568811470.

[14] J. Kajiya. "the rendering equation". In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20, pages 143–150, August 1986.

[15] J. Kautz. Hardware Lighting and Shading: A Survey. *Computers Graphics Forum*, 23(1):85–112, March 2004.

[16] J. Kautz, P.-P. Sloan, and J. Snyder. Fast, Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical Harmonics. In *13th Eurographics Workshop on Rendering*, pages 301–308, June 2002.

[17] E. Lafortune, S.-C. Foo, K. Torrance, and D. Greenberg. Non-linear approximation of reflectance functions. *Computer Graphics*, 31(Annual Conference Series):117–126, 1997.

[18] Jaakko Lehtinen and Jan Kautz. Matrix Radiance Transfer. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 59–64, 2003.

[19] X. Liu, P.-P. Sloan, H.-Y. Shum, and J. Snyder. All-Frequency Precomputed Radiance Transfer for Glossy Objects. In *Proceedings Eurographics Symposium on Rendering 2004*, pages 337–344, june 2004.

[20] T. Malzbender, D. Gelb, and H. Wolters. Polynomial texture maps. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 519–528, Los Angeles, CA, 2001.

[21] W. Matusik, H. Pfister, R. Ziegler, A. Ngan, and L. McMillan. Acquisition and rendering of transparent and refractive objects. In *Rendering Techniques '02*. Springer, Wien, 2002.

[22] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation. *ACM Transactions on Graphics*, 22(3):376–381, July 2003.

[23] R. Ng, R. Ramamoorthi, and P. Hanrahan. Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics*, 23(3):477–487, August 2004.

[24] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226. Eurographics Association, 2005.

[25] J. Nimeroff, E. Simoncelli, and J. Dorsey. Efficient Re-rendering of Naturally Illuminated Environments. In *Fifth Eurographics Workshop on Rendering*, pages 359–373, June 1994.

[26] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, 1992.

[27] R. Ramamoorthi and P. Hanrahan. Frequency Space Environment Map Rendering. In *Proceedings SIGGRAPH*, pages 517–526, July 2002.

[28] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered Principal Components for Precomputed Radiance Transfer. *ACM Transactions on Graphics*, 22(3):382–391, July 2003.

[29] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Transaction on Graphics*, 21(3):527–536, July 2002.

[30] P.-P. Sloan, B. Luna, and J. Snyder. Local, Deformable Precomputed Radiance Transfer. *ACM Transactions Graphics*, 24(3), August 2005.

[31] P. Tan, S. Lin, L. Quan, B. Guo, and H.-Y. Shum. Multiresolution Reflectance Filtering. In *Proceedings Eurographics Symposium on Rendering 2005*, pages 111–116, 2005.

[32] P. Teo, E. Simoncelli, and D. Heeger. Efficient Linear Re-rendering for Interactive Lighting Design. Technical Report STAN-CS-TN-97-60, Stanford University, 1997.

[33] R. Wang, J. Tran, and D. Luebke. All-Frequency Relighting of Non-Diffuse Objects using Separable BRDF Approximation. In *Proceedings Eurographics Symposium on Rendering 2004*, pages 345–354, june 2004.

[34] D. Zongker, D. Werner, B. Curless, and D. Salesin. Environment matting and compositing. In *Computer Graphics*, SIGGRAPH 99 Proceedings, pages 205–214, August 1999.

[illegible]