Bidirectional Lightcuts

Bruce Walter Pramook Khungurn Kavita Bala
Cornell University*

Figure 1: *This kitchen scene combines many different phenomena including glossy surfaces, subsurface BSSRDFs (e.g., milk and dragon), heterogeneous smoke, a highly detailed anisotropic volumetric cloth model (over billion voxel effective resolution, see bottom right), skylight through three windows and 25 local lights. Computing global illumination in such a scene is extremely challenging and standard VPL methods cannot capture many of the perceptually important illumination details. Our bidirectional method extends VPL-based techniques to handle a wider range of such phenomena (top row). A bidirectional path traced result of equal time is extremely noisy (see zoom ins) while bidirectional lightcuts maintains the low noise and scalability advantages of VPL-based methods. A probabilistic progressive photon map image (bottom left) of equal time shows visible noise (e.g., from glossy paths) and bias around small features (e.g., very thin cloth, <5mm).*

Abstract

Scenes modeling the real-world combine a wide variety of phenomena including glossy materials, detailed heterogeneous anisotropic media, subsurface scattering, and complex illumination. Predictive rendering of such scenes is difficult; unbiased algorithms are typically too slow or too noisy. Virtual point light (VPL) based algorithms produce low noise results across a wide range of performance/accuracy tradeoffs, from interactive rendering to high quality offline rendering, but their bias means that locally important illumination features may be missing.

We introduce a bidirectional formulation and a set of weighting strategies to significantly reduce the bias in VPL-based rendering algorithms. Our approach, *bidirectional lightcuts*, maintains the scalability and low noise global illumination advantages of prior VPL-based work, while significantly extending their generality to support a wider range of important materials and visual cues. We demonstrate scalable, efficient, and low noise rendering of scenes with highly complex materials including gloss, BSSRDFs, and anisotropic volumetric models.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism;

Keywords: bidirectional ray tracing, global illumination

Links: DL PDF

*email: bjw@graphics.cornell.edu, {pramook,kb}@cs.cornell.edu

1 Introduction

Advances in the quality and performance of rendering algorithms have seen an increasing use of accurate rendering algorithms in real-world applications where *predictive* rendering is needed. Example applications are product design and visualization, furniture

showrooms, and apparel visualization. The output of these rendering algorithms must match closely real world material appearance to be useful to the designers using them.

These applications include large and complex representations of shape, material and lighting, thus stressing existing rendering algorithms beyond the range at which they operate efficiently and robustly. For example, a scene of a modern interior may include fabrics such as drapes, sofas, clothes, where volumetric representations represent yarns at micron resolution; polished metals in appliances like refrigerators and faucets; wood in furniture and floors; complex lighting fixtures; and building materials like granite and marble, characterized by subsurface scattering. Computing the visually important effects in such scenes is challenging and requires unified support for light transport with high gloss reflections, subsurface scattering, volumetric scattering in anisotropic and isotropic media, and complex light sources.

For such scenes, designers are faced with unsatisfactory choices. They can choose unbiased Monte Carlo algorithms such as path tracing and Metropolis light transport, that correctly capture all phenomena, but practically often produce noisy images even with very large computation times (requiring hours or days on large clusters [Zhao et al. 2011]). On the other hand, designers can pick from a range of biased algorithms, that produce low noise results, but are often limited in the phenomena they can handle within a reasonable computational budget. In this work we focus on virtual point light (VPL)-based methods because they span a wide range of accuracy vs. cost tradeoffs while producing low noise results. But [Křivánek et al. 2010] show that material perception is negatively impacted by approximations made by VPL-based rendering algorithms. The key issue that drives the practical applicability of a rendering algorithm is the tradeoff between error and efficiency; a practical algorithm must find the right tradeoff between these competing goals.

In this paper we present a scalable, bidirectional extension of VPL-based approaches, *bidirectional lightcuts* (BDLC), that expands their range of reproducible material effects while keeping the key advantage of VPL methods: the efficient generation of low noise images. We extend the generation of virtual sensor points (VPS) using recursive eye tracing and use weighting functions to combine and optimize the contributions from the points. We find that standard unbiased weighting heuristics do not work well in this case, and instead propose a novel path weighting strategy based on four constraints to control the bias vs. cost tradeoffs. Our strategy has much lower bias than standard VPL rendering, but without the higher noise levels of strictly unbiased methods. Because our eye and light paths have different amortized cost, the weights are also designed to reduce cost. Our system uses an extension of the scalable multidimensional lightcuts algorithm to efficiently handle the increased number of VPS points. We also use an augmented path formulation that integrates surface, volumetric, and subsurface materials in one unified system.

This paper makes the following contributions:

- A general bidirectional VPL formulation for low-noise rendering with reduced bias that supports a wide range of real-world materials such as highly glossy materials, complex volumetric models like cloth, and subsurface scattering.
- Bidirectional VPL rendering combined with scalable illumination algorithms to efficiently reuse light subpaths and mitigate the combinatorial performance issues common in bidirectional approaches.
- Novel weighting strategies for handling paths that balance the noise vs. bias tradeoff for efficient rendering.

The advantages of our approach are that:

- It keeps the strengths of biased VPL algorithms (efficiency and scalability) while providing results that more closely match the generality of unbiased algorithms.
- The formulation and weighting strategies are general and independent of any particular material model.
- We preserve prior scalable rendering (multidimensional lightcuts) advantages such as unified handling of all illumination and effects, including direct, indirect, environment illumination, antialiasing, depth-of-field, and motion-blur.

We show results for a broad range of general materials in complex models with global illumination, such as shown in Figure 1.

2 Previous Work and Review

Unbiased Monte Carlo rendering algorithms such as bidirectional path tracing [Lafortune and Willems 1993; Veach and Guibas 1994] and Metropolis light transport [Veach and Guibas 1997], are generally considered the gold standard for reference solutions in complex scenes. By randomly sampling transport paths, in theory they can handle any scene. In practice however, complex scenes often contain at least some important paths that are hard for a particular algorithm to find. If such paths exist, unbiased algorithms produce noisy results that are slow to converge. Many variants exist that improve the finding of various path types, but no algorithm has yet been demonstrated to efficiently find all possible important paths.

In many practical applications, biased algorithms may be preferred as the bias error is often less visually objectionable than noise. Two of the most commonly used classes of biased algorithms are photon mapping and VPL-based techniques.

Photon mapping [Jensen 2001] is a powerful technique that handles all types of illumination and is especially good for difficult caustics on diffuse surfaces and in low density media [Jarosz et al. 2011]. Photon mapping traces many light paths to deposit photons in the scene. A blurring kernel (density estimation) is used to reconstruct the illumination from the photon density, where the minimum resolvable features are limited by the local density. Modern variants such as [Hachisuka and Jensen 2009; Knaus and Zwicker 2011], use multi-pass to avoid memory constraints so that the achievable photon density is limited only by the computational budget. Rendering cost is strongly correlated to the ratio between scene size and the minimum illumination feature size reconstructed. Thus photon mapping works best for small scenes or large features, but the rendering cost can be large to reconstruct small features inside a large scene. For example, the extremely detailed (micron resolution), high frequency, volumetric cloth models used in some of our examples can be problematic for photon mapping approaches.

VPL based methods, such as instant radiosity [Keller 1997], also trace light paths, but deposit virtual point lights rather than photons. Global illumination is estimated by computing the direct illumination from these VPLs instead of density estimation. VPL methods can reconstruct detailed illumination using far fewer light paths than photon mapping, but the cost of evaluating each VPL is significantly higher. Scalable methods such as [Walter et al. 2005; Walter et al. 2006; Hašan et al. 2007; Ou and Pellacini 2011] greatly reduce this cost by adaptively evaluating only a small fraction of the VPLs to accurately estimate the illumination. VPL methods tend to be more efficient than photon mapping in larger scenes with detailed illumination, but with the tradeoff that they cannot handle some types of illumination phenomena, such as caustics, that photon mapping can. Our goal in this paper is to formulate a new bidirectional VPL method that handles a wider variety of phenomena than standard VPL methods while retaining their efficiency.

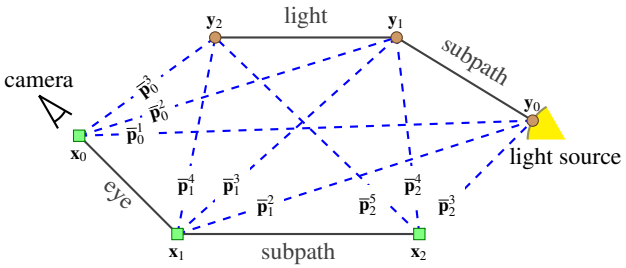


Figure 2: *Paths generated by generators used in bidirectional path tracing. Here, $\overline{\mathbf{p}}_k^\ell$ denotes the path with ℓ segments where the connection is made at the eye subpath vertex $\mathbf{x}_k$.*

2.1 Bidirectional Theory Background

The global illumination problem can be formulated as an integral over the paths, $\overline{\mathbf{p}}$, by which light can travel from the light sources to detectors (camera or eye) as: $\int f(\overline{\mathbf{p}})$, where $f(\overline{\mathbf{p}})$ describes the path's contribution to a pixel. Monte Carlo algorithms estimate this integral by generating paths with some probability $p()$. Finding good path generators with reliably low noise in all cases is hard.

Bidirectional path tracing [Veach and Guibas 1995] aims to combine the strengths of multiple different path generators within a single algorithm. The algorithm uses the random process of tracing partial paths starting from the lights and eye, called light and eye subpaths respectively. Complete paths can be formed by adding segments connecting a vertex on the eye subpath to a vertex on the light subpath as in Figure 2. We let $\overline{\mathbf{p}}_k^\ell$ denote the path of length ℓ (i.e., containing ℓ segments) generated by connecting eye subpath vertex $\mathbf{x}_k$ with light subpath vertex $\mathbf{y}_{\ell-k-1}$.

The (ℓ,k)-*generator* uses the above random process to generate paths $\overline{\mathbf{p}}_k^\ell$. Its contribution is given by the random variable:

$$X_k^\ell = w_k^\ell(\overline{\mathbf{p}}_k^\ell)\frac{f(\overline{\mathbf{p}}_k^\ell)}{p_k^\ell(\overline{\mathbf{p}}_k^\ell)} \tag{1}$$

where $p_k^\ell(\overline{\mathbf{p}}_k^\ell)$ is the probability density of generating $\overline{\mathbf{p}}_k^\ell$ via the (ℓ,k)-generator. The function $w_k^\ell()$ is the generator's *weighting function*. By convention, if a generator cannot generate a path, then its weight for that path is zero. Monte Carlo integration tells us that the expected value of X_k^ℓ is $E[X_k^\ell] = \int w_k^\ell(\overline{\mathbf{p}})f(\overline{\mathbf{p}})\,d\overline{\mathbf{p}}$.

The total estimate X is the sum of the contribution of all possible (ℓ,k)-generators: $X = \sum_{\ell=0}^{\infty}\sum_{k=0}^{\ell} X_k^\ell$ and its expected value is:

$$\begin{aligned} E[X] &= \sum_{\ell=1}^{\infty}\sum_{k=0}^{\ell} E[X_k^\ell] = \sum_{\ell=1}^{\infty}\sum_{k=0}^{\ell}\int w_k^\ell(\overline{\mathbf{p}})f(\overline{\mathbf{p}})\,d\overline{\mathbf{p}} \\ &= \int\left(\sum_{\ell=1}^{\infty}\sum_{k=0}^{\ell} w_k^\ell(\overline{\mathbf{p}})\right)f(\overline{\mathbf{p}})\,d\overline{\mathbf{p}} \end{aligned} \tag{2}$$

If the weights sum to one for all possible paths, $\sum_{\ell,k} w_k^\ell(\overline{\mathbf{p}}) = 1$, then this is an unbiased estimator, $E[X] = \int f(\overline{\mathbf{p}})\,d\overline{\mathbf{p}}$. Otherwise the estimator is biased and the bias attributable to a particular path $\overline{\mathbf{p}}$ depends on one minus its summed weights and can be expressed as: $(1-\sum_{\ell,k} w_k^\ell(\overline{\mathbf{p}}))f(\overline{\mathbf{p}})$.

The choice of the weighting functions strongly affects the efficiency and variance of the estimator. Ideally, the most efficient generator for any path should be given the highest weight for that path. Also, skipping the generation of zero weight paths can reduce cost. Standard bidirectional path tracing uses unbiased weighting heuristics such as "balanced" and "power" [Veach and Guibas 1995] based on the reasonable assumption that generators that have the highest probability of sampling a path have low noise. However, these heuristics still suffer from noise and cost issues because:

1. There are often paths for which none of the bidirectional estimators have a sufficiently low variance, especially in scenes with highly glossy and complex materials, and thus high sampling rates are often still needed.

2. Due to the weighting heuristics used and lack of 'a priori' knowledge, all possible generators for a path have to be checked, even the high variance ones, increasing the cost for paths that have many potential generators.

In VPL-based methods, the light tracing for VPL generation is equivalent to the light subpaths in bidirectional path tracing, and the sensor (a.k.a. gather) point generation is equivalent to the eye subpaths but usually limited to length 1. Thus VPL-based rendering can be viewed as a type of bidirectional method that sets $w_k^\ell(\overline{\mathbf{p}}) = 0$ if $k \neq 1$. Moreover their bias, such as from clamping, can be expressed as constraints on the path weights that cause them to sum to less than one for some paths.

VPL-based rendering as a variant of bidirectional tracing was first recognized by [Kollig and Keller 2004], who proposed an unbiased VPL-based algorithm by extending the eye subpath generation. This is equivalent to including additional path generators with weights to compensate for the bias from clamping whenever their summed weights is less than one. They demonstrated that their algorithm was up to 25% faster than standard bidirectional path tracing in two test scenes. However, their focus was on being unbiased, thus they have the same fundamental noise convergence problems as standard bidirectional path tracing in scenes with glossy or complex materials. Simplified approximate versions of this approach have been developed for interactive applications using screen space [Novák et al. 2011] and volumetric [Engelhardt et al. 2010] approximations, but these are not suitable for high complexity models such as our kitchen scene with high frequency volumetric cloth.

[Segovia et al. 2006] used bidirectional tracing during VPL generation process to try to improve the distribution and locations of the VPLs. [Hašan et al. 2009] used modified VPLs (called VSLs) to approximate glossy reflection while [Davidovič et al. 2010] use recursive eye tracing to generate additional "local" VPLs in glossy regions. These VPLs can nicely reproduce broad glossy reflections but their visibility approximations make then unsuitable for high frequency regions such as volumetric cloth. Unlike these methods, we do not change the VPL generation process, but instead generate additional virtual point sensors (VPS) to better capture a range of phenomena including glossy reflections, subsurface, and volumes.

The system of [Dammertz et al. 2010] combines standard VPL-based, density estimation, and path tracing rendering algorithms. They partition path space by preclassifying materials as diffuse or specular/glossy, and assign each path to the most appropriate method. We believe that decomposing path space and combining multiple algorithms is the right approach, but in this work we concentrate on extending the set of paths and materials that can be efficiently handled by a VPL-based method. Improved hybrid methods to solve for the remaining missing energy is left for future work.

3 Our Approach

Our method uses the bidirectional framework and deeper eye subpaths to extend the range of materials that VPL-based rendering can efficiently support. As shown in Figure 3, standard VPL methods only trace paths of length one from the camera when creating vir-

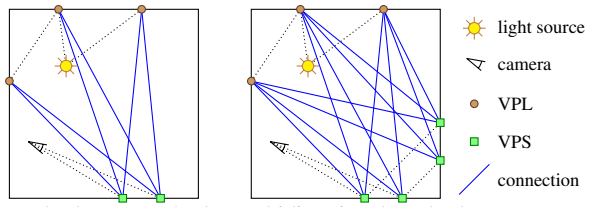


Figure 3: *Standard VPL methods use recursive tracing only for VPL generation but not VPS. Our method uses recursive tracing for both, creating many more VPSs and potential connections.*

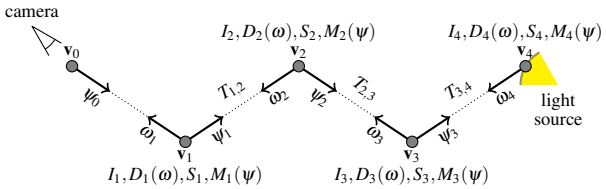


Figure 4: *A complete path from the camera to a light source. The path's components are annotated with associated quantities. Note that* $\mathbf{v}_0$ *does not have VPS/VPL attributes associated with it as we never generate a VPS on the camera's aperture.*

tual sensor points (VPS). Using recursive eye tracing, our method generates more VPSs and more potential connections to the VPLs. There are now multiple ways to generate the same path, which has both benefits and potential problems. Segments with high contribution are subject to clamping, which introduces bias; for example, for segments connecting a VPS and VPL that are too close together, or segments with large BRDFs due to glossy reflections. Standard VPL methods have no way to remove this bias except by using more VPLs to reduce clamping. However, our method can often reduce the bias by including energy from additional VPSs.

One potential drawback of recursive VPS generation is that it may introduce noise especially if the VPSs are spread over a large region. Since low noise is one of the main advantages of VPL-based methods, this would be undesirable. We estimate the spread of the eye paths and stop using the VPSs (and terminate their eye subpaths) if they become too spread out. This process is encoded in our weighting heuristics that will be described in Section 4.

The first pass of our algorithm generates the VPLs using exactly the same methods as in standard VPL methods (i.e., stochastic light tracing and russian roulette). To compute each pixel, the second pass generates a set of VPSs using stochastic eye tracing, which is very similar the VPL generation process. We allow the eye paths to recurse through multiple bounces. However, we also try to proactively terminate the recursion once it starts producing useless VPSs (i.e., ones whose weights are guaranteed to be zero). Once all a pixel's VPSs are generated, we compute the pixel's value by estimating the weighted sum of all VPS/VPL pairs. Since there are commonly 100+ million pairs per pixel, we use the scalable multidimensional lightcuts algorithm to accurately estimate this sum while only actually evaluating a small fraction of the pairs.

The largest change between our method and prior VPL methods is the combination of extended VPS generation and new weighting strategies to decide how much each VPL/VPS pair contributes. Our strategy consists of four constraints, each with a specific purpose:

1. The first constraint favors using shorter eye paths whenever possible. VPLs are shared across all pixels and thus have lower amortized generation cost than VPSs. Thus we prefer to use shorter eye paths and fewer VPSs when possible.

2. The second constraint implements the clamping that is required by all VPL methods to control noise from the VPL generation. We show how to define this consistently for a broad range of materials including high gloss, subsurface scattering, and volumes.

3. The third constraint limits the directionality of the generated VPLs. Highly directional VPLs are less useful because they affect few pixels and are highly likely to be clamped, while they increase evaluation cost, especially in scalable rendering algorithms.

4. The last constraint is the most sophisticated; it introduces a bias in the difficult case where none of the path generators have low noise. This excludes paths that would otherwise be likely to cause visible noise based on the current rendering settings. This is very important for complex materials as shown in Figure 8.

Although we use multidimensional lightcuts because it naturally supports large numbers of both VPS and VPL points, our weighting strategies can also be used with other VPL methods. If far fewer VPLs are used though, some parameter adjustments may be needed.

3.1 Bidirectional VPL Formulation

To express our weighting strategies, we need to formulate the path integral in terms of quantities used in VPL-based rendering. A VPL can be characterized by its position, an intensity I and a directionality function $D(\omega)$ which describes its emission as a function of direction ω. The intensity is determined by the history of its path back to the light source, while the directionality depends on the type of scattering at its location (e.g., BRDF for surface reflection or phase function for volumetric scattering) and the direction from which it arrived. Similarly, a VPS, can characterized by its position, a strength S based on its eye subpath, and a material term $M(\psi)$ which describes its relative sensitivity to light arriving from various directions, ψ. We also define a transit term T to account for factors related to the connecting segment between a VPS-VPL pair including distance, visibility, and attenuation. More details on how these terms are defined for various cases and materials is given in Appendix A. The unweighted contribution of a VPS-VPL pair is just a product of these terms: $SM(\psi)\,T\,D(\omega)\,I$.

Suppose we connect a VPS to a VPL to form a complete part $\bar{\mathbf{p}}$ of length ℓ, and the connecting VPS is the kth vertex ($k \geq 1$) from the camera. In order to compute its weighted contribution, we need to compute its weight $w_k^\ell(\bar{\mathbf{p}})$. To do so, we introduce some notation. We number the vertices on this path starting from the camera ($\mathbf{v}_0$) and increasing towards the light as shown in Figure 4. For any j, we let ψ_j denote the direction from $\mathbf{v}_j$ to vertex $\mathbf{v}_{j+1}$, and let ω_j denote the direction from $\mathbf{v}_j$ to $\mathbf{v}_{j-1}$.

For the weighting computations, we consider every vertex on the path as both a potential VPS and VPL location and define the strength S, material M, intensity I, and directionality D terms at every vertex. Although $\mathbf{v}_k$ and $\mathbf{v}_{k+1}$ correspond to the actual VPS and VPL pair used to generate the path, when computing weights we also need to consider the possibility that we could have generated the same path using a different pair. For example if $k = 2$ then $\mathbf{v}_2$ was the actual VPS we selected, but we could also have generated the same path by connecting a VPS at $\mathbf{v}_1$ to a VPL at $\mathbf{v}_2$. The quantities are designed so that the weighted contribution from any segment is:

$$w_j^\ell(\bar{\mathbf{p}})\frac{f_j^\ell(\bar{\mathbf{p}})}{p_j^\ell(\bar{\mathbf{p}})} = w_j^\ell(\bar{\mathbf{p}})S_jM_j(\psi_j)T_{j,j+1}D_{j+1}(\omega_{j+1})I_{j+1} \qquad (3)$$

for any $1 \leq j < \ell$ (including $j = k$ which is the one we actually use).

To reduce clutter, we will omit the directional parameters to the material M_j and the directionality D_j terms in later equations as their values can be inferred from context. We will also drop the path parameter $\bar{\mathbf{p}}$ and its length ℓ as they are both constant and not needed within the computation of the weight. Henceforth we will denote $w_j^{\ell}(\bar{\mathbf{p}})$ more simply as just w_j.

4 Four Weight Constraints

Given a path $\bar{\mathbf{p}}$, created by connecting a VPS, at the kth vertex from the camera, to a VPL, we need to compute the weight w_k that is needed to compute the path's contribution. In our weighting strategy the weight for a segment depends on the weights for the prior segments, so we first compute w_1, then w_2, then w_3, and so on until we reach w_k. (Our algorithm always set $w_0 = 0$.) Each w_j is computed based on four weighting constraints $(w_j^{\mathrm{E}}, w_j^{\mathrm{C}}, w_j^{\mathrm{D}}, w_j^{\mathrm{V}})$ that limit the maximum value of w_j. Once we have computed the four constraints, the actual weight w_j is the minimum of all the constraints, but cannot be less than zero:

$$w_j = \max(0, \min(w_j^{\mathrm{E}}, w_j^{\mathrm{C}}, w_j^{\mathrm{D}}, w_j^{\mathrm{V}})). \tag{4}$$

Before describing the constraints, we note that the computation of w_j depends only on previous weights and the quantities stored in $\mathbf{v}_j$ and $\mathbf{v}_{j+1}$. Since we only compute the weights w_1 through w_k, we do not need any information about vertices beyond the actual VPL location, $\mathbf{v}_{k+1}$. As such, VPLs do not need to store their path history, and our system can treat all VPLs identically regardless of how they were generated. On the other hand, some information about the eye path must be stored with each VPS. Our four weighting constraints are described below.

(1) Energy conservation and favoring shorter eye paths

$$w_j^{\mathrm{E}} = 1 - \sum_{i=1}^{j-1} w_i \tag{5}$$

This constraint has two goals: (1) to ensure the sum of the connection weights for a path cannot exceed one, and (2) if given a choice, to always assign the non-zero weights to the segments closest to the eye. If the summed weights over all segments for a path is one, then we have an unbiased estimator for that path. Other constraints may cause the sum to be less than one (negative bias) but it will never exceed one (positive bias or excess energy).

In our system, VPLs are shared across all pixels while the VPSs must be regenerated for each pixel. Thus segments on the light path have a lowered amortized cost in our system than eye segments, and it is more cost effective to use connecting segments closer to the eye when possible. By assigning zero weights to later eye segments, we can often terminate eye subpaths earlier (see pruning discussion later in this section) and reduce the cost of VPS generation.

(2) Clamping

$$w_j^{\mathrm{C}} = \min\left(\frac{C_{\mathrm{clamp}}}{M_j T_{j,j+1} D_{j+1}},\ 1\right) \tag{6}$$

Clamping is required in all VPL-based methods to ensure that the influence of any single VPL is never large enough to be visible. Otherwise noise from the stochastic VPL generation method could cause objectionable visual artifacts in the image. Early VPL methods used a clamping value based purely on distance but that does not handle glossy materials. We use a clamping formulation that is functionally identical to the clamping in [Walter et al. 2006] because it works across a wide range of materials and is widely used in VPL rendering.

The constant C_{clamp} is chosen based on the average VPL intensity I_{avg}. By design the light particle tracing should produce VPLs which all have very similar intensities. Currently we use $C_{\mathrm{clamp}} = L_a/(800 I_{\mathrm{avg}})$ where L_a is the estimated adaptation luminance for the scene. This is a stricter clamping threshold than used in most prior work, but is needed for high gloss and dynamic scenes. Static diffuse scenes can often tolerate higher values for C_{clamp} as the resulting noise is typically low frequency and less noticeable.

(3) Diffuse VPLs

$$w_j^{\mathrm{D}} = \min\left(\frac{C_{\mathrm{D}}}{D_{j+1}},\ 1\right) \tag{7}$$

This constraint's goal is to limit the illumination from a VPL to only include the diffuse components of a material. Most VPL-based algorithm include restrictions (sometimes implicitly) on the allowed directionality of VPLs. The most common is restricting them to only account for the Lambertian component of a material. However we want to support a wider range of diffuse materials. Instead of assuming any particular material model, our constraint limits the maximum effective value of the VPLs directionality term, D to C_{D}. The advantage of this formulation is that it works equally well with non-Lambertian diffuse material models, such as the fiber phase function in our volumetric cloth examples.

Directionality terms D are normalized to integrate to approximately one over all directions. The directionality term for a pure Lambertian material is always $\leq 1/\pi$ while for glossy materials the maximum can easily be in the hundreds or higher. By setting C_{D} to a suitable intermediate value (we use $C_{\mathrm{D}} = 1$), we can remove most of the glossy components while preserving the diffuse components of arbitrary materials in D.

Restricting the VPLs to be diffuse, while not strictly necessary, has advantages. Glossy, highly directional, VPLs are often ineffective as they only illuminate a very small region and are very likely to be clamped anyway due to their large D values. They also make the VPLs appear less similar, complicating the light clustering and bounding used in our scalable solver and reducing its performance.

(4) Exclusion of high-variance eye subpaths

$$w_j^{\mathrm{V}} = \mu_j - \sum_{i=1}^{j-1} w_i \tag{8}$$

This constraint aims to eliminate high noise estimators by introducing bias if necessary. It is similar in form to the energy conservation constraint (Equation 5) except that if $\mu_j \leq 1$ it can force the maximum weight assigned to a path to be less than one (causing negative bias). Before we can define μ_j, we will need to introduce a few new quantities, but the intuition is that there exist paths for which none of our bidirectional estimators is very efficient and that could introduce objectionable noise if included. In such cases, accepting some negative bias is often preferable to the alternatives of producing noisy images or greatly increased rendering cost. Reducing μ_j on problematic paths, such as caustics, is how we achieve this goal. The importance of this constraint is demonstrated in Figure 8.

If the early eye segments have low weights, typically due to clamping, the recursive VPS generation can effectively degenerate into a form of path tracing with its noise problems, especially if the eye paths become spread across a large area and broad range of directions. To avoid this we use heuristics to estimate the spread of the recursive eye tracing, and exclude their contribution if they spread too far apart. However we do allow energy even from deeper recursion in some cases such as for sharp glossy reflections and for short range diffuse effects.

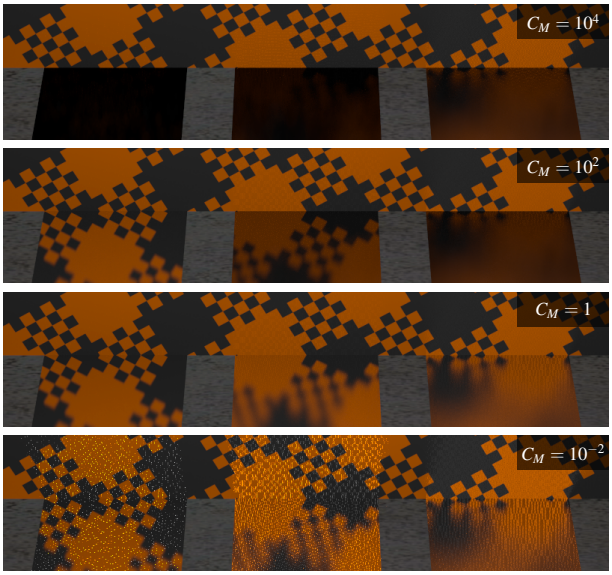


Figure 5: *The effect of different values of our C_M parameter. This scene contains three surfaces with decreasing gloss sharpness from left to right. Setting C_M too high (e.g., top) excludes energy even from sharp reflections, while setting it too low (e.g., bottom) results in objectionable noise from caustics and other problematic paths. Our results use the somewhat conservative value of $C_M = 10^2$.*

A directional spread term Θ is used to estimate the directional spread of the recursive eye subpaths. This term sums over each vertex from the eye and is computed as:

$$\Theta_j = \sum_{i=1}^{j-1} \sqrt{\frac{1}{\pi M_i}} \quad \text{with} \quad \Theta_0 = \Theta_1 = 0 \tag{9}$$

This term can be intuitively explained as follows: since the material term is normalized to integrate to approximately one over the sphere of directions, it cannot exceed some value a over a solid angle more than roughly $1/a$ steradians. For a single bounce, restricting ourselves to directions ψ where $M(\psi) \geq a$ would ensure we only recurse for directions within a solid angle less than $1/a$. To estimate the spread potentially through multiple bounces, we estimate the angle of a cone that would subtend the same solid angle and then sum the angles over each subsequent scattering. For small angles the corresponding cone angle is $\Theta = \sqrt{1/(\pi a)}$, where setting $a = M$ results in our heuristic.

Next we introduce a term Γ to define two behaviors depending on how large the estimated directional spread Θ is. If the spread is sufficiently small then we allow unrestricted recursive eye tracing by setting $\Gamma = \infty$. However if the spread is too large (for example after scattering from a diffuse surface), then we set Γ to a small value that strongly restricts further recursive eye tracing. Specifically we set Γ_j as follows based on these two cases:

$$\Gamma_j = \begin{cases} \infty & \text{if } \Theta_j \leq \sqrt{\frac{1}{\pi C_{\mathrm{M}}}} \\ C_{\mathrm{V}} & \text{otherwise} \end{cases} \tag{10}$$

where C_{M} corresponds to a level of gloss considered sharp enough to be reliably computed using recursive eye subpath tracing. We currently use $C_{\mathrm{M}} = 100$ and $C_{\mathrm{V}} = 1$. The effect of C_M on rendered image quality is illustrated in Figure 5.

The intuition here is that if eye subpaths are contained within a small solid angle (highly coherent), then we want to set $\Gamma = \infty$ to allow the recursive eye tracing to include features such as sharp glossy reflections. However, if they have diverged too much we set Γ to some small value C_{V} that limits the recursive eye tracing to only include the diffuse material components below this threshold (in practice one should always set $C_{\mathrm{V}} \leq C_{\mathrm{D}}$). Because we use a large number of VPLs (1 million), diffuse components will only be clamped at very short ranges and such recursive tracing will be limited to short distances.

The final value of μ_j is computed from Γ as follows:

$$\mu_j = \min\left(\frac{\Gamma_j}{D_{j+1}}, 1\right) \min\left(\frac{\Gamma_j}{D_j}, \frac{\Gamma_j}{M_j}, 1\right) \prod_{i=1}^{j-1} \min\left(\frac{\Gamma_{i+1}}{D_i}, \frac{\Gamma_{i+1}}{M_i}, 1\right)$$

If $\Gamma_j = \infty$, then $\mu_j = 1$ and the constraint does not introduce any bias. When Γ is finite, then μ will decrease whenever the material M or directionality terms D exceeds Γ, effectively limiting them to only their diffuse components ($\leq C_V$). In this case μ_j becomes somewhat similar to a much more stringent version of the diffuse VPL constraint. Because μ_j is a non-increasing function of j, we can also easily show that if $w_j = w_j^{\mathrm{V}}$ then $w_p = w_p^{\mathrm{V}} = 0$ for all $p > j$. Thus once this constraint applies, it effectively terminates the eye subpath as all subsequent segments have zero weight.

Pruning. The particular form of this constraint has been specially designed to enable us to proactively prune and terminate eye subpaths that are no longer useful. When we generate a sensor point, we know everything about the path up to $\mathbf{v}_j$ except for M_j and D_j which depend on the light direction ψ_j, that is not determined until we connect it to particular VPLs. This means that we cannot yet compute w_{j-1} exactly, and thus sometimes cannot eliminate useless VPSs (and eye paths) whose weight will always be zero. However, if we can prove that $w_{j-1} \leq w_{j-1}^{\mathrm{V}}$, then we can guarantee that $w_p = 0$ for all $p \geq j$ and safely terminate the path. This can often be proven even if we do not know w_{j-1} exactly yet and eliminates much potential wasted computation.

5 Scalable Implementation

Now that we know how to compute the weight and contribution for an individual VPS/VPL pair, we describe how to estimate the sum of many such pairs. A direct implementation of bidirectional VPL rendering would use the weights above to evaluate every VPS/VPL pair. However, in scenes with millions of VPLs and hundreds of VPS, this pairwise evaluation is prohibitively expensive. Instead we choose to adapt a scalable VPL rendering algorithm that can handle such large numbers of VPLs and VPSs.

Formulation. Multidimensional lightcuts can accurately estimate the sum of many point pairs while only actually evaluating a tiny fraction of the total pairs. It does so by grouping the points (VPS and VPL) into clusters that are estimated by evaluating one representative pair from the cluster. The set of clusters used is dynamically determined by an error bound driven refinement process to guarantee that the error from each cluster estimate is below a perceptual threshold (See [Walter et al. 2006] for details).

In its original notation lightcuts estimates the sum over all points in a sensor set $\mathbb{S}$ and light set $\mathbb{L}$ as:

$$\sum_{\langle s,l\rangle \in \mathbb{S}\times\mathbb{L}} S_s \, M_{\langle s,l\rangle} V_{\langle s,l\rangle} G_{\langle sl\rangle} I_l \tag{11}$$

We introduced weights to the sum, and modified the terms, so that we estimate this modified sum:

$$\sum_{\langle s,l\rangle \in \mathbb{S}\times\mathbb{L}} w_{\langle s,l\rangle} S_s \, M_{\langle s,l\rangle} T_{\langle s,l\rangle} D_{\langle s,l\rangle} I_l \tag{12}$$

The first change we made is that we factored out the distance related terms from G and combined them with the visibility V to create our new transit term T. Since $VG = TD$, this refactoring does not change the sum's value, but is important to make this formulation compatible with our weighting strategies. We also added our new weighting term w for each pair.

Fortunately the lightcuts algorithm is fairly agnostic about the exact phenomena each point represents, allowing us to add new points and phenomena easily. The only requirements are that the algorithm must be able to evaluate point pairs, cluster points together, and bound their maximum possible contribution. Evaluation with weighting, has already been discussed. For each new type of material, we also need to be able to compute corresponding bounds on their M and D functions. We support the materials from prior lightcut's methods (Lambertian, Phong, and Ward) and have added support for BSSRDF materials and two volumetric phase functions Henyey-Greenstein and the Gaussian fiber for cloth [Zhao et al. 2011]. Since these are based on similar components as in Phong and Ward (cosines and half-angles respectively), it was straightforward to adapt existing methods to bound these new materials.

Bounding functions. To cluster points together we need to be able to combine bounding functions from all points in a cluster. In prior work, cube-maps, computed at fixed resolutions, were used for VPS points (called gather points) to project all M bounds into a common representation that could easily be combined for clusters. We also use cube-maps, but use an adaptive quad-tree on each cube-map face rather than a fixed resolution. This gives tighter bounding across a wider range of materials, especially sharp gloss.

Lightcuts originaly only supported a very limited set of VPL types (Lambertian and omnidirectional) that were combined and bounded using specialized functions (bounding cones). However, that approach does not generalize easily to support new materials. Instead, in this paper, we reuse the same cube-map with adaptive quad-tree approach as we use for bounding M. For each VPL, we compute and store a cube-map that bounds its D. Then it is simple to compute maxima over multiple cube-maps, allowing us to cluster VPLs together regardless of type (e.g., we can combine surface and volumetric VPLs in a single cluster). This allows us to create VPLs for a much wider range of materials, however the bounds are not quite as tight or compact as with the much more specialized methods used earlier. Thus, there is some performance cost for this generality.

We also want to integrate information about the weights into the bounding functions. We could use the trivial upper bound of $w \leq 1$ but this would be overly conservative causing extra expense and larger cut sizes. Instead we integrate some weight bounds into the M and D bounds. For VPLs we integrate the diffuse VPL constraint w^{D} directly into the bound by storing an upper bound on $\min(D, C_{\mathrm{D}})$ in their cube-maps. For sensor points, we store information about weights from earlier path segments and other information needed to compute the weights. If $\Gamma \leq C_{\mathrm{D}}$, then we can often prove the weight has to be less than the diffuse VPL weight, and can multiply this additional restriction into the upper bound that we store for M. For each new segment added to an eye subpath, we also check to see if we can prove that its weight will be zero for all subsequent segments and if so we terminate the eye subpath.

6 Results

In this section, we present results of applying our approach and weighting strategies to several scenes. All our code is written in Java, except for the ray intersection testing which is in C++. The code is parallelized to use multiple CPU cores but does not use GPU or SIMD acceleration. All timings are for one machine with a single Intel Core i7-2600K processor with 4 cores at 3.4GHz and 16GB of main memory. Unless otherwise noted, all results are for 512×512 images using 1 million VPLs with parameters $C_{\mathrm{D}} = C_{\mathrm{V}} = 1.0$, $C_{\mathrm{M}} = 100$, and $C_{\mathrm{clamp}} = \frac{L_{\mathrm{a}}}{800 I_{\mathrm{avg}}}$ where L_{a} is the estimated adaption luminance (brightness of a mid-gray surface) for the image, and I_{avg} is the average VPL intensity (by design all VPLs have roughly equal intensities). Multidimensional lightcuts was configured to use a 2% error threshold, 64 representatives per cluster, and a max cut size of 20000. Both standard VPL and our bidirectional method use 64 eye rays per pixel. Timings for rendering each scene are given in Figure 6. Comparisons are given to both standard multidimensional lightcuts and a bidirectional path traced image of equal time or quality. The bidirectional path tracing used multiple importance sampling with the balance heuristic and optimized with russian roulette.

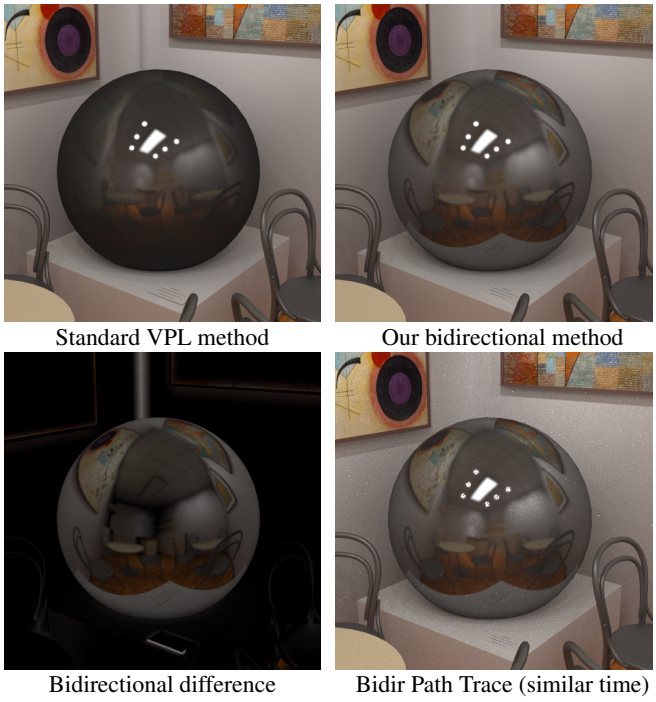
	Pass 1	Pass 2		Bidir Path Trace	
model	VPL gen	Standard	Bidirectional	≈ Time	≈ Quality
cafe-ball	22s	324s	457s	682s	> 11900s
cafe-dragon	21s	338s	724s		7660s
San Miguel	27s	407s	591s		8030s
kitchen	138s	938s	1665s	2203s	> 51900s

Figure 6: *Timings for result images. The first pass to generate VPLs is the same in both standard and bidirectional VPL methods. For each pixel, pass 2 traces 64 eye subpaths to generate VPS points and uses multidimensional lightcuts to compute the summed contribution of all VPS/VPL pairs. Our bidirectional method has a higher cost in this pass because it uses deeper eye subpaths and generates more VPS points per pixel. We also show timings for comparison bidirectional path traced images configured for equal time or equal quality (except cafe-ball and kitchen scenes where even the longest path tracings we ran had objectionable noise artifacts).*

Our first example (see Figure 7) is the cafe scene from [Křivánek et al. 2010] that was used to test the limitations of VPL-based rendering and their effects on image and material perception. They found that in some cases, especially smooth objects with sharp gloss, the clamping in VPL rendering algorithms significantly altered the image and user's perceptions of its material properties. We use their G0 test object (a sphere) and a material based on their MS (metal smooth) material except with sharper gloss $(\rho_d, \rho_s, \alpha) = (0.03, 0.22, 0.015)$. Their results predict that this combination cannot be well handled by standard VPL rendering even if using very large numbers of VPLs, and indeed that is what we observe in Figure 7. Adding our bidirectional estimators nicely fills in the missing glossy reflection on the ball and provides a much more accurate portrayal of its material. A similar time bidirectional path tracing also fills in the reflection but has objectionable noise artifacts.

In Figure 8, we show a comparison between bidirectional lightcuts using our biased weighting strategy and an unbiased weighting based on the method of [Kollig and Keller 2004]. For the unbiased strategy we replace our weight constraint w_j^{V} with their proposed weight constraint: $w_j^{KK} = \prod_{i<j}(1 - w_i)$. Although both strategies are more accurate than standard VPL rendering, the unbiased weights result in significantly higher noise and would require much higher sampling rate and computational cost to converge. In many cases, accepting some bias can significantly reduce the rendering cost while preserving most of the energy and visual fidelity.

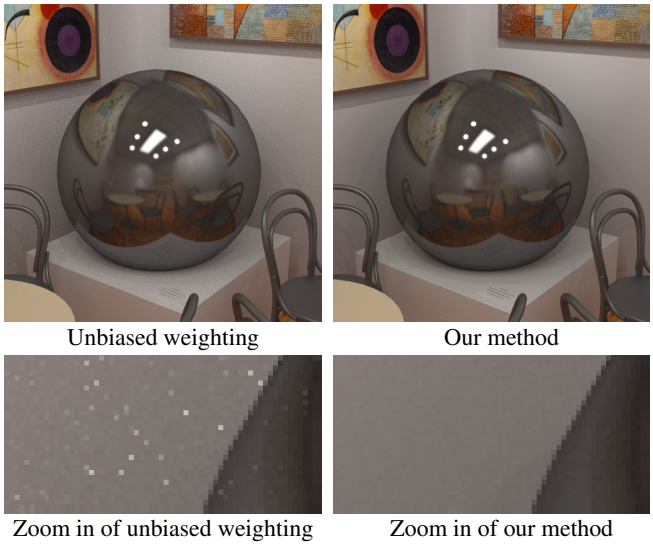
Our next example shows that our bidirectional method can handle BSSRDFs which are used to model materials with dense subsurface scattering such as jade or milk. Figure 9 shows a dragon with exponential falloff BSSRDF material. Because subsurface scattering is a relatively short range effect, the BSSRDF contribution is strongly clamped and largely absent in standard VPL methods. However, our bidirectional method handles it naturally by extending the eye



Standard VPL method | Our bidirectional method

Bidirectional difference | Bidir Path Trace (similar time)

Figure 7: *Cafe scene rendered with standard VPL rendering compared with our bidirectional approach, both using 1 million VPLs. A difference image between standard VPL and our method is in lower left. Also shown is a bidirectional path tracing of roughly equal render time. It closely matches our bidirectional lightcuts result but contains objectionable noise especially near the ball.*



Unbiased weighting | Our method

Zoom in of unbiased weighting | Zoom in of our method

Figure 8: *Comparison between using our biased weighting and an unbiased weighting based on the method proposed by Kollig-Keller Both images use bidirectional lightcuts with 1 million VPLs and 64 eye paths per pixel, however the unbiased weighting results in significantly more noise at the same sampling rates. This is because for difficult paths, such as the weak caustic from the ball onto the wall, the unbiased method effectively reduces to a form of path tracing which is a high variance estimator. Our biased weights fill in nearly all the energy missing from standard VPL methods but without the greatly increased noise of the unbiased weights.*

subpath farther and generating sensor points where it exits the BSSRDF surface. Our result closely matches a bidirectional path traced image of similar image quality, while being >10x faster.

The San Miguel scene in Figure 10 is large, with 9.8 million triangles and complex illumination from a sun/sky model. We modified the original scene by replacing the placemats on one table with ones using the detailed volumetric cloth model ($\approx$ billion voxel effective resolution). Because this scene is mostly diffuse, the standard VPL rendering already captures most of the lighting, but there are places where local interreflection energy is missing, especially the placemats. Our bidirectional method more closely matches the reference.

Our final example is a kitchen scene (see Figure 1) that contains 3.5 million triangles, skylight from three windows and 25 other lights, and a wide variety of different materials and effects. Nearly all the surfaces in this scene have gloss components. Several objects including the dragon and milk are modeled using BSSRDFs. There is some heterogeneous smoke near the dragon and the red cloth is modeled using the anisotropic volumetric cloth model from [Zhao et al. 2011]. The garbardine cloth data is based on measured data with a million voxels that is tiled and warped to create an effective resolution of billions of voxels. The volumetric data is extremely high frequency (see the close up view) and highly problematic for any rendering algorithm that assumes local smoothness or homogeneity. It also uses a Gaussian fiber phase function that, while far from being Lambertian, is actually fairly diffuse (max $M \approx 0.6$). This is by far our most challenging scene.

While the standard VPL method reproduces the overall lighting well, it is missing many of the detailed visual cues that help us understand the materials. Our bidirectional method does a much better job of reproducing important cues such as the reflections that make the countertop look glossy, the subsurface scattering that makes the dragon green and the milk white, and the local scattering within the cloth that produce its bright red color. An equal time bidirectional path tracing also reproduces these effects but is very noisy.

An equal time probabilistic photon map [Knaus and Zwicker 2011] image (Figure 1, bottom left) exhibits visible noise (from difficult glossy paths) and bias where the features are smaller than its kernel (e.g., very thin volumetric cloth <5mm). This method uses kernel density estimation for diffuse materials and recursive eye tracing for non-diffuse. Mixed materials use a random decision to select the diffuse or non-diffuse component based on their relative albedos. We also tried using density estimation for the glossy components, but overall image quality was worse (not shown), and better automatic material heuristics is an open problem in these methods. This image used 1000 passes, 2 million photons/pass, $\alpha = 2/3$, and an initial radius hand-tuned for best results.

7 Conclusions

In this paper, we have presented a bidirectional formulation of VPL rendering along with a novel set of weighting strategies to optimize the new bias vs. noise tradeoffs enabled by this extension. Standard VPL-based rendering is often very good at capturing the general global illumination in complex scenes, but their clamping bias removes some portions of the illumination. These omissions can remove visually important cues for material perception and scene understanding. As shown in our results, our approach allows VPL-based methods to reproduce a wider range of illumination paths, thus restoring many of these visual cues such as glossy reflections, subsurface scattering via BSSRDFs, and short-range interactions in dense, high frequency media such as volumetric cloth models. To preserve the low noise and efficiency advantages of VPL methods, we use a biased weighting strategy unlike most prior bidirectional

Standard VPL method | Our bidirectional method

Bidirectional difference | Bidir Path Trace (similar quality)

Figure 9: *Cafe scene with a BSSRDF/subsurface dragon rendered with standard VPL rendering compared with our bidirectional approach. Both versions used 1 million VPLs. A bidirectional path trace image of similar image quality took >10× longer to compute.*

Standard VPL method | Our bidirectional method

Bidirectional difference | Bidir Path Trace (similar quality)

Figure 10: *San Miguel scene modified to add red placemats using a volumetric cloth model. The standard VPL rendering with 1 million VPLs is already quite close to the reference solution for this mostly diffuse scene. Our bidirectional method adds some of the missing local interreflection illumination, especially in the highly detailed cloth, to more closely match the bidirectional path traced reference which took more than 10× longer to compute.*

work. This allows us to significantly reduce the bias relative to standard VPL rendering without introducing the high noise problems common in purely unbiased methods. Our results show that we are able to much more closely match path traced reference images but without their noise problems while paying usually less than twice the cost of standard, scalable VPL rendering.

Limitations and future work. Although our method broadens the types of paths that can be handled by VPL rendering, there are still some types of illumination paths that are still missing, resulting in a negative bias. One example which our method cannot reproduce is caustics such as those from light shining through a glass ball onto a nearby surface. In future, we would like to explore hybrid algorithms similar to the approach of [Dammertz et al. 2010] to remove this remaining bias. Since in our method, the bias for any path can easily be computed as $(1-\sum w_k(\bar{\mathbf{p}}))f(\bar{\mathbf{p}})$, it is possible to solve for part or all of this bias using algorithms such as Metropolis or caustic photon mapping, that are better suited to such paths.

Deciding how much bias is needed to prevent visible noise is a challenging problem. Our current heuristics work well in our tests but do not provide guarantees on low noise. In future, we would like to explore a wider space of weighting strategies to see if there are alternatives that might better optimize the bias vs. noise tradeoff or provide such guarantees.

The performance of all lightcuts methods is tied to the tightness of their bounding functions. Thus, adding new materials requires developing the corresponding bounding functions, though one can often adapt from existing bounds for similar materials. Scenes with high occlusion are more expensive because the simple visibility bounds used become very loose in this case. In future, adding occlusion culling or tightening the visibility bounds could significantly improve performance.

Acknowledgements. This work was funded by NSF (CAREER 1041534 and IIS 1011919), Autodesk, and conducted in conjunction with the Intel Science and Technology Center Visual Computing. Models courtesy of Guillermo M. Leal Llaguno (San Miguel), Jaroslav Krivanek (cafe), Autodesk (kitchen), Shuang Zhao (gabardine cloth) and Stanford Computer Graphics Lab (dragon). Also thanks to Edgar Velazquez-Armendariz and Shuang Zhao for help in model conversion and coding.

References

ARBREE, A., WALTER, B., AND BALA, K. 2008. Single-pass scalable subsurface rendering with lightcuts. *Computer Graphics Forum 27*, 2 (Apr.), 507–516.

DAMMERTZ, H., KELLER, A., AND LENSCH, H. P. A. 2010. Progressive point-light-based global illumination. *Computer Graphics Forum 29*, 8, 2504–2515.

DAVIDOVIČ, T., KŘIVÁNEK, J., HAŠAN, M., SLUSALLEK, P., AND BALA, K. 2010. Combining global and local virtual lights for detailed glossy illumination. *ACM Trans. Graph. 29* (December), 143:1–143:8.

ENGELHARDT, T., NOVAK, J., AND DACHSBACHER, C. 2010. Instant multiple scattering for interactive rendering of heterogeneous participating media. Tech. Rep. December, KIT - Karlsruhe Institut of Technology.

HACHISUKA, T., AND JENSEN, H. W. 2009. Stochastic progressive photon mapping. *ACM Transactions on Graphics 28*, 5 (Dec.), 141:1–141:8.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics 26*, 3 (July), 26:1–26:10.

HAŠAN, M., KŘIVÁNEK, J., WALTER, B., AND BALA, K. 2009. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics 28*, 5 (Dec.), 143:1–143:6.

JAKOB, W., ARBREE, A., MOON, J. T., BALA, K., AND MARSCHNER, S. 2010. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Transactions on Graphics 29*, 4 (July), 53:1–53:13.

JAROSZ, W., NOWROUZEZAHRAI, D., SADEGHI, I., AND JENSEN, H. W. 2011. A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics 30*, 1 (Jan.), 5:1–5:19.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001*, 511–518.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA.

KELLER, A. 1997. Instant radiosity. In *SIGGRAPH '97*, 49–56.

KNAUS, C., AND ZWICKER, M. 2011. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph. 30* (May), 25:1–25:13.

KOLLIG, T., AND KELLER, A. 2004. Illumination in the Presence of Weak Singularities. In *Monte Carlo and Quasi-Monte Carlo Methods*, 245–257.

KŘIVÁNEK, J., FERWERDA, J. A., AND BALA, K. 2010. Effects of global illumination approximations on material appearance. *ACM Transactions on Graphics 29*, 4 (July), 112:1–112:10.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In *Compugraphics '93*, 145–153.

NOVÁK, J., ENGELHARDT, T., AND DACHSBACHER, C. 2011. Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Symposium on Interactive 3D Graphics and Games*, ACM, 119–124.

OU, J., AND PELLACINI, F. 2011. Lightslice: matrix slice sampling for the many-lights problem. *ACM Trans. Graph. 30* (Dec.), 179:1–179:8.

SEGOVIA, B., IEHL, J.-C., MITANCHEY, R., AND PÉROCHE, B. 2006. Bidirectional instant radiosity. In *Proceedings of the 17th Eurographics Workshop on Rendering*.

VEACH, E., AND GUIBAS, L. 1994. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*.

VEACH, E., AND GUIBAS, L. J. 1995. Optimally combining sampling techniques for monte carlo rendering. In *Proceedings of SIGGRAPH 95*, 419–428.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *SIGGRAPH '97*, 65–76.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics 24*, 3 (Aug.), 1098–1107.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. *ACM Transactions on Graphics 25*, 3 (July), 1081–1088.

YUE, Y., IWASAKI, K., CHEN, B.-Y., DOBASHI, Y., AND NISHITA, T. 2010. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. *ACM Trans. Graph. 29* (December), 177:1–177:8.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro ct imaging. *ACM Trans. Graph. 30* (Aug.), 44:1–44:10.

A VPL and VPS Attributes

We model all paths from light sources to sensors as a set of vertices connected by segments (see Figure 4). The vertices represent points on light sources and sensors as well as locations where important intermediate scattering events—such as surface reflections or volume scattering—take place. This path abstraction enables us to support a wide variety of different lighting phenomena within a single unified framework.

We write the contribution of a path as a product of three types of terms: emission e, scattering s, and transit τ. The emission term describes how much light is being emitted from each source, as well as the responsivity of each sensor. The scattering term $s(\mathbf{v}, \psi, \omega)$ describes how much of the light arriving at a vertex $\mathbf{v}$ from direction ψ leaves in direction ω. The transit term $\tau(\mathbf{v}_i, \mathbf{v}_{i+1})$ accounts for any changes in the light as it travels from $\mathbf{v}_{i+1}$ to $\mathbf{v}_i$. Such changes include occlusion and volumetric attenuation. Below we give definitions of s and τ for several common cases.

Surface scattering: When a path vertex lies on a surface, the scattering function is closely related to the surface material's BSDF (Bidirectional Scattering Distribution Function) f_s

$$s(\mathbf{v}, \psi, \omega) = |\psi \cdot \mathbf{n}^\mathrm{g}| \, f_\mathrm{s}(\mathbf{v}, \psi, \omega) \, |\omega \cdot \mathbf{n}^\mathrm{g}| \tag{13}$$

where $\mathbf{n}^\mathrm{g}$ is the geometric surface normal. For example, for a Lambertian surface using shading normal $\mathbf{n}^\mathrm{s}$ and albedo ρ, its BSDF is $f_\mathrm{s} = (\rho \, |\psi \cdot \mathbf{n}^\mathrm{s}|)/(\pi \, |\psi \cdot \mathbf{n}^\mathrm{g}|)$ and we get:

$$s_\mathrm{lambertian}(\mathbf{x}, \psi, \omega) = \frac{\rho \, |\psi \cdot \mathbf{n}^\mathrm{s}| \, |\omega \cdot \mathbf{n}^\mathrm{g}|}{\pi} \tag{14}$$

if both directions lie on the same side of the surface.

In special cases such as mirror reflection, the scattering function (and BSDF) can include a dirac delta function and have infinite value. Segments containing such a vertex cannot be connecting segments in bidirectional methods and are assigned a zero weight. However, such paths can be handled by tracing the eye or light path past the mirror and connecting at a different segment.

Volumetric scattering: At a volumetric point, the scattering function is defined in terms of the phase function f_p and scattering coefficient σ_s:

$$s(\mathbf{v}, \psi, \omega) = \sigma_\mathrm{s}(\omega) \, f_\mathrm{p}(\mathbf{v}, \psi, \omega) \tag{15}$$

This general formulation holds even for anisotropic media [Jakob et al. 2010] where σ_s varies with direction.

Transit in ordinary media: For ordinary media where light travels in straight lines such as air, fog, and anisotropic media, the transit can be defined in terms of the visibility V along the segment (usually 0 or 1) and the line integral of the attenuation coefficient σ_t along the segment as:

$$\tau(\mathbf{v}_i, \mathbf{v}_{i+1}) = V(\mathbf{v}_i, \mathbf{v}_{i+1}) \, e^{-\int_{\mathbf{v}_{i+1}}^{\mathbf{v}_i} \sigma_\mathrm{t}(\mathbf{x}, \omega) d\mathbf{x}} \tag{16}$$

For transparent media, $\sigma_t = 0$ and τ is just the visibility V. In some media, the line integral either cannot be computed or is too expensive. In such cases, Woodcock tracking [Yue et al. 2010] is used instead. Here, the line integral is replaced by a random variable that is either zero or one depending on whether the random scattering point selected by the Woodcock process occurs before the end of the segment.

Subsurface transport (BSSRDFs) can also be modeled in terms of paths and the corresponding s and τ are described in Appendix B. Now we can define the quantities used in our VPL rendering formulation using the above path scattering and transit terms.

Material M and directionality D terms: D describes the directional pattern of the light emitted from a VPL, and M is the analogous term for sensor points. At a vertex $\mathbf{v}$, we define them as:

$$M(\psi) = \frac{s(\mathbf{v},\psi,\omega)}{\alpha_M(\mathbf{v},\omega)}, \qquad \alpha_M(\mathbf{v},\omega) \approx \int s(\mathbf{v},\psi,\omega)\, d\psi \quad (17)$$

$$D(\omega) = \frac{s(\mathbf{v},\psi,\omega)}{\alpha_D(\mathbf{v},\psi)}, \qquad \alpha_D(\mathbf{v},\psi) \approx \int s(\mathbf{v},\psi,\omega)\, d\omega \quad (18)$$

where the α terms are normalization terms that are intended to approximate the corresponding spherical integrals. For example, $\alpha_M(\mathbf{v},\omega) = \rho\,|\omega\cdot\mathbf{n}^{\mathrm{g}}|$ for a Lambertian surface and $\alpha_M(\mathbf{v},\omega) = \sigma_s(\omega)$ for volumetric scattering.

These normalizing terms can be approximate because canceling α multipliers exist in the strength S and intensity I terms. Thus, exact solutions (which are often unknown) are not required. We propose using approximately normalized M and D terms to make it easier to reason about the effects of weighting strategies.

Transit term T: The transit term T accounts for all effects that can occur along the line segment connecting a VPS to a VPL. It combines τ with a distance factor as follows:

$$T_{i,i+1} = \frac{\tau(\mathbf{v}_i,\mathbf{v}_{i+1})}{\|\mathbf{v}_i - \mathbf{v}_{i+1}\|^2} \quad (19)$$

In the special case of lights at infinity (environment map lighting) or BSSRDF media, we omit the distance term in the denominator.

Strength S and intensity I terms: The strength and intensity terms depend on the eye and light subpaths used to generate the VPS and VPLs, respectively. We follow standard convention in stochastic particle tracing, and the computation of the terms is only briefly summarized here. The strength is given by:

$$S_i = \frac{e_c(\mathbf{v}_0)}{p_c(\mathbf{v}_0)} \left(\prod_{m=0}^{i-1} \frac{s(\mathbf{v}_m,\psi_m,\omega_m)\,\tau(\mathbf{v}_m,\mathbf{v}_{m+1})}{p(\psi_m)\,p(\mathbf{v}_{m+1})} \right) \frac{\alpha_M(\mathbf{v}_i,\psi_i)}{p_{\mathrm{survive}}} \quad (20)$$

where e_c and p_c are the area sensitivity and the probability of selecting an initial vertex on the camera, respectively. The product accounts for all the vertices and segments between the initial vertex and the current location. It includes the probabilities of selecting segment directions ψ and of choosing the next vertex, given that direction. Finally, we multiply by an α term to cancel the one in the material term M and divide by the probability that the particle is not killed by the russian roulette used to prune weak particles. A similar equation determines the intensities of the VPLs.

B BSSRDF in Path-Based Algorithms

A BSSRDF [Jensen et al. 2001] describes light scattering in a medium where light changes direction and leaves from a different point on the surface. Thus a BSSRDF is a function of two positions and two directions, $f\mathrm{ss}(\mathbf{v}_a,\psi_a,\mathbf{v}_b,\omega_b)$ and cannot be expressed in our path notation as a single scattering function $s(\mathbf{v},\psi,\omega)$. However, we can fit it into our formulation by modeling it as two path vertices plus a special type of segment such that:

$$\begin{aligned} &|\psi\cdot\mathbf{n}^{\mathrm{g}}|\, f\mathrm{ss}(\mathbf{v}_{i+1},\psi_{i+1},\mathbf{v}_i,\omega_i)\,|\omega\cdot\mathbf{n}^{\mathrm{g}}| \\ &\quad = s(\mathbf{v}_i,\psi_i,\omega_i)\,\tau(\mathbf{v}_i,\mathbf{v}_{i+1})\,s(\mathbf{v}_{i+1},\psi_{i+1},\omega_{i+1}). \end{aligned} \quad (21)$$

For example, given a BSSRDF based on a distance falloff function $g()$, Lambertian directional behavior, and albedo ρ such that:

$$f\mathrm{ss}(\mathbf{v}_{i+1},\psi_{i+1},\mathbf{v}_i,\omega_i) = \frac{\rho}{\pi}\, g(\|\mathbf{v}_i - \mathbf{v}_{i+1}\|), \quad (22)$$

we can model this behavior in our framework by setting:

$$s(\mathbf{v}_i,\psi_i,\omega_i) = \frac{\sqrt{\rho}}{\pi}\,\left|\omega_i\cdot\mathbf{n}_i^{\mathrm{g}}\right| \quad (23)$$

$$\tau(\mathbf{v}_i,\mathbf{v}_{i+1}) = \pi g(\|\mathbf{v}_i - \mathbf{v}_{i+1}\|) \quad (24)$$

$$s(\mathbf{v}_{i+1},\psi_{i+1},\omega_{i+1}) = \frac{\sqrt{\rho}}{\pi}\,\left|\psi_{i+1}\cdot\mathbf{n}_{i+1}^{\mathrm{g}}\right| \quad (25)$$

where we keep the scattering functions as similar as possible to those for Lambertian BSDFs. This τ does not include visibility so there is no shadowing within this special segment type. Also, note that it can only connect vertices on the same BSSRDF surface or object. Otherwise, these vertices and segments behave similarly to the more conventional surface or volume ones, and our system required only small modifications to support them. This formulation allows path and VPL-based methods to include BSSRDFs without major algorithmic modifications (e.g., [Arbree et al. 2008] had to extend point pairs to triplets and required specialized BSSRDF caches).

When generating eye and light subpaths, one may need to randomly generate new vertices for BSSRDF segments on the same surface with probability based on a distance function $g()$. We have developed a novel ray-based solution to this problem based on uniform random lines. A brief outline is given below.

We generate two random points and connect them by a line segment to generate a uniform distribution of lines within the sphere. If we find all intersections between such a line segment and any surfaces inside the sphere, the probability density of finding a surface point $\mathbf{x}$ is simply $p(\mathbf{x}) = 1/(2\pi r^2)$ where r is the radius of the sphere. Thus it will uniformly sample all surface points within some distance r, which would be exactly what we want if our distance function $g()$ were constant within some distance and zero outside it. The distance functions used in practice are usually smoothly decreasing functions of distance, but we can also match such a function by randomly generating the radius of the sphere as well as the points on the sphere. In this case the probability of finding a point is:

$$p(\mathbf{x}) = \int_{d_{\mathbf{x}}}^{\infty} \frac{p_r(r)}{2\pi r^2}\, dr \quad (26)$$

where $d_{\mathbf{x}}$ is the distance between $\mathbf{x}$ and the center of the sphere and p_r is the probability of choosing a radius. By setting this equal to our target distance function $g()$ and inverting (analytically or numerically) to generate p_r, we can sample points with a distribution that matches any decreasing distance function. Each segment may generate zero, one, or multiple points though, so the sampling still has some variance. The advantages are that it is unbiased, works with any ray-intersectable surface, and requires no precomputation or extra storage.