Scalable Realistic Rendering with Many-Light Methods

Carsten Dachsbacher, Karlsruhe Institute of Technology
Jaroslav Křivánek, Charles University, Prague
Miloš Hašan, UC Berkeley
Adam Arbree, Autodesk, Inc.
Bruce Walter, Cornell University
Jan Novák, Karlsruhe Institute of Technology

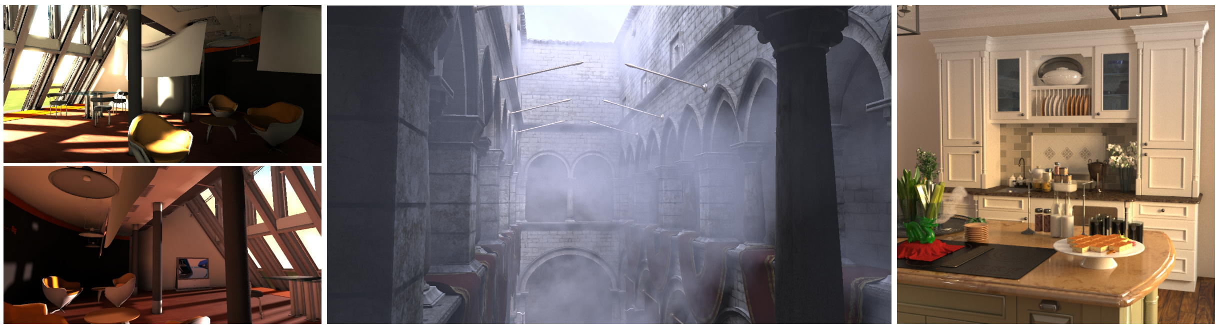

Figure 1: *Many-light rendering methods, covered in this report, yield good results at different points along the quality-speed trade-off axis. The images on the left were rendered in real-time with [REH*11] (courtesy of Tobias Ritschel) and capture diffuse interreflections. The center image took 52 minutes to render and demonstrates many-light methods for participating media (adapted from [ENSD12]). The image on the right combines different phenomena such as glossy surfaces, subsurface BSSRDFs and a detailed anisotropic volumetric cloth model rendered with Bidirectional Lightcuts [WKB12] in about 46 minutes.*

Abstract
Recent years have seen increasing attention and significant progress in many-light rendering, a class of methods for the efficient computation of global illumination. The many-light formulation offers a unified mathematical framework for the problem reducing the full lighting transport simulation to the calculation of the direct illumination from many virtual light sources. These methods are unrivaled in their scalability: they are able to produce artifact-free images in a fraction of a second but also converge to the full solution over time. In this state-of-the-art report, we have three goals: give an easy-to-follow, introductory tutorial of many-light theory; provide a comprehensive, unified survey of the topic with a comparison of the main algorithms; and present a vision to motivate and guide future research. We will cover both the fundamental concepts as well as improvements, extensions, and applications of many-light rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

1. Introduction

Various industries, including game development, film production, industrial design, architecture, and e-commerce, require realistic image rendering with global illumination. The importance of this topic is underscored by the numerous research activities and papers in this field. Despite significant progress, many speed and/or image quality requirements still remain out of reach for even state-of-the-art realistic rendering algorithms. One class of methods—many-light rendering derived from the instant radiosity algorithm of Keller [Kel97]—comes close to this goal and has received significant attention in recent years.

Many-light methods are attractive because they offer a simple solution to many difficult rendering problems. Their core insight is that the general light transport problem can be approximated by the simpler problem of calculating the direct illumination from many virtual light sources. This gives many-light algorithms two distinct advantages. First, it provides a unified mathematical framework for the calculation of global illumination. Second, it makes many-light algorithms very adaptable: the same algorithm can be adjusted to meet a wide range of quality and performance goals. By using a few virtual sources as a coarse approximation, many-light methods can be used to produce biased, but artifact free, images in a fraction of a second (see Fig. 1, left) making them attractive for real-time rendering applications such as computer games. On the other hand, by using sufficiently many virtual sources and a highly scalable evaluation algorithm, any bias from the virtual source approximation can be reduced to a negligible level. These algorithms produce results comparable to other unbiased methods in much less time and make them an attractive option for even high-fidelity applications (see Fig. 1, right). Finally, because their images are produced not only quickly but scalably—with predictable and reliable costs across a wide range of model, lighting, and material complexity—many-light algorithms have been an excellent choice for large scale rendering applications like Autodesk 360 Rendering [Aut13].

This report presents a coherent summary of the state-of-the-art in many-light rendering. We start with a hands-on introduction of the basic many-light concept and then define in detail the underlying theory. Afterwards, we provide a comprehensive overview of the field categorizing the related work into four topics:

- many-light generation,
- rendering and scalability,
- image fidelity, and
- performance considerations for interactive rendering.

2. Introduction to the Basic Instant Radiosity Method

The instant radiosity (IR) algorithm [Kel97] is the basis of all many-light methods. Like that original work, we will introduce the algorithm by describing the method for surfaces only and later generalize to include participating media as a part of the precise, mathematical formulation in Section 3.

We begin with the rendering equation [Kaj86] which describes the light transport in a scene. The radiance L leaving the surface at point $\mathbf{x}$ in direction ω is expressed as:

$$L(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{\mathcal{H}^+} f_r(\mathbf{x},\omega,\omega')L_i(\mathbf{x},\omega')\cos\theta'\,d\omega',$$

The integration is over the upper hemisphere $\mathcal{H}^+$, f_r denotes the bidirectional reflectance distribution function (BRDF), L_i is the incoming radiance, and θ' is the angle between ω' and the surface normal at $\mathbf{x}$.

In principle, all (Markov chain) Monte Carlo global illumination methods evaluate this integral equation numerically and expand it recursively to construct *light transport paths* on which light travels from light sources via reflections off the surfaces to the camera or eye sensors. Path tracing [Kaj86], for example, builds the paths by tracing rays starting from the camera. At each intersection with the scene, the path is continued by randomly sampling an outgoing direction in the hemisphere above the intersected surface (usually direct illumination is also estimated at each intersection point). Building on path tracing, bidirectional methods trace both camera sub-paths and light sub-paths and then (deterministically) connect them to form full paths.

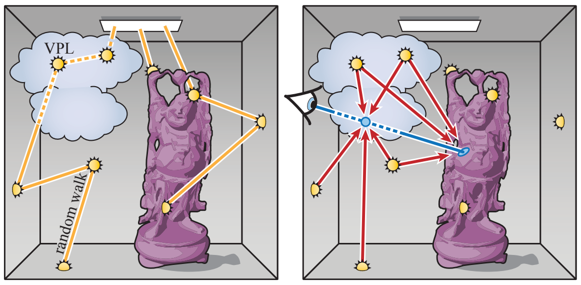

Figure 2: *Many-light algorithms operate in two passes: first, they distribute a number of VPLs (left) and then use them to illuminate the scene and by this approximate indirect illumination (right).*

IR is a bidirectional method that constructs these two sets of sub-paths in a specific manner. First, a large set of arbitrary length light sub-paths is generated and stored. For each vertex of these sub-paths, the complete local environment is recorded: the position, the normal, the incident direction, the BRDF and the current "flux" (i.e. emitted radiance at the light source multiplied by the path throughput to the vertex divided by the probability density of the path to that point). The intention is that the data stored for each vertex suffice to compute the outgoing illumination scattered from this vertex into any direction. If this is true, we can discard the notion of the original path and instead model the vertex as an unusual type of point light source. Since these do not correspond to any physical light sources in the scene, we call them *virtual point lights* (VPLs).

In order to complete the IR algorithm, camera sub-paths are constructed for each pixel in a second phase. Since the light sub-paths were arbitrarily long, it is sufficient to consider only length-one camera paths. Then, like all bidirectional algorithms, IR connects vertices of these camera sub-paths to vertices of the light paths to form full paths. However, because of the VPL generation pre-process, this connecting step is elegantly equivalent to the direct illumination of the first camera hit point by the VPLs.

The two phase IR algorithm is more efficient than general bidirection methods for two reasons. First, since each VPL is used to illuminate surface points in all (or many) image pixels, the effort invested into generating the VPLs is well amortized. This is an instance of sub-path reuse, similar to, for example, photon mapping [PH10]. Additionally, the use of a single set of VPLs to illuminate all surface points produces correlated pixel values. This property is extremely efficient at visually suppressing the image noise present in traditional Monte Carlo approaches, which build *independent* paths for each pixel. Note that this latter advantage has little to do with reducing numerical error – it is purely of perceptual nature.

A basic many-light method can be summarized as follows (see Section 3 for the full mathematical background):

Phase 1 VPL Generation (Section 4)

- Randomly choose one of the primary light sources in the scene, sample a random position $\mathbf{x}$ and direction ω (create a VPL at this location if direct illuminaton is not handled otherwise).
- Trace the ray $\mathbf{x} + t\omega$. If it intersects a surface then create a VPL at this intersection location.
- Decide randomly whether or not to terminate the path using Russian roulette. If continued, sample outgoing direction, update path throughput based on BRDF and direction, and continue tracing.

Phase 2 Rendering with VPLs (Section 5 and 6)

- For shading a surface point, simply iterate over all VPLs, test whether illumination thereof is blocked, and if not, compute the respective contribution.

Participating Media. The same principle works analogously for scenes with participating media. For this, VPL tracing has to account for light scattering within the media, yielding VPLs that might not be located on surfaces. Multiple scattering at a point in a medium can then be computed by summing up the single-scattering (i.e. direct illumination) contributions from all the VPLs. Note that in media, binary visibility is replaced by transmittance, as light traveling through space can be absorbed or outscattered.

The original instant radiosity method [Kel97] was not an all-end solution to the global illumination problem. The decision for this particular strategy for constructing transport paths has advantages as well as disadvantages. Common to most variants is that they are relatively simple to implement and provide fast convergence, progressive rendering, and predictable rendering costs. On the other hand, IR methods are prone to singularities (imagine the entire light energy in a scene contracted to the locations of the VPLs), and have difficulties in scenes, where a large number of light subpaths is required, e.g. in scenes with glossy surfaces. All of these aspects (and many more) have been addressed in works in this field and are discussed, or pointed to, in this article.

3. Global Illumination with Many-Light Rendering

In the following we introduce a more formal definition of the instant radiosity algorithm, and also take participating media into account. Our end goal is to calculate pixel values, given by an integral of the incoming radiance incident at camera sensors over the pixel area, and optionally also over the camera aperture.

Here, we focus on the calculation of incoming radiance $L_\mathrm{i}(\mathbf{x}_0, \omega)$ at a given point $\mathbf{x}_0$ from a given direction ω. In a scene with no participating media, this would simply be equal to the outgoing radiance $L_\mathrm{s}(\mathbf{x}_\mathrm{s} \to \mathbf{x}_0)$ at a surface

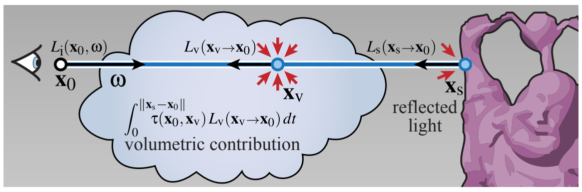
$L_\mathrm{i}(\mathbf{x}_0, \omega)$
$\mathbf{x}_0$
ω
$L_\mathrm{v}(\mathbf{x}_\mathrm{v} \to \mathbf{x}_0)$
$\mathbf{x}_\mathrm{v}$
$L_\mathrm{s}(\mathbf{x}_\mathrm{s} \to \mathbf{x}_0)$
$\mathbf{x}_\mathrm{s}$
$\int_0^{\|\mathbf{x}_\mathrm{s} - \mathbf{x}_0\|} \tau(\mathbf{x}_0, \mathbf{x}_\mathrm{v}) L_\mathrm{v}(\mathbf{x}_\mathrm{v} \to \mathbf{x}_0)\, dt$
volumetric contribution
reflected light

Figure 3: *The radiance L_i reaching the eye at $\mathbf{x}_0$ from direction ω is computed by integrating the inscattered light along the eye ray and adding the light reflected or transmitted towards the eye from the nearest visible surface point.*

point $\mathbf{x}_\mathrm{s}$, visible from $\mathbf{x}_0$ along direction ω. In a scene with participating media, this outgoing radiance needs to be attenuated by the transmittance $\tau(\mathbf{x}_0, \mathbf{x}_\mathrm{s})$ between $\mathbf{x}_0$ and $\mathbf{x}_\mathrm{s}$. Furthermore, we add the integral of the inscattered radiance L_v along the $\overline{\mathbf{x}_0\mathbf{x}_\mathrm{s}}$ line (see Fig. 3):

$$\begin{aligned} L_\mathrm{i}(\mathbf{x}_0, \omega) &= \tau(\mathbf{x}_0, \mathbf{x}_\mathrm{s}) L_\mathrm{s}(\mathbf{x}_\mathrm{s} \to \mathbf{x}_0) \\ &\quad + \int_0^{\|\mathbf{x}_\mathrm{s} - \mathbf{x}_0\|} \tau(\mathbf{x}_0, \mathbf{x}_\mathrm{v}) L_\mathrm{v}(\mathbf{x}_\mathrm{v} \to \mathbf{x}_0)\, dt, \end{aligned} \tag{1}$$

with $\mathbf{x}_\mathrm{v} = \mathbf{x}_0 + t\omega$. Note that the actual interactions will be created in a stochastic sampling process and denoted as $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ... starting from the camera (Fig. 4).

3.1. Outgoing Radiance as an Integral over Light Paths

The outgoing radiance at a surface point, denoted L_s in the equation above, is given by the rendering equation shown in Section 2. Outgoing radiance due to inscattering in a medium, L_v, is defined similarly and we refer the reader to [PH10] for details. By recursively expanding these equations k times (derivation omitted here), we obtain a general formula for outgoing radiance $L(\mathbf{x}_1 \to \mathbf{x}_0)$ at $\mathbf{x}_1$ due to light transport corresponding to exactly $k-1$ bounces (i.e. scattering events in media and/or reflection/refraction events on surfaces) including the bounce at $\mathbf{x}_1$:

$$L^k(\mathbf{x}_1 \to \mathbf{x}_0) = \int .. \int \left[T_1^k L_\mathrm{e}(\mathbf{x}_k) \right] d\mu(\mathbf{x}_2) \ldots d\mu(\mathbf{x}_k). \tag{2}$$

Thanks to the generalized notation, defined in Fig. 5, Eq. (2) can be used for outgoing radiance from media as well as from surfaces. As such, we no longer need to distinguish between L_v and L_s and simply denote the outgoing radiance as L. The differential measure $d\mu(\mathbf{x})$ amounts to the differential area $dA(\mathbf{x})$ for path vertices $\mathbf{x}$ that are surface points. If $\mathbf{x}$ is a volumetric point, $d\mu(\mathbf{x})$ corresponds to differential volume $dV(\mathbf{x})$. In the integrand, $L_\mathrm{e}(\mathbf{x}_k)$ is radiance emitted from $\mathbf{x}_k$ towards $\mathbf{x}_{k-1}$ and T_j^k is the throughput of the light path $\mathbf{x}_j \ldots \mathbf{x}_k$ defined as the product of the *generalized scattering distribution function* $f(\mathbf{x}_i)$ (one term for each vertex except $\mathbf{x}_k$), and the *generalized geometry* $G(\mathbf{x}, \mathbf{y})$ and *generalized visibility* $\hat{V}(\mathbf{x}, \mathbf{y})$ terms (one term for each segment); see Fig. 4 for an illustration and Fig. 5 for a definition of these terms.

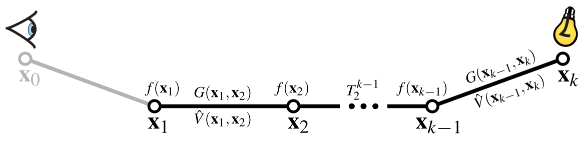
$\mathbf{x}_0$
$f(\mathbf{x}_1)$
$\mathbf{x}_1$
$G(\mathbf{x}_1,\mathbf{x}_2)$
$\hat{V}(\mathbf{x}_1,\mathbf{x}_2)$
$f(\mathbf{x}_2)$
$\mathbf{x}_2$
T_2^{k-1}
$f(\mathbf{x}_{k-1})$
$\mathbf{x}_{k-1}$
$G(\mathbf{x}_{k-1},\mathbf{x}_k)$
$\hat{V}(\mathbf{x}_{k-1},\mathbf{x}_k)$
$\mathbf{x}_k$

Figure 4: *The path throughput T_1^k from Eq. (2) is a product of the generalized scattering function $f(\mathbf{x}_i)$ for path vertices, and the generalized geometry $G(\mathbf{x},\mathbf{y})$ and visibility $\hat{V}(\mathbf{x},\mathbf{y})$ terms for path segments.*

Eq. (2) yields outgoing radiance after exactly $k-1$ bounces. The complete global illumination solution is given by the equilibrium radiance, which is equal to the sum of radiance from paths of all lengths:

$$L(\mathbf{x}_1 \to \mathbf{x}_0) = \sum_{k=1}^{\infty} L^k(\mathbf{x}_1 \to \mathbf{x}_0). \tag{3}$$

Note that the zero-bounce radiance (i.e. $k = 1$) corresponds to the emitted radiance $L_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_0)$, the one-bounce radiance at $\mathbf{x}_1$ corresponds to direct illumination for surfaces and single-scattering for media; indirect illumination and multiple scattering correspond to more than one light bounce.

As a side note, we would like to point out that our definitions so far are in line with the extension of the path integral formulation of light transport [Vea97] to account for participating media [JM12]. The difference is that we consider a part of the transport path held fixed (specifically vertex $\mathbf{x}_0$ and direction $\mathbf{x}_0 \to \mathbf{x}_1$) and we only integrate over the remaining subspace of the path space.

$$\begin{aligned}
T_j^k &= \prod_{i=j}^{k-1} f(\mathbf{x}_i)\, G(\mathbf{x}_i,\mathbf{x}_{i+1})\, \hat{V}(\mathbf{x}_i,\mathbf{x}_{i+1}) \\
f(\mathbf{x}_i) &= \begin{cases} f_r(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1}) & \text{if } \mathbf{x}_i \text{ is on surface} \\ f_p(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1})\sigma_s(\mathbf{x}_i) & \text{if } \mathbf{x}_i \text{ is in medium} \end{cases} \\
G(\mathbf{x},\mathbf{y}) &= D_\mathbf{x}(\mathbf{y}) D_\mathbf{y}(\mathbf{x}) / \|\mathbf{x}-\mathbf{y}\|^2 \\
\hat{V}(\mathbf{x},\mathbf{y}) &= \tau(\mathbf{x},\mathbf{y}) V(\mathbf{x},\mathbf{y}).
\end{aligned}$$

The path throughput T_j^k is a product of the generalized scattering function $f(\mathbf{x}_i)$, geometry $G(\mathbf{x},\mathbf{y})$ and visibility $\hat{V}(\mathbf{x},\mathbf{y})$ terms. $f(\mathbf{x}_i)$ describes the probability of light traveling from point $\mathbf{x}_{i+1}$ to $\mathbf{x}_i$ being scattered towards point $\mathbf{x}_{i-1}$. It is defined by the BRDF f_r for surface points, and the product of the phase function f_p and the scattering coefficient σ_s for points in the medium. The geometry term $G(\mathbf{x},\mathbf{y})$ represents the geometric coupling between a pair of path vertices. If $\mathbf{x}$ is on a surface, then $D_\mathbf{x}(\mathbf{y}) = \max(0, \cos\theta)$, where θ is the angle between the surface normal at $\mathbf{x}$ and the direction towards $\mathbf{y}$. If $\mathbf{x}$ is a volumetric point, then $D_\mathbf{x}(\mathbf{y})$ reduces to 1. Finally, $V(\mathbf{x},\mathbf{y})$ and $\tau(\mathbf{x},\mathbf{y})$ represent the mutual visibility and transmittance between $\mathbf{x}$ and $\mathbf{y}$, respectively.

Figure 5: *Definition of the path throughput and generalized scattering, geometry, and visibility terms.*

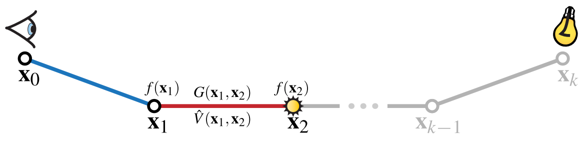

Figure 6: *When connecting a shading point ($\mathbf{x}_1$) to a VPL ($\mathbf{x}_2$) we need to evaluate both corresponding generalized scattering functions, $f(\mathbf{x}_1)$ and $f(\mathbf{x}_2)$, and account for the mutual geometric configuration $G(\mathbf{x}_1,\mathbf{x}_2)$ and visibility $\hat{V}(\mathbf{x}_1,\mathbf{x}_2)$ between the two points.*

3.2. Using VPLs to Evaluate the Radiance Integral

Most light transport simulation algorithms employ Monte Carlo quadrature to evaluate the integral in Eq. (2), which amounts to drawing a random light path $\mathbf{x}_2 \ldots \mathbf{x}_k$ from some joint probability distribution $p(\mathbf{x}_2, \ldots, \mathbf{x}_k)$, and evaluating the following Monte Carlo estimator:

$$\langle L^k(\mathbf{x}_1 \to \mathbf{x}_0) \rangle = T_1^k L_\mathrm{e}(\mathbf{x}_k) / p(\mathbf{x}_2, \ldots, \mathbf{x}_k). \tag{4}$$

In practice, we would generate N random paths and average their contribution given by this estimator. We omit this fact in our formulas for notation clarity.

Keller's [Kel97] key observation in the instant radiosity algorithm is: we can split the evaluation of the estimator in Eq. (4) into terms that are independent of the vertex $\mathbf{x}_1$, at which we calculate the radiance, and the remaining terms illustrated in Fig. 6:

$$f(\mathbf{x}_1)\, G(\mathbf{x}_1,\mathbf{x}_2)\, \hat{V}(\mathbf{x}_1,\mathbf{x}_2)\, f(\mathbf{x}_2). \tag{5}$$

In this way, a single path prefix $\mathbf{x}_2 \ldots \mathbf{x}_k$ can be reused to calculate illumination at a large number of different points $\mathbf{x}_1$, allowing for an efficient path reuse. Specifically, instant radiosity precomputes a number of path prefixes of the form $\mathbf{x}_2 \ldots \mathbf{x}_k$ using a random walk starting from a light source ($\mathbf{x}_k$) and storing their end vertices $\mathbf{x}_2$ as *virtual point lights*. With each VPL the algorithm also stores the partially evaluated estimator from Eq. (4), which is usually referred to as the VPL "flux" (specific formulas are given in Section 4). Other information saved with the VPL include the terms necessary to use the VPL to "illuminate" or to "connect to" a given point $\mathbf{x}_1$, i.e. to evaluate the terms in Eq. (5). This includes a reference to the scattering function at the VPL location $\mathbf{x}_2$, the incident direction $\mathbf{x}_3 \to \mathbf{x}_2$, as well as the surface normal $\mathbf{n}_{\mathbf{x}_2}$ if $\mathbf{x}_2$ resides on a surface. Note that for VPLs on surfaces with anisotropic BRDFs we also need to store the tangent vector.

In practice, it is often assumed that the BRDF at a surface VPL $\mathbf{x}_2$ is Lambertian, i.e. that $f_r(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3)$ is independent of $\mathbf{x}_1$ and $\mathbf{x}_3$. In this case, we do not need to store the incident direction $\mathbf{x}_3 \to \mathbf{x}_2$ and we can premultiply the VPL "flux" [W] by the BRDF value [sr^{-1}], to obtain the VPL "intensity" [W.sr^{-1}]. It is often simpler and less error-prone to think about VPLs in terms of partial evaluation of the estimator in Eq. (4) than in terms of flux or intensity.

4. Generation of Virtual Point Lights

In this section we describe the common random walk procedure used to distribute VPLs and discuss some of its variants whose purpose is to direct the VPLs into visually important parts of the scene.

4.1. Random Walk VPL Distribution

In order to distribute VPLs in a scene, we can use the same random walk procedure that is used to distribute photons in photon mapping [Jen01]. We start by tracing N light paths, originating from light sources, which yield M VPLs; one at each bounce (vertex) of the light path. Since the VPL tracing procedure proceeds from the light source, it will be more natural to change the path vertex indexing scheme for a short while, and start with 0 at the light source and increment by one for each scattering event. To avoid confusion with the previous notation, we use $\mathbf{v}$ to denote path vertices indexed from the light source.

The random walk starts by generating the first path vertex on a light source with a probability density function (PDF) $p(\mathbf{v}_0)$ (usually proportional to radiant exitance) and proceeds as follows:

1. Initialize $j := 0$.
2. *Sample the next path vertex.* First, we sample direction ω_j from a PDF $p(\omega_j)$ proportional to the directional emission of the light source (for the first path vertex, $j = 0$) or to the generalized scattering function (for other vertices, $j > 0$). In scenes without participating media, this direction uniquely determines the next path vertex. To account for media, we need to sample the scattering distance t_j along the ray $\mathbf{r}_j(t) = \mathbf{v}_j + t\omega_j$. We usually use a PDF $p(t_j)$ proportional to the medium transmittance. For homogeneous media an analytic expression for the PDFs exist [LW96], otherwise we can use Woodcock tracking [WMHT65]. If the sampled distance is beyond the nearest surface along ω_j, the next path vertex $\mathbf{v}_{j+1}$ will be the surface intersection point. Otherwise, the next vertex resides in the medium.
3. *Create a VPL.* A virtual point light is stored at the position of the generated path vertex $\mathbf{v}_{j+1}$. The VPL "flux" is calculated as

$$\Phi_{j+1} = \begin{cases} \dfrac{L_e(\mathbf{v}_0 \!\to\! \mathbf{v}_1)\, D_{\mathbf{v}_0}(\mathbf{v}_1)\, \tau(\mathbf{v}_0, \mathbf{v}_1)}{p(\mathbf{v}_0)\, p(\omega_0)\, p(t_0)} & \text{if } j = 0 \\[2ex] \Phi_j\, \dfrac{f(\mathbf{v}_j)\, D_{\mathbf{v}_j}(\mathbf{v}_{j+1})\, \tau(\mathbf{v}_j, \mathbf{v}_{j+1})}{p(\omega_j)\, p(t_j)\, q_j} & \text{if } j > 0. \end{cases} \qquad (6)$$

   As mentioned before, for every VPL we also store a reference to the scattering function $f(\mathbf{v}_{j+1})$ at the VPL location, the incident direction ω_j, as well as the surface normal $\mathbf{n}_{\mathbf{v}_{j+1}}$ if $\mathbf{v}_{j+1}$ resides on a surface.
4. *Terminate path.* Use Russian roulette to terminate the random walk with probability $(1 - q_{j+1})$, where the survival probability q_{j+1} is usually proportional to the albedo of the surface or volumetric point $\mathbf{v}_{j+1}$ [PH10].
5. If the random walk survives the Russian roulette, set $j := j + 1$ and go to step 2.

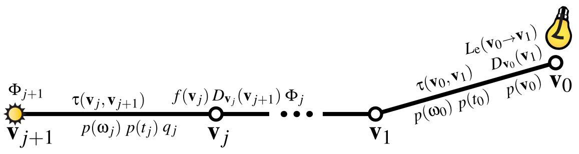
Φ_{j+1}
$\tau(\mathbf{v}_j, \mathbf{v}_{j+1})$
$p(\omega_j)\ p(t_j)\ q_j$
$f(\mathbf{v}_j)\ D_{\mathbf{v}_j}(\mathbf{v}_{j+1})\ \Phi_j$
$\mathbf{v}_{j+1}$
$\mathbf{v}_j$
$\mathbf{v}_1$
$L_e(\mathbf{v}_0 \to \mathbf{v}_1)$
$D_{\mathbf{v}_0}(\mathbf{v}_1)$
$\tau(\mathbf{v}_0, \mathbf{v}_1)$
$p(\omega_0)\ p(t_0)$
$p(\mathbf{v}_0)$
$\mathbf{v}_0$

Figure 7: *This illustration shows the terms that are required to compute the "flux" of a VPL during the construction of random walks.*

After finishing all random walks we divide each VPL's "flux" by the total number of generated light paths N.

Switching back to our original notation and path vertex indexing scheme, one random walk described above creates a number of path prefixes $\mathbf{x}_2\mathbf{x}_3$, $\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$, etc., that can be used to evaluate the estimator in Eq. (4). Indeed, the "flux" of the VPLs created by the above procedure exactly corresponds to all the terms in the estimator that are independent of $\mathbf{x}_1$.

4.2. Improved VPL Generation

We now discuss various approaches for generating VPLs where they are most needed for a given camera view. The need to develop such alternate VPL distribution approaches stems from the fact that the basic VPL tracing algorithm, described above, may often generate many VPLs in regions where they do not have a significant contribution to the visible parts of the scene. As an example, consider a large environment with only a small portion of it visible to the camera. In addition, the density of VPLs generated by the basic VPL tracing algorithm along concave geometry features is usually insufficient to faithfully render local interreflections.

4.2.1. Rejection of Unimportant VPLs

A simple approach to improve VPL distribution, proposed by Georgiev and Slusallek [GS10], is to reject VPLs that do not significantly contribute to the image. The goal of the algorithm is to generate a number of VPLs, all having approximately similar total contribution to the image, denoted Φ_v. The algorithm generates candidate VPLs using the VPL tracing algorithm described above. For each candidate VPL the total image contribution Φ_i is estimated. The VPL is then accepted with the probability P_i^{acc} given by the ratio of the actual VPL contribution Φ_i to the target "average" contribution Φ_v:

$$P_i^{\text{acc}} = \min\left\{ \frac{\Phi_i}{\Phi_v} + \varepsilon, 1 \right\}. \qquad (7)$$

If accepted, the VPL flux is divided by P_i^{acc} to ensure unbiasedness.

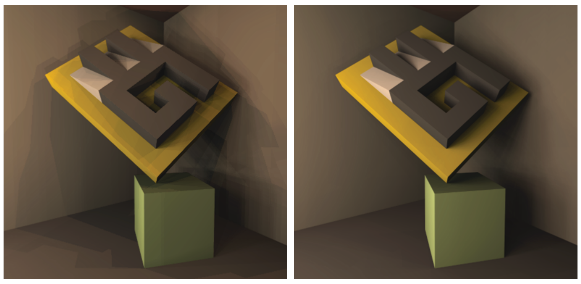

Figure 8: *A scene rendered with the same number of VPLs by standard VPL sampling (left) and a VPL rejection sampling algorithm [GS10] (right). Images courtesy of Iliyan Georgiev.*

A simple way to estimate the target "average" VPL contribution Φ_v is to run a number of pilot VPLs and render a low-resolution image. When rendering an animation, it is also possible to use information from the previous frame. To estimate the image contribution Φ_i of a candidate VPL, we can render a low-resolution image by picking only a couple of pixels. The estimates of Φ_v and Φ_i do not need to be particularly accurate: the algorithm will produce correct results no matter how accurately the VPL contribution is estimated. Fig. 8 shows an example of the result of this algorithm.

4.2.2. Metropolis Instant Radiosity

The rejection sampling approach described above is simple but suffers from an important disadvantage: many candidate VPLs may need to be generated before one VPL is accepted. The Metropolis Instant Radiosity algorithm, proposed by Segovia et al. [SIP07], addresses this problem by replacing the standard VPL tracing algorithm, based on independent random walks, by a Metropolis-Hastings sampler.

Consider a light path that connects a light source to the camera. As discussed in Section 3, the second vertex from the camera can be interpreted as a VPL. Because the path connects the light to the camera, a VPL generated this way is likely to have an important image contribution. We can now use the Metropolis-Hastings procedure, as in the Metropolis Light Transport algorithm [Vea97], to explore the space of all possible light paths by proposing and probabilistically accepting path mutations. Every time a path is mutated, the second vertex from the camera of the mutated path yields a new VPL. Segovia et al. [SIP07] shows that all VPLs created in this way contribute the exact same total flux to the image.

Discussion. Though very different, both the VPL rejection algorithm [GS10] and Metropolis Instant Radiosity [SIP07] generate VPL sets where each VPL has roughly the same contribution to the image. From this, we can expect that the VPL sets generated by both algorithms will be of similar quality. For complex scenes, where light bounces many times to reach the camera, the rejection algorithm may perform poorly because it will reject many VPLs. On the other hand, while the VPL rejection approach is trivial, Metropolis Instant Radiosity requires a substantial implementation effort. Finally, none of the two algorithms addresses the local interreflection problem because the VPL distribution is driven by the contribution of VPLs to the entire image.

4.2.3. Sampling VPLs from the Camera

The density of VPLs generated by the VPL distribution algorithms discussed so far is usually insufficient to faithfully render local interreflections in geometry cavities. To deal with this problem, it is more reasonable to distribute the VPLs by tracing paths from the camera instead of starting from the light sources. This approach is bound to produce VPLs in locations important for the image to be rendered but there are some important technical issues that need to be taken care of: first, we need to explicitly connect these VPLs to the light sources so that they can form complete light transport paths. Second, computing the VPL flux involves the evaluation of the probability density of generating the particular VPL position, which is more complex and costly when the VPLs are generated from the camera.

The idea of generating VPLs by tracing paths from the camera appeared in [SIMP06a] under the name Bidirectional Instant Radiosity. It was used by Davidovič et al. [DKH*10], who refer to the VPLs generated from the camera as local VPLs (as opposed to global VPLs, generated by tracing paths form the light sources). We describe the latter approach in more detail in Section 5.3.

5. Lighting in Many-Light Methods

Once the VPLs are generated, many-light algorithms use them to illuminate the scene. This amounts to evaluating the outgoing radiance, Eq. (2), at a number of locations in the scene using the estimator in Eq. (4). As discussed in Section 3, this can be done by summing over all M VPLs, adding together their "flux" Φ_i weighted by the terms from Eq. (5):

$$\langle L(\mathbf{x}_1 \to \mathbf{x}_0) \rangle = \sum_{i=1}^{M} f(\mathbf{x}_1)\, G(\mathbf{x}_1, \mathbf{x}_2^i)\, \hat{V}(\mathbf{x}_1, \mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i, \quad (8)$$

where $\mathbf{x}_2^i$ is the position of i-th VPL.

Although the equation above—the core part of many-light methods—is fairly simple and can be evaluated efficiently, closer inspection reveals one major problem of these algorithms. The geometry term $G(\mathbf{x}, \mathbf{y})$ used when connecting a VPL to a surface point contains a singularity, namely the inverse squared distance between the shading point and the VPL. Since the distance can be arbitrarily small—in the limit zero—the contribution of a VPL to a shading point, and thus the variance of the estimator, is possibly unbounded. In practice, this leads to degraded rendering quality due to distracting artifacts visible as high intensity splotches (Fig. 9). Note

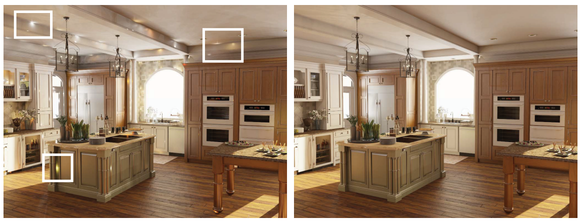

Figure 9: *When not accounted for, the singularity in the geometry term produces high intensity splotches, which degrade the rendering quality (left). Several techniques address this problem producing images that do not suffer from these artifacts (right). Images courtesy of Edgar Velázquez-Armendáriz.*

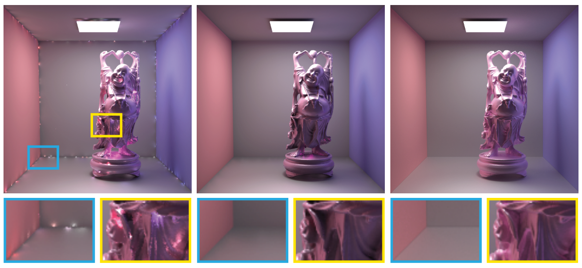

Figure 10: *Comparison of the unbounded (left) and bounded (middle) VPL renderings to an unbiased solution (right).*

that correlating the estimates of indirect illumination (by always connecting to the same set of VPLs) does not compromise the mathematical correctness of the algorithm: the estimation is unbiased, but the variance for some points $\mathbf{x}_1$ is very high and the rendering suffers from structured artifacts.

A straightforward and popular approach to avoid these artifacts is to bound the geometry term to a user-defined maximum b. We will denote this as the *bounded geometry term*:

$$G_b(\mathbf{x},\mathbf{y}) = \min(G(\mathbf{x},\mathbf{y}), b).$$

By substituting $G_b(\mathbf{x},\mathbf{y})$ for $G(\mathbf{x},\mathbf{y})$ in the estimator in Eq. (8), we avoid the artifacts, however, we partly suppress the short distance light transport, and thus obtain a biased solution. Images rendered with the bounded geometry term suffer from darkened regions, in particular in cavities and corners (see Fig. 10 for examples). Furthermore, selectively suppressing light transport can have a severe impact on the material appearance [KFB10] and should thus be avoided.

5.1. Bias Compensation using Final Gathering

Kollig and Keller [KK04] express the bias $B(\mathbf{x}_1 \to \mathbf{x}_0)$, i.e. the energy removed due to bounding, as the difference between the transport obtained with the original and the bounded geometry term. Adapting Eq. (8) to estimate only the *removed VPL lighting* yields:

$$\langle B(\mathbf{x}_1 \to \mathbf{x}_0) \rangle = \sum_{i=1}^{M} f(\mathbf{x}_1)\, G_r(\mathbf{x}_1, \mathbf{x}_2^i) \hat{V}(\mathbf{x}_1, \mathbf{x}_2^i)\, f(\mathbf{x}_2^i)\, \Phi_i,$$

with the *residual geometry term* $G_r(\mathbf{x},\mathbf{y})$ defined as:

$$\begin{aligned} G_r(\mathbf{x},\mathbf{y}) &= G(\mathbf{x},\mathbf{y}) - G_b(\mathbf{x},\mathbf{y}) \\ &= \max(G(\mathbf{x},\mathbf{y}) - b, 0). \end{aligned}$$

In order to correct for the bias, Kollig and Keller evaluate a compensation term which is very similar to the rendering equation; the only difference is that the original geometry term is replaced by $G_r(\mathbf{x},\mathbf{y})$. This compensation corresponds to removed VPL lighting only, and is added to the result of the bounded VPL lighting. The authors propose to evaluate the compensation using localized final gathering: they shoot rays to nearby surfaces, estimate the incident light at the intersection points, and transport back to the original shading point only such amount of energy that corresponds to the removed VPL lighting. Although the compensation is localized, the incident light is estimated by adding the contribution from all light sources, including VPLs. As bounding can also occur during the compensation, the technique is recursive and can quickly degenerate to path tracing (see Fig. 11). Raab et al. [RSK08] extended this strategy to handle scenes with participating media. Note that the cost of bias compensation to obtain unbiased results can exceed the computation time of the bounded transport by orders of magnitude, which makes it less attractive for practical applications.

5.2. Approximate Bias Compensation

In order to preserve one of the prominent characteristics of many-light approaches—short and predictable rendering times—two publications address the high computational cost of bias compensation by making the integration of the residual transport more efficient.

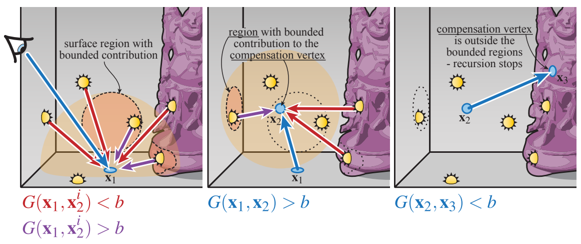
surface region with bounded contribution
$\mathbf{x}_1$
$G(\mathbf{x}_1,\mathbf{x}_2^i) < b$
$G(\mathbf{x}_1,\mathbf{x}_2^i) > b$
region with bounded contribution to the compensation vertex
$\mathbf{x}_2$
$\mathbf{x}_1$
$G(\mathbf{x}_1,\mathbf{x}_2) > b$
compensation vertex is outside the bounded regions - recursion stops
$\mathbf{x}_3$
$\mathbf{x}_2$
$G(\mathbf{x}_2,\mathbf{x}_3) < b$

Figure 11: *Since the contribution of nearby VPLs is bounded (purple arrows) (left), Kollig and Keller [KK04] construct a new path segment for gathering the residual light transport (middle), and continue recursively until the vertex is outside the regions where bounding occurs (right).*

Novák et al. [NED11] note that the missing energy can be restored by computing additional light transport on a once-computed bounded solution. To intuitively explain their approach we will use the recursive formulation of the rendering equation and use an operator notation (in spirit of Arvo et al. [ATS94]) to define a generalized transport operator $\mathbf{T}$:

$$(\mathbf{T}L)(\mathbf{x}\to\mathbf{z}) = \int f(\mathbf{x})\, G(\mathbf{x},\mathbf{y})\, \hat{V}(\mathbf{x},\mathbf{y})\, L(\mathbf{y}\to\mathbf{x})\, d\mu(\mathbf{y}). \quad (9)$$

Additionally, we also define a bounded transport operator $\mathbf{T}_b$ and a residual transport operator $\mathbf{T}_r$, which both differ from $\mathbf{T}$ only in the geometry term that is replaced by $G_b(\mathbf{x},\mathbf{y})$ and $G_r(\mathbf{x},\mathbf{y})$, respectively. Assuming that VPLs represent all light in the scene except for the emission, denoted as $\hat{L} \approx L - L_\mathrm{e}$, the radiance leaving an arbitrary point can be concisely expressed as:

$$\begin{aligned} L &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}_b\hat{L} + \mathbf{T}_r\hat{L} \\ &\approx L_\mathrm{e} + \mathbf{T}L_\mathrm{e} + \mathbf{T}_b\hat{L} + \mathbf{T}_r(L - L_\mathrm{e}) \\ &\approx L_\mathrm{e} + \sum_{i=0}^{\infty} \mathbf{T}_r^i \left(\mathbf{T}L_\mathrm{e} + \mathbf{T}_b\hat{L}\right). \end{aligned}$$

The first step in the derivation above is to replace the approximation of the residual indirect illumination, $\mathbf{T}_r\hat{L}$, by the accurate corresponding light transport $\mathbf{T}_r(L - L_\mathrm{e})$, and then recursively expand the equation. The result shows that the full transport can be computed from direct illumination and bounded transport $\mathbf{T}L_\mathrm{e} + \mathbf{T}_b\hat{L}$ only. However, it has to be stored first to apply the residual transport operator iteratively. Novák et al. chose to store the bounded transport in screen-space which allows for an efficient, hierarchical computation of the residual energy using filter-like operations. The drawback is that the compensation energy from invisible surfaces is lacking, and surfaces seen under grazing angles require special treatment due to undersampling in the image.

Engelhardt et al. [ENSD12] address bias compensation in participating media. Their approach is inspired by the original path tracing-based method of Raab et al. [RSK08], simply because in volumes it is impractical to store the bounded transport for later compensation. They carefully analyze the residual energy and provide approximations leading to significant speedups in rendering (scalable up to interactive frame rates) with very little impact on the visual quality. First, they show that two compensation steps usually recover almost the entire residual energy, and that a locally homogeneous medium can be assumed during compensation. They demonstrate that visibility tests to new compensation vertices can be omitted, making the approach GPU-friendly; lastly, they also discuss an optimized sampling strategy to decrease variance when connecting to VPLs.

5.3. Bias Compensation Using Local Lights

Similarly to the previously mentioned compensation techniques, Davidovič et al. [DKH*10] separate light transport into the bounded global component, and the residual local component. The global component accounts for the long-distance light transport, while the local component corresponds to the short-range interreflections and indirect glossy highlights. They take advantage of the specific structure of each component and design a solution tailored for each of them independently. Specifically, they handle as much energy as possible in the global component, which leaves only the local interreflections for the local component. This approaches turns out to be substantially more efficient than a general global illumination solution.

The global component is calculated by the classic, “global” VPLs, traced from the light sources. To handle a high number of global VPLs efficiently, any of the scalable many-light methods, described in Section 6, could be used. Davidovič et al. [DKH*10] propose a GPU-based, visibility clustering approach, similar to Matrix Row Column sampling [HPB07].

The local component is handled by the so called “local” VPLs, distributed by tracing paths from the camera. Since the local VPLs are designed to calculate localized transport in regions with concave geometry, a local VPL only contributes to a small tile of pixels around the pixel through which the path used to sample that local VPL was traced. In addition, it is assumed that there is no occlusion between the local VPLs and the pixel that it contributes to, and thuso the visibility checking can be skipped when calculating the contribution of local VPLs to the pixels in the tile. This approximation, which is at the origin of the method’s efficiency, is made possible by the global component containing most of the energy and handling most of the indirect shadows. Finally, the complete global illumination solution is obtained by summing the local and the global components.

5.4. Avoiding the Singularity: Virtual Spherical Lights

Rather than compensating for the bias due to the bounded light transport, Hašan et al. [HKWB09] try to avoid the singularity in the first place. They notice that the intensity of bright splotches can be exacerbated by glossy BRDFs, and the compensation can thus become arbitrarily expensive. This happens when the shading point and the VPL are aligned such that either of the two BRDF terms (or both) have a large value in that direction. It is worth noting that while the artifacts due to the geometry term are generally localized (occurring mostly in corners), the latter can happen across large distances if the materials in the scene are sufficiently glossy.

The authors first introduce the concept of a photon light: a point light that distributes its energy over surfaces within a sphere with a certain radius. The name was inspired by the connection to photon mapping, where each photon contributes its energy to nearby surfaces. In order to efficiently evaluate the contribution, they use the visibility of the original VPL for all points inside the photon light and also assume that surfaces around the VPL are planar. This yields a new lighting primitive called the virtual spherical light (VSL). The advantage of VSLs is that they replace the point-to-point evaluation, which is the cause of the singularity, by an integration over the solid angle subtended by the spherical light. Additionally, the integration averages the product of the two BRDFs over the solid angle, and thus, VSLs effectively avoid the singularity artifacts by blurring them over nearby surfaces. The concept of photon light introduces bias, similar to photon mapping with final gathering. But unlike bounding, it preserves the energy by redistributing it spatially over nearby surfaces. VSLs can be combined with scalable many-light methods, e.g. Lightcuts [WFA*05] or Matrix Row-Column Sampling [HPB07] (see Section 6).

5.5. Avoiding the Singularity: Virtual Ray/Beam Lights

For scenes containing participating media, Novák et al. [NNDJ12b] propose to utilize entire segments of the light and eye paths. Instead of creating discrete VPLs at vertices of the random-walk, they turn each linear segment of the walk into a continuous virtual ray light (VRL). The point-to-point evaluation is then replaced by an integration over the length of the ray light, which is evaluated numerically using importance sampling. Spreading the energy along line segments provably reduces the degree of the singularity, and in practical scenarios no, or very little bounding is required, reducing the need for subsequent bias compensation.

In order to remove the singularity completely, the same authors propose to inflate VRLs into virtual beam lights (VBLs) [NNDJ12a]. The idea is to distribute the energy of the infinitesimal ray light over a cylindrical region with finite thickness (see Fig. 12). In the spirit of VSLs, this avoids the singularity at the cost of slightly blurring the illumination, but the appearance of materials and media is better preserved than when bounding the transport. The authors also formulate the rendering algorithm progressively, ensuring that the thickness of each VBL reduces over time, the blurring diminishes, and the integration achieves consistent results.

6. Scalability with Many-Light Methods

In the preceding sections, we have discussed how VPLs are generated, represented, and evaluated. In this section, we shift our focus to another problem: given a large set of VPLs, how can we compute their contribution efficiently? This is a critical question as the accuracy of many light methods strongly depends on the number of VPLs used. With only a few VPLs, the illumination can only be coarsely reconstructed. While this may be sufficient for small scenes or in real-time applications, generating high-quality renderings in complex scenes requires capturing many detailed, highly localized, indirect illumination effects such as glossy highlights, indirect shadows and proximity color bleeding. Accurately simulating these effects may require thousands or millions of VPLs. Since the effects of individual VPLs would often be imperceptible, a linear, brute force evaluation of Eq. (8) for a million VPLs would be prohibitively expensive and inefficient. Instead, one would prefer an accurate but approximate evaluation that requires far less computation. We will call algorithms *scalable* if their cost increases slowly, or sub-linearly, with the number of VPLs used.

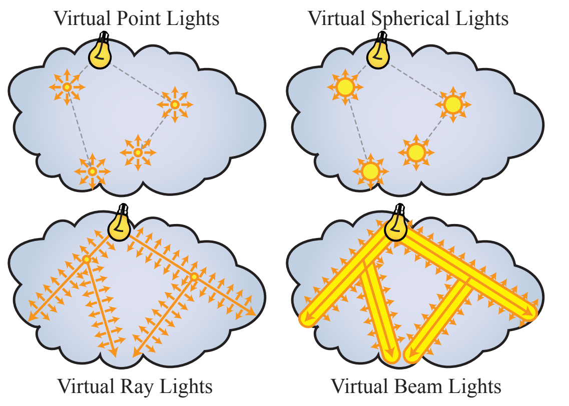

Figure 12: *Virtual point lights can be inflated into virtual spherical lights to avoid singularities (especially with glossy BRDFs), as suggested by Hašan et al. [HKWB09]. For rendering scenes with participating media, Novák et al. propose to turn linear segments of light paths into ray lights [NNDJ12b] and inflate them into beam lights [NNDJ12a], effectively avoiding singularities.*

Alternatively, increasing scalability can be viewed as variance reduction. If an algorithm handles a million VPLs while only spending the resources for a few hundreds, this is equivalent to decreasing the noise in the estimate while holding the amount of computation fixed. The main Monte Carlo variance reduction techniques are stratification, adaptive sampling, and importance sampling. The scalable algorithms discussed below are based on carefully combining stratification and adaptivity, or caching of importance.

We will discuss several practical algorithms with sub-linear scalability. Because they can render images using even millions of VPLs efficiently, these scalable, many-light algorithms have several advantages [KHA*12]:

- They unify the integration of all illumination sources. Global illumination, environment lighting, and direct illumination from complex sources are simply converted to VPLs and evaluated with a single algorithm.
- Their sub-linear scalability makes them very efficient. They can achieve the quality of other methods at a fraction of the cost.
- They separate the quality of the lighting representation (how many VPLs are generated) from the quality of the evaluation (how much effort is spent evaluating those VPLs). By approximately evaluating a large set of VPLs, these algorithms can generate low cost, low noise preview images that accurately predict final results.

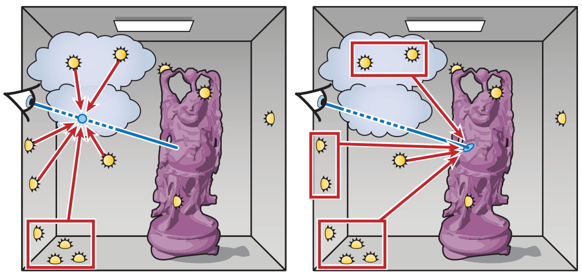

Figure 13: *Scalable many-light algorithms can achieve sub-linear cost when evaluating VPL lighting. They cluster VPLs to be able to adjust the accuracy of the many-light evaluation. For example, a group of distant VPLs might be replaced with a single brighter representative.*

- They provide scalability across a wide range of models with different geometry, lighting, occlusion, and material properties.

6.1. VPL Clustering

The methods discussed in this section exploit a common insight: within a large set of VPLs, each VPL does not contribute equally. Many VPLs have low importance because they contribute very little to a region of interest; for example if the VPLs are far away or occluded. However, typically a small number of VPLs are very important, such as VPLs that contribute to a glossy highlight, and these must be handled accurately. A general, scalable algorithm tries to exploit this non-uniform VPL importance to reduce computation. It seeks to identify and evaluate all of the most important VPLs while only sparsely evaluating the unimportant ones.

While the algorithms discussed below differ in how they estimate importance and select a set of VPLs to evaluate, they all use the same framework. Each algorithm clusters similar VPLs together. They choose a clustering that places unimportant VPLs in large clusters and important VPLs in smaller clusters, see Fig. 13 for an illustration. It is then assumed that the VPLs within a cluster are sufficiently similar that their aggregate effect can be approximated by evaluating just a single, brighter, representative VPL. If the representative is chosen randomly from the VPLs within the cluster and its power scaled appropriately, the sum of over these representative approximations is equivalent to a stratified, Monte Carlo evaluation of the path integral, where only a single sample is drawn from the domain of each cluster. If these algorithms can find a set of clusters—typically called a *cut*—that is much smaller than the number of all VPLs, the cut approximation becomes a scalable alternative to brute force evaluation. We will discuss three (classes of) methods:

- Lightcuts [WFA*05, WABG06, WKB12],
- Matrix Row-Column Sampling [HPB07, HVAPB08], and
- LightSlice [OP11].

They differ in how they estimate VPL importance and in how often cuts are computed to generate an image but they all share the fundamental idea of cut selection. Multidimensional Lightcuts [WABG06] extends the idea of cuts beyond lights to all light-receiver pairs to achieve scalability across a wider range of effects and dimensions. Finally, the last method discussed in the section chooses most relevant VPLs for a shading point rather than clustering them.

6.2. Lightcuts

Lightcuts [WFA*05] was the first practical, scalable many-light method. At each receiver point where illumination needs to be computed, Lightcuts generates a customized cut based on analytic per-cluster error bounds and a perceptual metric. As a first pass the VPLs are first organized into a binary tree based on spatial and directional similarity. To select a cut for a receiver, we start with a trivial, coarse clustering, such as putting all the lights in a single cluster. This corresponds to a cut consisting of only the root node of the light tree. Then we iteratively select the cluster in the current cut with the highest error bound and refine it replacing it by its children in the light tree. This process is repeated until the error bounds for all clusters in the cut are below a perceptual-based threshold, typically set as 2% of the total. The size of the cut is typically only weakly dependent on the number of VPLs, resulting in very large speed-ups compared to a full evaluation as the number of VPLs grows. The use of analytic error bounds also guarantees that the most important VPLs are always found and evaluated, making the estimation robust. Davidovič et al. [DGS12] describe a progressive, GPU-friendly variant of Lightcuts.

However, using error bounds also has some drawbacks. The error bounds ignore occlusion and are often overly conservative, resulting in larger than optimal cuts. Furthermore, developing good error bounds for novel material models is challenging and difficult if bounds have not yet been developed for sufficiently similar materials. Lightcuts also does not take advantage of the fact that the cuts for nearby receiver points are often quite similar, but instead regenerates a cut from scratch for each point.

6.3. Matrix Row-Column Sampling

In contrast to Lightcuts, which generates a cut per receiver point, Matrix Row-Column Sampling (MRCS) [HPB07] computes a single, global cut for the whole image. Because the cut generation is amortized over the whole frame, this approach has two advantages compared to Lightcuts.

1. As a measure of VPL importance, it replaces the error bounds, which may be expensive or unknown for general materials, with sparsely sampled direct estimates of each VPL's image contribution. This allows MRCS to easily add material and light types (including VSLs), and to include visibility information in the cut selection algorithm.

2. The VPL importance and the VPL contributions can both be estimated by using shadow maps. By using graphics hardware to accelerate their calculation, MRCS can achieve very fast, low-noise rendering.

Of course, there are some disadvantages: because the cut is computed once from sparsely sampled data, MRCS is less adaptive than Lightcuts, and may sometimes miss small features of the VPL illumination, such as glossy highlights that affect only a few pixels.

To compute the global cut, MRCS models the VPL evaluation problem as a large matrix **M**. The rows of **M** represent the surface points visible through each pixel and the columns of **M** represent VPLs. Each entry $\mathbf{M}(i, j)$ represents the fractional energy of the j-th VPL that reaches the camera through the i-th pixel. The key insight of MRCS is that **M** is usually a highly structured, often low-rank, matrix and can be well approximated by the reduced matrix **R** that subsamples the elements of **M**. The MRCS consists of computing **R**, using it to compute a cut and then using that cut to approximate the original matrix **M**.

To compute **R**, MRCS computes small, subsampled images of the scene illuminated by just one VPL. In the matrix model above, each of these images corresponds to a subsampling of one column of **M**. Since it is prohibitively expensive to generate even a small image for each VPL, MRCS also selects a small subset of the VPLs for this calculation. Concatenating all these small images together yields **R**. To compute a cut using **R**, MRCS first clusters the columns of **R** so that VPLs with similar image contributions are grouped together. Then all the other VPLs are mapped to these initial clusters by minimizing an error metric that encodes both the similarity of their columns in **R** and their geometric proximity. The authors prove that their metric is optimal for the given clustering problem and give an efficient two-phase approximation algorithm for computing the cut.

The MRCS approach has also been extended to render full animations [HVAPB08], concatenating the matrices of the animation frames into a 3D array (tensor), and exploiting temporal coherence to drive the number of required cluster representatives even lower. Davidovič et al. [DKH*10] propose a modification of MRCS called visibility clustering.

6.4. LightSlice

The LightSlice [OP11] algorithm combines the idea of locally adapted cuts from Lightcuts with the global optimization advantages of MRCS. The authors of LightSlice noticed that, while Lightcuts can capture detailed illumination effects by recomputing a cut at each receiver point, MRCS demonstrated that many of these cut calculations waste effort recomputing a shared set of global VPL clusters. The LightSlice algorithm improves performance by identifying these shared global clusters once and then reusing them as a starting point for local per-slice cluster refinements. The LightSlice algorithm works as follows:

1. Generate all the receiver points for an image and cluster them based on their geometric proximity into groups called *slices*.
2. Select a representative receiver point from each slice and then run MRCS on the set of all slice representatives. This forms both an initial, global cut for all slices and a reduced matrix **R** describing the light transport of the slice representatives.
3. For each slice i, restrict **R** to a smaller matrix $\mathbf{R}^i$ that contains only the row for slice i and the rows from other nearby slices. Iteratively refine the global clustering for i by using $\mathbf{R}^i$ to identify and split high-cost, local clusters to generated localized cuts for each slice.

By using localized per-slice cuts, LightSlice is able to reduce the average cut size needed compared to a single global cut while also reducing the chance that locally important VPLs will be missed due to the sparse sampling. Also, by reusing an initial global cut as the local starting point, it significantly reduces the cost of cut selection.

6.5. Multidimensional Lightcuts

Multidimensional Lightcuts [WABG06] extends the domain of clusters and cuts to include receiving points as well as lights, to achieve scalable performance across a much wider range of effects. When computing a pixel, we really want the average illumination over a region rather than its value at individual receiving points. For example, the region may extend over an image space for anti-aliasing, over the aperture for depth of field, over time for motion blur, and spatially for effects such a volume rendering. When rendering, these regions are typically converted to many receiver points using sampling, but separately evaluating a cut for each point is inefficient as they often have very different sensitivities to the lights and illumination accuracy requirements. Instead Multidimensional Lightcuts builds a hierarchy over all light-receiver point pairs for a pixel and its cut is a partition of this much larger point-pair space. To make this feasible, the point pair hierarchy, called the product graph, is implicitly represented as the Cartesian product of a light tree and a receiver tree, which is also called the gather tree. The cut selection process is similar to that in the Lightcuts method in that it uses analytic per cluster error bounds and refines the cut until a perceptual threshold is met. One major difference is that cluster refinement can now choose between light or receiver refinement at each step. Overall this technique greatly reduces the rendering cost when computing effects that require both many receiver points per pixel and large numbers of VPLs. The method also adaptively balances effort between different illumination components in each pixel. For example, a bright highlight if present automatically allows other components, such as the volumetric or diffuse illumination, to be computed at reduced accuracy and cost.

Bidirectional Lightcuts [WKB12] extends the previous approach to handle even more effects including glossy reflections, subsurface scattering, and short-range indirect illumination. While its goals are similar to the bias compensation methods mentioned previously, it functions by adding additional receiver points per pixel.

6.6. Importance Caching for VPL Selection

Georgiev et al. [GKPS12] propose a different approach to improve scalability: they choose the most relevant VPLs for any given position in the scene based on importance caching. In a preprocess, the contribution of all VPLs is computed at a number of locations in the scene and cached. When calculating the VPL contributions during image rendering, the cached contributions at a few nearby locations are used as a discrete probability distribution from which the most relevant VPLs are sampled randomly.

Together these scalable approaches have greatly increased the effective number of VPLs that can be used in many-light methods and thus increased the accuracy and image quality that can be achieved.

7. Interactive and Real-Time Many-Light Rendering

The obvious challenge in interactive and real-time rendering, compared to offline methods, is the tight time budget. This expectedly restricts the number of VPLs that can be created and used (typically several hundreds to thousands), and also the types of materials which can be faithfully rendered under such constraints. Conceptually, the main difference resides in the computation of visibility where rasterization is typically used instead of ray casting for VPL generation, shading, and shadowing. The following sections address these aspects, followed by an outlook of rendering participating media (which is particularly challenging at interactive frame rates) and techniques for improving image quality and temporal stability when using a limited number of VPLs.

7.1. Many-Light Generation in Interactive Rendering

In typical real-time scenarios the render times are dominated by shading and shadowing cost, and less by VPL generation. That is, as only a small number of VPLs can be handled, any spatial indexing structure—even if unoptimized or suboptimally built—is sufficient for tracing the few light paths for creating VPLs, and should be used if available.

VPLs can also be created using rasterization only: *reflective shadow maps* [DS05] (RSMs) render the scene from a primary light source similar to a shadow map and thus capture directly lit surfaces. In addition to depth, RSMs store the position, normal, and reflected flux and every pixel can be interpreted as a small light source, or a random subset of pixels can be chosen to serve as VPLs. A similar idea has previously been used to create virtual dipole light sources for rendering translucent objects [DS03]. In the original RSM work, the contribution of these VPLs was accumulated for a low-resolution image and refined at edges during upsampling. Dachsbacher and Stamminger [DS06] later proposed to accumulate the lights' contributions using deferred shading and with bounded regions of influence; they also introduced importance sampling of the underlying RSM.

Ritschel et al. [REH*11] propose to choose VPLs from RSMs by estimating their contribution to the image, and thus obtaining a probability factor for importance sampling the RSM. In both cases, longer light paths can be created by recursively computing further RSMs computed for the previously generated VPLs (used in [RGK*08]). Note that various approaches for casting rays using rasterization or based on voxelization exist, and can of course be used to trace light paths, but are not covered in this report.

7.2. Many-Light Shading in Interactive Rendering

Although programmable graphics hardware is able to shade from virtually arbitrary many light sources, many-light rendering is almost exclusively accompanied by deferred shading techniques [DWS*88, ST90]. Bounding the regions of influence of VPLs (as in [DS06]) to speed up shading is often not the preferred solution as it removes light transported over larger distances. Instead, the shading signal is often subsampled and subsequently interpolated. To this end, most of the techniques employ interleaved sampling [KH01], which has been first used for many-light rendering by Wald et al. [WKB*02]: shading of a single pixel is not performed using all VPLs, but instead each pixel is lit only by a disjoint subset of VPLs. The reasoning is that neighboring pixels often represent nearby and similarly oriented geometry and their shading would thus be similar as well. The interleaved shading is then transferred across neighboring pixels in a post-processing step using an edge-aware image filter. Segovia et al. [SIMP06b] describe a GPU-friendly implementation of interleaved sampling where the deferred shading buffers are reorganized such that pixels lit by the same subset of VPLs are stored in the same tiled sub-buffer. Interleaved sampling greatly speeds up the rendering, however, it is prone to aliasing artifacts with highly detailed geometry and normal mapping, and problematic with glossy BRDFs. These issues have been addressed by Segovia and Wald [SW10] who propose to filter the incoming radiance field, instead of the incident irradiance which is dependent on the surfaces' normals.

Nichols and Wyman [NW09, NW10] propose a hierarchical shading technique based on the observation that indirect illumination in regions with smooth surfaces varies slowly, while geometric detail requires more shading evaluation. Their technique makes use of a min-max mipmap of the depth buffer to detect discontinuities. Indirect illumination is then computed in multi-resolution deferred shading buffers where smooth regions (little variation in the min-max mipmaps) are shaded in low-resolution buffers, and detailed regions in high-resolution buffers. The resolution pyramid is finally combined to the final image using an adapted bilinear interpolation technique. Note that interleaved sampling approaches are orthogonal to shadow computation from VPLs, while multi-resolution splatting assumes a smooth shading signal (i.e., it does not detect shadow boundaries and thus blurs across them).

Further acceleration of the shading computation with a large number of light sources can be achieved by *tiled shading*: the image plane is subdivided into tiles, and for each tile a list of light sources potentially affecting the visible surfaces is built. Each tile can then be processed independently without evaluating all light sources [OA11]. Olsson et al. [OBA12] extended this idea and cluster light sources by their tile and depth.

7.3. Visibility Computation in Interactive Rendering

Whenever we compute shading from VPLs, we also have to determine whether the shading point is actually lit or occluded. To this end, almost all interactive methods compute the (hemi-)spherical visibility per VPL before shading. The original instant radiosity implementation employed a variant of shadow volumes [Kel97], but almost all later implementations rely on shadow mapping as this technique is robust, flexible, and its main disadvantage – jagged shadow edges – is not crucial when the contribution of hundreds of VPLs is accumulated. As VPLs illuminate a (hemi-)sphere, shadow maps are rendered using a suitable parameterization, e.g. a cube, paraboloid [HS98], or octahedron [ED08] map.

Computing the per-VPL visibility, or shadow map, is often the most time-consuming part of interactive many-light implementations. One way to speed up this step is to avoid repetitive computation and exploit temporal coherence. Laine et al. [LLK07] propose to keep VPLs for as long as they contribute to the image, and invalidate and reposition only few of them per frame. Their method maintains VPLs for single-bounce indirect illumination only (a limitation which could be relaxed when computing VPLs with ray casting), and it is restricted to static scenes, or indirect light from static on dynamic geometry only.

A different strategy is to compute visibility less accurately, which has been shown to be sufficient for indirect illumination in many cases [YCK*09]. Ritschel et al. [RGK*08] compute low-resolution and low-quality *imperfect shadow maps* (ISM): hundreds to thousands of shadow maps are rendered in parallel from a point representation of the scene geometry. Since only a few thousand point samples are used per each shadow map, the resulting maps contain holes and need to be filled using an image-space heuristic. The point representation is precomputed, but can be deformed with the scene to support animations. In a follow-up work, Ritschel et al. [REH*11] describe how these point sets can be chosen in a view-adaptive manner, improving the quality of the shadow maps. Holländer et al. [HREB11] also address the many-view rendering problem and present an incremental and GPU-friendly LoD algorithm, which can be used to compute shadow maps. Micro-rendering [REG*09] handles visibility computation using a point hierarchy and massively-parallel hybrid rasterization-raycasting technique to render water-tight hemispherical images. It also supports warping of the hemisphere, as it was originally designed for fast final gathering for both diffuse and glossy surfaces.

Popov et al. [PGSD13] propose a global visibility caching technique that can also be used for accelerating many-light rendering. They cache and reuse visibility queries by quantizing the endpoints of the two end points of the query. To control the error, the quantization adapts to variations in geometry, sampling densities and the light signal.

Recently, another approximate representation gained much interest: any scene geometry can be voxelized allowing for simple ray marching to compute visibility. Voxelization itself is a large research field and we thus refer to Crassin's PhD thesis [Cra11] for an exhaustive overview.

7.4. Visibility in Participating Media

When a scene contains participating media the mutual visibility between two points is non-binary, as light traveling along a ray can be partially absorbed or outscattered. Instead of depth, *deep shadow maps* [LV00] sample and store fractional visibility from one point through every pixel (corresponding to directions) at all possible depths. Such functional representation allows looking up transmittance or visibility at constant cost. Salvi et al. [SVLL10] present *adaptive volumetric shadow maps* (AVSMs), a GPU-friendly variant of deep shadow maps, which stores an approximation to the monotonic transmittance function. AVSMs has also been adapted by Engelhardt et al. [ENSD12] for interactive rendering of participating media. Note that gathering the contribution of VPLs along eye rays amounts to accumulating single-scattering due to point lights. Numerous methods exist to compute single-scattering efficiently, however, most of these works target high-quality rendering of crepuscular rays (see e.g. [ED10,CBDJ11,Wym11] or [RDGK12] for an overview), which is typically not necessary for VPLs in scattering media, or they rely on analytic integration restricting to homogeneous media and neglecting visibility [Peg09].

7.5. Improving Quality and Temporal Stability

A crucial factor for the quality of renderings is the number of VPLs: a low number of VPLs might be sufficient for (mostly) diffuse scenes, while significantly more is necessary for scenes with glossy surfaces [KFB10]. Moreover, VPLs are generated using random walks and even if the same random numbers are used, the generated VPLs can have different locations or contributions if the scene changes—and this is the source of distracting temporal flickering. Obviously the number of VPLs cannot be increased arbitrarily for interactive scenarios and thus special techniques had to be developed.

One possible solution, orthogonal to incrementally updating VPLs [LLK07], is to consider a VPL not as a point light, but instead as an area light that represents indirect illumination from a certain surface area. A straightforward approach to estimate the represented area is to initially create more VPLs and cluster them before shading. The size of the cluster serves is an estimate of the surface area, and this yields, together with the accumulated contribution of the VPLs, a virtual area light [DGRS09]. Visibility from area lights can efficiently be evaluated using soft shadow methods, e.g. [ADM*08]. When using RSMs, clustering can also be computed directly in image space of the RSM [PKD12].

8. Conclusions

It is not without reason that many-light rendering has been attracting attention in the past years: these methods are comparatively fast, often GPU-friendly, there exist variants with good, sub-linear scalability, they enable the rendering of high-quality images with a wide range of materials, and above all, the rendering cost is usually well predictable. However, this comes at a price: as we have seen, one inherent aspect is the generation of light paths yielding the virtual lights, which are then used to illuminate the scene. A coarse sampling of light paths can lead to problems with sharp modes in the scattering functions (BRDF on surfaces, phase functions in media), i.e. when the directionality of light transport is high. Moreover, temporal stability has to be explicitly addressed, e.g. in VPL generation or clustering. Fig. 14 presents an overview of the discussed methods providing a rough classification and description of goals and peculiarities.

In principle, many-light methods can be derived from the very same bidirectional path tracing framework as many other methods, including photon mapping. Photon mapping—which has also been an active research area recently (e.g. see [HOJ08, HJ09, KZ11, HJ11, JNSJ11, KD13b])—uses the same tracing of light paths, then followed by density estimation or final gathering (which is essentially the same as gathering contributions from VPLs). A recent trend in global illumination research is to subsume conceptually related methods into a generic framework to construct different sub-spaces of the path space, i.e. compute different light transport phenomena, with different path construction strategies (e.g. [GKDS12,HPJ12,KD13a]). The work of Dammertz et al. [DKL10] is one good example, where VPL rendering for diffuse interreflections is combined with specular gathering for glossy and specular reflections and photon tracing for caustics. This per se obvious idea of taking the best of all possible worlds is not just a straightforward combination of existing methods. We have seen such ideas influencing many-light methods, and believe that this will lead to more robust and efficient global illumination methods in the future.

Acknowledgements This work was partially supported by the Intel Visual Computing Institute in Saarbrücken/Germany, the DFG grant DA 1200/1-1, and the Czech Science Foundation grant P202-13-26189S.

References

[ADM*08] ANNEN T., DONG Z., MERTENS T., BEKAERT P., SEIDEL H.-P., KAUTZ J.: Real-time, all-frequency shadows in dynamic scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 27*, 3 (2008), 1–8. 14

[ATS94] ARVO J., TORRANCE K., SMITS B.: A framework for the analysis of error in global illumination algorithms. In *SIGGRAPH '94* (New York, NY, USA, 1994), ACM, pp. 75–84. 8

[Aut13] AUTODESK INC.: Autodesk® 360 Rendering, 2013. http://rendering.360.autodesk.com/. 2

[CBDJ11] CHEN J., BARAN I., DURAND F., JAROSZ W.: Real-time volumetric shadows using 1d min-max mipmaps. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2011), pp. 39–46. 13

[Cra11] CRASSIN C.: *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes.* PhD thesis, Université de Grenoble, 2011. 13

[DGRS09] DONG Z., GROSCH T., RITSCHEL T., SEIDEL H.-P.: Real-time indirect illumination with clustered visibility. *Proceedings of Vision, Modeling and Visualization* (2009), 187–196. 14

[DGS12] DAVIDOVIČ T., GEORGIEV I., SLUSALLEK P.: Progressive lightcuts for GPU. *ACM SIGGRAPH 2012 Talks* (2012). 10

[DKH*10] DAVIDOVIČ T., KŘIVÁNEK J., HAŠAN M., SLUSALLEK P., BALA K.: Combining global and local virtual lights for detailed glossy illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 143:1–143:8. 6, 8, 11

[DKL10] DAMMERTZ H., KELLER A., LENSCH H.: Progressive point-light-based global illumination. *Computer Graphics Forum 29*, 8 (2010), 2504–2515. 14

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Proc. Eurographics Workshop on Rendering* (2003), pp. 197–201. 12

[DS05] DACHSBACHER C., STAMMINGER M.: Reflective shadow maps. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2005), pp. 203–208. 12

[DS06] DACHSBACHER C., STAMMINGER M.: Splatting indirect illumination. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2006), pp. 93–100. 12

[DWS*88] DEERING M., WINNER S., SCHEDIWY B., DUFFY C., HUNT N.: The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *Computer Graphics (Proc. SIGGRAPH)* (1988), pp. 21–30. 12

[ED08] ENGELHARDT T., DACHSBACHER C.: Octahedron environment maps. In *Proceedings of Vision, Modeling and Visualization* (2008), pp. 383–388. 13

[ED10] ENGELHARDT T., DACHSBACHER C.: Epipolar sampling for shadows and crepuscular rays in participating media with single scattering. *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2010), 119–125. 13

[ENSD12] ENGELHARDT T., NOVÁK J., SCHMIDT T.-W., DACHSBACHER C.: Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proc. Pacific Graphics) 31*, 7 (2012), 2145–2154. 1, 8, 13

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 192:1–192:10. 14

Reference	Abbreviated Paper Title	Category / Goal	Speed	Materials	Clustering	Remark
[Kel97]	Instant Radiosity	diffuse GI	real-time	D	no	basic method
[DS05]	Reflective Shadow Maps	single-bounce GI	real-time	D	no	no VPL visibility, importance sampling of VPLs
[DS06]	Splatting Indirect Illumination	single-bounce GI	real-time	D,(G)	no	no VPL visibility, importance sampling of VPLs
[LLK07]	Incremental Instant Radiosity	single-bounce GI	real-time	D	no	reuse VPLs over frames, for static scenes only
[NW09,10]	Multiresolution Splatting	single-bounce GI	real-time	D,(G)	yes	hierarchical shading, no VPL visibility
[DGRS09]	Clustered Visibility	reduce banding	real-time	D,(G)	k-means	reduce banding with virtual area lights
[PKD12]	Reflective Shadow Map Clustering	single-bounce, reduce banding	real-time	D	k-means	virtual area lights with no visibility computation
[KK04]	Illumination Presence of Weak Singularities	bias compensation	offline	D,G	no	bias compensation using final gathering
[NED11]	Screen-Space Bias Compensation	bias compensation	real-time	D,G	no	image-space technique for approximate bias compensation
[RSK08]	Unbiased GI with Participating Media	bias compensation	offline	PM	no	basic bias compensation technique for media
[ENSD12]	Approximate Bias Compensation	bias compensation	offline/interactive	PM	no	approximate bias compensation for participating media
[WKB12]	Bidirectional lightcuts	bias compensation	offline	D,G,SSS,PM	yes	short-range indirect illumination, uses Multidimensional Lightcuts
[HKWB09]	Virtual Spherical Lights	avoid singularities	offline	D,G	yes	inflate VPLs to spherical lights, uses MRCS for clustering
[DKH∗10]	Combining Global and Local Virtual Lights	reduce singularities	offline	D,G	yes	local lights for short-range indirect illumination
[NNDJ12b]	Virtual Ray Lights	reduce singularities	offline	D,G,PM	no	use light path segments as virtual ray lights
[NNDJ12a]	Progressive Virtual Beam Lights	avoid singularities	offline	D,G,PM	no	inflate virtual ray lights
[HPB07]	Matrix Row-Column Sampling	scalability	precompute	D,G	yes	compute a global clustering of VPLs
[HVAPB08]	Tensor Clustering for Many-Light Animations	scalability	precompute	D,G	yes	compute a global clustering of VPLs, support for animations
[OP11]	Lightslice: Matrix Slice Sampling	scalability	precompute	D,G	yes	localized/clustered cuts
[WFA∗05]	Lightcuts	scalability	offline	D,G	yes	per-shading point local cut of VPL hierarchy
[WABG06]	Multidimensional Lightcuts	scalability	offline	D,G,PM	yes	cut of VPLs and sensor points for depth of field, motion blur
[DGS12]	Progressive Lightcuts for GPU	scalability	offline/interactive	D,G	yes	progressive, GPU-friendly variant of [WFA*05]
[SIMP06a]	Bidirectional Instant Radiosity	VPL generation	real-time	D,G	no	bidirectional VPL generation
[SIP07]	Metropolis Instant Radiosity	VPL generation	real-time	D,G	no	bidirectional VPL generation with Metropolis sampling
[GS10]	Iterative Importance Sampling of VPLs	VPL generation	real-time	D,G	no	rejection sampling of VPL paths
[GKPS12]	Importance Caching for Complex Illumination	VPL selection	offline	D,G	no	importance caching of VPL contributions and improved selection
[RGK∗08]	Imperfect Shadow Maps	visibility	real-time	D,G	no	fast approximate shadow maps for VPLs
[REH∗11]	Making ISMs View-Adaptive	visibility	real-time	D,G	no	extension of [RGK*08], view-adaptive ISMs and VPL placement
[REG∗09]	Micro-Rendering	visibility	interactive	D,G	dna	compute shadow maps for VPLs or final-gather from VPLs
[HREB11]	ManyLODs	visibility	real-time	D,G	dna	fast many-view rasterization
[PGSD13]	Adaptive Quantization Visibility Caching	visibility	offline/interactive	D,G	dna	general visibility cache

Figure 14: *This table provides an overview of the many-light rendering methods outlined in this report and is meant to serve as an orientation to find the right method for a given application or given needs.*

[GKPS12] Georgiev I., Křivánek J., Popov S., Slusallek P.: Importance caching for complex illumination. *Computer Graphics Forum (Proc. of Eurographics) 31*, 3 (2012), 701–710. 12

[GS10] Georgiev I., Slusallek P.: Simple and robust iterative importance sampling of virtual point lights. In *Eurographics short papers* (2010). 5, 6

[HJ09] Hachisuka T., Jensen H. W.: Stochastic progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), 141:1–141:8. 14

[HJ11] Hachisuka T., Jensen H. W.: Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics 30*, 5 (2011), 114:1–114:11. 14

[HKWB09] Hašan M., Křivánek J., Walter B., Bala K.: Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), 143:1–143:6. 8, 9

[HOJ08] Hachisuka T., Ogaki S., Jensen H. W.: Progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008), 130:1–130:8. 14

[HPB07] Hašan M., Pellacini F., Bala K.: Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26*, 3 (2007), 26. 8, 9, 10

[HPJ12] Hachisuka T., Pantaleoni J., Jensen H. W.: A path space extension for robust light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), 191:1–191:10. 14

[HREB11] Holländer M., Ritschel T., Eisemann E., Boubekeur T.: Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 30*, 4 (2011). 13

[HS98] Heidrich W., Seidel H.-P.: View-independent environment maps. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware* (1998), pp. 39–45. 13

[HVAPB08] Hašan M., Velázquez-Armendáriz E., Pellacini F., Bala K.: Tensor clustering for rendering many-light animations. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 27*, 4 (2008), 1105–1114. 10, 11

[Jen01] Jensen H. W.: *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001. 5

[JM12] Jakob W., Marschner S.: Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 58:1–58:13. 4

[JNSJ11] Jarosz W., Nowrouzezahrai D., Sadeghi I., Jensen H. W.: A comprehensive theory of volumetric radiance estimation using photon points and beams. *ACM Transactions on Graphics 30*, 1 (2011), 5:1–5:19. 14

[Kaj86] Kajiya J. T.: The rendering equation. In *Computer Graphics (Proc. SIGGRAPH)* (1986), pp. 143–150. 2

[KD13a] Kaplanyan A., Dachsbacher C.: Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proc. of Eurographics) 32*, 3 (2013). 14

[KD13b] Kaplanyan A. S., Dachsbacher C.: Adaptive progressive photon mapping. *ACM Transactions on Graphics 32*, 2 (2013). 14

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97* (1997), pp. 49–56. 1, 2, 3, 4, 13

[KFB10] KŘIVÁNEK J., FERWERDA J. A., BALA K.: Effects of global illumination approximations on material appearance. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 29*, 4 (2010), 112:1–112:10. 7, 13

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proc. Eurographics Workshop on Rendering* (2001), pp. 269–276. 12

[KHA*12] KŘIVÁNEK J., HAŠAN M., ARBREE A., DACHSBACHER C., KELLER A., WALTER B.: Optimizing realistic rendering with many-light methods. In *ACM SIGGRAPH 2012 Courses* (2012), pp. 7:1–7:217. 9

[KK04] KOLLIG T., KELLER A.: Illumination in the presence of weak singularities. *Monte Carlo and Quasi-Monte Carlo methods* (2004), 245–257. 7

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics 30*, 3 (2011), 25:1–25:13. 14

[LLK07] LAINE S., LEHTINEN J., KONTKANEN J.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering* (2007), pp. 4–8. 13, 14

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *SIGGRAPH '00* (2000), pp. 385–392. 13

[LW96] LAFORTUNE E. P., WILLEMS Y. D.: Rendering participating media with bidirectional path tracing. In *Proc. Eurographics Workshop on Rendering* (1996), pp. 91–100. 5

[NED11] NOVÁK J., ENGELHARDT T., DACHSBACHER C.: Screen-space bias compensation for interactive high-quality global illumination with virtual point lights. In *Proceedings of ACM Interactive 3D Graphics and Games* (2011), pp. 119–124. 8

[NNDJ12a] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Progressive virtual beam lights. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 31*, 4 (2012), 1407–1413. 9

[NNDJ12b] NOVÁK J., NOWROUZEZAHRAI D., DACHSBACHER C., JAROSZ W.: Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 60:1–60:11. 9

[NW09] NICHOLS G., WYMAN C.: Multiresolution splatting for indirect illumination. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2009), pp. 83–90. 12

[NW10] NICHOLS G., WYMAN C.: Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Transactions on Visualization and Computer Graphics 16*, 5 (2010), 729–741. 12

[OA11] OLSSON O., ASSARSSON U.: Tiled shading. *Journal of Graphics, GPU, and Game Tools 15*, 4 (2011), 235–251. 13

[OBA12] OLSSON O., BILLETER M., ASSARSSON U.: Clustered deferred and forward shading. In *Proc. of ACM SIGGRAPH / Eurographics conference on High Performance Graphics* (2012), pp. 87–96. 13

[OP11] OU J., PELLACINI F.: Lightslice: Matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* (2011). 10, 11

[Peg09] PEGORARO V.: *Efficient Physically-Based Simulation of Light Transport in Participating Media*. PhD thesis, School of Computing, University of Utah, 2009. 13

[PGSD13] POPOV S., GEORGIEV I., SLUSALLEK P., DACHSBACHER C.: Adaptive quantization visibility caching. *Computer Graphics Forum (Proc. of Eurographics) 32*, 3 (2013). 13

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers, 2010. 2, 3, 5

[PKD12] PRUTKIN R., KAPLANYAN A., DACHSBACHER C.: Reflective shadow map clustering for real-time global illumination. In *Eurographics short papers* (2012). 14

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Computer Graphics Forum 31*, 1 (2012), 160–188. 13

[REG*09] RITSCHEL T., ENGELHARDT T., GROSCH T., SEIDEL H.-P., KAUTZ J., DACHSBACHER C.: Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), 132:1–132:8. 13

[REH*11] RITSCHEL T., EISEMANN E., HA I., KIM J., SEIDEL H.-P.: Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 30*, 3 (2011). 1, 12, 13

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H. P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008), 1. 12, 13

[RSK08] RAAB M., SEIBERT D., KELLER A.: Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006* (2008), pp. 591–606. 7, 8

[SIMP06a] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Bidirectional instant radiosity. In *Proceedings of Eurographics Symposium on Rendering* (2006), pp. 389–398. 6

[SIMP06b] SEGOVIA B., IEHL J.-C., MITANCHEY R., PÉROCHE B.: Non-interleaved deferred shading of interleaved sample patterns. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware* (2006). 12

[SIP07] SEGOVIA B., IEHL J. C., PÉROCHE B.: Metropolis instant radiosity. *Computer Graphics Forum 26*, 3 (2007), 425–434. 6

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. *Computer Graphics (Proc. SIGGRAPH) 24*, 4 (1990), 197–206. 12

[SVLL10] SALVI M., VIDIMČE K., LAURITZEN A., LEFOHN A.: Adaptive volumetric shadow maps. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2010), pp. 1289–1296. 13

[SW10] SEGOVIA B., WALD I.: *Screen Space Spherical Harmonics Filters for Instant Global Illumination*. Tech. rep., Intel Corporation, 2010. 12

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. 4, 6

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 1081–1088. 10, 11

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24*, 3 (2005), 1098–1107. 9, 10

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *Proc. Eurographics Workshop on Rendering* (2002), pp. 15–24. 12

[WKB12] WALTER B., KHUNGURN P., BALA K.: Bidirectional lightcuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012), 59:1–59:11. 1, 10, 12

[WMHT65] WOODCOCK E., MURPHY T., HEMMINGS P., T.C. L.: Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Applications of Computing Methods to Reactor Problems* (1965), Argonne National Laboratory, pp. 557–579. 5

[Wym11] WYMAN C.: Voxelized shadow volumes. In *Proc. of ACM SIGGRAPH / Eurographics conference on High Performance Graphics* (2011), pp. 33–40. 13

[YCK*09] YU I., COX A., KIM M. H., RITSCHEL T., GROSCH T., DACHSBACHER C., KAUTZ J.: Perceptual influence of approximate visibility in indirect illumination. *ACM Transactions on Applied Perception 6*, 4 (2009), 1–14. 13