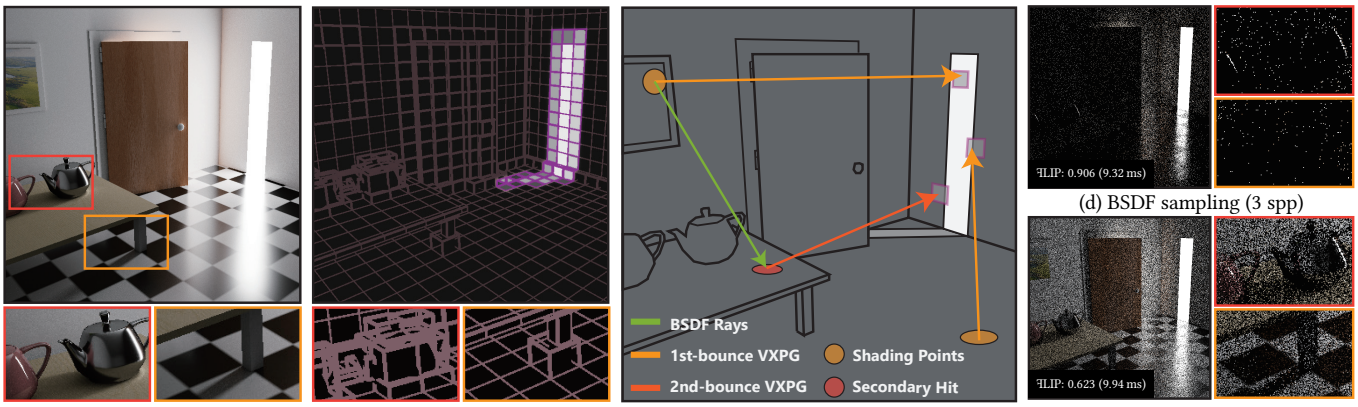
Real-Time Path Guiding Using Bounding Voxel Sampling

HAOLIN LU, University of California San Diego, USA
WESLEY CHANG, University of California San Diego, USA
TREVOR HEDSTROM, University of California San Diego, USA
TZU-MAO LI, University of California San Diego, USA



(a) Reference (b) Bounding Voxels (c) Illustration of VXPG (Ours) (d) BSDF sampling (3 spp) (e) VXPG sampling (2 spp)

Fig. 1. (a) The Veach Ajar scene is lit by a directional light source behind the door, and so only a small section of the room receives direct illumination. (b) An illustration of our bounding voxel data structure, which stores irradiance and geometry information for each voxel. (c) An illustration of our voxel path-guiding algorithm (VXPG), which guides paths to high-contribution voxels. (d-e) Comparison of BSDF sampling vs VXPG sampling for 2-bounce global illumination.

We propose a real-time path guiding method, Voxel Path Guiding (VXPG), that significantly improves fitting efficiency under limited sampling budget. Our key idea is to use a spatial irradiance voxel data structure across all shading points to guide the location of path vertices. For each frame, we first populate the voxel data structure with irradiance and geometry information. To sample from the data structure for a shading point, we need to select a voxel with high contribution to that point. To importance sample the voxels while taking visibility into consideration, we adapt techniques from offline many-lights rendering by clustering pairs of shading points and voxels. Finally, we unbiasedly sample within the selected voxel while taking the geometry inside into consideration. Our experiments show that VXPG achieves significantly lower perceptual error compared to other real-time path guiding and virtual point light methods under equal-time comparison. Furthermore, our method does not rely on temporal information, but can be used together with other temporal reuse sampling techniques such as ReSTIR to further improve sampling efficiency.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Global illumination, importance sampling, path guiding, real-time rendering

Authors' addresses: Haolin Lu, University of California San Diego, CA, USA, hal128@ucsd.edu; Wesley Chang, University of California San Diego, CA, USA, wec022@ucsd.edu; Trevor Hedstrom, University of California San Diego, CA, USA, tjhedstr@ucsd.edu; Tzu-Mao Li, University of California San Diego, CA, USA, tzli@ucsd.edu.

0730-0301/2024/7-ART125
https://doi.org/10.1145/3658203

ACM Reference Format:
Haolin Lu, Wesley Chang, Trevor Hedstrom, and Tzu-Mao Li. 2024. Real-Time Path Guiding Using Bounding Voxel Sampling. *ACM Trans. Graph.* 43, 4, Article 125 (July 2024), 14 pages. https://doi.org/10.1145/3658203

1 INTRODUCTION

Rendering scenes with complex visibility and strong indirect illumination in real-time requires finding good importance sampling distributions. Path guiding techniques are a popular importance sampling approach for offline rendering, but directly applying them is particularly challenging, since real-time computation budgets only allow for one to two light path samples per pixel per frame for learning the sampling distribution. Real-time rendering methods therefore often heavily rely on temporal information, but this can cause temporal artifacts and slow guiding distribution adaptation under complex dynamic scenes. Fig. 2 shows an example with challenging visibility where all the light paths need to go through the half-closed door to light the scene. Most samples do not help locate the sparse contribution as seen in Fig. 2(e). In this paper, we propose a new real-time path guiding method, Voxel Path Guiding (VXPG), which uses an efficient representation of the incoming radiance distribution which is simple to learn under limited sampling budgets and does not rely on temporal reuse.

Our key idea is to reuse a spatial irradiance representation across all shading points (Fig. 1 and Fig. 2(f)). We propose a bounding voxel data structure that stores both the irradiance and geometry information at each voxel. For each frame, we first populate the voxel data structure. Next, we sample from the voxel data structure for each shading point by first selecting a voxel and then shooting a ray from

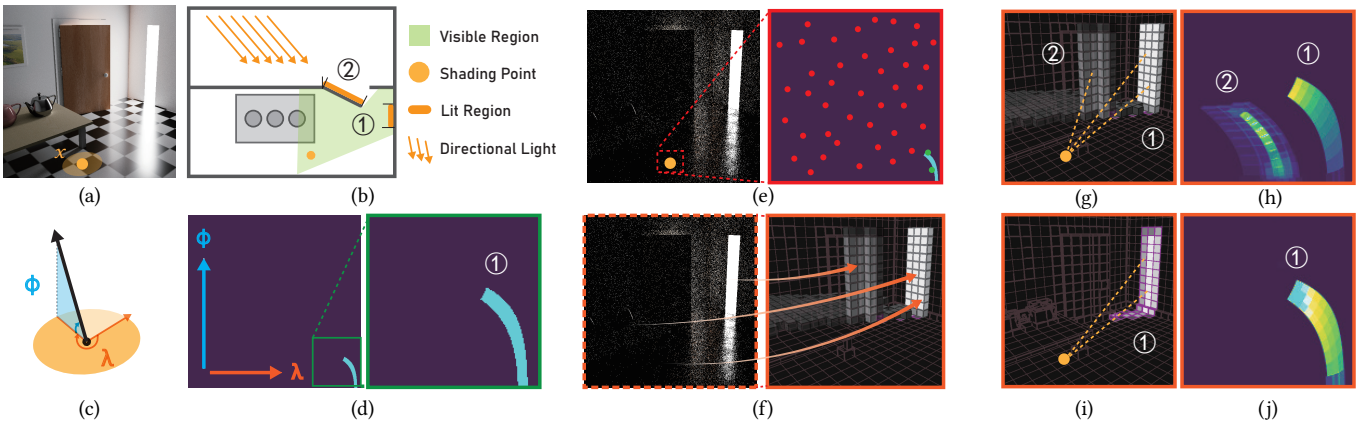


Fig. 2. Given a shading point x (a), our goal is to fit a target distribution (d), shown in cylindrical coordinates [Lambert 1770] (c), using a small amount of samples per pixel. Path guiding algorithms learning spatial-directional distributions often rely on local samples in a spatial partition. However, if the path contribution is sparse, local samples are not sufficient for reconstructing the target distribution (e), as only green samples have positive contributions. In contrast, our pipeline (f) uses all samples across the whole image to construct a global spatial distribution, mitigating the issue of insufficient information caused by using only local samples. However, blindly selecting a voxel (g) could result in connections with invisible surfaces (h). Notably, in the case of the door, it is only illuminated on the backface. Therefore, we propose a new voxel selection strategy (i) which incorporates visibility to achieve a good distribution (j).

the shading point towards a point in the sampled voxel. Compared to the traditional 5D spatial-directional distribution, our bounding voxel data structure significantly improves fitting efficiency since it reuses samples across all shading points.

Naïvely sampling from the voxels using only the irradiance information is suboptimal since it ignores both visibility and materials (Fig. 2(g) and (h)). We adapt techniques in offline many-lights sampling [Hašan et al. 2007; Ou and Pellacini 2011] by clustering pairs of shading points and voxels and building sampling distributions.

Finally, once we select a voxel, we need to sample a point to connect the path to the surface inside the voxel. This introduces two challenges: 1) We need to compute the probability density of a ray landing on a surface in the voxel, which can be intractable for real-time rendering. 2) The limited resolution of the voxel representation can cause the rays to miss the actual geometry inside. To tackle the challenges, we store the bounding volume of the geometry for each voxel. We then propose an unbiased and efficient sampling strategy to sample from the solid angle subtended by the bounding volumes.

While our method is primarily designed for first-bounce indirect illumination, we have observed its utility in addressing multiple-bounce indirect lighting. The irradiance-based representation imposes limitations on guiding caustics, but we can effectively handle most specular transport through approximate product sampling.

Our equal-time comparisons show that our method significantly improves the perceptual error (we use ꟻLIP [Andersson et al. 2020]) over other real-time path guiding and virtual point light methods for challenging scenes with complex visibility. Furthermore, our method does not rely on temporal reuse, but can be used together with other temporal reuse methods such as ReSTIR [Ouyang et al. 2021], to further improve sampling efficiency.

In summary, our contributions are:

- We introduce a bounding voxel data structure that stores the irradiance and geometry information for reusing information across all shading points for real-time path guiding.
- For sampling a voxel from a shading point while taking visibility and materials into consideration, we adapt offline many-light sampling methods to cluster pairs of shading points and voxels in real-time.
- We propose an efficient and unbiased way to sample a point inside a bounding voxel while taking the geometry inside the voxel into account.

2 RELATED WORK

Path guiding. Path guiding methods reuse previously traced light paths to fit an importance sampling distribution [Vorba et al. 2019].

Previous work often represents incoming radiance using a 5D spatial-directional distribution. Popular representations include adaptive histograms [Jensen 1995; Lafortune and Willems 1995; Müller et al. 2017], cosine lobes [Bashford-Rogers et al. 2012], Gaussian mixture models (GMMs) [Vorba et al. 2014], von-Mises-Fisher (vMF) distributions [Ruppert et al. 2020], or neural networks [Müller et al. 2019]. While originally designed for offline rendering, these methods have been adapted to real-time rendering. The real-time variants have to use much simpler distributions (e.g., a Gaussian or vMF lobe per-pixel [Derevyannykh 2022; Dittebrandt et al. 2023], or a coarse histogram [Dittebrandt et al. 2020; Kim and Kim 2021]), and often have to rely on temporal reuse [Derevyannykh 2022; Dittebrandt et al. 2020, 2023; Pantaleoni 2020] or offline training [Kim and Kim 2021] to improve sampling efficiency. We instead represent irradiance in 3D bounding voxels. Our representation significantly increases sample reuse across pixels, and does not require temporal reuse or offline training. Furthermore, our spatial representation is automatically "parallax-free" and does not require rotation of a directional distribution [Ruppert et al. 2020].

Our representation is related to the spatial representation used in the Focal Path Guiding work [Rath et al. 2023], in which they store the spatial density of light paths to automatically find "focal points" where many light paths meet. Our method differs in three main ways: 1) Focal Path Guiding is designed for offline rendering, while our method runs in real-time with a more efficient process to inject light paths to voxels. As a result, our voxels store irradiance instead of light path density. 2) We additionally take the contribution between the shading point and the spatial distribution in consideration when sampling. 3) We propose an efficient and unbiased sampling strategy for sampling points inside a bounding voxel, without the need to compute an integral for the probability density.

Virtual point lights and many-lights sampling. Our method can be broadly seen as a virtual point light (VPL) method [Dachsbacher et al. 2014; Keller 1997]. These methods first trace a batch of light paths, then in a second pass connect to the vertices of the light paths from the shading points. In a sense, VPL methods can be seen as path guiding with spatial Dirac delta distributions as the representation. For instance, power-based sampling orients the path towards regions with high radiance without considering visibility.

Traditionally, the VPLs are created by light tracing [Keller 1997]. It is also possible to create VPLs from path tracing. These VPLs are often only connected to nearby pixels [Bekaert et al. 2002; Davidovič et al. 2010]. This is because connecting these VPLs to a large set of shading points requires costly probability density function estimation [Segovia et al. 2006] and usually leads to bias.

Many-lights sampling is crucial for effectively sampling from a substantial number of lights, including both direct light sources and VPLs. A popular way is to build a hierarchical data structure that encodes information of a cluster of VPLs [Lin and Yuksel 2020; Liu et al. 2019; Moreau et al. 2019; Pantaleoni 2019; Paquette et al. 1998; Shirley et al. 1996; Vévoda et al. 2018; Walter et al. 2005; Wang et al. 2021; Yuksel 2020]. Alternatively, some methods cluster the VPLs by sampling the light transport matrix [Hašan et al. 2007; Ou and Pellacini 2011; Wu and Chuang 2013], where each element of the matrix accounts for the contribution of a light to a pixel sample. We adapt the light transport matrix sampling methods in a real-time setting to importance sample a bounding voxel from a shading point, while taking visibility and materials into consideration.

Our method is connected to VPL methods that "enlarge" the VPLs into an area [Hašan et al. 2009; Li et al. 2022; Simon et al. 2015; Tokuyoshi 2015]. Typically, the goal of these methods is to handle glossy reflections with VPLs, while our goal is to handle indirect illumination with complex visibility in real-time. Our method is also highly related to Stochastic Substitute Trees [Tatzgern et al. 2020], which is a real-time VPL sampling technique based on a continuous distribution with a spatial partition. Our method differs from Stochastic Substitute Trees in a few ways: we take the visibility between shading points and the spatial distribution into account, and our intra-voxel sampling is aware of the solid angles subtended by the bounding voxels. We generate the spatial distribution using path tracing, instead of light tracing. Lastly, our sampling is unbiased.

ReSTIR. In real-time rendering, it has become popular to reuse temporal information for sampling. In particular, ReSTIR [Bitterli et al. 2020] and its global illumination variants ReSTIR-GI [Ouyang et al. 2021] and ReSTIR-PT [Lin et al. 2022] employ resampling [Talbot et al. 2005] using samples drawn from spatial-temporal neighbors for each pixel. Our method alone does not rely on any temporal information. However, our method can also be used as a candidate sampling distribution for ReSTIR-GI to incorporate temporal reuse, which we demonstrate in the evaluation.

Table 1. Summary of paper notation.

x	A general path vertex on the scene surface $\mathcal{M}$.
x'	The previous path vertex to x.
x_0	The first path vertex at the camera.
x_1, x_{i_1}	The second path vertex, i.e. shading points.
x_2, x_{i_2}	The third path vertex, visible from x_1.
$\omega_\mathrm{i}, \omega_\mathrm{o}$	The incoming/outgoing direction.
y	The next vertex sampled to extend current path at x.
y'	A sample used for finding vertex y.
v_i	A bounding voxel.
I_i	The average irradiance of $\mathcal{M}_{v_i}$.
b_i	An AABB bounding $\mathcal{M}_{v_i}$.
Ω	The upper hemisphere.
$\mathcal{M}$	The scene surface.
$\mathcal{M}_{v_i}$	The portion of the scene surface $\mathcal{M}$ enclosed by voxel v_i.

3 BACKGROUND

We summarize the notation we use in Table 1.

Rendering Equation. The reflected radiance at a point x in the outgoing direction ω_o is described by the integral [Kajiya 1986]

$$L(x, \omega_\mathrm{o}) = L_e(x, \omega_\mathrm{o}) + \int_\Omega L_i(x, \omega_\mathrm{i}) f_r(x, \omega_\mathrm{o}, \omega_\mathrm{i}) \, |\cos\theta_\mathrm{i}| \, \mathrm{d}\omega_\mathrm{i}, \quad (1)$$

where L_e is the light emission, L_i is the incoming radiance, and f_r is the Bidirectional Scattering Distribution Function (BSDF). The radiance is computed by integrating over all incoming directions ω_i, and recursively evaluating L_i. This integral can also be rewritten to instead integrate over all points y on all surfaces $\mathcal{M}$:

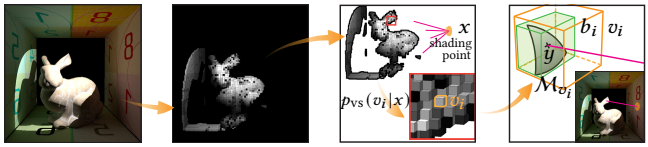
$$L(x \to x') = L_e(x \to x') + \int_\mathcal{M} L_i(y \to x) f_r(y \to x \to x') G(y \leftrightarrow x) V(y \leftrightarrow x) \, \mathrm{d}A(y), \quad (2)$$

where x' is the previous path vertex, G is the geometry term, and V is the visibility term.

Monte Carlo Integration. Equation 1 can be estimated using Monte Carlo integration with a single sample by sampling a direction ω_i, evaluating the integrand, and dividing by the PDF of the direction:

$$\langle L(x, \omega_\mathrm{o}) \rangle = L_e(x, \omega_\mathrm{o}) + \frac{\langle L_i(x, \omega_\mathrm{i}) \rangle f_r(x, \omega_\mathrm{o}, \omega_\mathrm{i}) \, |\cos\theta_\mathrm{i}|}{p(\omega_\mathrm{i} | x, \omega_\mathrm{o})}. \quad (3)$$

Eq. (2) can be estimated in a similar way by sampling and evaluating y instead. If $p > 0$ whenever the integrand is positive, then the estimator is unbiased and $\mathbb{E}[\langle L \rangle] = L$. When the estimator is unbiased, the error of Monte Carlo integration is given by the variance of the estimator, which is zero when p is exactly proportional to



(a) (b) (c) (d)

Fig. 3. (a) A bunny in a box lit by a spotlight. (b) Before VXPG sampling, we construct a voxel representation of the scene, including irradiance and geometry information (Section 5.1). (c) During VXPG sampling, for every shading point x, we first randomly select one voxel from the entire set to connect with (Section 5.2). (d) After that, we find the next vertex y within the selected voxel in a fast and unbiased way (Section 5.3).

the integrand. Importance sampling therefore aims to construct a density p which is approximately proportional to the integrand.

Path Guiding. Path guiding methods learn to sample high contribution paths by using previously sampled paths to fit the density p so that it approximates the integrand in Equation 1, to obtain low variance. The full integrand is 7D however (3D for x, 2D for ω_i, and 2D for ω_o), and so previous work typically instead aims to fit a 5D distribution $p(\omega_i|x)$ which is independent of the outgoing direction. This is done by discretizing the dimensions using subdivision structures [Derevyannykh 2022; Müller 2019; Ruppert et al. 2020; Vorba et al. 2014], or directly learning in 5D [Dodik et al. 2022; Dong et al. 2023]. However, the low number of samples generated per frame in real-time rendering is still insufficient to accurately fit this 5D distribution (Fig. 2(c)). As a result, previous work [Derevyannykh 2022; Dittebrandt et al. 2020] relies on temporal reuse to improve the guiding distribution, but then the distribution lags in time and is unable to quickly adapt to dynamic scenes.

4 OVERVIEW

We propose a real-time path guiding algorithm. Our method can fit the sampling distribution at low sample counts, and does not rely on temporal information. To achieve this, instead of learning a local 5D distribution $p(\omega_i|x)$ representing the directional sampling density for ω_i at each point x, we propose to learn a global 3D spatial distribution similar to $p(y)$, shared by all shading points. Inspired by next event estimation, this 3D distribution guides the location of the next path vertex y.

Our approach offers two advantages: First, a spatial distribution naturally eliminates parallax issues during sampling [Ruppert et al. 2020], and second, fewer samples are needed to learn the distribution due to its reduced dimensionality, and so it can be rebuilt from scratch at every frame to quickly adapt to dynamic scenes.

However, simply sampling from $p(y)$ for different shading points can be inefficient, since it ignores the visibility and BSDF terms [Rath et al. 2023]. In complex scenes, many paths will be occluded, limiting the guiding efficiency (Fig. 2(g)). Instead, we sample from a conditional distribution $p(y|x)$ for different shading points, which approximates the desired 5D density, while only storing a 3D spatial distribution.

To do this efficiently in real-time, we propose a novel two-stage sampling method, bounding voxel sampling (Section 5, Fig. 2(e)). We bound the scene in a set of voxels $\{v_0, \ldots, v_{n-1}\}$, each of which stores irradiance and geometry information. In the first stage, for each shading point x, we select one voxel from the entire set by sampling a conditional probability $p_{vs}(v_i|x)$. This stage is akin to many-lights sampling [Bitterli et al. 2020; Hašan et al. 2007; Moreau et al. 2022; Ou and Pellacini 2011; Shirley et al. 1996; Yuksel 2019]. In the second stage, we sample the point y within the selected voxel with $p(y|v_i)$, analogous to area light sampling [Arvo 1995; Gamito 2016; Peters 2021; Ureña et al. 2013].

Bounding voxel sampling therefore samples:

$$p(y|x) = p(y|v_i)p_{vs}(v_i|x), \tag{4}$$

with the conditioning in the voxel selection stage providing local adaptivity and accounting for both visibility and BSDF.

Fig. 3 illustrates an overview of our pipeline. It begins by populating voxels with irradiance and geometry information (Fig. 3(b), Section 5.1). Then, for each shading point, we select one voxel according to a distribution p_{vs} considering power, BSDF, and visibility (Fig. 3(c), Section 5.2). Finally, we sample the next path vertex y within the selected voxel (Fig. 3(d), Section 5.3).

5 BOUNDING VOXEL SAMPLING

In Sections 5.1-5.3, we discuss the construction and sampling of bounding voxels for our VXPG algorithm. We focus on the case of first-bounce indirect illumination. Section 5.4 then details combining bounding voxel sampling with BSDF sampling using multiple importance sampling to make the algorithm robust. Section 5.5 discusses extending the first-bounce case to direct illumination, as well as further bounces. We provide more details of our implementation in the supplementary material.

5.1 Construct Bounding Voxels

Bounding voxels. To construct our spatial distribution, we uniformly partition the scene using voxels $\{v_0, \ldots, v_{n-1}\}$. Each voxel v_i encloses a portion of the scene surface $\mathcal{M}$, which we denote as $\mathcal{M}_{v_i} = \mathcal{M} \cap v_i$. Each voxel v_i stores the average irradiance of $\mathcal{M}_{v_i}$, I_i, and an axis-aligned bounding box (AABB) $b_i \subseteq v_i$, which tightly bounds $\mathcal{M}_{v_i}$. We refer to this structure as "bounding voxel", emphasizing its dual characteristics: the bounding volume feature provided by b_i and the spatial partitioning nature of the voxels. Both characteristics are essential for unbiased sampling in Section 5.3.

Light injection. We call the process of assigning irradiance to voxels "light injection", a concept borrowed from voxel-based global illumination (VXGI) [Crassin et al. 2011]. In our method, assigning positive irradiance to completely invisible voxels is undesirable, since paths can be guided to occluded regions. Therefore, injecting light information through voxelization [Crassin et al. 2011] or light tracing [Hašan et al. 2009; Keller 1997] is suboptimal.

Instead, we inject light by tracing paths from the camera and place a virtual light at the first hit of the indirect bounce[1]. These points, which we denote as x_2, are guaranteed to contribute to the image, since they are visible from some shading points. In particular, we trace a BSDF ray for each shading point and populate the irradiance of vertex x_2 into the voxel in which x_2 is located. We define the

[1]This is similar to Reverse Instant Radiosity [Segovia et al. 2006], but we do not need to compute an intractable PDF for unbiased results.

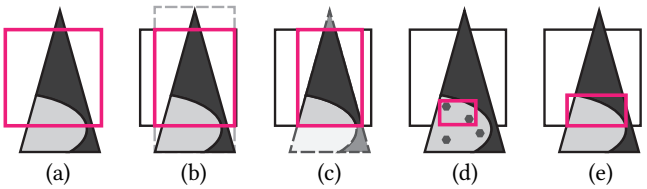


Fig. 4. Different geometry compaction strategies to compute the bounding box b_i. In this example, a triangle intersects with a voxel, and only the light gray portion of the triangle has non-zero irradiance. (a) Voxel bounds. (b) Intersection of the voxel and the bounding box of the triangle. (c) Our method: The triangle is clipped against the voxel to produce tight bounds. (d) A non-conservative bound using all second-bounce vertices in the voxel. (e) Optimal bounding box.

voxel irradiance I_i as the average irradiance of all vertices injected to it:

$$I_i = \frac{1}{n_i} \sum_{x_{j_2} \in v_i} E(x_{j_2}), \tag{5}$$

where n_i is the number of vertices injected into voxel v_i, and $E(\cdot)$ is the irradiance of each vertex[2]. We only evaluate direct lighting for $E(x_{j_2})$, as we are focusing on one-bounce indirect illumination.

Geometry injection. The second part of the bounding voxel is the AABB b_i. This is important because as shown in Fig. 3(d), the voxel v_i often only provides a loose bound for the surface $\mathcal{M}_{v_i}$. In the intra-voxel sampling step, further explained in Section 5.3, we need to sample a point within the voxel. Sampling a point on the more compact b_i rather than directly on v_i increases the chance of selecting a point that lies on the surface.

To inject geometry information and create b_i, we apply rasterization-based voxelization [Crassin and Green 2012]. When we detect that a triangle overlaps with a voxel, we compute an AABB of their intersection and then combine them using atomic operations.

However, simply intersecting the voxel with the bounding box of the triangle often does not result in full compactness, as illustrated in Fig. 4(b). We use the Sutherland–Hodgman algorithm [1974] to clip the triangle against the voxel and compute its own bounds accordingly, as demonstrated in Fig. 4(c).

In principle, the ideal AABB b_i should bound only the portion of the geometry with non-zero irradiance, as illustrated in Fig. 4(e). However, obtaining this AABB exactly is not feasible in practice. Estimating it by, for example, calculating a bounding volume for all vertices within the voxel, as depicted in Fig. 4(d) leads to a non-conservative bound and may lead to temporal instability, especially when the vertex count is low. We need a conservative bound so that we do not miss sampling geometry with non-zero radiance.

5.2 Voxel Selection

Once the bounding voxels are constructed, we sample a voxel v_i for each shading point x_1 using a probability mass function (PMF) $p_{\text{vs}}(v_i|x_1)$. Ideally, we want the PMF to be proportional to the square root of the second moment of the contribution estimator [Pantaleoni and Heitz 2017; Rath et al. 2020; Vévoda et al. 2018], where the contribution is defined as:

$$\int_{\mathcal{M}_{v_i}} L_i(y \to x_1) f_r(y \to x_1 \to x_0) G(y \leftrightarrow x_1) V(y \leftrightarrow x_1) dA(y). \tag{6}$$

However, estimating this contribution for all voxel-shading point pairs is costly. Instead, we observe that selecting voxels is similar to selecting light sources in many-light sampling, and it can, in fact, be viewed as selecting a "virtual voxel light" with emission I_i and bounds b_i. Consequently, we can leverage existing work in many-lights sampling [Bitterli et al. 2020; Hašan et al. 2007; Moreau et al. 2022; Ou and Pellacini 2011; Shirley et al. 1996; Yuksel 2019] to construct the voxel selection distribution p_{vs}.

A simple method is to select a voxel with probability proportional to its power [Shirley et al. 1996]:

$$\Phi(v_i) = I_i \cdot A(v_i), \tag{7}$$

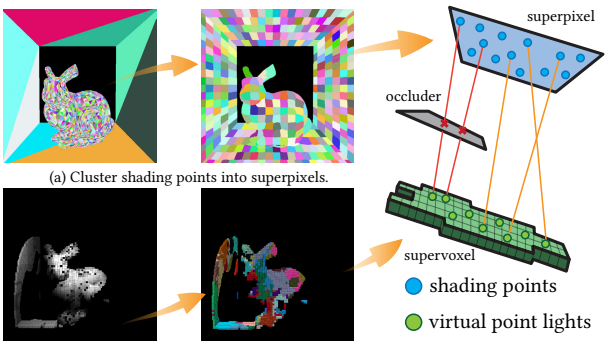
where $A(v_i)$ is the surface area of $\mathcal{M}_{v_i}$. We approximate A using the surface area of the largest of the 6 faces of b_i.

Since the bounding voxels are shared across all shading points, sampling them based on power ignores the visibility, geometry, and BSDF terms in Eq. (6). While our light injection method partially mitigates this by ensuring that voxels with non-zero irradiance are visible to some shading points, they can still be occluded to others. We aim to estimate the contribution without querying visibility between each pair of shading points and voxel, which would be prohibitively expensive. Inspired by previous work in many lights sampling [Hašan et al. 2007; Ou and Pellacini 2011; Wu and Chuang 2013] and probabilistic connections of bidirectional path tracing [Popov et al. 2015; Su et al. 2022], we assume the contribution is locally similar, and use clustering to reduce the number of visibility queries. We group pixels and voxels into clusters, referred to as superpixels and supervoxels, respectively. Then, we estimate an average throughput for each superpixel-supervoxel pair, providing an approximation of the product of visibility, geometry, and BSDF. This process is depicted in Fig. 5. Below we detail each step.

Superpixel and supervoxel clustering. We base our method on the SLIC superpixel algorithm [Achanta et al. 2012] for superpixel clustering, taking into account geometry similarity. Meanwhile, we adopt a simplified version of the K-means algorithm to group voxels into supervoxels, based on visibility and irradiance information. The supplementary material provides the details of the clustering.

Average throughput estimation. After clustering, we estimate the average throughput for each superpixel-supervoxel pair. In a preparation stage, we figure out which shading points and x_2 vertices correspond to each superpixel and supervoxel. This stage also adds surface normal information to each point so that backfacing surfaces can be rejected. Then, for each superpixel-supervoxel pair, we select 32 pairs of shading points and x_2 vertices within the cluster pair, and query their binary visibility by tracing a ray (Fig. 5(c)). As both ends of these rays are expected to be close, they are typically

[2]Strictly speaking, $E(\cdot)$ here does not represent irradiance, which should be integrated over the hemisphere. In our case, we cannot guarantee full coverage of the domain but still borrow the term for simplicity.



(a) Cluster shading points into superpixels.
(b) Cluster bounding voxels into supervoxels.
(c) Average throughput

Fig. 5. The overview of our visibility-aware voxel selection algorithm. (a-b) To reduce the number of visibility queries, we cluster the shading points and voxels respectively. Then, for each pair of superpixel and supervoxel, we cast multiple rays to estimate an average throughput that is the product of visibility, BSDF, and geometry terms.

coherent and can be efficiently traced. The average throughput is:

$$\bar{\mathrm{T}} = \frac{1}{32}\sum_{i=0}^{31} f_r(x_2^i \to x_1^i \to x_0) G(x_2^i \leftrightarrow x_1^i) V(x_2^i \leftrightarrow x_1^i). \quad (8)$$

Voxel selection. Next, we select voxels for each shading point as follows. First, we identify the superpixel $\mathrm{SP_i}$ of the shading point. Next, we select a supervoxel $\mathrm{SV_j}$ with probability proportional to the product of average throughput and total power, $\bar{\mathrm{T}}_{i,j} \cdot \sum_{v_k \in SV_j} \Phi(v_k)$, which approximately product sample the full contrbution: irradiance, visibility and BSDF terms. Finally, we choose one voxel within the supervoxel $\mathrm{SV_j}$ based on power Φ as in Eq. (7).

5.3 Intra-Voxel Sampling

Bounding volume sampling. Once a voxel v_i is selected, we need to select the position of the path vertex within the voxel. The challenge is that we cannot simply sample any point $y' \in b_i$, since the path vertex must lie on the scene surface $\mathcal{M}$.

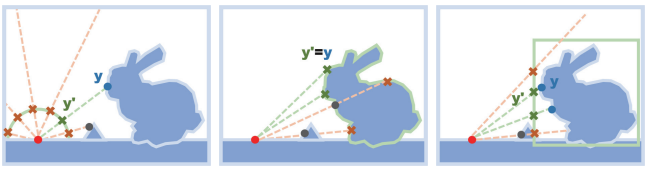
Stochastic Substitute Trees [Tatzgern et al. 2020] handle this by treating the sampled y' as a VPL which introduces bias. The probability density of this method is also intractable. Focal Path Guiding [Rath et al. 2023] first maps y' to a direction ω_i, where $\omega_i = \frac{y'-x}{\|y'-x\|}$, and finds the exact vertex y by ray casting.

However, since there are infinitely many y' that map to the same ω_i, an integration is need to compute the probability density [Rath et al. 2023; Simon et al. 2017]:

$$p_d(\omega_i|x) = \int_0^\infty p_s(x + t\omega_i) t^2 \mathrm{d}t. \quad (9)$$

This is prohibitively expensive to compute for real-time applications, as it requires traversing the entire voxel structure.

Instead, we propose bounding volume sampling, as illustrated in Fig. 6(c). We first generate the primal sample y' on the surface of the AABB b_i associated with the voxel. Next, we cast a ray from x towards y' in direction ω_i to obtain the path vertex y. Finally, we check the position of y, accepting it if $y \in b_i$ and discarding it otherwise.



(a) directional sampling (b) next event estimation (c) bounding voxel sampling

Fig. 6. (a) In directional sampling, a primary sample y' is taken from the upper hemisphere, and a ray is cast to determine the vertex y on the surface. (b) NEE acquires the primary sample y' directly from the geometry's surface, which corresponds to vertex y itself. The ray is traced to assess visibility. (c) In our bounding voxel sampling, the primary sample y' is drawn from a bounding voxel. We then trace a ray to locate vertex y.

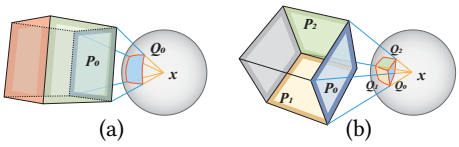


Fig. 7. When the shading point x is outside of the bounding box, there is at least one (a) and at most three (b) surfaces facing x. During sampling, we project the forward-facing surfaces P_{0-2} onto spherical rectangles Q_{0-2}.

The probability density of selecting the vertex y can then be computed in closed form[3] as:

$$p(y|x) = p(y|y', x) p(y'|x, v_i) p_{\mathrm{vs}}(v_i|x), \quad (10)$$

due to two factors. First, due to the rejection step, each vertex y can only be sampled when we select the voxel v_i within which it resides, since voxels are disjoint. This eliminates the need to marginalize over v_i. Second, we sample the boundary of b_i rather than the interior. Consequently, given x and a specific v_i, the mapping $y' \mapsto \omega_i$ is injective, removing the need to marginalize over y'.

Our formulation trades off the costly PMF integration, as shown in Eq. (9), with sample rejection. Therefore, having a compact bounding volume is crucial to reduce sample rejection, as discussed in Section 5.1 on geometry injection.

Spherical voxel sampling. To efficiently generate samples on the surface of the bounding voxel, we project all forward-facing surfaces of the AABB b_i into spherical rectangles [Arvo 1995; Ureña et al. 2013]. We only attempt to select a sample if the vertex x is outside the AABB b_i, and so there are at least one and at most three spherical rectangles, Q_0, Q_1, Q_2, as illustrated in Fig. 7.

To draw the primary sample y', we first select one spherical rectangle proportionally to its surface area, $p(Q_i) = \mathrm{Area}(Q_i)/\sum_j \mathrm{Area}(Q_j)$, and then apply spherical rectangle sampling [Ureña et al. 2013] on Q_i. This allows us to sample the boundary of the AABB with a probability density proportional to the solid angle:

$$p(y'|x, v_i) = \frac{1}{\sum_j \mathrm{Area}(Q_j)}. \quad (11)$$

[3]The $p(y|y', x)$ term in Eq. (10) is essentially a Jacobian of the mapping from y to y'.

5.4 Multiple Importance Sampling with BSDF Sampling

To make our algorithm robust, we combine samples drawn using VXPG with BSDF importance sampling using Multiple Importance Sampling (MIS) with the balance heuristic [Veach and Guibas 1995]. This step is crucial since, like in many path guiding methods, light injection is not guaranteed to find every voxel that contributes to the image and can results in bias. Combining BSDF sampling ensures all surfaces have a non-zero probability of being selected.

5.5 Path Guiding for Further Bounces

Our bounding voxel sampling method can also be applied to direct illumination and multi-bounce indirect illumination. To guide direct illumination, we inject light source emission instead of irradiance of x_2 in the light injection stage. For further bounces, we can extend our method by injecting $x_3, x_4, \ldots$ and further vertices to guide the corresponding bounces. A challenge for second bounce onwards is that our voxel selection strategy estimates contribution between superpixels and supervoxels, while path vertices may lie on parts of the scene outside the image. To obtain information for these vertices, we need to estimate voxel-voxel contribution. Alternatively, we can skip the contribution estimation for second bounce onwards.

In our implementation, we include up to second-bounce indirect illumination. To minimize overhead, we reuse the bounding voxel structure built for the first bounce, as seen in Fig. 1(c), and use power-based sampling for voxel selection. Our experiments show that this simple strategy still helps with sampling the second bounce.

6 EVALUATION

We implemented our algorithm in a custom renderer using the Vulkan API with hardware-accelerated ray tracing. All results are rendered at a 1280×720 resolution, on a laptop with an NVIDIA GeForce RTX 3070 GPU. The reference images are rendered using standard unidirectional path tracing with a high sample count. All reported timings are measured by averaging at least 300 frames.

Our test scenes are based off Bitterli's [2016] rendering resources. In most rendered images, we visualize indirect illumination only, omitting direct illumination, to emphasize the improvement of our algorithm for guiding indirect illumination.

We provide more comparisons in the supplementary material.

6.1 Static Scene Comparisons

Image quality comparisons. We assess image quality using ꟻLIP (↓) [Andersson et al. 2020], by comparing to the following methods: BSDF importance sampling, real-time stochastic lightcuts (SLC) [Lin and Yuksel 2020], stochastic substitute trees (SST) [Tatzgern et al. 2020], and screen space path guiding (SSPG) [Derevyannykh 2022]. All scenes are rendered with 2-bounce global illumination, with the exception of Veach Mis, which only includes direct illumination. Our method always uses one VXPG and one BSDF path, combined using MIS, resulting in a constant 2 samples-per-pixel. We use 64^3 voxel resolutions for all our results.

SSPG maintains one Gaussian lobe per pixel and uses a random mixture of the Gaussian lobe and BSDF lobe for first-bounce path guiding. In our implementation, we adopt the balance heuristic [Veach 1998] instead of their learned random mixture and directly fit the Gaussian lobe to the one-bounce incident radiance. This setting always provides superior quality across all tested configurations. Meanwhile, since SSPG heavily relies on temporal reuse to learn the lobe, all results are captured after the guiding distributions have been fully learned.

Fig. 8 and Table 2 presents an equal-time comparison. In scenes with complex visibility, our approach outperforms SLC and SST since we incorporate visibility in our sampling process. However, the Veach Mis scene has low visibility complexity, and so VPL-based methods can outperform our approach in some regions. On the other hand, VPL-based methods are poor at handling highly specular surfaces, while our approach addresses this through approximate product sampling and combining with BSDF sampling.

Guiding distribution comparison. In Fig. 9, we provide a visualization of the learned directional distributions, which approximate the direct or one-bounce radiance. We compare our method (VXPG) with two established techniques for real-time path guiding: the Compressed Directional Quadtree (CDQ) [Dittebrandt et al. 2020] and the one-lobe Gaussian mixture model (GMM) [Derevyannykh 2022].

When comparing with CDQ, we use the implementation with 64 bit counts, as recommended in Section 7.3 of the paper [Dittebrandt et al. 2020]. For the sake of simplicity and fairness, in every frame, we exclusively draw 256 samples from the shading point to update the CDQ structure, rather than using all samples within a leaf of an adaptive octree as described by the authors. This modification improves the quality of the learned CDQ compared with the original implementation, as it effectively eliminates parallax issues, so that it is easier to compare with our approach and screen space path guiding [Derevyannykh 2022] as both are inherently parallax-free.

The Veach Mis scene consists of three emitting spherical lights of increasing size. The smallest light is a challenge for both CDQ and a one-lobe GMM to capture and represent. In the Fireplace scene, there are over eight distinct high-contribution regions, but both CDQ and GMM are unable to represent complex distributions. It shows that our method is able to represent challenging distributions. It excels in identifying and representing small as well as multiple high-contribution regions while preserving sharp boundaries.

Combination and comparison with ReSTIR. Directly comparing our method with ReSTIR is challenging, since ReSTIR heavily relies on reusing samples across frames to refine the sampling distribution. However, as a path guiding technique, our approach can be used for generating the candidate samples for ReSTIR.

In Fig. 10, we compare VXPG with ReSTIR GI [Ouyang et al. 2021] and a combination of both methods. We combine VXPG and BSDF sampling using MIS to generate candidate samples for ReSTIR. Notably, in the Veach Ajar scene, where BSDF sampling may generate poor candidate samples, our approach can outperform ReSTIR when the number of temporal samples is insufficient. Furthermore, using VXPG as a candidate distribution of ReSTIR GI improves the result.

Adaptation speed comparison. In Fig. 9, we illustrate that VXPG can rapidly adapt to sudden illumination changes in a single frame, whereas CDQ and GMM exhibit only minor adjustments.

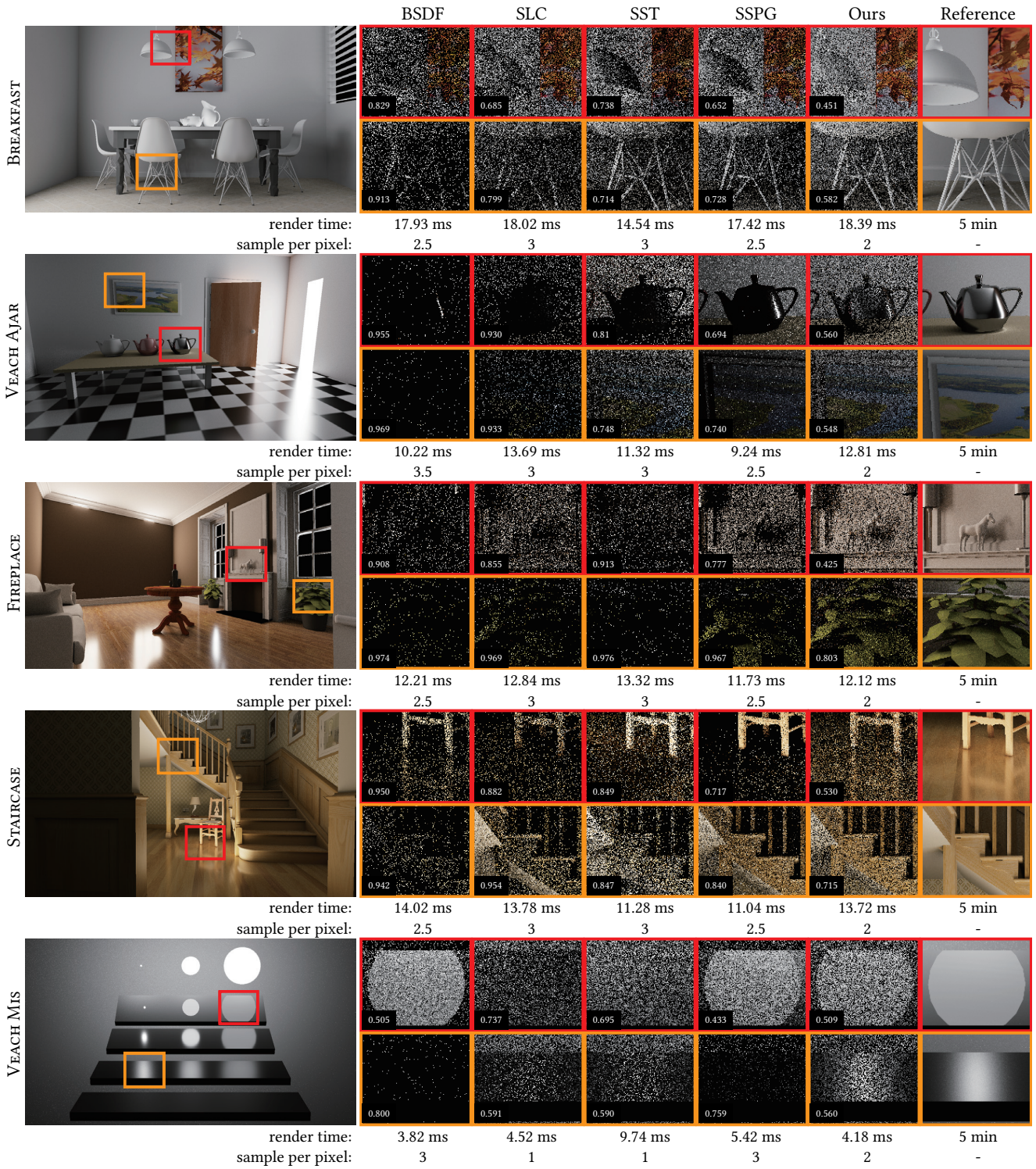


Fig. 8. Approximate equal time comparisons of our method (VXPG) with previous works. ꟻLIP values are inset in each image. We compare with BSDF sampling, real-time Stochastic Lightcuts (SLC) [Lin and Yuksel 2020], stochastic substitute trees (SST) [Tatzgern et al. 2020], and screen-space path guiding (SSPG) [Derevyannykh 2022]. All images are captured under static lighting conditions, with a fixed scene and camera pose. For BSDF and SSPG, we sometimes include an additional path that only evaluates the first-bounce indirect lighting to align with the time budget, shown as an extra 0.5 sample per pixel.

Table 2. Approximate equal time comparison of our method with previous works across a broader set of test scenes. We report mean ꟻLIP and average render time. For all scenes, we evaluated 2-bounce indirect illumination, except for the Veach, Mis scene, where we assess direct illumination without next event estimation. The best-performing entries are highlighted in **bold letters**. Note that we precompute the geometry injection step once at scene initialization for VXPG, and so omit it from the per-frame render times.

		[Lin and Yuksel 2020]	[Tatzgern et al. 2020]	[Derevyannykh 2022]	Ours
	BSDF	SLC	SST	SSPG	VXPG
Veach Ajar	0.912 10.22 ms	0.904 13.69 ms	0.801 11.32 ms	0.785 09.24 ms	**0.573** 12.81 ms
Fireplace	0.895 12.58 ms	0.840 12.84 ms	0.856 13.32 ms	0.800 11.73 ms	**0.599** 12.12 ms
Staircase	0.963 14.02 ms	0.910 13.78 ms	0.869 11.28 ms	0.850 11.04 ms	**0.768** 13.72 ms
Teapot	0.704 11.08 ms	0.665 11.74 ms	0.781 12.09 ms	0.588 10.98 ms	**0.489** 10.85 ms
Kitchen	0.867 14.63 ms	0.736 12.18 ms	0.841 10.04 ms	0.796 12.68 ms	**0.618** 13.02 ms
Bedroom	0.919 18.38 ms	0.645 14.18 ms	0.823 17.12 ms	0.806 17.95 ms	**0.450** 15.48 ms
Breakfast	0.831 17.93 ms	0.701 18.02 ms	0.678 14.54 ms	0.683 17.42 ms	**0.496** 18.39 ms
Veach Mis	0.847 03.82 ms	0.594 04.52 ms	**0.573** 09.74 ms	0.766 05.42 ms	0.589 04.18 ms

We also directly compare adaptation speed for temporal information, as shown in Fig. 11, where we compare our method to SSPG and ReSTIR GI. Both SSPG and ReSTIR GI improve sampling through the temporal accumulation of information.

The effectiveness of SSPG is limited since the guiding distribution is restricted to a single lobe Gaussian. It also converges the slowest, as BSDF sampling alone does not generate enough high contribution samples to effectively train the model. In contrast, our approach is able to learn a better guiding distribution at each frame, but does not further refine using previous frames. On the other hand, ReSTIR GI takes more than 20 frames to achieve stable quality. When using VXPG as the candidate distribution, ReSTIR GI converges faster.

Denoising. Fig. 12 shows that our approach also improves rendering results with A-SVGF denoising [Schied et al. 2018]. It can further mitigate temporal artifacts in A-SVGF due to disocclusion or rapid lighting changes (see supplementary video).

6.2 Dynamic Scene Comparisons

Dynamic objects and lighting. We evaluate our method on three dynamic scenes. Fig. 13 illustrates how our approach can adapt to dynamic environments and improve the quality of ReSTIR GI and A-SVGF. Additional results are presented in the supplemental video.

6.3 Performance

Pipeline time breakdown. Table 3 displays the average execution times of each stage of the pipeline. In particular, in light tree building stage we build a binary tree for power-based voxel selection, and in path tracing stage we execute all sampling and path tracing. Besides the geometry injection stage, the preparation stages of VXPG typically introduce an overhead of less than 1ms in the test scenes. This overhead is outweighed by the improved sampling efficiency as shown in the equal-time comparison.

The cost of geometry injection is heavily dependent on scene complexity. In practice, we precompute geometry information for static objects. During runtime, we perform geometry injection solely for dynamic objects and then merge the results.

Dynamic geometry injection. As geometry injection requires per-frame voxelization of dynamic objects, the overhead can be concerning. Fortunately, we observe that sub-voxel details are usually filtered during injection, therefore creating level of details (LoDs) for high-poly models often do not lead to any quality loss under low voxel resolution, while significantly reducing the injection overhead. Table 4 shows how various combinations of LoDs and voxel resolutions influence the overhead, where LoDs are generated by quadric mesh simplification [Garland and Heckbert 1997]. Higher-resolution grids may actually accelerate geometry injection by reducing the number of conflicts for atomic operations. However, the increase in resolution implies a larger number of voxels to be processed, consequently increasing the overhead of the entire pipeline. Furthermore, we implemented a spatial hashing [Binder et al. 2021] version for geometry injection, which usually involves an overhead less than 0.15ms caused by resolving linear probing, but can be helpful when memory is the bottleneck.

6.4 Ablation Studies

Various voxel resolutions for large scenes. Fig. 14 illustrates the significance of voxel resolution in the quality of VXPG. In moderately large scenes like Zero-Day, a grid size of 64^3 voxels may struggle to accurately capture the characteristics of irradiance and geometry. Consequently, noticeable quality improvement can be observed with increased voxel resolution.

Multi-bounce path guiding. Fig. 15 presents an equal-sample comparison of second-bounce-only indirect illumination between BSDF and VXPG sampling. When the scene is dominated by indirect illumination and visibility is less crucial, VXPG can also improve multi-bounce sampling quality simply by using power-based [Shirley et al. 1996] voxel selection. In contrast, screen-space techniques [Derevyannykh 2022; Ouyang et al. 2021] cannot guide for further bounces.

Voxel selection strategies. Fig. 16 shows how various voxel selection strategies affect image quality. We compare our visibility-aware algorithm with power-based sampling [Shirley et al. 1996] and stochastic lightcuts [Lin and Yuksel 2020]. Our algorithm consistently outperforms others, particularly when visibility is a crucial factor

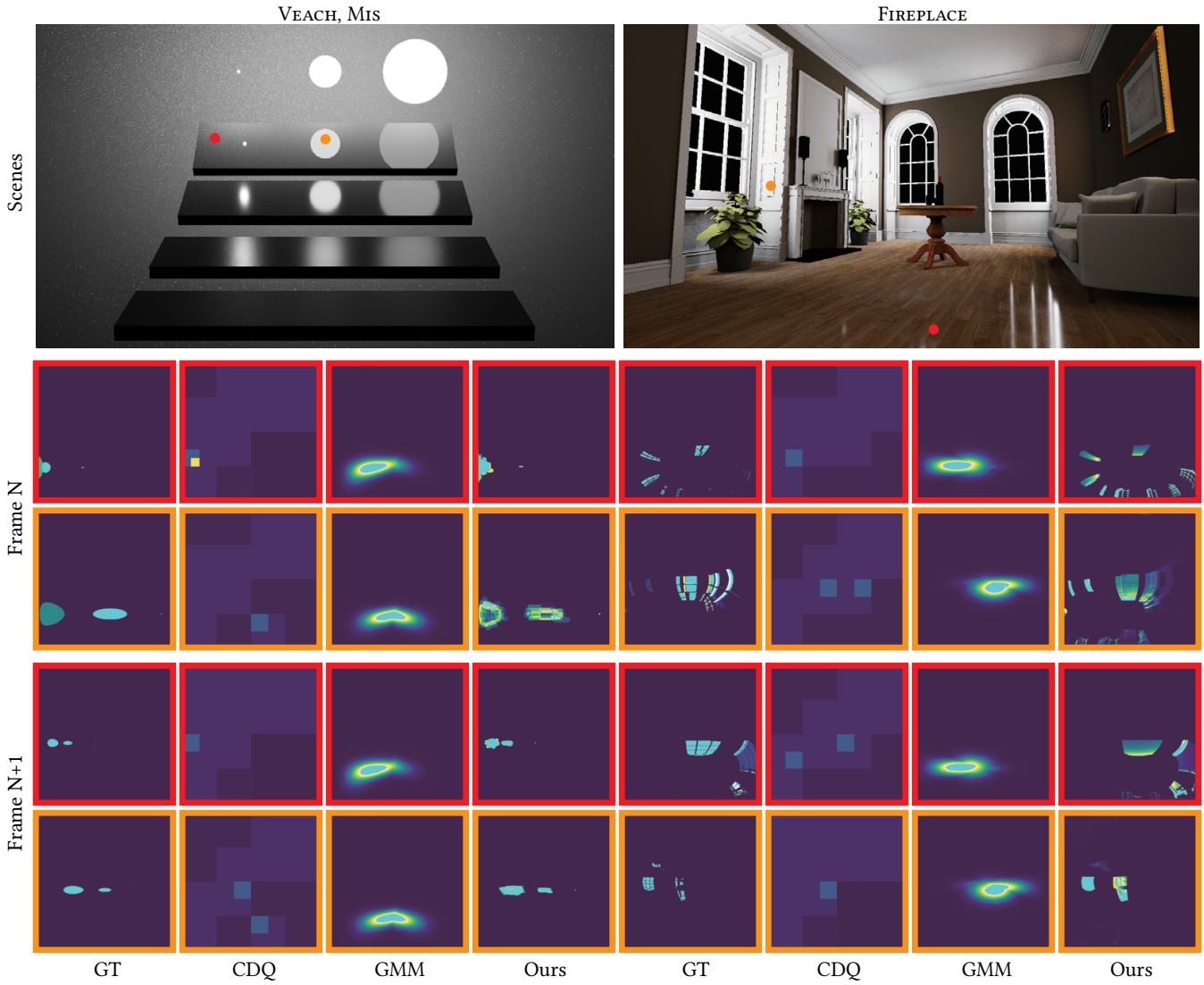


Fig. 9. We analyze the directional distribution that fits the one-bounce radiance, represented in cylindrical coordinates [Lambert 1770]. Our comparative study involves our method (VXPG) and the reference radiance (GT), as well as Compressed Directional Quadtree (CDQ) [Dittebrandt et al. 2020] and a one-lobe Gaussian mixture model (GMM) [Derevyannykh 2022] (which is learned under a hemispherical mapping [Shirley and Chiu 1997]). The scene and lighting remain static until frame N, where N is sufficiently large to ensure stable convergence across all methods. However, on frame N+1, a sudden movement of the light source results in a change in radiance, and it is evident that CDQ and SSPG exhibit slow adaptation.

Table 3. Breakdown of the VXPG pipeline execution time for 2-bounce global illumination. (*: run only once for static scenes geometries.)

	Visibility Buffer	Light Injection	Geometry Injection*	Superpixel Clustering	Supervoxel Clustering	Light Tree Building	Evaluating Average Visibility	Path Tracing
Fireplace	0.39 ms	0.09 ms	0.57 ms	0.13 ms	0.16 ms	0.13 ms	0.21 ms	10.77 ms
Veach Ajar	0.40 ms	0.09 ms	2.15 ms	0.13 ms	0.16 ms	0.12 ms	0.17 ms	7.01 ms
Staircase	0.76 ms	0.09 ms	2.78 ms	0.13 ms	0.17 ms	0.13 ms	0.29 ms	12.03 ms
Breakfast	0.68 ms	0.09 ms	2.38 ms	0.14 ms	0.18 ms	0.14 ms	0.40 ms	15.15 ms
Kitchen	0.52 ms	0.09 ms	11.52 ms	0.13 ms	0.16 ms	0.14 ms	0.24 ms	9.37 ms
Bedroom	0.68 ms	0.11 ms	5.02 ms	0.13 ms	0.17 ms	0.14 ms	0.31 ms	13.33 ms

Fig. 10. Image quality comparisons for our approach (VXPG), ReSTIR GI [Ouyang et al. 2021] (ReSTIR), and their combination (ReSTIR + VXPG). We also show the effect of the ReSTIR variants after both small and large amounts of temporal reuse. The top row represents images captured after 5 frames of temporal reuse, while the bottom row corresponds to images captured after more than 500 frames with M-cap = 20.

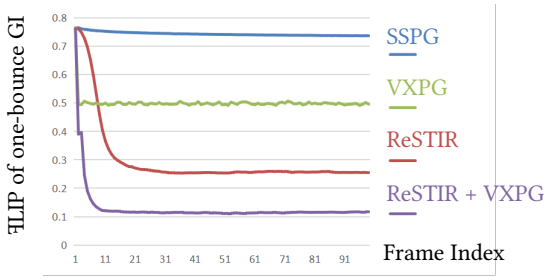


Fig. 11. To assess adaptation speed, we studied first-bounce indirect illumination in the Veach, Ajar scene over 10 frames. SSPG utilizes 10 neighbor samples to accelerate convergence, as detailed in [Derevyannykh 2022]. ReSTIR reuses 1 spatial sample and the temporal sample at every frame, with M-cap = 20.

(a) BSDF + A-SVGF (b) VXPG + A-SVGF (c) Reference

Fig. 12. (a) Noisy inputs can cause shadow and highlight detail loss and color distortion, even with A-SVGF denoising [Schied et al. 2018]. (b) Our approach improves the denoised image quality.

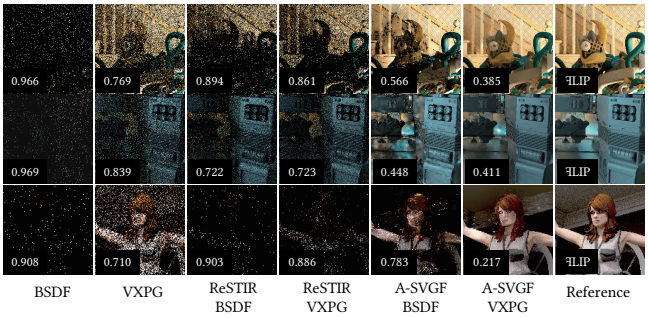
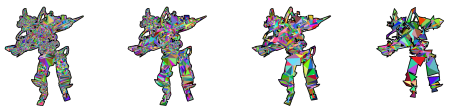
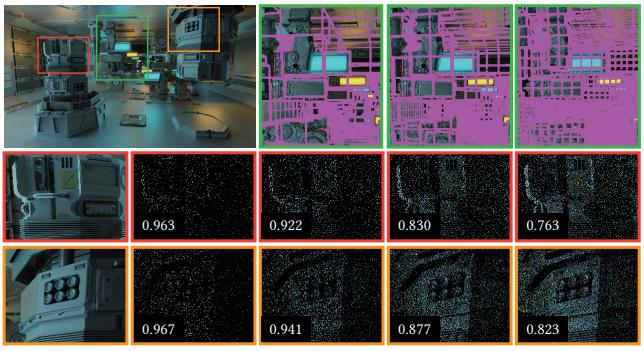


Fig. 13. Our approach can effectively adapt in dynamic scenes, enhancing the quality of ReSTIR GI and A-SVGF. BrainStem @Keith Hunter, Zero Day ©beeple [Winkelmann 2019], Disco @Md Mahmudur Rahman Bappy

Table 4. The mean ꟻLIP and average time of geometry injection for the Staircase BrainStem scene are assessed across various combinations of voxel resolution and mesh level of detail. We also show the time for the rest of the pipeline to show the impact of increasing voxel resolution. Additionally, we show the mesh at different LoDs and the number of the triangles.

	Resolution 64^3	Resolution 128^3	Resolution 256^3
LoD-0	0.872 1.042 ms	0.849 0.628 ms	0.811 0.454 ms
LoD-1	0.872 0.313 ms	0.854 0.256 ms	0.804 0.190 ms
LoD-2	0.875 0.112 ms	0.838 0.087 ms	0.803 0.107 ms
LoD-3	0.870 0.078 ms	0.843 0.065 ms	0.822 0.082 ms
Rest of the pipeline	15.88 ms	16.87 ms	20.23 ms
LoD level			
# of triangles	123,332	36,998	12,328	6,164



(a) Reference (b) BSDF (c) VXPG 64^3 / 128^3 / 256^3

Fig. 14. Direct illumination and 1st bounce indirect lighting without NEE by VXPG with various voxel resolutions. Error shown in ꟻLIP.

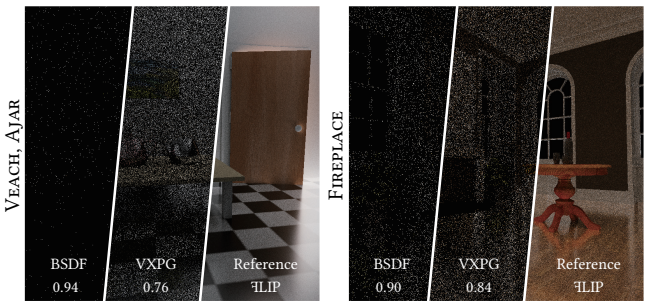


Fig. 15. An equal sample comparison of the second-bounce-only indirect illumination. VXPG uses power-based sampling [Shirley et al. 1996] for voxel selection, as the visibility information is in screen-space and thus not available during the second bounce.

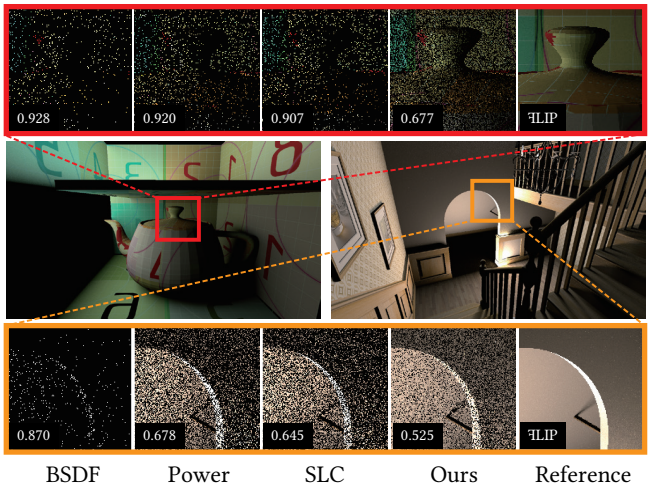


Fig. 16. The first bounce indirect illumination sampled by BSDF sampling and VXPG with various voxel selection strategies.

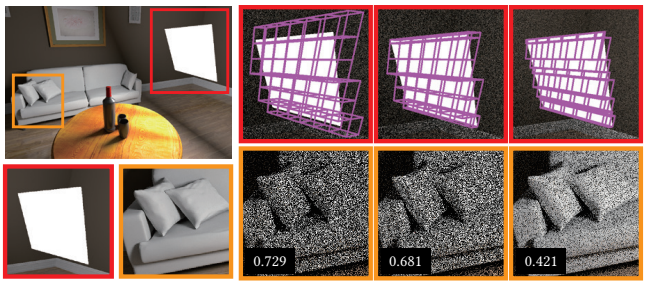


Fig. 17. Direct illumination without NEE and guided by VXPG, with various geometry compaction strategies. Error shown in ꟻLIP.

(e.g. the Veach Ajar scene). Note that VXPG with naïve voxel selection strategies is still able to outperform BSDF sampling.

Geometry compaction strategies. Geometry compaction has a substantial impact on the rejection rate during sampling. Fig. 17 shows the different compaction strategies discussed in Fig. 4. Using a more compact AABB can yield significantly better results.

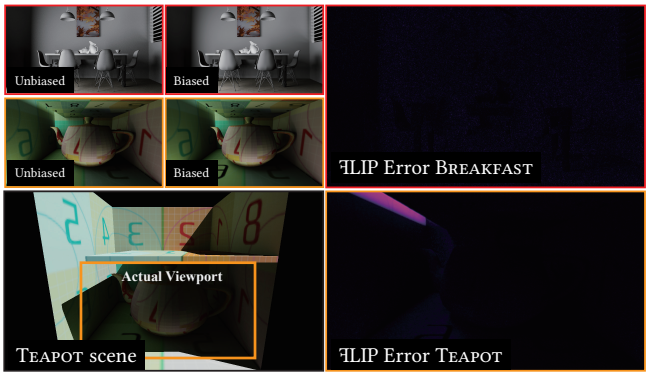


Fig. 18. The bias of naïve balance heuristic MIS estimator in Breakfast and Teapot scenes, as measured by ꟻLIP.

7 DISCUSSION, LIMITATIONS AND FUTURE WORK

Unbiasedness. As mentioned in Section 5.4, BSDF samples serve dual purposes: light injection and MIS. While reusing a single BSDF path for both purposes is attractive, it should be approached with caution. In particular, if, for each frame, we spawn a BSDF path and successively use it for light injection and MIS, the outcome will be biased. Other adaptive sampling methods [Kirk and Arvo 1991] share the concern as well.

There are two ways to address the problem: 1) Using x_2 of BSDF paths from the previous frame for light injection, resulting in a one-frame lag for path guiding. All previous path guiding approaches use this strategy, given their primary focus on static scenes. 2) Spawn two BSDF rays for light injection and MIS, respectively. This introduces extra overhead but can be beneficial in highly dynamic scenarios.

In practice, we can also simply ignore the bias. Experiments suggest that the bias is closely related to the light injection process. As illustrated in Fig. 18, when light injection successfully captures most of the voxels that contribute to the image, such as in the Breakfast scene, no bias is observed. Conversely, the Teapot scene presents an extreme case. The specific viewport configuration complicates the identification of all contributing voxels in the upper section, with only a few pixels in the upper-left corner of the image being affected. In scenarios like this, the missed voxels will bring bias to corresponding pixels. However, our light injection strategy tends to identify voxels that contribute to a greater number of pixels, which results in any bias being relatively small and localized.

Temporal Stability. In most cases, our approach is temporally stable. That being said, different voxel clustering can lead to different amounts of variance across frames, especially in scenes with complex visibility. An option is to improve the clustering by refining the local clusters for each superpixel [Ou and Pellacini 2011].

Caustics Transport. In our approach, we only estimate irradiance values for bounding voxels, and assume each voxel is a diffuse emitter during sampling. While our approximate product sampling during voxel selection can handle simple specular paths, this assumption limits our ability to guide caustics. To be more specific,

as we neglect the potential specular effect at x_2 vertices, guiding general indirect light reflected from a glossy surface towards a primary vertex would be challenging. At the same time, VXPG can potentially be used to guide photon tracing [Jensen 1995].

Scalability to large-scale scenes. While we have showcased promising results on moderately large scenes like Zero-Day, which includes 5.2 million triangles. There are two main concerns regarding further scalability: 1) In terms of performance, we have noted that the cost of geometry injection generally scales linearly with the number of triangles, see Table 4. To mitigate this, we have adopted LoD as a solution. 2) Regarding quality, large scenes may lead to less precise bounds of geometries, as illustrated in Fig. 14. To remedy this, a higher resolution voxel is essential. We have implemented spatial hashing for sparse storage, but a hierarchical structure such as a clipmap [Tanner et al. 1998] or sparse voxel octrees [Laine and Karras 2011] might be useful for very large-scale scenes.

8 CONCLUSION

We presented a path guiding algorithm that can learn and adapt complex distributions in real time. At the heart of our method, bounding voxel sampling demonstrates the feasibility of using a pure spatial distribution for path guiding, which is visibility-aware and can easily adapt to dynamic scenes. It removes the need for integration that arises when using spatial distributions from previous work, by sampling on voxel boundaries and employing rejection sampling.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their invaluable feedback, and to Yash Belhe for useful reviews and discussions. This work was supported in part by NSF grant 2105806 and gifts from Adobe and Google.

REFERENCES

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. 2012. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Trans. Pattern Anal. Mach. Intell. 34, 11 (2012), 2274–2282.

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. Proc. ACM Comput. Graph. Interact. Tech. 3, 2, Article 15 (2020), 23 pages.

James Arvo. 1995. Stratified Sampling of Spherical Triangles. In SIGGRAPH. 437–438.

Thomas Bashford-Rogers, Kurt Debattista, and Alan Chalmers. 2012. A significance cache for accelerating global illumination. Comput. Graph. Forum 31, 6 (2012), 1837–1851.

Philippe Bekaert, Mateu Sbert, and John H Halton. 2002. Accelerating Path Tracing by Re-Using Paths. Rendering Techniques (Proc. EGWR) 2 (2002), 125–134.

Nikolaus Binder, Sascha Fricke, and Alexander Keller. 2021. Massively Parallel Path Space Filtering. arXiv:1902.05942 [cs.GR]

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. ACM Trans. Graph. (Proc. SIGGRAPH) 39, 4 (2020), 148.

Cyril Crassin and Simon Green. 2012. Octree-based sparse voxelization using the GPU hardware rasterizer. OpenGL Insights (2012), 303–318.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive Indirect Illumination Using Voxel Cone Tracing. Computer Graphics Forum (Pacific Graphics) 30, 7 (2011).

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable Realistic Rendering with Many-Light Methods. Comput. Graph. Forum (Proc. Eurographics STAR) 33, 1 (2014), 88–104.

Tomáš Davidovič, Jaroslav Křivánek, Miloš Hašan, Philipp Slusallek, and Kavita Bala. 2010. Combining global and local virtual lights for detailed glossy illumination. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 29, 6 (2010), 1–8.

Mikhail Derevyannykh. 2022. Real-Time Path-Guiding Based on Parametric Mixture Models. In Eurographics Short Papers.

Addis Dittebrandt, Johannes Hanika, and Carsten Dachsbacher. 2020. Temporal Sample Reuse for Next Event Estimation and Path Guiding for Real-Time Path Tracing. In Eurographics Symposium on Rendering - DL-only Track.

Addis Dittebrandt, Vincent Schüß ler, Johannes Hanika, Sebastian Herholz, and Carsten Dachsbacher. 2023. Markov Chain Mixture Models for Real-Time Direct Illumination. Comput. Graph. Forum (Proc. EGSR) (2023).

Ana Dodik, Marios Papas, Cengiz Öztireli, and Thomas Müller. 2022. Path Guiding Using Spatio-Directional Mixture Models. Comput. Graph. Forum 41, 1 (2022), 172–189.

Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. In SIGGRAPH Conference Proceedings (Los Angeles, CA, USA) (SIGGRAPH '23). Article 29, 10 pages.

Manuel N. Gamito. 2016. Solid Angle Sampling of Disk and Cylinder Lights. Comput. Graph. Forum 35, 4 (jul 2016), 25–36.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In SIGGRAPH. 209–216.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix row-column sampling for the many-light problem. ACM Trans. Graph. (Proc. SIGGRAPH) 26, 3 (2007), 26.

Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. 2009. Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes. ACM Trans. Graph. (Proc. SIGGRAPH Asia), Article 143 (2009), 6 pages.

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. In Rendering Techniques (Proc. EGWR). 326–335.

James T. Kajiya. 1986. The Rendering Equation. Comput. Graph. (Proc. SIGGRAPH) 20, 4 (1986), 143–150.

Alexander Keller. 1997. Instant Radiosity. In SIGGRAPH. 49–56.

Juhyeon Kim and Young Min Kim. 2021. Fast and Lightweight Path Guiding Algorithm on GPU. In Pacific Graphics Short Papers, Posters, and Work-in-Progress Papers, Sung-Hee Lee, Stefanie Zollmann, Makoto Okabe, and Burkhard Wünsche (Eds.).

David Kirk and James Arvo. 1991. Unbiased sampling techniques for image synthesis. Comput. Graph. (Proc. SIGGRAPH) 25, 4 (1991), 153–156.

Eric P. Lafortune and Yves D. Willems. 1995. A 5D tree to reduce the variance of Monte Carlo ray tracing. In Eurographics Workshop on Rendering. 11–20.

Samuli Laine and Tero Karras. 2011. Efficient Sparse Voxel Octrees – Analysis, Extensions, and Implementation. Technical Report. NVIDIA Technical Report NVR-2010-001.

J.H. Lambert. 1770. Beiträge zum Gebrauche der Mathematik und deren Anwendung. Vol. 1. Im Verlag der Buchhandlung der Königl. Realschule.

Tianyu Li, Wenyou Wang, Daqi Lin, and Cem Yuksel. 2022. Virtual Blue Noise Lighting. Proc. ACM Comput. Graph. Interact. Tech. (Proc. HPG) 5, 3, Article 23 (2022), 26 pages.

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: foundations of ReSTIR. ACM Trans. Graph. (Proc. SIGGRAPH) 41, 4 (2022), 1–23.

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. Proc. ACM Comput. Graph. Interact. Tech. (Proc. I3D) 3, 1, Article 5 (2020), 18 pages.

Yifan Liu, Kun Xu, and Ling-Qi Yan. 2019. Adaptive BRDF-Oriented Multiple Importance Sampling of Many Lights. Comput. Graph. Forum (Proc. EGSR) 38, 4 (2019), 123–133.

Pierre Moreau, Matt Pharr, and Petrik Clarberg. 2019. Dynamic Many-Light Sampling for Real-Time Ray Tracing.. In High Performance Graphics (Short Papers). 21–26.

P. Moreau, M. Pharr, and P. Clarberg. 2022. Dynamic Many-Light Sampling for Real-Time Ray Tracing. In High Performance Graphics. 21–26.

Thomas Müller. 2019. "Practical Path Guiding" in Production. In ACM SIGGRAPH Courses: Path Guiding in Production, Chapter 10. 18:35–18:48.

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical Path Guiding for Efficient Light-Transport Simulation. Comput. Graph. Forum (Proc. EGSR) 36, 4 (2017), 91–100.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural importance sampling. ACM Transactions on Graphics (ToG) 38, 5 (2019), 1–19.

Jiawei Ou and Fabio Pellacini. 2011. LightSlice: matrix slice sampling for the many-lights problem. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 30, 6 (2011), 179:1–179:8.

Y. Ouyang, S. Liu, M. Kettunen, M. Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. Computer Graphics Forum (Proc. HPG) 40 (12 2021), 17–29.

Jacopo Pantaleoni. 2019. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. arXiv preprint arXiv:1911.10217 (2019).

Jacopo Pantaleoni. 2020. Online Path Sampling Control with Progressive Spatio-temporal Filtering. SN Comput. Sci. 1, 5 (2020), 16 pages.

Jacopo Pantaleoni and Eric Heitz. 2017. Notes on optimal approximations for importance sampling. arXiv preprint arXiv:1707.08358 (2017).

Eric Paquette, Pierre Poulin, and George Drettakis. 1998. A Light Hierarchy for Fast Rendering of Scenes with Many Lights. Comput. Graph. Forum (Proc. Eurographics)

(1998), 63–74.
Christoph Peters. 2021. BRDF Importance Sampling for Polygonal Lights. ACM Trans. Graph. 40, 4, Article 140 (jul 2021), 14 pages. https://doi.org/10.1145/3450626.3459672
Stefan Popov, Ravi Ramamoorthi, Fredo Durand, and George Drettakis. 2015. Probabilistic connections for bidirectional path tracing. In Computer Graphics Forum, Vol. 34. Wiley Online Library, 75–86.
Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-Aware Path Guiding. ACM Transactions on Graphics (Proceedings of SIGGRAPH 2020) 39, 4 (July 2020), 151:1–151:12. https://doi.org/10.1145/3386569.3392441
Alexander Rath, Ömercan Yazici, and Philipp Slusallek. 2023. Focal Path Guiding for Light Transport Simulation. In SIGGRAPH Conference Proceedings. Article 30, 10 pages.
Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. 2020. Robust Fitting of Parallax-Aware Mixtures for Path Guiding. ACM Trans. Graph. 39, 4, Article 147 (aug 2020), 15 pages. https://doi.org/10.1145/3386569.3392421
Christoph Schied, Christoph Peters, and Carsten Dachsbacher. 2018. Gradient Estimation for Real-Time Adaptive Temporal Filtering. Proc. ACM Comput. Graph. Interact. Tech. (Proc. HPG) 1, 2, Article 24 (2018), 16 pages.
B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche. 2006. Bidirectional Instant Radiosity. Rendering Techniques (Proc. EGSR) (2006), 389–397.
Peter Shirley and Kenneth Chiu. 1997. A Low Distortion Map Between Disk and Square. Journal of Graphics Tools 2, 3 (1997), 45–52.
Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo Techniques for Direct Lighting Calculations. ACM Trans. Graph. 15, 1 (1996), 1–36.
Florian Simon, Johannes Hanika, and Carsten Dachsbacher. 2015. Rich-VPLs for Improving the Versatility of Many-Light Methods. Comput. Graph. Forum (Proc. Eurographics) 34, 2 (May 2015), 575–584.
Florian Simon, Johannes Hanika, Tobias Zirr, and Carsten Dachsbacher. 2017. Line Integration for Rendering Heterogeneous Emissive Volumes. Comput. Graph. Forum (Proc. EGSR) 36, 4 (July 2017), 101–110.
Fujia Su, Sheng Li, and Guoping Wang. 2022. SPCBPT: subspace-based probabilistic connections for bidirectional path tracing. ACM Trans. Graph. (Proc. SIGGRAPH) 41, 4 (2022), 1–14.
Ivan E Sutherland and Gary W Hodgman. 1974. Reentrant polygon clipping. Commun. ACM 17, 1 (1974), 32–42.
Justin F. Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. Rendering Techniques (Proc. EGSR) (2005), 139–146.
Christopher C. Tanner, Christopher J. Migdal, and Michael T. Jones. 1998. The Clipmap: A Virtual Mipmap. In SIGGRAPH. 151–158.
Wolfgang Tatzgern, Benedikt Mayr, Bernhard Kerbl, and Markus Steinberger. 2020. Stochastic Substitute Trees for Real-Time Global Illumination. In Symposium on Interactive 3D Graphics and Games. Article 2, 9 pages.
Yusuke Tokuyoshi. 2015. Virtual Spherical Gaussian Lights for Real-time Glossy Indirect Illumination. Comput. Graph. Forum (Proc. PG) 34, 7 (2015), 89–98.
Carlos Ureña, Marcos Fajardo, and Alan King. 2013. An Area-Preserving Parametrization for Spherical Rectangles. Comput. Graph. Forum (Proc. EGSR) 32, 4 (2013), 59–66.
Eric Veach. 1998. Robust Monte Carlo Methods for Light Transport Simulation. Ph. D. Dissertation. Stanford, CA, USA. Advisor(s) Guibas, Leonidas J. AAI9837162.
Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In SIGGRAPH. 419–428.
Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian Online Regression for Adaptive Direct Illumination Sampling. ACM Trans. Graph. (Proc. SIGGRAPH) 37, 4 (2018), 125:1–125:12.
Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. 2019. Path Guiding in Production. In ACM SIGGRAPH Courses. Article 18, 77 pages.
Jirí Vorba, Ondrej Karlík, Martin Sik, Tobias Ritschel, and Jaroslav Krivánek. 2014. On-line learning of parametric mixture models for light transport simulation. ACM Trans. Graph. (Proc. SIGGRAPH) 33, 4 (2014), 101:1–101:11.
Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. 2005. Lightcuts: A Scalable Approach to Illumination. ACM Trans. Graph. (Proc. SIGGRAPH) 24, 3 (2005), 1098–1107.
Yu-Chen Wang, Yu-Ting Wu, Tzu-Mao Li, and Yung-Yu Chuang. 2021. Learning to Cluster for Rendering with Many Lights. ACM Trans. Graph. (Proc. SIGGRAPH Asia) 40, 6, Article 277 (2021), 10 pages.
Mike Winkelmann. 2019. Zero-Day, Open Research Content Archive (ORCA). https://developer.nvidia.com/orca/beeple-zero-day
Yu-Ting Wu and Yung-Yu Chuang. 2013. VisibilityCluster: Average directional visibility for many-light rendering. IEEE Trans. Vis. Comput. Graph. 19, 9 (2013), 1566–1578.
Cem Yuksel. 2019. Stochastic Lightcuts. In High Performance Graphics. 27–32.
Cem Yuksel. 2020. Stochastic lightcuts for sampling many lights. IEEE Trans. Vis. Comput. Graph. 27, 10 (2020), 4049–4059.