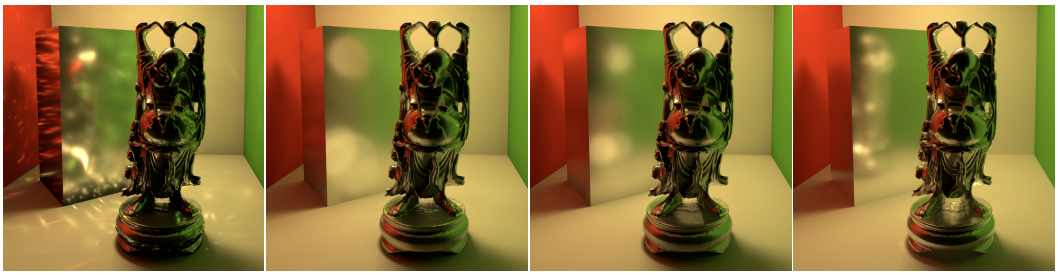
Virtual Blue Noise Lighting

TIANYU LI*, WENYOU WANG*, DAQI LIN, and CEM YUKSEL, University of Utah, USA



(a) *VPLs* **(b)** *VSLs* **(c)** *Rich-VSLs* **(d)** ***VBNL (Ours)***

Fig. 1. *Different methods for estimating indirect illumination using virtual lights: (a) virtual point lights [Keller 1997] producing illumination spikes, (b) virtual spherical lights [Hašan et al. 2009] blurring those spikes, (c) Rich-VSLs [Simon et al. 2015] improving the results by storing a non-uniform emission profile, and (d) our virtual blue noise lighting. All images rendered using 12K virtual lights.*

We introduce virtual blue noise lighting, a rendering pipeline for estimating indirect illumination with a blue noise distribution of virtual lights. Our pipeline is designed for virtual lights with non-uniform emission profiles that are more expensive to store, but required for properly and efficiently handling specular transport.

Unlike the typical virtual light placement approaches that traverse light paths from the original light sources, we generate them starting from the camera. This avoids two important problems: wasted memory and computation with fully-occluded virtual lights, and excessive virtual light density around high-probability light paths. In addition, we introduce a parallel and adaptive sample elimination strategy to achieve a blue noise distribution of virtual lights with varying density. This addresses the third problem of virtual light placement by ensuring that they are not placed too close to each other, providing better coverage of the (indirectly) visible surfaces and further improving the quality of the final lighting estimation.

For computing the virtual light emission profiles, we present a photon splitting technique that allows efficiently using a large number of photons, as it does not require storing them. During lighting estimation, our method allows using both global power-based and local BSDF important sampling techniques, combined via multiple importance sampling. In addition, we present an adaptive path extension method that avoids sampling nearby virtual lights for reducing the estimation error.

We show that our method significantly outperforms path tracing and prior work in virtual lights in terms of both performance and image quality, producing a fast but biased estimate of global illumination.

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

*Joint first authors: both authors contributed equally to this work.

Authors' address: Tianyu Li, ltyucb@gmail.com; Wenyou Wang, wenyouwang@outlook.com; Daqi Lin, daqi@cs.utah.edu; Cem Yuksel, cem@cemyuksel.com, University of Utah, USA.

2577-6193/2022/7-ART23 $15.00
https://doi.org/10.1145/3543872

Additional Key Words and Phrases: Virtual lights, virtual point lights, virtual spherical lights, many lights, instant radiosity, global illumination, light sampling, blue noise sampling, sample elimination

ACM Reference Format:
Tianyu Li, Wenyou Wang, Daqi Lin, and Cem Yuksel. 2022. Virtual Blue Noise Lighting. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 23 (July 2022), 26 pages. https://doi.org/10.1145/3543872

1 INTRODUCTION

Indirect illumination estimation using *virtual lights* has been a popular technique since the introduction of *instant radiosity* [Keller 1997]. Instead of directly sampling the path space during rendering, they cache light paths into virtual lights prior to rendering. During rendering, camera paths can be terminated in advance by connecting shading points to virtual lights, providing an efficient global illumination estimation. Obviously, the estimation quality is correlated with the number of virtual lights. Using a large number effectively converts the indirect illumination estimation problem to the many-lights problem.

On the other hand, simply increasing the number of virtual lights does not solve all problems with this approach. For example, using *virtual point lights* (VPLs) for illuminating highly-specular surfaces produces undesirable illumination spikes (Figure 1a). *Virtual spherical ligths* (VSLs) [Hašan et al. 2009] effectively blur such illumination spikes (Figure 1b). Combining them with directional emission profiles [Simon et al. 2015] improves the result (Figure 1c) at a cost of significantly inflated storage demand. This, in turn, limits the number of virtual lights that can be used in practical applications. Also, with more virtual lights, efficiently sampling them becomes a challenge.

In this paper, we introduce *virtual blue noise lighting* (VBNL), a complete redesign of the rendering pipeline using virtual lights. We begin with virtual light placement, a crucial factor determining the rendering quality, and present efficient methods for computing their emission profiles and sampling them during rendering. We rely on a similar virtual light storage as *Rich-VPLs* [Simon et al. 2015], containing non-uniform emission profiles. Our pipeline includes the following departures from prior methods:

- **Virtual Light Generation:** Unlike typical methods that generate virtual lights from the original scene lights, we start from the camera. This guarantees that *all* virtual lights with non-zero emission contribute to the final image, preventing a substantial amount of unnecessary computation and storage. Notably, this leads to improved quality with a set number of virtual lights, as our virtual light density is correlated with camera importance.
- **Virtual Light Distribution:** Simon et al. [Simon et al. 2015] have shown that virtual light distribution with blue noise characteristics improves the rendering quality. However, the relaxation method they use for achieving this is difficult to control and does not always converge to a desirable distribution. It also requires storing all photons used during the emission profile computation. We replace it with a form of *sample elimination* [Yuksel 2015], including a new parallel computation approach and a novel adaptive density factor that automatically adjusts the virtual light density based on a spatial importance measure. This also helps with assigning a radius to each virtual light.
- **Emission Profile Computation:** We present a *photon splitting* technique that efficiently computes the incident illumination for each virtual light without storing any photons. This incident illumination profile per light is then converted to an emission profile. This approach allows efficiently using a large number of photons for computing the emission profiles.
- **Virtual Light Sampling:** We combine three strategies via multiple importance sampling. We use the traditional power-based importance sampling for distant illumination only. We employ a BSDF importance sampling strategy that relies on our virtual light representation

for efficient illumination estimation. In addition, we use adaptive path extension, as needed, to improve the accuracy of indirect illumination estimation from nearby surfaces.

Our results show that our VBNL approach provides substantial improvement in both quality and performance over the state-of-the-art in virtual lights (Figure 1). Preliminary experiments for our virtual light generation and distribution approaches were conducted by Montazer [2017], using VPLs without emission profiles on Lambertian surfaces.

2 BACKGROUND

Since the introduction of instant radiosity [Keller 1997], estimating indirect illumination with virtual lights has been a popular solution [Dachsbacher et al. 2014]. This approach can be considered a variant of bidirectional path tracing [Veach and Guibas 1995], in which the same light paths are shared by all camera paths. Therefore, relatively few VPLs are enough to approximate indirect illumination without noise. Virtual lights have also been used for approximating volumetric scattering using ray [Novák et al. 2012b] or beam [Novák et al. 2012a] lights.

The rendering efficiency of virtual lights stems from the fact that they are a set of light path vertices that can be reused globally. Density estimation techniques like photon mapping [Jensen 1996] and path-space filtering [Keller et al. 2014] allow path vertices to be reused locally, which have the strength of smoothing out low-probability path samples like caustics samples. These two types of reuse can be combined in frameworks like Vertex-Connection and Merging (VCM) [Georgiev et al. 2012] or Unified Path Sampling (UPS) [Hachisuka et al. 2012] to improve the sampling efficiency further. Our method shares some similarities with VCM/UPS in the sense that our virtual lights can be reused both globally and locally (Section 4). Different from VCM/UPS, our method reuses virtual lights (instead of path vertices) that have a spatial extent [Hašan et al. 2009] and enriched emission profiles [Simon et al. 2015]. While our method advances the frontier of virtual lights rendering, it is also related to radiance caching [Krivánek et al. 2005; Müller et al. 2021; Ward et al. 1988] and point-based global illumination (e.g., surfel) [Christensen 2010].

2.1 Virtual Lights and Light Transport

The types of virtual lights used in indirect illumination estimation vary in their ability to handle different forms of light transport:

$L_{D\to D}$: Light from diffuse surfaces illuminating diffuse surfaces
$L_{D\to S}$: Light from diffuse surfaces illuminating specular surfaces
$L_{S\to D}$: Light from specular surfaces illuminating diffuse surfaces
$L_{S\to S}$: Light from specular surfaces illuminating specular surfaces

Virtual point lights (VPLs), a popular choice, provide a simple representation using a constant emission profile. VPLs can only handle diffuse transport (i.e. $L_{D\to D}$) and they lead to illumination spikes with any form of specular transport (i.e. $L_{D\to S}$, $L_{S\to D}$, or $L_{S\to S}$), as shown in Figure 1a. In addition, they suffer from singularity in the geometry term due to inverse-square attenuation, resulting in illumination spikes with diffuse transport as well. Clamping the geometry term leads to energy loss, which requires expensive treatment to compensate, such as introducing path tracing to handle nearby indirect illumination [Kollig and Keller 2006]. Alternatively, VPLs close to the shading point are treated as photons, avoiding the geometric singularity without introducing energy loss [Sriwasansak et al. 2018]

Virtual spherical lights (VSLs) [Hašan et al. 2009] avoid the singularity by replacing VPLs with simplified spherical lights, which reduces the energy loss. For handling specular transport, each VSL stores the incoming illumination and material/surface information at its center. The outgoing illumination towards any given direction is computed at render time. This provides an effective

solution to $L_{D\to D}$ and $L_{D\to S}$. It also helps with $L_{S\to D}$ and $L_{S\to S}$, but may suffer from excessive blurring (Figure 1b).

A more general solution to specular transport is provided by *Rich-VPLs* and *Rich-VSLs* [Simon et al. 2015] by storing an outgoing illumination profile per virtual light. Unlike a VSL that is generated from a single light path, the emission profile of a Rich-VSL is computed from multiple light paths using a photon map [Jensen 2001]. As a result, Rich-VSLs provide a more effective solution to $L_{S\to D}$ and $L_{S\to S}$ (Figure 1c). On the other hand, the emission profile significantly increases the storage cost per virtual light, thereby limiting the number of virtual lights that can be used in practice.

2.2 Evaluating Many Lights

Instant radiosity effectively converts the indirect illumination problem into the many-lights problem. Brute-force computation of many lights can be highly inefficient. Fortunately, there are various efficient methods for estimation, including clustering using a light tree and finding important clusters [Davidovič et al. 2012; Paquette et al. 1998; Walter et al. 2006, 2005, 2012], refactoring the light-surface interaction into a lighting matrix to evaluate it approximately [Davidovic et al. 2010; Hašan et al. 2007; Huo et al. 2015; Ou and Pellacini 2011], and using an multiresolutional grid representation [Lin and Yuksel 2019; Yuksel and Yuksel 2017]. Monte Carlo sampling provides a more general solution and can be used for interactive rendering, but requires an effective importance sampling strategy to achieve an estimation with low noise. Many recent methods build light trees to guide sampling [Estevez and Kulla 2018; Lin and Yuksel 2020; Moreau et al. 2019; Tatzgern et al. 2020; Vévoda et al. 2018; Vévoda and Křivánek 2016; Yuksel 2019]. Recently, spatiotemporal reservoir resampling (ReSTIR) [Bitterli et al. 2020] is found to be highly effective for sampling direct illumination on primary hit points.

Our VBNL method can be used with any of these many-lights solutions. However, simple intensity-based sampling [Shirley et al. 1996] combined with BSDF sampling turns out to be highly efficient for our method, enabling high convergence rate and low overhead which are ideal for interactive pre-visualization. MIS between BSDF and light sampling has also found to be highly effective by recent work in light tree sampling [Liu et al. 2019].

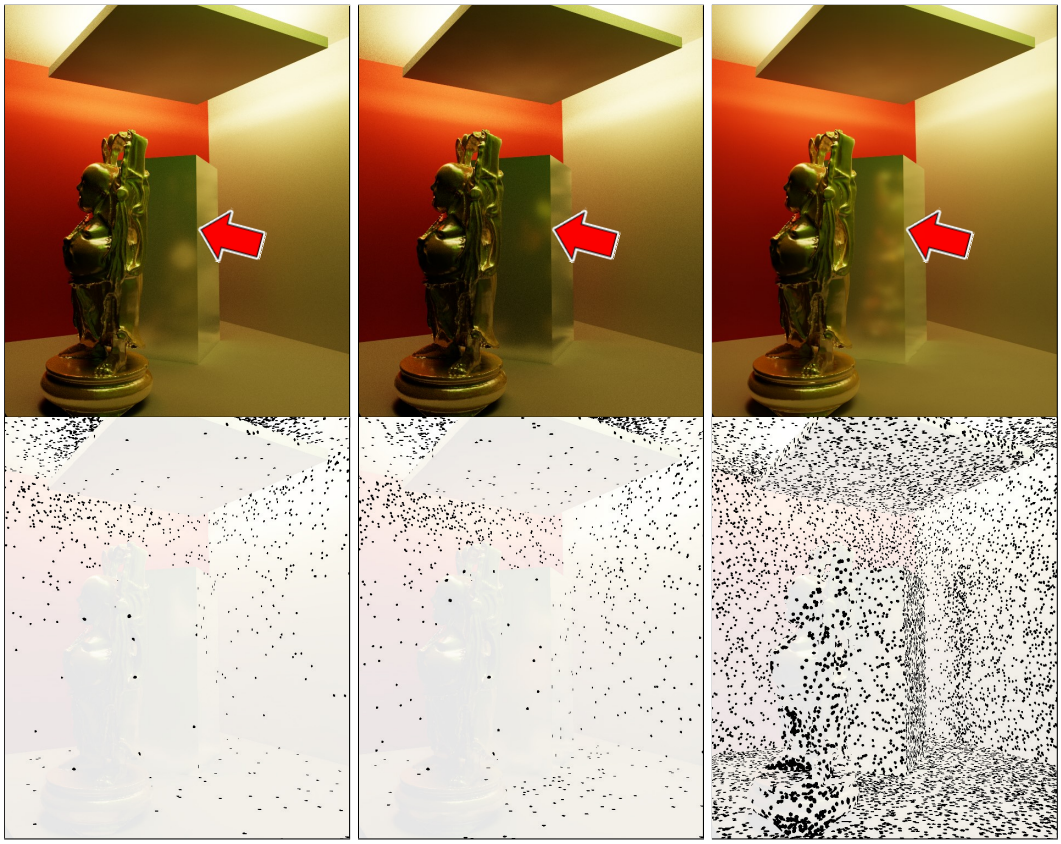
Our virtual light sampling also involves extending the camera paths, which is conceptually similar to Bidirectional Lightcuts [Walter et al. 2012] that combines light paths and camera paths.

2.3 Virtual Light Placement

Virtual lights are typically placed by tracing light paths from the original light sources [Georgiev and Slusallek 2010; Keller 1997; Segovia et al. 2007]. However, this approach has the following fundamental problems:

Problem 1: It can generate a large number of virtual lights that have no contribution to the final image, depending on the camera position and the light sources in the scene. This not only leads to wasted storage and computation, but also, by including many lights with no visibility in the lighting estimation, it hinders the efficiency and effectiveness of the many-lights solution used. This problem is demonstrated in Figure 2a with only a small portion of virtual lights contributing to the rendered image, resulting in a low-quality indirect illumination estimation.

Problem 2: "Hot zones," where a significant portion of the light paths go through, are often over-sampled with many virtual lights, even when their contributions to the final image could be effectively approximated using fewer virtual lights. On the other hand, light paths with lower probability but with more significant contributions to the final image are not



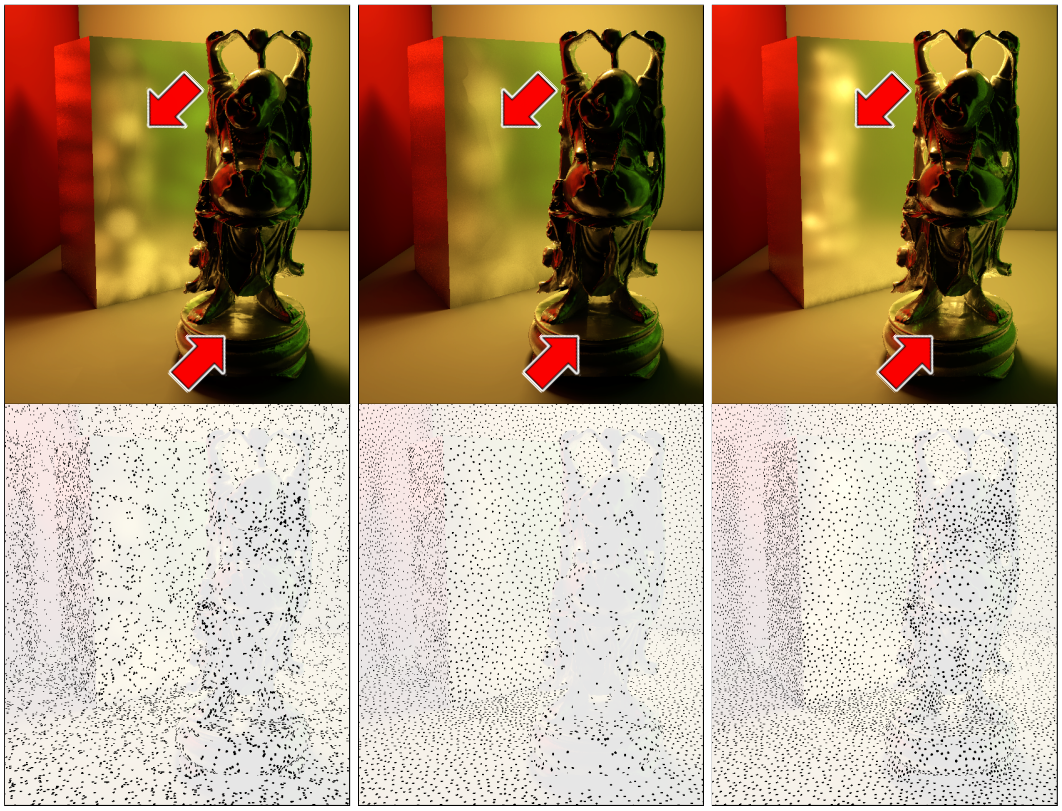
(a) *VSLs from light* **(b)** *with importons* **(c)** ***VSLs from camera***

Fig. 2. ***Comparison of VSLs placement methods.*** *(a) When VSLs are generated from the light sources, their distribution can be inadequate for visible surfaces to camera. (b) Generating VSLs from the light sources using an importon map [Simon et al. 2015] improves the results, but does not always produce a desirable placement for VSLs. (c) Generating VSLs from camera ensures that they all contribute to the final image, improving the quality of the indirect illumination estimation. The bottom row shows the VSL distributions. All images use 30400 VSLs with non-uniform emission profiles computed using our method.*

always sampled adequately. Figure 2a also demonstrates this with a large portion of virtual lights placed close to the original light source.

Problem 3: Random sampling leads to many virtual light pairs (or larger groups) that are placed close to each other and forms large gaps with no virtual lights on parts of illuminated surfaces (Figure 3a). This not only causes wasted storage and computation but also results in a low-frequency noise that is difficult to filter out.

Bidirectional instant radiosity [Segovia et al. 2006] addresses Problem 1 by introducing *reverse VPLs* that are generated by traversing camera paths with a single bounce (Figure 2c). To estimate the emission of reverse VPLs, a set of regular VPLs (generated with light paths) are used. Then, these two sets of VPLs are combined and a subset of them are used during rendering, selected via resampling based on the contribution of each VPL to the final image, estimated by sampling a set of



(a) *Random* **(b)** *Relaxation* **(c)** ***Sample Elimination***

Fig. 3. ***Comparison of VSL distributions.*** *(a) Randomly generating VSLs leads to an uneven distribution with tightly packed groups and relatively large spaces without VSLs. (b) Relaxation [Simon et al. 2015] improves the distribution in some places, but not everywhere. (c) Our sample elimination approach leads to a more even distribution and improves the indirect illumination estimation. The bottom row shows the VSL distributions. All images use 28K VSLs that are generated from camera and store non-uniform emission profiles computed using our method.*

camera paths connecting to them. This also involves computing the probability of generating each reverse VPL, which is first estimated by generating a small number of paths from the camera to each reverse VPL and then refined with more paths after resampling. Reverse VPLs help with Problem 1 and Problem 2, though the resampled regular VPLs still exhibit these problems (Figure 2a). This approach does not offer a solution to Problem 3. Another benefit of reverse VPLs is that a small set of VPLs that are indirectly visible to the camera can be maintained in a temporally coherent manner for interactive rendering [Hedman et al. 2017].

Other strategies for VPL placement include rejection sampling [Georgiev and Slusallek 2010], which suffers from rejecting too many real VPLs for a complex scene, and metropolis instant radiosity [Segovia et al. 2007], which produces high-variance with glossy surfaces.

Rich-VPLs [Simon et al. 2015] address Problem 1 using an *importon map* [Peter and Pietrek 1998] generated with camera paths. Virtual lights are placed by first generating a large number of photons as VPL candidates and then selecting a subset of them based on the importon map.

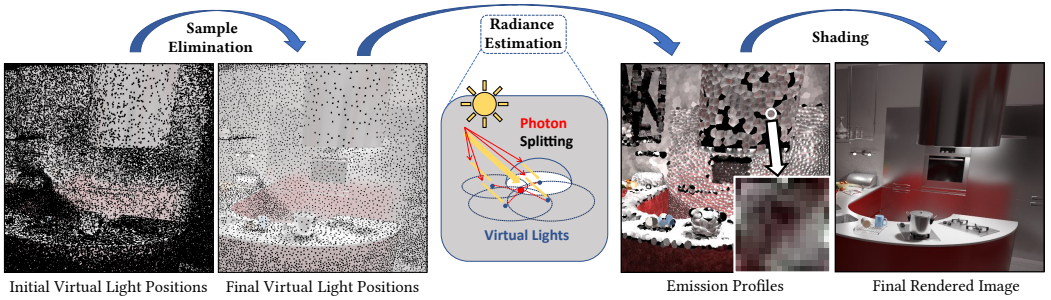


Fig. 4. *A visualization of our rendering pipeline. Initial virtual light positions are generated from the camera. A subset of them are selected as the final virtual light positions. Virtual light emission profiles are generated through radiance estimation using photon splitting. The final image is rendered by sampling the virtual lights during shading.*

This may also help with Problem 2 when high-probability light paths appear at places with no importons, but otherwise they can still get oversampled (Figure 2b). For addressing Problem 3, an iterative relaxation process [Spencer and Jones 2009; Turk 1991] is employed, which moves the selected VPLs and then snaps them back onto the closest photon after each step, effectively selecting a different photon location to improve the distribution. Though the relaxation process improves the virtual light distribution, it can easily get stuck in a local minimum (Figure 3b). The final emission profiles are computed using irradiance estimation from the photon map.

Our virtual light placement approach begins with a similar process as the reverse VPL generation of bidirectional instant radiosity to address Problem 1 (Figure 2c). However, we do not use VPLs generated with light paths to evaluate their intensities and avoid the complexity of resampling and probability computation. More importantly, this addresses Problem 2. We address Problem 3 using sample elimination [Yuksel 2015]. The resulting virtual light distribution significantly improves the lighting estimation (Figure 3c).

3 VIRTUAL BLUE NOISE LIGHTING (VBNL)

The rendering pipeline with our VBNL method is centered around the idea of storing a non-uniform emission profile per virtual light, akin to Rich-VPLs. When a virtual light stores an emission profile, it represents more than a single light sub-path. Instead, it corresponds to a collection of light sub-paths that reach the position of the virtual light. This concept allows decoupling the virtual light placement from its emission profile computation.

A visualization of our pipeline can be seen in Figure 4. In our VBNL pipeline we propose generating the virtual light positions by tracing camera sub-paths, instead of light sub-paths, to address Problem 1 and Problem 2 (Section 3.1). We handle Problem 3 by using a parallel and non-uniform version of the sample elimination method [Yuksel 2015] that results in a blue noise distribution of virtual light positions (Section 3.2). We compute the emission profile of virtual lights by first tracing a large number of photons from the light sources to estimate the incoming radiance field at each virtual light and then converting it to an outgoing radiance field (Section 3.3).

3.1 Initial Virtual Light Placement

With any rendering method using virtual lights, light paths are formed by explicitly connecting camera sub-paths to the virtual lights, each virtual light representing one or more light paths, as shown in Figure 5. Therefore, only the virtual lights at points $\mathbf{y}$ that are visible from the end points $\mathbf{x}$ of camera sub-paths can form valid light paths. Let $\mathbb{X}$ represent the set of all points $\mathbf{x}$ at the ends

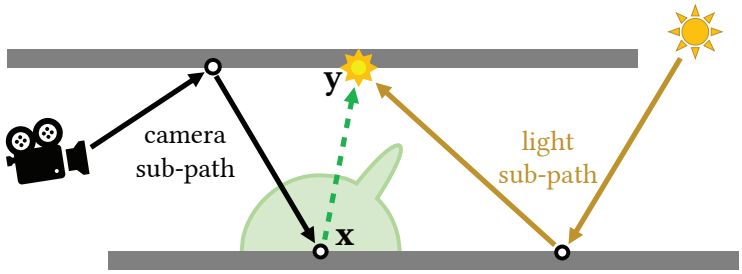


Fig. 5. *A light path generated by connecting the end point* $\mathbf{x}$ *of a camera sub-path with the light sub-path of a virtual light at* $\mathbf{y}$.

of camera sub-paths that can be generated during rendering, and $\mathbb{Y}$ be the set of all scene points $\mathbf{y}$ visible from any point $\mathbf{x} \in \mathbb{X}$. Then, a virtual light that is *not* on a surface point $\mathbf{y} \in \mathbb{Y}$ cannot contribute during rendering and leads to wasted computation and storage (i.e. Problem 1).

To avoid this problem, we place virtual lights only on points $\mathbf{y} \in \mathbb{Y}$ by randomly sampling $\mathbb{Y}$. We achieve this by generating M random camera sub-paths, each ending at a point $\mathbf{x}$, and then randomly picking a reflection direction at $\mathbf{x}$ and tracing a ray to find a point $\mathbf{y} \in \mathbb{Y}$. To improve the sample distribution, we use BSDF importance sampling at $\mathbf{x}$.

Note that this process does not sample $\mathbb{Y}$ with a uniform probability. Instead, points $\mathbf{y} \in \mathbb{Y}$ that are visible from more points $\mathbf{x} \in \mathbb{X}$ are chosen with a higher probability, including the geometry term between $\mathbf{x}$ and $\mathbf{y}$ and the BSDF at $\mathbf{x}$. This leads to a sampling density correlated with the importance brought to the camera.

Nonetheless, the exact probability used for selecting a point $\mathbf{y} \in \mathbb{Y}$ is not needed. This is because these samples merely represent the selected virtual light positions, but not their emissions. Our emission profile generation approach with photon splitting (Section 3.3) avoids the complexity of estimating the probability of picking each virtual light sample $\mathbf{y}$.

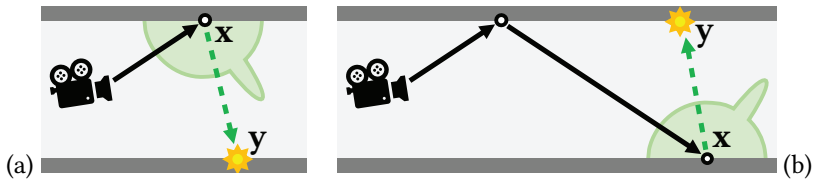
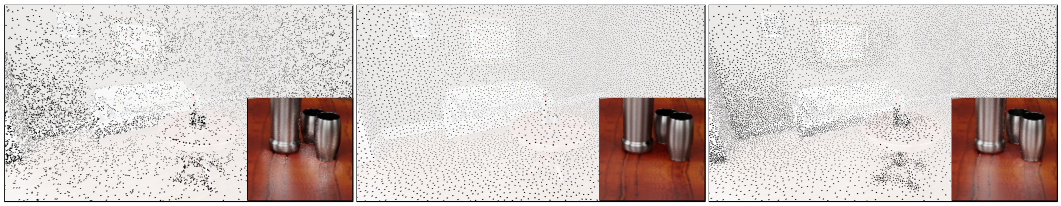


Fig. 6. *Virtual lights can be placed (a) at the secondary hit after the primary hit or (b) at the next hit after a secondary hit for specular reflections on highly-specular materials.*

The camera path generation process used for generating the initial virtual light placement should match the camera path generation used during rendering. For taking full advantage of virtual lights, it is advisable to keep the camera paths short. The shortest camera path would only contain the primary hit. Then, a virtual light can be placed at the secondary hit by picking a random reflection direction at the primary hit via BSDF sampling (as in Figure 6a). For highly-specular materials, however, the indirect illumination from these virtual lights may not provide sufficient resolution. Therefore, when BSDF sampling picks specular reflection on a highly-specular material, we extend the camera path to the secondary hit point, and a virtual light is placed at the next hit point after this secondary hit (as in Figure 6b). When the secondary hit is also on a highly-specular material,



(a) *No Sample Elimination* **(b)** *Uniform Sample Elimination* **(c)** *Adaptive Sample Elimination*

Fig. 7. *Virtual light distribution in the Fireplace scene (a) without sample elimination, (b) with uniform sample elimination, and (c) with adaptive sample elimination. The $r_{\max}$ value for uniform sample elimination is hand tuned in this test. Notice that adaptive sample elimination preserves the higher virtual light density on the objects on the table and, therefore, properly resolves their specular reflections on the table.*

we can extend the camera path further. In general, wherever we decide to end the camera path, we trace one more reflection ray and place a virtual light at that next hit point.

3.2 Virtual Light Placement with Sample Elimination

Our goal is to generate virtual light positions that form a blue noise distribution, which can substantially improve the quality of the results given the same number of lights, particularly with specular transport (Figure 3c). This addresses Problem 3.

We develop a variant of the sample elimination method [Yuksel 2015] to achieve a blue noise distribution. Given M randomly generated virtual light positions (as explained above), sample elimination selects a subset of N with $N < M$ that exhibits blue noise characteristics.

Sample elimination uses weights computed using the maximum possible Poisson disk radius $r_{\max}$ that can be achieved with N samples within the sampling domain. Assuming perfect packing on a 2D surface, we can write

$$r_{\max} = \sqrt{\frac{A}{2\sqrt{3}N}} \,. \tag{1}$$

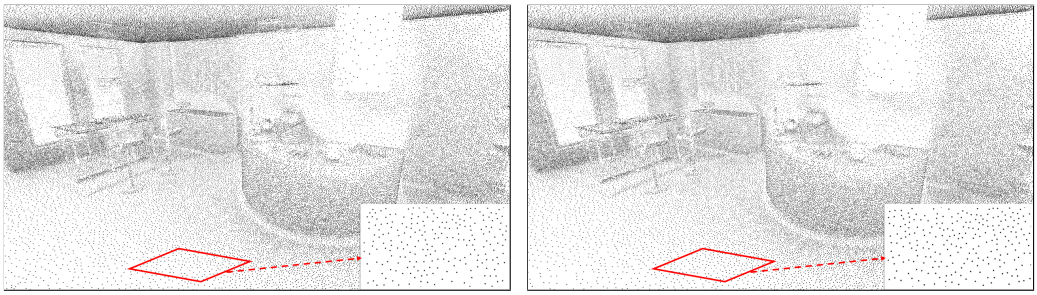
where A is the total area of the surface. Unfortunately, we cannot directly apply this formula to our sampling problem, since we do not know the total area of $\mathbb{Y}$, the scene points where we can place virtual lights. Also, a fixed $r_{\max}$ value would lead to uniform distribution (i.e. *uniform sample elimination*), which would overwrite the desired sample density we achieve during initial sample generation described in Section 3.1.

We resolve these issues with our *adaptive sample elimination* by separately computing the maximum radius value for each initial virtual light position sample, based on the local density, prior to sample elimination. More specifically, we consider a local sample elimination problem where out of m closest samples to a sample ($m \ll M$), we target keeping n of them, such that $n/m = N/M$. Let d_i be the distance to the m^{th} closest sample to sample i. We can write the maximum radius for sample i as

$$r_{i,\max} = \sqrt{\frac{A_i}{2\sqrt{3}n}} = \sqrt{\frac{M\,A_i}{2\sqrt{3}N\,m}} \,, \tag{2}$$

where A_i is the estimated surface area around sample i within d_i distance. Assuming that all m samples around sample i are on a flat surface, we can estimate the area as $A_i = \pi\, d_i^2$.

When using VSLs, we assign their radii as $r_i = \alpha r_{i,\max}$, where α is a user-defined parameter controlling the overlap of neighboring VSLs (we use $\alpha = 2$ in our tests).



(a) *Sequential Sample Elimination (26.3 s)* **(b)** *Parallel Sample Elimination (2.7 s)*

Fig. 8. *Sequential Sample Elimination vs Parallel Sample Elimination(1.3M samples eliminated to 0.13M samples). Parallel Sample Elimination is noticeably faster than Sequential Sample Elimination without any noticeable distribution degeneration.*

Figure 7 shows virtual light distributions in an example scene without sample elimination, with uniform sample elimination, and with our adaptive sample elimination. Notice that without sample elimination the distribution includes many virtual lights that are closely placed together (i.e. oversampling) and insufficient density elsewhere. Uniform sample elimination produces a blue noise distribution, but loses the density variation of the original distribution. This results in loss of detail for some parts of the image, particularly with specular transport. Our adaptive sample elimination preserves the original density variations and achieves a blue noise distribution that prevents placing virtual lights too close to each other.

To parallelize sample elimination, we split the sampling domain into a number of boxes and perform sample elimination independently within each box in parallel. Since our sampling domain $\mathbb{Y}$ is complex and our initial sample distribution is non-uniform, simply splitting the bounding box of the initial samples into boxes of equal size leads to a highly non-uniform work distribution. For better load balancing, we recursively split the bounding box into two boxes of (approximately) equal number of samples along the axis that provides the most spatially even split. This process is repeated until we reach a desired number of boxes, forming a (balanced) k-d tree.

Performing sample elimination independently for each box (i.e. leaf node) does not eliminate samples near the boundaries of boxes that are too close to the samples of adjacent boxes. To avoid this, we terminate sample elimination before we reach the target sample count in each box, leaving some excessive samples. In our implementation this is controlled by a parameter determining what percentage of excessive samples are eliminated before we terminate the process. We empirically find that setting this parameter to 80% consistently gives us similar quality as sequential sample elimination across all test scenes (Figure 8, Figure 12). Then, we continue sample elimination using the parent nodes of the leaf nodes. We repeat this process until we reach the root node and eliminate all of the remaining excessive samples in a single thread.

Each time we restart sample elimination using a parent node's box, we must recompute the sample elimination weights for each sample in the box. To minimize this overhead, our implementation skips a few levels by joining 8 nodes under a common parent node. As can be seen in Figure 8, our parallel sample elimination process produces a similar result as (though different than) the serial implementation.

3.3 Computing Virtual Light Emission Profiles

The virtual lights positions generated from the camera subpaths do not carry any illumination information. To compute their illumination profiles, we must estimate the incident radiance field at each virtual light and use it to compute the reflected radiance field.

We estimate the incident light via *photon tracing* by generating random light paths. At each light path vertex, we convert the photon's energy to a reflected radiance field and distribute it to the nearby virtual lights, if any. We call this process *photon splitting*. Let w_i be the portion of the radiance field added to the virtual light i. We must have a partition of unity (i.e. $\sum_i w_i = 1$) to preserve energy or, if there are no nearby virtual lights, $w_i = 0$ for all virtual lights. A virtual light i is considered nearby, if the distance to its center d_i is less than its radius r_i. We use $1 - (d_i/r_i)^2$ as relative weights to compute w_i.

This process involves tracing a large number of photons. Fortunately, it does not require storing any photons. Since photon tracing (without photon storage) is relatively cheap, we can efficiently use a large number of photons. Obtaining the nearby virtual lights at any photon hit point can also be performed efficiently, using a spatial partitioning structure for the virtual lights.

The most expensive component of this computation is evaluating the outgoing radiance, as it involves all outgoing directions and performed for each photon hit. To accelerate this step, we delay the outgoing radiance field computation. First, we complete tracing all photons and store an incident radiance field per virtual light. Then, we convert the resulting incident radiance field into an outgoing radiance field that represents the emission profile of the virtual light. This later conversion can be performed in a lazy fashion, as needed, during rendering, though in our implementation we complete the conversion for all lights in a separate pass, as the final step of virtual light preparation.

4 RENDERING WITH VIRTUAL BLUE NOISE LIGHTING

Our virtual lights can be used with any existing light sampling technique. However, the fact that they provide a relatively uniform coverage of the indirectly visible surfaces $\mathbb{Y}$ opens up possibilities for alternative light sampling and virtual light evaluation techniques for improving the lighting estimation quality. More specifically, we propose treating virtual lights as *photon lights* with non-uniform emission profiles and combining three light sampling strategies via multiple importance sampling (MIS): *power-based light sampling*, *BSDF sampling*, and *adaptive camera path extension*.

A photon light is a VSL that does not include the simplifications of the VSL formulation [Hašan et al. 2009]. VSLs approximate photon lights by treating them as point sources for visibility purposes and an extra cosine factor to account for possible rays that intersect with the light's sphere but not the surface on which the light is placed. Instead, we use the photon light formulation by adding a non-uniform emission function I^e_j, such that the outgoing illumination at a point $\mathbf{y}$ towards a direction $\boldsymbol{\omega}$ from a virtual light j placed at $\mathbf{y}_j$ with radius r_j can be written as

$$L^e_j(\mathbf{y}, \boldsymbol{\omega}) = I^e_j(\boldsymbol{\omega})(\|\mathbf{y} - \mathbf{y}_j\| \leq r_j) \ . \tag{3}$$

4.1 Virtual Light Sampling

Because we are generating a large number of virtual lights, we must efficiently sample them during rendering. Power-based importance sampling [Shirley et al. 1996] is a standard technique for handling many lights. More recently, a number of more effective sampling techniques have been proposed, such as light trees [Lin and Yuksel 2020; Moreau et al. 2019; Yuksel 2019] and *ReSTIR* [Bitterli et al. 2020]. These methods are shown to outperform power-based light sampling. However, when combined with our BSDF sampling and adaptive camera path extension, in our test we have found that the simple power-based importance sampling strategy provides sufficient sampling

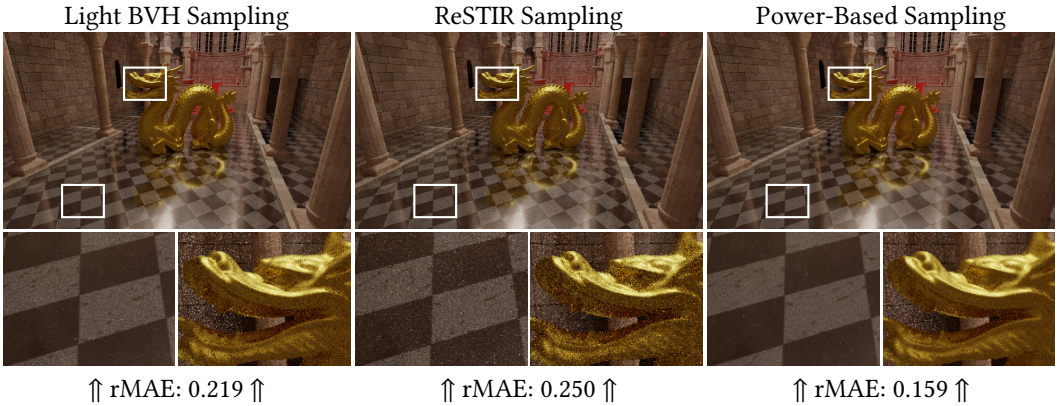


Fig. 9. *Comparison of different light sampling techniques combined with our BSDF sampling via MIS. All results rendered in 200 ms. Because of the runtime overhead of Light BVH [Moreau et al. 2019] and ReSTIR [Bitterli et al. 2020], power-based sampling outperforms them, when combined with our BSDF sampling. (140K Virtual Lights, 20M light Paths)*

quality without the additional cost of these more advanced techniques (see Figure 9 for an example). Nonetheless, we do not dismiss the possibility that in some scenes these more advanced techniques might offer a practical advantage, in which case our power-based importance sampling can be safely replaced with any of them.

While estimating the illumination at a shaded point $\mathbf{x}$, our power-based sampling strategy picks a photon light with probability proportional to its emissive power P_j. Then, we pick a random direction $\boldsymbol{\omega}_j$ towards the light within the solid angle that contains the light's sphere with uniform probability. Thus, the resulting probability of picking a light sample can be written as

$$p_j(\boldsymbol{\omega}_j) = \frac{P_j}{\sum_k P_k} \left(\frac{1}{2\pi(1-\cos\theta_j)} \right) , \tag{4}$$

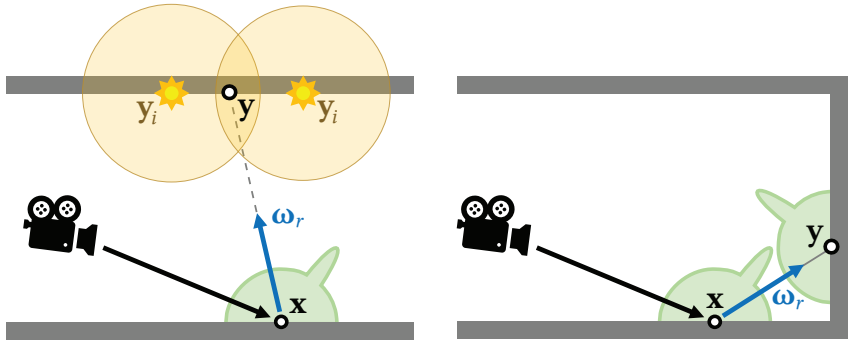
where θ_j is the half-angle of the cone that contains the light.

We compute visibility (i.e. shadows) by tracing a ray from the shaded point $\mathbf{x}$ along $\boldsymbol{\omega}_j$. Unless the hit point $\mathbf{y}$ of this ray is within the light's sphere, we consider the light occluded (as in Equation 3).

We combine our power-based importance sampling with BSDF sampling, which is a highly-effective strategy, particularly with relatively specular materials. This strategy is enabled as a direct outcome of our virtual light placement technique that provides a relatively uniform coverage of $\mathbb{Y}$. For BSDF sampling, we generate a random reflection direction $\boldsymbol{\omega}_r$ with probability $p_{\text{BSDF}}(\boldsymbol{\omega}_r)$ and trace it to find the corresponding hit point $\mathbf{y}_r \in \mathbb{Y}$ in the scene. Then, we gather all virtual lights that contain $\mathbf{y}_r$ within their radii, as shown in Figure 10a. We use these nearby virtual lights to estimate the outgoing light at $\mathbf{y}_r$ using Equation 3 (i.e. $L_i^e(\mathbf{y}_r, -\boldsymbol{\omega}_r) = I_i^e(-\boldsymbol{\omega}_r)$ for light i).

We combine our BSDF sampling with power-based light sampling using MIS. We compute an MIS weights w_i for each light i near $\mathbf{y}_r$ and w_j for the power-based light sample j using power heuristic

$$w_i = \frac{p_{\text{BSDF}}^2(\boldsymbol{\omega}_r)}{p_{\text{BSDF}}^2(\boldsymbol{\omega}_r) + p_i^2(\boldsymbol{\omega}_r)} \qquad \text{and} \qquad w_j = \frac{p_j^2(\boldsymbol{\omega}_j)}{p_{\text{BSDF}}^2(\boldsymbol{\omega}_j) + p_j^2(\boldsymbol{\omega}_j)} . \tag{5}$$



(a) *BSDF Sampling* **(b)** *Camera path extension*

Fig. 10. *Illumination estimation with our VSLs. (a) A random direction ω_r is generated by BSDF-based or power-based importance sampling and the nearby VSLs to the hit point* $\mathbf{y}$ *are sampled. (b) When* $\mathbf{y}$ *is too close to* $\mathbf{x}$*, we extend the camera path and shade* $\mathbf{y}$ *first to find the reflected light towards* $\mathbf{x}$ *along* $-\omega_r$*.*

4.2 Adaptive Camera Path Extension

Virtual lights can provide a good estimate for distant illumination, but they often fall short when it comes to properly estimating indirect illumination from nearby surfaces. There are two fundamental reasons behind this shortcoming. The first one is the virtual light formulation. Indeed, VPLs are notorious with geometry terms going to infinity, as the distance between the shaded point and the virtual light goes to zero. VSLs improve this behavior and using photon lights with non-uniform emission profiles computed using our method provide further improvement. The other reason is virtual light density. Even though our virtual light placement approach significantly improves the distribution, density can still be inadequate for handling nearby illumination, even when using a large number of virtual lights.

Our solution is to avoid using virtual lights for nearby illumination and extending the camera path, as needed, for handling nearby indirect illumination, similar to path tracing. Fortunately, our virtual light sampling technique described above provides an efficient mechanism for adaptively extending the camera path with minimal additional cost, separating our solution from similar ideas in prior work [Kollig and Keller 2006; Walter et al. 2012].

We can write the incoming illumination $L_i(\mathbf{x}, \boldsymbol{\omega}_r)$ to the shaded point $\mathbf{x}$ from a direction $\boldsymbol{\omega}_r$ as a weighted combination of illumination estimation from virtual lights L_v and illumination sampled via camera path extension L_p, using

$$L_i(\mathbf{x}, \boldsymbol{\omega}_r) = w_\mathrm{v} L_\mathrm{v}(\mathbf{y}, -\boldsymbol{\omega}_r) + w_\mathrm{p} L_\mathrm{p}(\mathbf{y}, -\boldsymbol{\omega}_r) \,, \tag{6}$$

where $w_\mathrm{v} + w_\mathrm{p} = 1$ are the blending weights and $\mathbf{y}$ is the hit point of the secondary ray from $\mathbf{x}$ along $\boldsymbol{\omega}_r$. We adjust the blending weights such that virtual lights are only used for distant illumination.

For determining the distance beyond which we can rely on virtual lights, we consider the material at $\mathbf{x}$ and the virtual light density at $\mathbf{y}$. This is because we can safely use relatively nearby virtual lights when $\mathbf{x}$ is on a relatively diffuse surface and when there is sufficient virtual light density at $\mathbf{y}$. More specifically, we use the estimated sample footprint [Bekaert et al. 2003; Müller et al. 2021]

$$a(\mathbf{x}, \boldsymbol{\omega}_r) = \frac{\|\mathbf{x} - \mathbf{y}\|^2}{p_\mathrm{BSDF}(\boldsymbol{\omega}_r) |\cos \theta_r|} \,, \tag{7}$$

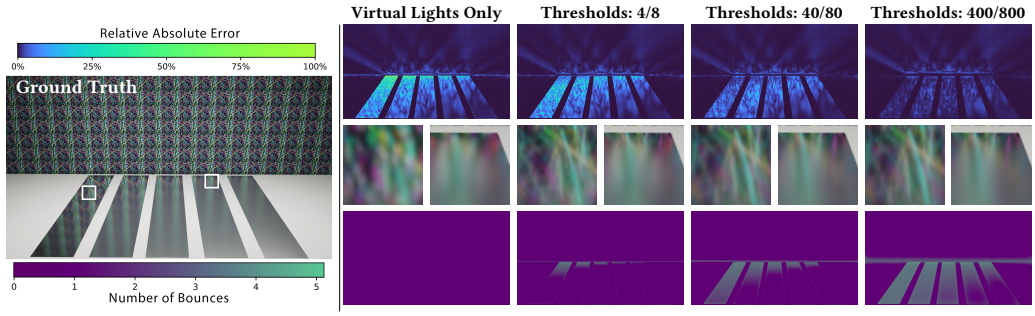


Fig. 11. *Our results without camera extension or with camera path extension but using different $n_{\min}$ and $n_{\max}$. Top row: Relative squared errors compared to the path tracing reference. Middle row: results of our method. Bottom row: visualization of the average path depth. Notice our method is capable of handling materials with different roughness adaptively. For rougher materials, our method is able to extend the path less frequently. On the bottom row, the transition zones on different metal plates indicate an interpolation of path extension and termination.(125K Virtual Lights, 5M Light Paths)*

where θ_r is the angle between $-\boldsymbol{\omega}_r$ and the surface normal at $\mathbf{y}$. Multiplying $a(\mathbf{x}, \boldsymbol{\omega}_r)$ with the estimated virtual light density $\bar{\rho}(\mathbf{y})$ at $\mathbf{y}$ gives the estimated number of virtual lights that correspond to the sampled specular lobe. We compute $\bar{\rho}(\mathbf{y})$ by averaging the density estimation of each virtual light ρ_i, derived from Equation 2, such that

$$\bar{\rho}(\mathbf{y}) = \frac{1}{n} \sum_{i}^{n} \rho_i \qquad \text{and} \qquad \rho_i = \frac{\alpha^2}{2\sqrt{3}\, r_i^2}\,, \tag{8}$$

where n is the number of virtual lights that overlap with $\mathbf{y}$. Using this, we define the blending weight as

$$w_{\mathrm{v}} = \mathrm{clamp}(\tilde{w}_{\mathrm{v}}, 0, 1) \qquad \text{with} \qquad \tilde{w}_{\mathrm{v}} = \frac{a(\mathbf{x}, \boldsymbol{\omega}_r)\, \bar{\rho}(\mathbf{y}) - n_{\min}}{n_{\max} - n_{\min}}\,, \tag{9}$$

where $n_{\min}$ and $n_{\max}$ are user-defined parameters, controlling the minimum and maximum number of virtual lights that would be sufficient for relying on the virtual light estimation.

Our virtual light sampling method allows implementing camera path extension with minimal additional cost. For each virtual light sample we generate via either power-based sampling or BSDF sampling, we compute the corresponding blending weight w_{v}. With power-based sampling, we estimate the density using only the selected light. When $w_{\mathrm{v}} < 1$, we simply use the hit point $\mathbf{y}$ as the next vertex of the camera path (see Figure 10b). Note that when $w_{\mathrm{v}} = 0$, we do not need to evaluate the virtual lights at all.

Our virtual light placement approach provides an almost complete coverage of $\mathbb{Y}$, but a complete coverage is not guaranteed. In some rare cases, BSDF sampling can hit a point $\mathbf{y}$ that does not overlap with any virtual lights. When this happens, our camera path extension is automatically triggered (with $w_{\mathrm{v}} = 0$).

Our camera path extension not only improves the accuracy of the lighting estimation, but also prevents the energy loss that is common with virtual light formulations near corners. We show a detailed study about threshold parameter's effects in Figure 11. From our experiments, we found $n_{\min} = 40$ and $n_{\max} = 80$ works well in most cases, because these settings provide a good balance between performance and rendering quality.

5 IMPLEMENTATION AND RESULTS

We have implemented our virtual blue noise lighting approach in a GPU ray tracer, using NVIDIA GameWorks's framework Falcor [Benty et al. 2020]. All methods we compare against are also implemented within the same rendering framework, using the same functions and data structures when possible. All results are rendered with unlimited light bounces via Russian-Roulette. The performance results are obtained on a computer with an AMD 3900X CPU with 3.8 GHz/12 Cores and 32 GB RAM, and an NVIDIA RTX 3090 GPU.

For the initial virtual light generation, we split the screen space into tiles and generate an equal number of random samples within each tile. For computing the emission profiles, we build a BVH that contains all virtual lights. The nearest neighbour queries are handled using the GPU ray tracing functionalities [Evangelou et al. 2021]. We use 16×16 directional entries per virtual light (unless otherwise specified) to store the incoming radianceusing octahedron environment mapping [Engelhardt and Dachsbacher 2008]. These are then converted to the final emission profiles of the same resolution. For virtual lights on diffuse (i.e. Lambertian) surfaces, we do not explicitly store directional emission profiles, since emission towards any direction can be easily computed from a single light intensity value.

For accelerating power-based virtual light sampling, after we pick a random direction $\boldsymbol{\omega}_j$ towards light j, before we trace a ray in that direction to determine visibility, we first intersect the ray with the disk that is centered at the light and aligned with the surface normal at the light's position. If the ray does not intersect the light's disk, we assume that the light is not visible along the direction $\boldsymbol{\omega}_j$. While this test can produce false negatives that results in a small amount of error, it also provides an average of 12% reduction of render time in our tests.

For our camera path extension, we use $n_{\text{min}} = 40$ and $n_{\text{max}} = 80$ in all examples (see Figure 11).

We use relative mean absolute error (rMAE) as our error metric, measuring the error relative to the reference pixel intensity. Let I and I_{ref} represent the intensities of a rendered pixel and the corresponding reference image pixel. rMAE is defined as the mean pixel error, which is calculated using $|I - I_{\text{ref}}|/(I_{\text{ref}} + \epsilon)$, where $\epsilon = 0.01 \cdot \text{mean}(I_{\text{ref}})$ is a scene-adaptive small bias value to avoid division by zero.

5.1 Comparisons to Rich-VSLs

We begin with directly comparing our VBNL method with Rich-VPLs [Simon et al. 2015], as it is the state-of-the-art for indirect illumination estimation with virtual lights. Our implementation of Rich-VSLs use the same data structures for storing VSLs and their emission profiles. The differences are in VSLs generation, distribution, emission profile computation, and VSL sampling during rendering. Unlike our method, Rich-VSLs are generated from scene lights using photon tracing, selected based on an importance map generated from the camera. The Rich-VSL distribution is computed via an iterative relaxation algorithm and the emission profiles are computed with irradiance estimation from a photon map. In our tests with Rich-VSLs, we cast 8 importons per pixel, generate a photon map on the GPU that contains up to 100× photons than VPLs, and perform 20 relaxation iterations. We use Falcor's Light BVH sampling with Rich-VSLs, unless stated otherwise.

Figure 12 shows comparisons of our method to Rich-VSLs, using images rendered with the same number of VSLs. In these comparisons we allow Rich-VSLs to use our entire virtual light initialization time (i.e. for virtual light preparation) plus render time just for rendering. We also allow additional initialization time for Rich-VSLs, which is significantly longer than our initialization time. Visualizations of VSL distributions used in these images are presented in Figure 13. Notice that, even though most scenes do not include particularly challenging illumination conditions, our

VSLs generated from the camera clearly improve the indirect illumination estimation, producing closer results to the path tracing reference.

In the Kitchen scene (Figure 12 top row) and the Sibenik scene (Figure 12 second row), Rich-VSLs struggle with resolving specular reflections and produce an overall darker scene. This darkening can be remedied by generating a lot more VSLs. In the Modern Room Scene (Figure 12 third row), Rich-VSLs still have low convergence due to the inefficiencies of the virtual light sampling method. In the Veach Door scene, Rich-VSLs suffer from a sparse virtual light distribution (Figure 13) inside the room. In this scene, we use 10 million photons generated from 1.5 million light paths for Rich-VSLs, but only a small percentage of photons enter the room and the others remain outside. This severely limits the possible virtual light positions for Rich-VSLs inside the room, resulting in undersampled illumination. Instead, most virtual lights generated by our method are well distributed inside the rendered room.

One important advantage of our method over Rich-VSLs is the VSL placement/distribution (Figure 13), playing a significant role in the resulting differences (Figure 12). Note that our method automatically places VSLs with a higher density where prominent specular reflections appear. In particular, notice the high-density VSL distribution with our method in the Sibenik scene on parts of the pillars, which are indirectly visible on the specular reflections on the floor. Similar increase in VSL density can be observed in all scenes that are indirectly visible via specular reflections. Such adaptive density variations are needed for properly resolving specular transport. Therefore, using our method specular reflections of the objects are properly resolved and corner darkening on the floor near the walls are avoided, producing closer results to path tracing in all scenes.

Another source of improvement is the number of photons used for the emission profile computation of our method. Since our method does not require storing any photons, we could use significantly more photon paths than the Rich-VSL examples in Figure 12 in much shorter virtual light initialization time in Table 1. In Figure 12, Rich-VSLs use 1.5 million photon paths for all scenes, while our method uses 20 million photon paths for the first three scenes and 200 million for the Veach Door scene. As can be seen in Figure 14, using a small number of photons in the Veach Door scene does not provide sufficient convergence for the virtual light emission profiles, resulting in visible fluctuations in the illumination estimation. Our method can efficiently use a massive number of light paths to handle such challenging illumination cases.

In comparison to Rich-VSLs, our method also consumes less memory and provides faster initialization. Detailed breakdowns of computation times and storage cost for both methods are provided in Table 1. Notice that the emission computation of Rich-VSLs using photon density estimation consumes most of the initialization time in most scenes. In comparison, our emission computation can be performed significantly faster. In terms of memory cost, our method does not need to store a photon map or an importon map. This results in storage savings with our method, even though both methods have identical storage per VSL.

Our method also benefits from our adaptive path extension strategy. A visualization of the average path lengths with our method is shown in Figure 15. In the supplemental document, we show a study of applying our light sampling method to Rich-VSLs. Although the rendering quality of Rich-VSLs can be improved using our light sampling, the resulting images still contain excessive noise and sampling artifacts as compared to our method due to the differences in virtual light distribution and emission profile generation.

5.2 Virtual Light Placement

Generating virtual light positions starting from the camera is an important component of our method. Figure 2 shows comparisons of different methods for placing VSLs.

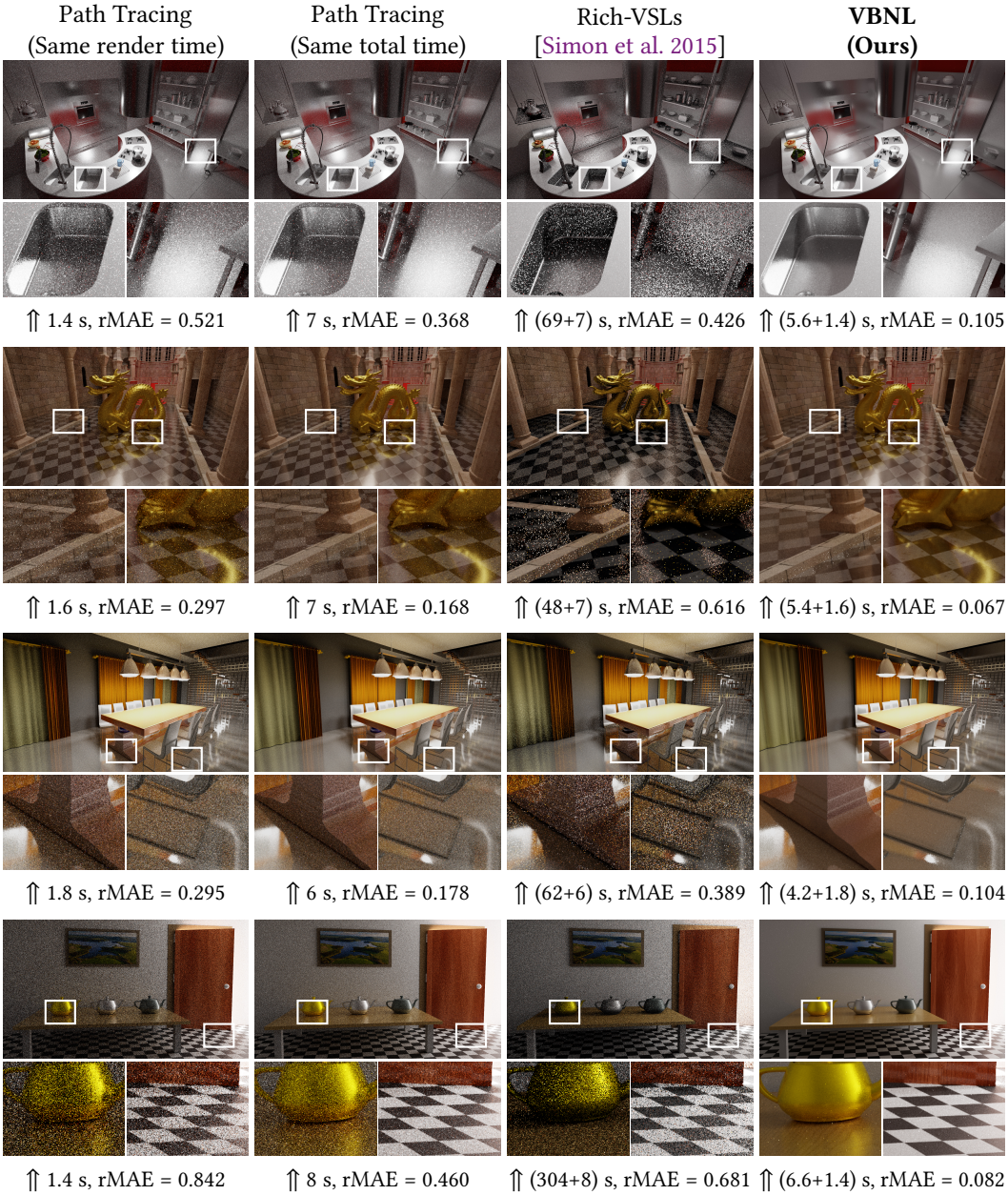


Fig. 12. *Comparison of our virtual blue noise lighting method to Rich-VSLs in the Kitchen, Sibenik, ModernRoom, and Veach Door scenes. Notice that, unlike our VBNL method, Rich-VSLs miss various important illumination details, mainly because of virtual light placement/distribution. Both methods use 140K VSLs in all scenes.For ours and Rich-VSLs, the total time is given as a sum of preparation time plus render time. Note that in all scenes the render time we use for Rich-VSLs is equal to the sum of our reparation and render time. In addition, Rich-VSLs in these examples are given much longer (7× to 38×) extra time for preparation.*

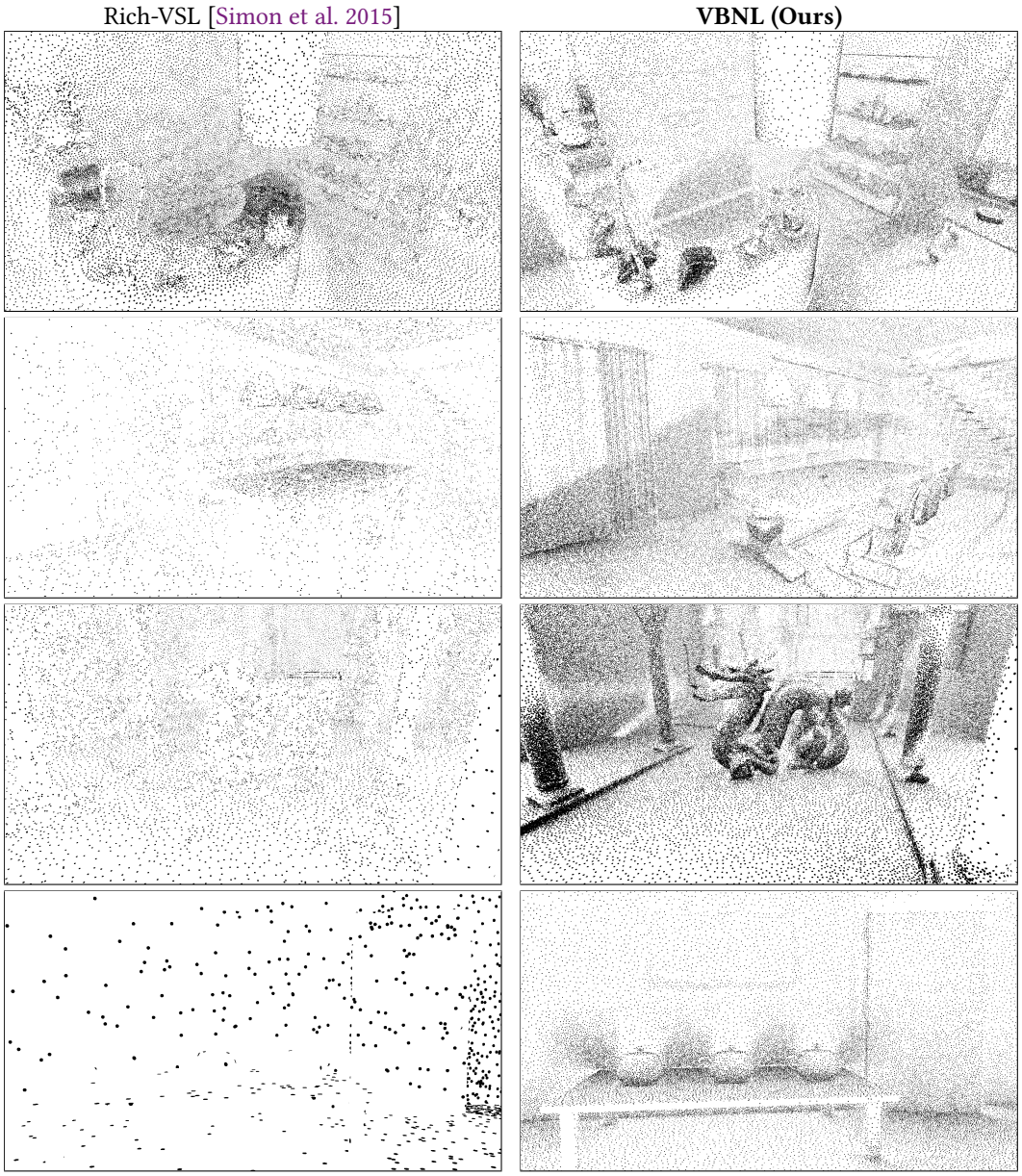
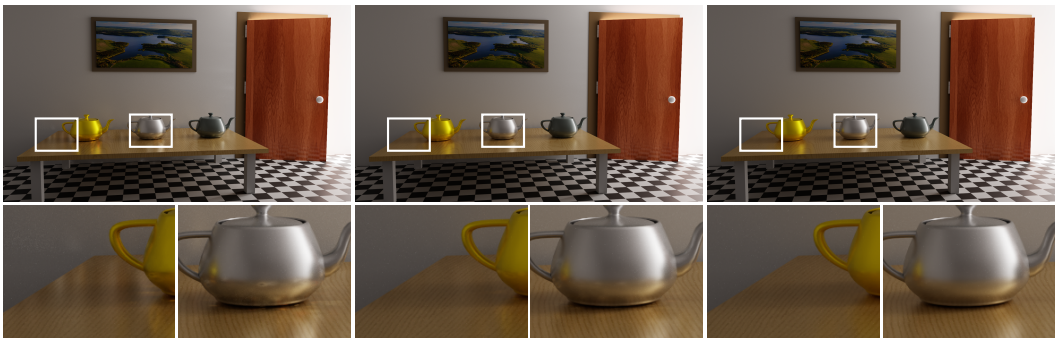


Fig. 13. *VSL distributions of Rich-VSLs [Simon et al. 2015] and our VBNL method for the images in Figure 12. Notice that our method provides a more uniform blue noise distribution and better adjusts the VSL density to camera importance.*

Notice that the typical method of using light paths (Figure 2a) can waste virtual lights by placing them where they have no contribution to the rendered image (Problem 1) and oversample high-probability light paths (Problem 2). As a result, the indirect illumination estimation in the rendered



10,000,000 paths
Photon Tracing Time: 0.70s

100,000,000 paths
Photon Tracing Time: 1.02s

1,000,000,000 paths
Photon Tracing Time: 4.02s

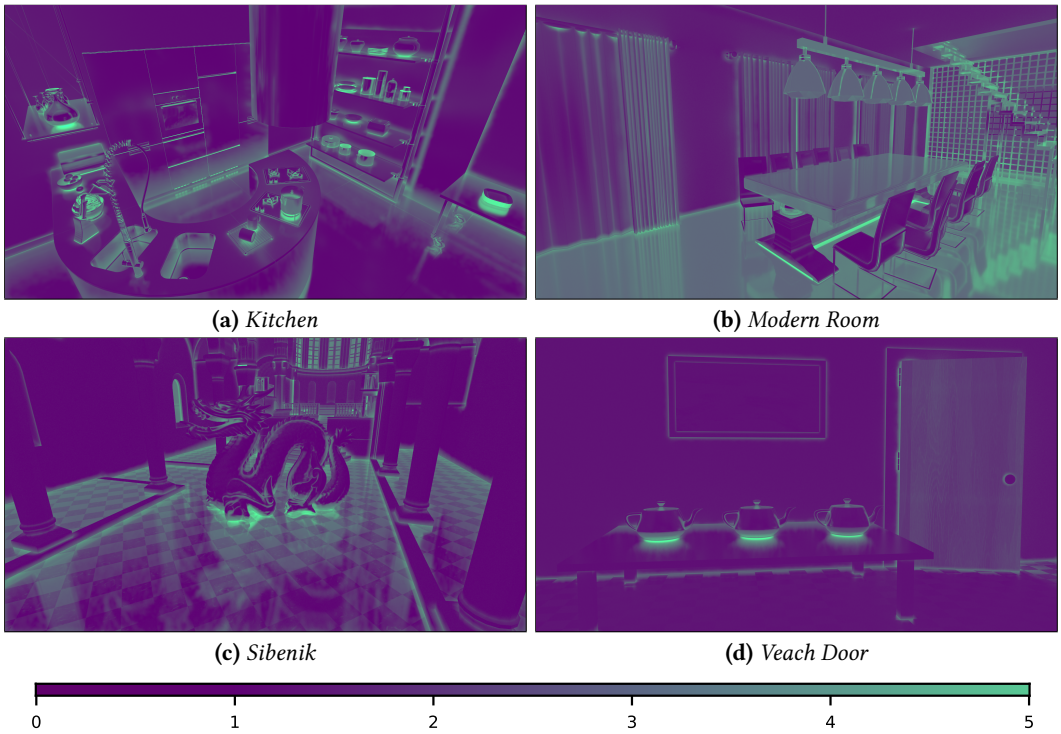
Fig. 14. *Results rendering with different numbers of photon paths used for computing the emission profiles of virtual lights in the Veach Door scene (using $n_{\min} = 40$ and $n_{\max} = 80$). Due to complex light transport of this scene, most light paths are terminated in advance. The emission profiles of virtual lights still have high variance even after 10M light paths are traced. As the number of the light paths grows, the illumination from virtual lights converges. All images are rendered using 140K virtual lights in this example.*

Table 1. *Performance breakdown of Rich-VSLs and our VBNL for the results in Figure 12.*

		Kitchen	*Modern Room*	*Sibenik*	*Veach Door*
					
Rich-VSLs	VSL Placement	8.46 sec	4.95 sec	5.05 sec	225.76 sec
	Relaxation	9.99 sec	5.36 sec	6.75 sec	6.61 sec
	Emission Comp.	50.10 sec	51.40 sec	35.74 sec	71.70 sec
	Shading	7.00 sec	6.00 sec	7.00 sec	8.00 sec
	Total Time	**75.55 sec**	**67.71 sec**	**54.54 sec**	**312.07 sec**
	VSL Storage	333.31 MB	349.16 MB	103.43 MB	568.45 MB
	Importon Storage*	427.38 MB	374.48 MB	215.00 MB	206.96 MB
	Photon Storage*	361.76 MB	67.65 MB	401.86 MB	341.11 MB
Ours	VSL Placement	1.19 sec	1.09 sec	1.19 sec	1.05 sec
	Sample Elim.	2.55 sec	2.00 sec	2.46 sec	2.28 sec
	Photon Tracing.	1.42 sec	0.76 sec	1.54 sec	1.30 sec
	Convert.	0.42 sec	0.37 sec	0.26 sec	2.02 sec
	Shading	1.42 sec	1.78 sec	1.55 sec	1.35 sec
	Total Time	**7.00 sec**	**6.00 sec**	**7.00 sec**	**8.00 sec**
	VSL Storage	298.21 MB	234.61 MB	145.85 MB	548.56 MB
	Initial Candidate Storage*	192.31 MB	168.50 MB	193.52 MB	186.31 MB

* Used during initialization and can be freed prior to rendering.

image lacks smaller-scale details, particularly on glossy surfaces, due to undersampling, as only a relatively smaller percentage of virtual lights are actually used during rendering.



(a) *Kitchen* **(b)** *Modern Room*

(c) *Sibenik* **(d)** *Veach Door*

Fig. 15. *Visualizations of average path length in above scenes. For Kitchen and Sibenik, our method is able to terminate most of camera paths early. For Modern Room, because the floor is highly specular, we extend the camera path more on it with high probability. Zero means no path extension.*

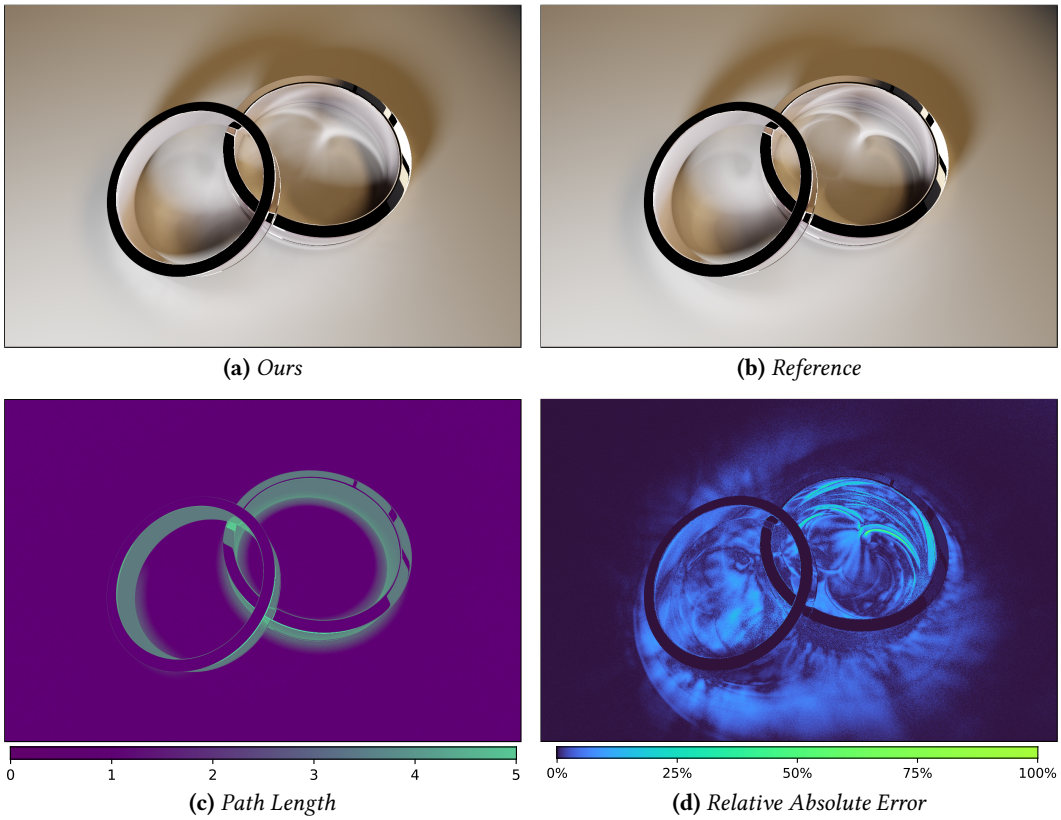
Using an importon map [Simon et al. 2015] to guide the virtual light placement using light paths can help (Figure 2b), but it can still oversample high-probability light paths (Problem 2), as can bee seen in the Veach Door scene (Figure 13 bottom-left). If light paths going through hot zones (Problem 2) with high photon density rarely directly contribute to illumination of the final image, it will lead to an overall darker result as the Veach Door result of Rich-VSLs shows in Figure 12 bottom.

Our virtual light placement using camera paths (Figure 2c) resolves both of these problems, producing a desirable coverage of the indirectly-visible scene surface (i.e. $\mathbb{Y}$).

5.3 Emission Profiles

Storing an emission profile per virtual light [Simon et al. 2015] can significantly improve the quality. Our supplemental document includes a comparison of our method to regular VPLs and VSLs generated using the same number of photon paths, showing that our method improves both the rendering quality and performance.

Our delayed computation of the emission profiles (by first accumulating the directional incoming radiance data per light and then converting it to emission profiles, as explained in Section 3.3) results in some minor error due to quantization of the directional information, but it significantly improves the computation time. An example showing two orders of magnitude speed difference is included in our supplemental document.



(a) *Ours* **(b)** *Reference*

(c) *Path Length* **(d)** *Relative Absolute Error*

Fig. 16. *Our method can handle high frequency caustics with some blurring: (a) our method with 53k* 32×32 *virtual lights placed on these metal rings (250M Light Paths), (b) reference results, (c) average path length during shading, and (d) relative absolute error (rMAE = 0.0126).*

In our implementation, we store all emission profiles of all virtual lights in a single texture, where each 16×16 pixel block belongs to a virtual light. To implement highly specular reflection effects, it might be preferred to useemission profiles with higher resolution. For example, we show our method is capable of handling detailed caustics to some extent using 32×32 blocks for emission profiles as Figure 16 shows.

During photon splitting, this texture stores the incoming radiance. Then, we convert it to the outgoing radiance. Note that not all emission values stored in this texture are used during rendering, because not all virtual lights are sampled from all directions. Therefore, a lazy conversion that computes parts of the emission profiles as needed can provide further optimization. Nonetheless, in our tests we have found that 70% of the data in our texture are accessed during rendering. Therefore, any savings with lazy initialization might be limited, especially considering potential overhead it might incur.

5.4 VSL Sampling

Figure 17 shows results rendered by different configuration of our sampling method in 2 seconds. We can clearly notice the individual effect of a single technique and their composition. BSDF sampling significantly improves the overall sampling quality. Multiple Importance sampling compensates

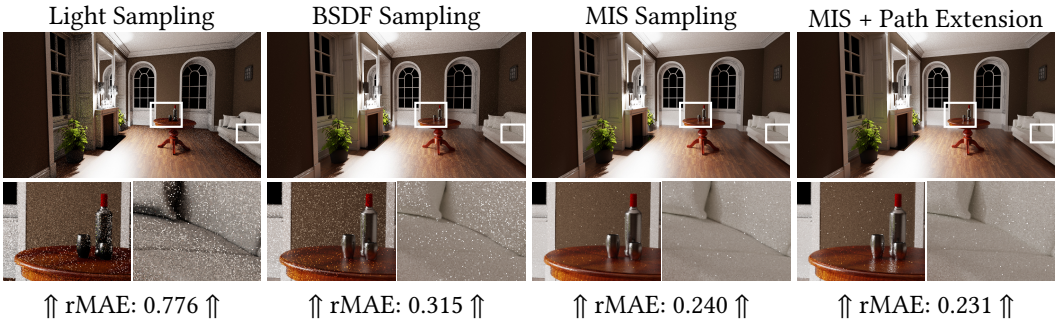


Fig. 17. *Ablation of our sampling method. All images are rendered in 2s. We can find that BSDF sampling can significantly improve the efficiency of virtual light sampling. In addition, we can find that the path extension strategy can improve the corner quality a lot. All images are rendered using 140K virtual lights and 50M light paths.*

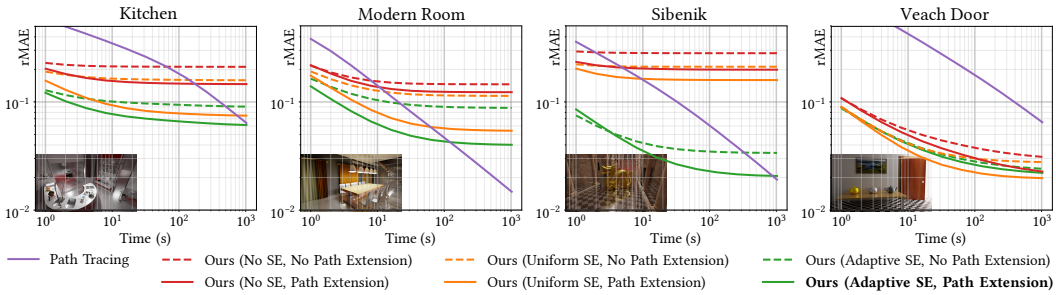


Fig. 18. *Log-log plots of render time vs. rMAE, showing the convergence of path tracing and variants of our method with different virtual light distribution (no sample elimination, uniform sample elimination, and our adaptive sample elimination) and sampling during rendering (with and without adaptive path extension).*

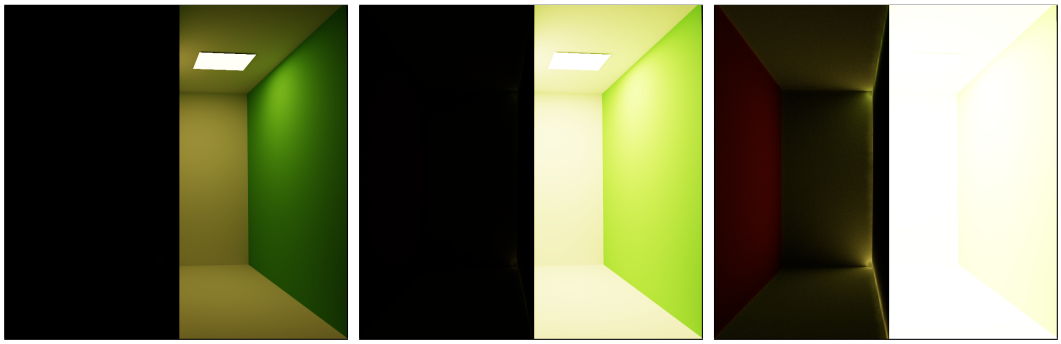
BSDF sampling results by sampling the virtual lights based on power. Path Extension fixes the undersampling of the corner part of the scene thus improve the details of rendering results.

Our sampling method is designed to be coupled with our virtual light generation method. Without a stable blue noise distribution spreading over the indirectly visible surface, the quality of BSDF sampling drops significantly. In addition, our path extension terminates a path based on whether the virtual light distribution is sufficient enough (controlled by the $n_{\min}$ and $n_{\max}$ parameters).

Noticing the similarity between our BSDF sampling of virtual lights and photon mapping with final gathering, we provide an equal-time, equal-photon comparison with photon mapping in the supplemental document where our method shows clearly better quality.

5.5 Convergence

In Figure 18, we visualize longer time convergence (up to 1000 seconds) using the scenes in Figure 12 (with the same settings). Our method (green solid line) produces much lower error than path tracing (purple line) within pre-visualization time range (1-10 seconds), but eventually converges to a biased result. Note that the error of our full method is still less than path tracing within 100 seconds in all scenes, and even after 1000 seconds in the Veach Door scene.



(a) *Exposure = 0* **(b)** *Exposure = 4* **(c)** *Exposure = 8*

Fig. 19. *Light leakage of our method in a Cornell Box scene split down the middle by an infinitely thin wall, separating the half that contains the light source from the other half that is supposed to be completely dark. The same rendered image is displayed with different exposures (rMAE = 0.003). In this example, some small amount of light leakage can be seen with exposure = 8.*

In the same plots, we also perform an ablation study to understand the importance of adaptive sample elimination for virtual light distribution (different line colors) and path extension for rendering (dashed vs. solid lines). Sample elimination (green and orange lines) produces results with lower error than a random distribution of virtual lights (red lines). Adaptive sample elimination produces results with lower error when emission profiles virtual lights are sufficiently converged (Kitchen, Modern Room, and Sibenik scenes). Otherwise, it is possible that adaptive sample elimination produces higher error than uniform sample elimination, since regions with high virtual light density can have higher variance with insufficient photon path count (Veach Door scene). Note that having a higher error in a single rendering does not mean the bias is higher. Our method generally produces blurring in the results (as can be seen in Figure 16), and adaptive sample elimination can reduce blurring compared to uniform sampling elimination, thus having lower bias. In all scenes and virtual light distributions, we can see that path extension (solid lines) reduces error faster than not using path extension (dashed lines) as the render time increases and eventually produces results of lower error.

6 CONCLUSIONS AND FUTURE WORK

We have introduced the virtual blue noise lighting that improves the indirect illumination estimation pipeline using virtual lights. Unlike typical virtual light generation that relies on light paths, our method uses camera paths for virtual light placement. We combine this approach with an adaptive sample elimination strategy that leads to a blue noise distribution with a varying density. These two methods combined provide an effective algorithm for virtual light placement that produces superior illumination sampling quality for virtual lights. Furthermore, we describe a photon splitting method that can efficiently use a large number of photon paths for generating the emission profiles of virtual lights.

We have also presented techniques that improve the illumination estimation from our virtual lights during shading. More specifically, we combine BSDF sampling and power-based sampling with MIS. Moreover, we have described an adaptive camera path extension technique for improving the near under-sampled illumination.

Our results show clear improvements over prior indirect illumination estimation methods using virtual lights. Yet, since our method is based on virtual lights, it inherits some limitations of this

approach. First and foremost, our VBNL approach does not provide an unbiased rendering method. Distributing and gathering energy from virtual lights with non-zero radius can introduce some smoothing and light leaking in the results (see Figure 19). Storing emission profiles for a large number of virtual lights is expensive. Our implementation uses a small table per light with a fixed number of quantized directions, which limits the estimation quality. An adaptive emission profile storage would be an interesting direction for future research. This is particularly important, because virtual lights with emission profiles can consume a considerable amount of memory and rendering with them can quickly become memory bound.

ACKNOWLEDGMENTS

We thank Mohi Montazer for his preliminary experiments of our virtual light generation and distribution approaches. This project was supported in part by a grant by the Epic MegaGrants program.

REFERENCES

Philippe Bekaert, Philipp Slusallek, Ronald Cools, Vlastimil Havran, and Hans-Peter Seidel. 2003. A custom designed density estimation method for light transport. (2003).

Nir Benty, Kai-Hwa Yao, Petrik Clarberg, Lucy Chen, Simon Kallweit, Tim Foley, Matthew Oakes, Conor Lavelle, and Chris Wyman. 2020. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor

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 Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020).

Per H Christensen. 2010. Point-based global illumination for movie production. In *ACM SIGGRAPH*. 40.

Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. 2014. Scalable realistic rendering with many-light methods. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 88–104.

Tomáš Davidovič, Iliyan Georgiev, and Philipp Slusallek. 2012. Progressive lightcuts for GPU. In *ACM SIGGRAPH 2012 Talks*. ACM, 1.

T Davidovic, J Krivnek, M Hasan, P Slusallek, and K Bala. 2010. Combining global and local lights for high-rank illumination effects. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 29, 5 (2010).

Thomas Engelhardt and Carsten Dachsbacher. 2008. Octahedron Environment Maps.. In *VMV*. 383–388.

Alejandro Conty Estevez and Christopher Kulla. 2018. Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1, 2 (2018), 25.

I. Evangelou, G. Papaioannou, K. Vardis, and A. A. Vasilakis. 2021. Fast Radius Search Exploiting Ray Tracing Frameworks. *Journal of Computer Graphics Techniques (JCGT)* 10, 1 (5 February 2021), 25–48.

Iliyan Georgiev, Jaroslav Krivánek, Tomas Davidovic, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6 (2012), 192–1.

Iliyan Georgiev and Philipp Slusallek. 2010. Simple and Robust Iterative Importance Sampling of Virtual Point Lights.. In *Eurographics (Short Papers)*. 57–60.

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–10.

Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. 2009. Virtual spherical lights for many-light rendering of glossy scenes. In *ACM SIGGRAPH Asia 2009 papers*. 1–6.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2007. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 26.

Peter Hedman, Tero Karras, and Jaakko Lehtinen. 2017. Sequential monte carlo instant radiosity. *IEEE transactions on visualization and computer graphics* 23, 5 (2017), 1442–1453.

Yuchi Huo, Rui Wang, Shihao Jin, Xinguo Liu, and Hujun Bao. 2015. A matrix sampling-and-recovery approach for many-lights rendering. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 210.

Henrik Wann Jensen. 1996. Global illumination using photon maps. In *Eurographics workshop on Rendering techniques*. Springer, 21–30.

Henrik Wann Jensen. 2001. *Realistic image synthesis using photon mapping*. Vol. 364. Ak Peters Natick.

Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 49–56.

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2014. Path space filtering. (07 2014).

Thomas Kollig and Alexander Keller. 2006. Illumination in the presence of weak singularities. In *Monte Carlo and Quasi-Monte Carlo Methods 2004*. Springer, 245–257.

Jaroslav Krivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. 2005. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics* 11, 5 (2005), 550–561.

Daqi Lin and Cem Yuksel. 2019. Real-Time Rendering with Lighting Grid Hierarchy. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 1 (2019), 8–1.

Daqi Lin and Cem Yuksel. 2020. Real-Time Stochastic Lightcuts. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 1 (2020), 1–18.

Yifan Liu, Kun Xu, and Ling-Qi Yan. 2019. Adaptive BRDF-oriented multiple importance sampling of many lights. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 123–133.

Mohi Montazer. 2017. *Blue noise virtual point lights for global illumination.* Master's thesis. University of Utah.

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.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021).

Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. 2012a. Progressive virtual beam lights. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1407–1413.

Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. 2012b. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–11.

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

Eric Paquette, Pierre Poulin, and George Drettakis. 1998. A Light Hierarchy for Fast Rendering of Scenes with Many Lights. *Computer Graphics Forum* 17, 3 (1998), 63–74.

Ingmar Peter and Georg Pietrek. 1998. Importance driven construction of photon maps. In *Eurographics Workshop on Rendering Techniques*. Springer, 269–280.

Benjamin Segovia, Jean Claude Iehl, Richard Mitanchey, and Bernard Péroche. 2006. Bidirectional Instant Radiosity.. In *Rendering Techniques*. 389–397.

Benjamin Segovia, Jean Claude Iehl, and Bernard Péroche. 2007. Metropolis instant radiosity. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 425–434.

Peter Shirley, Changyaw Wang, and Kurt Zimmerman. 1996. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics (TOG)* 15, 1 (1996), 1–36.

Florian Simon, Johannes Hanika, and Carsten Dachsbacher. 2015. Rich-VPLs for improving the versatility of many-light methods. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 575–584.

Ben Spencer and Mark W Jones. 2009. Into the blue: Better caustics through photon relaxation. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 319–328.

Jamorn Sriwasansak, Adrien Gruson, and Toshiya Hachisuka. 2018. Efficient Energy-Compensated VPLs using Photon Splatting. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1 (2018), 16–1.

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*. 1–9.

Greg Turk. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. *Acm Siggraph Computer Graphics* 25, 4 (1991), 289–298.

Eric Veach and Leonidas J. Guibas. 1995. Optimally Combining Sampling Techniques for Monte Carlo Rendering.

Petr Vévoda, Ivo Kondapaneni, and Jaroslav Křivánek. 2018. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 125.

Petr Vévoda and Jaroslav Křivánek. 2016. Adaptive direct illumination sampling. In *SIGGRAPH ASIA 2016 Posters*. ACM, 43.

Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. 2006. Multidimensional Lightcuts. *ACM Transactions on Graphics (Proceedings of SIGGRAPH '06)* 25, 3 (2006), 1081–1088.

Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P Greenberg. 2005. Lightcuts: a scalable approach to illumination. In *ACM SIGGRAPH 2005 Papers*. 1098–1107.

Bruce Walter, Pramook Khungurn, and Kavita Bala. 2012. Bidirectional Lightcuts. *ACM Transactions on Graphics* 31, 4, Article 59 (2012), 11 pages.

Gregory J Ward, Francis M Rubinstein, and Robert D Clear. 1988. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 85–92.

Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)* 34, 2 (2015), 25–32.

Cem Yuksel. 2019. Stochastic Lightcuts. In *High-Performance Graphics (HPG 2019)* (Strasbourg, France). The Eurographics Association, 27–32.

Can Yuksel and Cem Yuksel. 2017. Lighting grid hierarchy for self-illuminating explosions. *ACM Trans. Graph.* 36, 4 (2017), 110–1.