TU Delft

Delft University of Technology

Next Event Estimation++

Visibility Mapping for Efficient Light Transport Simulation

Guo, J.; Eisemann, M.; Eisemann, E.

DOI
10.1111/cgf.14138

Publication date
2020

Document Version
Final published version

Published in
Computer Graphics Forum

Citation (APA)
Guo, J., Eisemann, M., & Eisemann, E. (2020). Next Event Estimation++: Visibility Mapping for Efficient Light Transport Simulation. *Computer Graphics Forum*, *39*(7), 205-217. https://doi.org/10.1111/cgf.14138

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

DOI: 10.1111/cgf.14138
Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L. Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Next Event Estimation++: Visibility Mapping for Efficient Light Transport Simulation

Jerry Jinfeng Guo[†1], Martin Eisemann[‡2] and Elmar Eisemann[§1]

[1]Delft University of Technology, the Netherlands
[2]TH Köln, University of Applied Sciences, Germany

Abstract

Monte-Carlo rendering requires determining the visibility between scene points as the most common and compute intense operation to establish paths between camera and light source. Unfortunately, many tests reveal occlusions and the corresponding paths do not contribute to the final image. In this work, we present next event estimation++ (NEE++)*: a visibility mapping technique to perform visibility tests in a more informed way by caching voxel to voxel visibility probabilities. We show two scenarios: Russian roulette style rejection of visibility tests and direct importance sampling of the visibility. We show applications to next event estimation and light sampling in a uni-directional path tracer, and light-subpath sampling in Bi-Directional Path Tracing. The technique is simple to implement, easy to add to existing rendering systems, and comes at almost no cost, as the required information can be directly extracted from the rendering process itself. It discards up to 80% of visibility tests on average, while reducing variance by ~20% compared to other state-of-the-art light sampling techniques with the same number of samples. It gracefully handles complex scenes with efficiency similar to Metropolis light transport techniques but with a more uniform convergence.*

CCS Concepts

• ***Computing methodologies*** → *Ray tracing;*

Keywords: *Visibility, path tracing, bi-directional path tracing, shadowray, rendering*

1. Introduction

Unbiased Monte Carlo (MC) rendering algorithms, such as Path Tracing (PT), Bi-Directional Path Tracing (BDPT) and Metropolis Light Transport (MLT) are among the most versatile and robust rendering algorithms today. They use importance sampling techniques to efficiently solve the rendering equation [Kaj86] by integrating over light paths between the camera sensor and light emitter. An integral part, and often the most expensive, is building vertex connections between camera and light subpaths. These are solved by tracing rays between vertices to determine visibility. While an acceleration structure (usually a *bounding volume hierarchy*, BVH) can lower the cost of such queries, these tests are numerous and, depending on the outcome, there is no contribution to the final image. In fact, any occluded connection is discarded. The problem is even more aggravated by the fact that the number of camera and light vertices is generally linear in the number of pixels and light sources, whereas the number of visibility tests can be quadratic in the number of path vertices. For example, in a BDPT scenario for subpaths of length 8, internal connections comprise already ~80% of the acceleration structure visits.

We present a novel and versatile visibility mapping technique to importance sample the camera/light path connections. We precompute a voxel-to-voxel visibility within the scene, which is the probability of an unoccluded connection between a random point in the first voxel and a random point within the second voxel. For efficiency, we reuse information gathered during the initial light/camera path generation, leading to a construction overhead in the range of a few hundred milliseconds to a few seconds – a fraction of the total rendering time. We save the visibility information in a matrix-like map where the rows and columns represent different scene voxels and each entry reflects their approximate mutual visibility. During rendering, we can use the structure to estimate the probability of the success of camera-to-light vertex connections before any explicit tests. Our visibility map can be applied in various ways, e.g., Russian roulette style rejection sampling or direct importance sampling.

We name our method *next event estimation*++ (NEE++). Next event estimation (NEE) refers to direct illumination sampling, for which explicit light emitter information is given and sampled to form a potentially connected path segment. With our proposed

† Email: J.Guo-3@tudelft.nl
‡ Email: Martin.Eisemann@th-koeln.de
§ Email: E.Eisemann@tudelft.nl

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

method, we not only estimate the next event(s), but also provide an estimate of the likelihood that the connection succeeds. Our approach can drastically reduce the number of point-to-point visibility tests by up to 80% and shows an approximate 20% variance reduction compared to other techniques with the same number of samples. Specifically, our main contributions are:

- A view-independent visibility mapping, encoding voxel-to-voxel visibility, generated at a low additional cost;
- Unbiased estimators built atop of our visibility mapping;
- Several example applications of our technique.

2. Related Work

Visibility is a key component of rendering and the topic too broad to cover in this paper. Consequently, we focus only on publications reducing the amount of tests and improving sampling in the context of solving the rendering equation [Kaj86].

In his seminal work Veach [Vea98] already proposed using Russian roulette to reduce visibility tests between camera and light subpaths but without global scene information. This idea was recently extended to stratified sampling in the connection space to avoid sampling connections, which are too similar and contribute little new information to the image [CBH*18]. The opposite of early termination has been demonstrated with splitting in [VK16].

PCBPT by Popov *et al.* [PRDD15] and the close followup work by Nabata *et al.* [OHHD18] importance sample connections from a given eye vertex to a set of presampled light subpaths. They compute probabilities of connecting to the light vertices for a subset of eye vertices and interpolate these for the rest. Their technique provides a trade-off between accuracy in the probabilities and computation time but, contrary to ours, requires an expensive search query in a kd-tree for each camera vertex and is recomputed iteratively to avoid visible artifacts during convergence. The necessary adjustments for this approach also can make it difficult to integrate into existing pipelines. The information needed for our approach requires only a few simple look-ups per camera vertex to estimate the probabilities, is computed only once (though it could be updated as well) and comes basically with negligible additional cost for creation.

For a somewhat simpler problem of precomputing binary visibility for direct lighting (e.g., a surrounding environment map), computing sparse hemispherical caches that are interpolated during rendering to steer importance sampling of the map has been shown to work well [CAM08]. Augmenting these caches with additional shading information, like depth, normals, material information, they can even serve as an approximate scene representation to accelerate later bounces in unidirectional path tracing [UNRD13]. However, the direct usage, especially in earlier bounces, introduces bias, which is also present when clustering visibility into discrete bins and using these directly for visibility testing [PGSD13]. Still, visibility tests are significantly reduced and accelerated, a motivation that our work shares. In contrast, our technique is unbiased, more versatile (as it is applicable to many rendering techniques) and easier to integrate into existing rendering systems.

Caching pre-computed visibility information between spatial subdivisions and arranging such data in memory for a second render pass has been investigated in [War94, TB94, ZS95, SWZ96, FBG02, LBBS08]. Direct illumination is treated in [War94] by using a ratio of prior shadow tests as the final contribution discounter for other light samples. Hard thresholding and the direct use of the tests is biased and only addresses direct illumination. Tellier *et al.* introduces scene voxelization to form a binary visibility graph. In format this is similar to our visibility map but binary. The direct use of the binary information in the hierarchy for direct illumination calculation inevitably leads to biased results [TB94]. Fernandez *et al.* introduces similar idea by sparsely caching visibility status (fully visible, partially visible and fully occluded) for light sources. Shadow rays are only tested for partially visible light sources according to the cache. This introduces bias in the other two cases [FBG02]. Lacewell *et al.* propose to cache such information in BVH nodes, providing an early out criteria for going down the hierarchy. In principle, this re-allocates shadow-ray tests but the employed hard thresholding introduces system bias [LBBS08]. These related works are biased and have very specific application, while our way of caching visibility as probability does not suffer from bias and is more general.

As an alternative to influence the sampling process, some work also proposes a two-pass strategy [ZS95]: first a radiosity solution is computed using simplified geometry, which is view independent. The resulting surfaces are considered light sources. Next, a regular grid is produced, in which each grid cell contains the list of visible and strongly-contributing light sources. To test visibility a single point-point test is performed. During rendering, a gathering is performed at each pixel by only testing the lights connected to the cell containing the pixel. The method is biased, handles only Lambertian materials, and applies a coarse visibility sampling. Our method does not store light-sources per cell but more general visibility relations between regions of space. We avoid bias and handle arbitrary materials. Further, the cost of our preprocess is negligible.

A trend in production rendering, quickly established as the state of the art, is to guide the sampling when extending camera or light subpaths. Here, we mention only a few examples and refer the interested reader to [VHH*19]. A successful option is to learn optimal directions for ray sampling [MGN17, VKŠ*14, SJHD18, VKK18, GBBE18]. Kroes *et al.* [vRKEE16] process environmental light and visibility jointly to improve direct light estimation at scattering events in volumetric data. Similarly, Herholz *et al.* [HZE*19] sample scattering directions according to the product of phase function and incident radiance, which includes visibility and supports multiple scattering. Keller *et al.* utilize visibility information that is encoded in pre-cached photon maps in section "*Guiding and Shadow Rays*" in [VHH*19]. Visibility is reused for illumination sampling and the technique is demonstrated to improve convergence. Our technique could be viewed as a generalization as we target spatial-spatial visibility, regardless of light source or not. This enables the multiple usage as demonstrated in this paper. Though related, as these techniques reduce the number of required samples, path guiding is essentially orthogonal to our technique. It guides the sampling usually during path creation, whereas ours improves the success rate of shadow tests and could be integrated to improve next event estimation.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

3. Visibility Mapping

Our method consists of three steps: visibility-map creation (Sec.3.1), visibility caching (Sec.3.2) and visibility use (Sec.3.3), see Fig.1.

The visibility map encodes visibility relations between points in the scene. This information is gathered during the rendering process and a fast preprocess. Once established this visibility information steers the sampling process by probabilistically rejecting visibility tests that have little chance to succeed. This solution can be used for unbiased estimates.

3.1. Data Structure

The visibility map $\tilde{V} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ is a function that, given two events $x', x'' \in \mathbb{R}^3$, returns the estimated probability of a ray from x' to x'' to not intersect the scene. To render the encoding practical, $\tilde{V}$ uses an implicit uniform voxel grid, englobing the scene of resolution D^3 (Fig.1 (0.)) with each voxel having a unique index in $\{0, ...D^3 - 1\}$. For each pair of voxels (V_i, V_j), we have $\tilde{V}(x', x'') = \tilde{V}(y', y'')$ for all events $x', y' \in V_i$ and $x'', y'' \in V_j$. We encode $\tilde{V}$ as a 2D matrix, where each entry stores the mutual visibility estimate between the corresponding voxels given by the row and column indices (Fig.1 (1.)). We obtain these visibility estimates as described in the following section.

3.2. Obtaining Visibility Information

The ratio of unoccluded visibility between events in two voxels can be naively determined by randomly choosing respective points within the voxels and testing their mutual visibility. The averaged visibility converges to the wanted result. The quadratic complexity of this step and inclusion of empty voxels makes it suboptimal. Similarly, voxels within opaque watertight objects are of no relevance but would still be tested.

To determine relevant connections and reduce the overhead, we update $\tilde{V}$ stochastically. Each entry is initialized to an $\varepsilon > 0$ (in practice, 10^{-4}), which ensures that no paths are entirely rejected in the later stages to maintain unbiasedness. We then update $\tilde{V}$ during the initial camera and light path creation (including the correspondingly tested shadow rays) of the standard rendering step. We accumulate the successful and unsuccessful visibility tests (Fig.1(1.), Scenario 1) and save their ratio in $\tilde{V}$. This step comes at almost no additional costs in a standard rendering pipeline.

We can perform additional visibility tests to refine the result (Fig.1(1.), Scenario 2). This second step is especially useful for offline rendering as it improves precision and has negligible cost compared to the overall rendering time. In practice, we use 16 additional tests but restrict them to entries that have been increased from ε after the first step.

Fig.2(A) shows a visualization of our visibility map where brighter intensity values represent a higher ratio of successful visibility tests between the respective voxels. We save only the lower-triangular part of the map as a linearized vector due to the inherent symmetry, Fig.2(A, left). The visibility map converges as more visibility samples are used. Convergence behaviour appears to be similar to typical MC integration techniques. In our experiments we find that, with 64 samples per entry the result is already close to the reference solution ($1k$ samples). As $\tilde{V}$ can be reused for any viewpoint, it can be applied over several frames of a static scene without updates. This fact and the low amount of necessary rays, makes the creation overhead of the visibility map negligible in most rendering scenarios.

3.3. Using Visibility Information

Sec.1 mentioned two possible ways to use our map: rejection sampling or direct sampling. Since we are targeting a full light-transport simulation, using cached visibility to calculate illumination would result in a biased estimator. Instead, we opt for a stochastic use to solve the rendering equation (here, slightly reformulated to keep the visibility function V separate):

$$L(x', x) = L_e(x', x) + \int_A f_s(x'', x', x) L(x'', x') G(\cdot) V(x'', x') dA''(x''), \tag{1}$$

where $G(\cdot) = G(x'', x') = \cos\theta_i'' \cos\theta_o' / \left\|x'' - x'\right\|^2$. The equation describes the recursive transport of light energy from area A via scattering event x' to event x, as well as the direct energy from event x' to event x. The visibility term $V(x'', x')$ is 1 if x'' and x' are mutually visible, or 0 otherwise.

3.3.1. Rejection Sampling

Russian roulette early termination is widely used during path construction to randomly terminate paths with low throughput [PJH17]. With respect to our problem, we can apply the same technique to the visibility term. With simplification of notations from Eq.1 by leaving out parameters , we can write the estimator as:

$$\hat{L}_{rr} = L_e + \begin{cases} \frac{f_s LGV(x'', x')}{\tilde{V}(x'', x') p(x'')} & \xi < \tilde{V}(x'', x') \\ 0 & else \end{cases}, \xi \sim U[0, 1] \tag{2}$$

This estimator is unbiased:

$$\begin{aligned} E[\hat{L}_{rr}] &= L_e + \tilde{V}(x'', x') \times E[\frac{f_s LGV(x'', x')}{\tilde{V}(x'', x') p(x'')}] + (1 - \tilde{V}(x'', x')) \times 0 \\ &= L_e + E[f_s LGV] = L. \end{aligned}$$

Russian roulette does *not* reduce variance – actually it is known to increase variance. Nevertheless, by taking early outs, we avoid expensive visibility tests when they are not likely to contribute to the final image. By pruning unnecessary tests, we can allocate resources to constructing additional paths. This is a typical scenario for constant operation time optimization.

3.3.2. Direct Sampling

Our visibility map can also be used to guide importance sampling. Given a set of M candidates $\{x_i | x_i \in R^3, i = 1, \cdots, M\}$ to connect to a vertex x_0 (e.g., vertex connections in BDPT or light sampling in PT), we can build a 1D discrete distribution $P_v(x')$. We can sample this distribution and get candidate x_j and its associated probability $p_v(x_j) = \tilde{V}(x_j, x') / \sum_i \tilde{V}(x_i, x')$. The NEE++ estimator for Eq.1 is then $\hat{L} = L_e + \hat{L}_v$, where:

$$\hat{L}_v = \frac{1}{N} \sum_{i=1}^{N} \frac{f_s LGV(x_i, x')}{p_v(x_i)}. \tag{3}$$

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

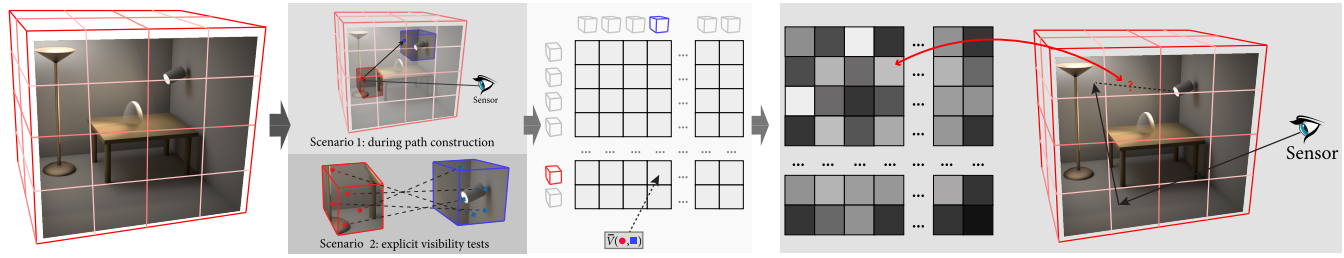


Figure 1: *NEE++ Overview. Visibility information is obtained during regular path construction and explicit spatial-spatial testing. The highlighted voxel pair in (1.), correspond to an index in the visibility map, where the average visibility is accumulated. As an application (2.), the quality of a shadow ray can be estimated by querying the estimated visibility of the voxel pair containing the path vertices.*

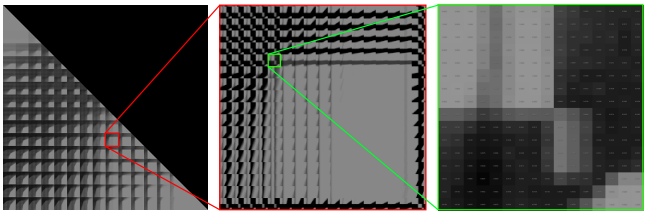


Figure 2: *A* 4096×4096 *visibility map for the* Cornell Box*, which uses* 32 *MB, including closeup images.*

Since by design our $p_v(x)$ is guaranteed to satisfy $p_v(x) \geq 0$ and $\sum p_v(x) = 1$, $p_v(x)$ is a valid pdf and our estimator unbiased.

4. Applications of Visibility Mapping

In this section we present three example applications of NEE++. Additional possibilities will be discussed in Sec.7.

4.1. NEE++ Rejection Sampling Shadow Test

Following Sec.3.3.1, we present a direct application of *rejection sampling*. When constructing a backwards-traced path from the sensor to a possible light source in unidirectional path tracing, direct illumination can be explicitly sampled at each scattering event (NEE). For each newly added path vertex, instead of explicitly testing the connection directly, we reject the test based on the probability p_{map} in $\tilde{V}$. For unbiasedness the kept samples need to be weighted with $1/p_{map}$.

The pseudo code of the NEE++ rejection sampling is given in Alg.1. The difference of our estimator from a standard path tracer lies in lines 4–6: stochastic visibility test with probability p_{map} and light-sample weighting.

4.2. NEE++ Direct Light Sampling Strategy

Using $\tilde{V}$ we can importance sample the visibility term in the rendering equation by adapting light sampling. LIGHTSAMPLING uses a prebuilt index list of voxels containing light emitters (Alg.2, line 1)

Algorithm 1 Pseudo code of NEE++ Rejection Sampling Shadow Test in unidirectional path tracing

1: **procedure** LIGHTSAMPLING(scene, path, visMap)
2: $L' \leftarrow \mathbf{0}$
3: $[sample_{light}, p_{light}] \leftarrow$scene.sampleEmitter()
4: $p_{map} \leftarrow$visMap.query(path.last, $sample_{light}$)
5: **if** $\xi \leq p_{map}$ **then**
6: $L' \leftarrow \frac{f_s LGV(\text{path.last}, sample_{light})}{p_{light} p_{map}}$
7: **return** L'

which is computed before rendering the image. To generate a light sample for a given path vertex x, we first create a 1D distribution of visibility from x to the light-containing voxels using $\tilde{V}$ (Alg.2, line 4–5). Next, we draw a sample from this distribution to choose a voxel (Alg.2, line 6), from which the light sample is drawn (Alg.2, line 7). In the degenerate case, the 1D distribution is constant, we obtain a uniform light sampling.

Algorithm 2 Pseudo code of NEE++ as a Direct Light Sampling Strategy in unidirectional path tracing

1: lightIDs$\leftarrow$BUILDLIGHTINDEX(scene)
2: **procedure** LIGHTSAMPLING(scene, path, visMap)
3: lightDistro$\leftarrow \mathbf{0}$
4: **for** i **in range** (lightToVoxels.size()) **do**
5: lightDistro[i]+=visMap.query(path.last, lightIDs[i])
6: voxID $\leftarrow$ SAMPLE1D(lightDistro, ξ)
7: $[sample_{light}, p_{light}] \leftarrow$ scene.sampleEmitterInVoxel(voxID)
8: **return** $\frac{f_s LGV(\text{path.last}, sample_{light})}{p_{light}}$

4.3. NEE++ Bi-Directional Subpath Sampling

Our visibility mapping can also be applied to BDPT by using NEE++ as a sampling strategy for choosing light subpaths out of a set of sampled light subpaths for a given camera subpaths (Fig.3). We reformulate the rendering equation in *path space* as:

$$L = \int_{\mathbb{X}} \int_{\mathbb{Y}} \text{CONNECT}(\bar{X}, \bar{Y}) d\bar{X} d\bar{Y}, \tag{4}$$

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

where the symbols are respectively:

- $\mathbb{X}$: the space of all camera subpaths;
- $\mathbb{Y}$: the space of all light subpaths;
- $\bar{X}$: a differential camera subpath in space $\mathbb{X}$;
- $\bar{Y}$: a differential light subpath in space $\mathbb{Y}$;
- CONNECT$(\cdot)$: weighted bi-directional subpath connection that connects two subpaths with multiple importance sampling.

The full path space is the bi-directional connection of two subpath-spaces, where all camera subpaths are connected to all light subpaths. As a condition, we assume that camera and light subpaths are generated independently from each other, as is the case for standard BDPT.

For integration of Eq.4, we derive the following MC estimator:

$$\hat{L}^{BiDir} = \frac{1}{M}\frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N}\bar{X}_i \bar{\circledast} \bar{Y}_j. \tag{5}$$

M and N denote the number of samples for camera and light subpaths respectively. As $\hat{L}^{BiDir}$ is an unbiased estimator, we can think of the solution of $\hat{L}^{BiDir}$ as a progressive solution that can be iterated and accumulated on demand until convergence. Consequently, the set of light paths $\bar{\mathbf{Y}}$ in each iteration can be generated beforehand for each camera sample $\hat{X}_i$. Instead of connecting $\bar{X}_i$ to all subpaths in the set $\bar{\mathbf{Y}}$, we can rewrite Eq.5 as an importance-sampling problem, where only one light subpath is selected with a certain probability for each camera subpath:

$$\hat{L}_v^{BiDir} = \frac{1}{M}\sum_{i=1}^{M}\frac{\bar{X}_i \bar{\circledast} \bar{Y}_j}{p_v^{BiDir}(\bar{Y}_j|\bar{X}_i))}, \tag{6}$$

where $\bar{Y}_j$ is the sample drawn from $\bar{\mathbf{Y}}$ with probability $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$. Here $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$ is a valid probability mass function, a discrete probability distribution (the set of light subpaths $\bar{\mathbf{Y}}$, from which to sample $\bar{Y}_j$), rather than a continuous pdf.

We define the value of $p_v^{BiDir}(\bar{Y}_j|\bar{X}_i)$ as the normalized sum of visibility values in $\tilde{V}$ to connect all vertices in $\bar{Y}_j$ to $\bar{X}_i$ to increase the likelihood for vertices to connect. $\hat{L}_v^{BiDir}$ is still an unbiased estimator.

To obtain $\bar{X}_i$, $\bar{Y}_j$ and p_v^{BiDir}, we generate one camera subpath at a time and N light subpaths that get reused for all camera samples of one iteration and construct p_v^{BiDir} from this. The step by step pseudocode for BDPT with NEE++ and building p_v^{BiDir} is given in Alg.3 and described in the following.

In each iteration, we create a subset $\bar{\mathbf{Y}}$ of light subpaths. For each camera subpath $\bar{X}_i$, we build p_v^{BiDir} (Alg.3, line 5) to select a corresponding light subpath $\bar{Y}_j$ to connect to (Alg.3, line 6). To build p_v^{BiDir} (Alg.3, line 9–13), we estimate a probability for $\bar{X}_i$ to connect successfully to each of the light subpaths $\bar{Y}_j \in \mathbb{Y}$. The probability for selecting a light subpath $\bar{Y}_j$ is the normalized accumulated visibility to connect all vertices of $\bar{X}_i$ to $\bar{Y}_j$ (Alg.3, line 14–18).

Algorithm 3 Pseudo code of NEE++ for Bi-Directional Subpath Sampling in a BDPT

```
1:  procedure RenderIteration(scene, image, visMap)
2:      Ȳ ←LightTracing(scene, N)
3:      for i in range(M) do
4:          X̄_i ←RandomWalkFromSensor(scene)
5:          P_v^BiDir ←BuildDistribution(visMap, X̄_i, Ȳ)
6:          [Ȳ_j, p_v^BiDir(Ȳ_j)] ←Sample1D(Ȳ, P_v^BiDir, ξ)
7:          L_i ← Connect(X̄_i,Ȳ_j) / p_v^BiDir(Ȳ_j)
8:          image.add(L_i)
9:  procedure BuildDistribution(visMap, X̄, Ȳ)
10:     subpathDistro← 0
11:     for all Ȳ_j in 𝕐 do
12:         subpathDistro[j] += QueryPaths(visMap, X̄, Ȳ_j)
13:     return Normalize(subpathDistro)
14: procedure QueryPaths(visMap, X̄, Ȳ)
15:     v ← 0
16:     for all connections (x, y) in [X̄, Ȳ] do
17:         v += visMap.query(x,y)
18:     return v
```

5. Results

All implementations and experiments are carried out within the framework of PBRT [PJH17] on an i7 6800K 3.4GHz six core PC with 32GB RAM. We implemented test cases for all three applications (Sec.4). PBRT uses tile rendering, so we do not create light subpaths for the whole image but per tile and iterate instead with the number of light subpaths equal to the number of camera samples per tile in each iteration. We used the default PRB setting (16×16), an optimization of this parameter is left for future work. Both camera subpaths and light subpaths are generated with only local importance sampling, i.e., BSDF sampling, without any guiding information, such as the position of their counterparts. This way we ensure that both sets are generated independently from each other and identically distributed within the same set.

We use a voxelization of $16 \times 16 \times 16$, resulting in a visibility map of 4096×4096 for all our test cases. For all tests, we initially run a regular render iteration and, in between two iterations, we create the visibility map. Timings for the construction are given in Tab.1, which shows the negligible costs.

5.1. Experiment Objectives

Since we apply our technique in three different scenarios, three different objectives are considered. They are:

- Reducing shadowray tests for Sec.5.2 while achieving similar results;
- Robustly reducing variance comparing to other light sampling strategies (*Uniform, Power* and *Spatial*) for Sec.5.3, given sample rate or runtime;
- Robustly reducing variance comparing to other unbiased estimators (*Path, (M) MLT* and *BDPT*) for Sec.5.4, given sample rate or runtime.

5.2. NEE++ Rejection Sampling Shadow Test

We test our first technique (Sec.4.1) in a unidirectional path tracer with the *Veach ajar* scene (Fig.4). Most of the light comes from the

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

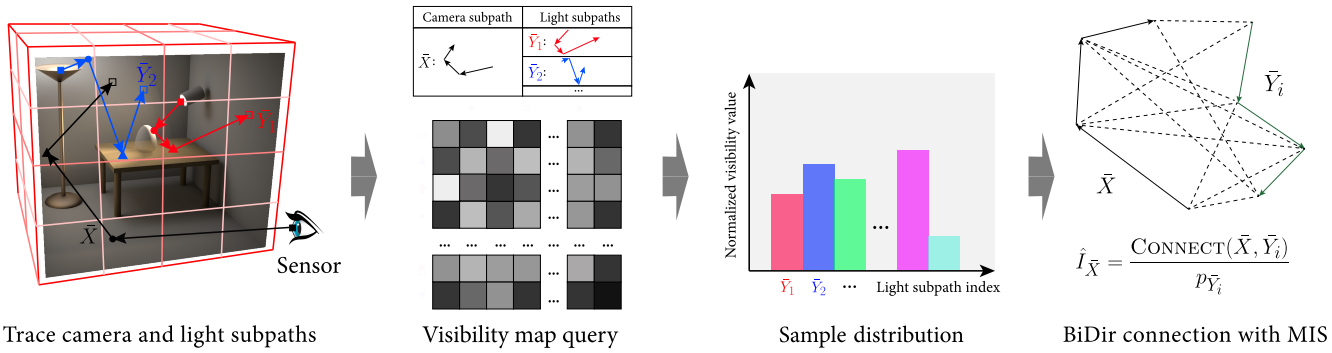


Figure 3: *An illustration of NEE++ in the context of BDPT. By constructing a distribution for all candidate light subpaths, which is a true subset of all light subpaths, we are able to utilize the precomputed visibility information and select a light subpath that has a higher chance of establishing successful connections with a higher probability.*

Scene	*Test Corridor*	*Ajar*	*Kitchen*
Timing	231.4 ms	331.5 ms	401.2 ms
Scene	*Staircase*	*N. Classroom*	*Bathroom*
Timing	3413.4 ms	2350.1 ms	1579.6 ms

Table 1: *Timing for caching visibility maps during our experiments. As a reference, the typical render time for these scenes with a relatively noise free output ranges from thousands of seconds to tens of hours.*

small gap at the door and the vast majority of the scene receives no direct light, which leads to many unnecessary shadow rays. The main objective of this experiment is to verify whether our technique could reduce shadowray tests (and subsequently respective runtime) while achieving similar results. Notice that plots in Fig.4 are roughly *equal quality*, of which the rMSE can be found in Tab.2

This scene was mostly chosen for its particular configuration and as it is well-known. It has a particularly simple geometry, which results in the main cost being related to shading computations. In consequence, one cannot expected a strong impact on the total rendering cost (our solution results in a $1-4\%$ speedup). Nevertheless, as can be seen in the figures and the table, we manage to cut around 80% of the shadow rays and use a fraction $(28-34\%)$ of the runtime for visibility tests. The cutting of shadowrays is desirable for complex scenes.

5.3. NEE++ Direct Light Sampling Strategy

We test NEE++ as a light sampling strategy (Sec.4.2) with the *Kitchen* scene (Fig.5) and the *Staircase* scene Fig.7. Both scenes feature multiple light sources with varying intensity and size. Some sources are very small and located in a relatively complex region (e.g., above the microwave, under the staircase), which can lead to undesirable samples.

We compare our strategy against three existing light sampling

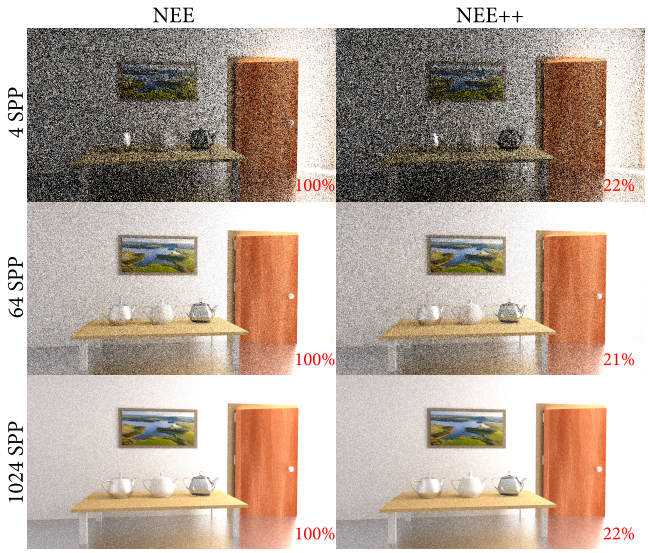


Figure 4: *NEE++ Rejection Sampling Shadow Test. We achieve similar visual results with less shadow-ray queries and lower render time. Ratios of tests compared to regular path tracing are in the bottom-right corner of each image. Tab.2 contains detailed statistics.*

strategies: *uniform distribution*, *power distribution* and *spatial distribution*. Uniform distribution treats each light source equally: $P(\text{Light}_i) = P(\text{Light}_j) = \frac{1}{\text{number of lights}}$. Power distribution favours light sources with larger area and/or stronger emitting intensity: $P(\cdot) \propto A(\cdot)E(\cdot)$. These distributions are globally static distributions (invariant to the used query points). Spatial distribution takes distances as well as emitter surface and energy into account. Details about all distributions can be found in [PJH17]. Results are given in Fig.5, Fig.6 and Fig.7. The timing and rMSE are reported in Tab.3 and Tab.4 respectively . We notice:

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

	rMSE			Shadowrays			Runtime Visibility Tests		
	4 SPP	64 SPP	1024 SPP	4 SPP	64 SPP	1024 SPP	4 SPP	64 SPP	1024 SPP
NEE	5.82e−2	3.24e−2	2.12e−2	5.45e+6	8.72e+7	1.40e+9	1.26s	5.47s	82.4s
NEE++	5.91e−2	3.33e−2	2.18e−2	1.19e+6	1.92e+7	3.07e+8	0.43s	1.54s	24.9s
Ratio	1.01	1.03	1.03	0.22	0.21	0.22	0.34	0.28	0.30

Table 2: *Statistics for rMSE, shadow-ray queries and runtime for Fig.4. We achieve similar results (rMSE) with significantly reduced shadow rays (around* 80% *less) and much shorter runtime.*

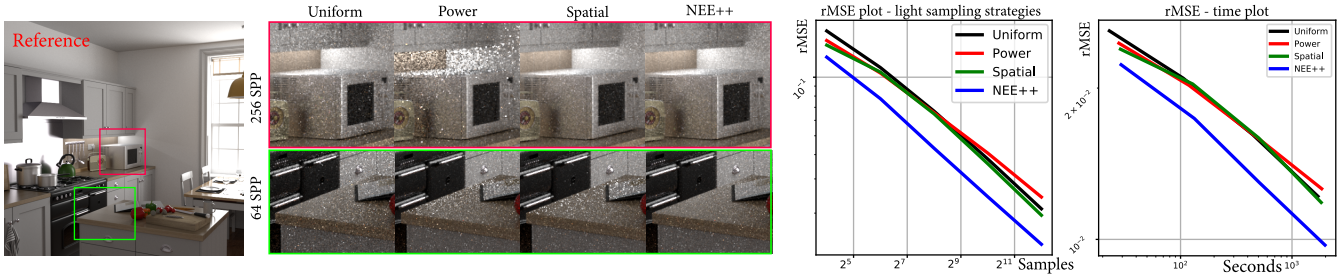


Figure 5: *NEE++ Direct Light Sampling Strategy. Our strategy is more robust with* $\sim$ 20% *variance reduction overall. Details are given in Tab.3 and Tab.4. Fig.6 shows an equal variance comparison.*

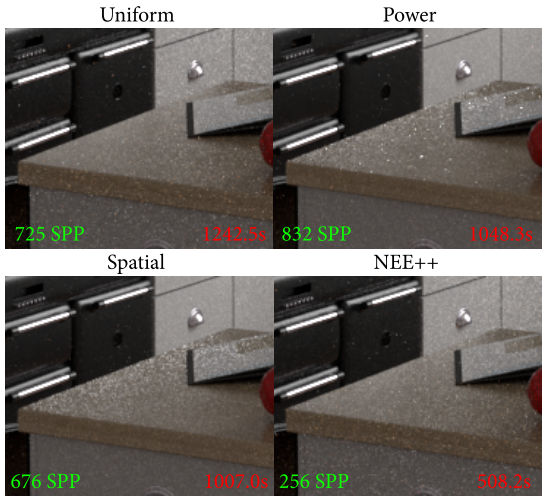


Figure 6: *Equal rMSE comparison corresponding to Fig.5. Sample rate is given in green, while the runtime performance is red. For similar variance, our method takes* 35%, 30% *and* 38% *the sample rate of* uniform, power *and* spatial *respectively with runtime ratios of* 40.1%, 48.5%, *and* 50.5%.

Sampl. Strategy	16 SPP	64 SPP	256 SPP	1024 SPP
Uniform	23.0s	105.6s	442.6s	1710.4s
Power	28.0s	118.9s	474.6s	1846.4s
Spatial	28.9s	128.6s	502.8s	1827.1s
NEE++	29.3s	132.8s	508.2s	1977.1s
Uniform	10.6s	50.2s	204.2s	768.7s
Power	10.2s	46.9s	184.8s	712.5s
Spatial	35.8s	77.3s	237.8s	855.3s
NEE++	11.5s	56.9s	231.3s	968.8s

Table 3: *NEE++ as light sampling strategy (Fig.5 and Fig.7). Statistics for runtime. Top: the* Kitchen *scene; bottom: the* staircase *scene. NEE++ has a slight* 1% $\sim$ 8% *runtime overhead compared to the other three strategies with the same number of samples.*

- NEE++ outperforms other strategies with the same sample rate in terms of error (Fig.5 and Fig.7);
- NEE++ better handles visibility-crucial regions and is less likely to sample strong light sources that are less visible (Fig.5, middle);
- NEE++ achieves 20% variance reduction with 1% $\sim$ 8% computational overhead (Tab.3);
- NEE++ requires only about a fraction of the number of samples compared to other light sampling techniques to match the error (Fig.6 and Fig.7).

The power distribution fails to handle the regions above the microwave, and causes severe fireflies due to the low probability of reaching this light source. The spatial distribution creates fireflies on the table for the same reason. The runtime overhead could be even further reduced by caching and reusing light distributions, which we do not do in our current implementation. We notice an unexpectedly long runtime for strategy *Uniform*, our repeated experiments confirm this behaviour. We use the original code from

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

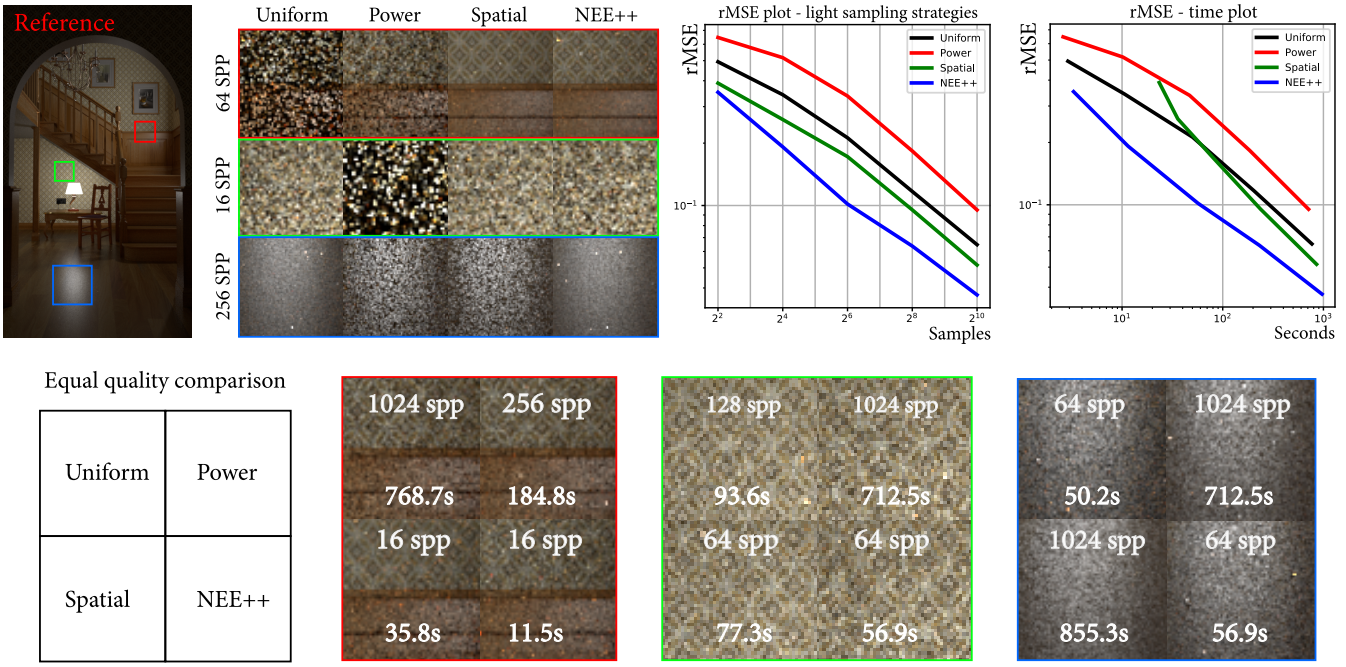


Figure 7: *Top left:* Staircase *reference. Top centre: equal sample rate comparison. Top right: plot of rMSE against sample rate and runtime. Bottom: equal quality comparison. NEE++ outperforms other methods in terms of convergence and is more robust.*

Strategy	16 SPP	64 SPP	256 SPP	1024 SPP
Uniform	2.60e−2	2.09e−2	1.62e−2	1.22e−2
Power	2.46e−2	2.03e−2	1.60e−2	1.26e−2
Spatial	2.40e−2	2.04e−2	1.59e−2	1.18e−2
NEE++	2.23e−2	1.74e−2	1.29e−2	9.74e−3
Uniform	3.43e−1	2.12e−1	1.17e−1	6.47e−2
Power	5.17e−1	3.38e−1	1.83e−1	9.52e−2
Spatial	2.60e−1	1.72e−1	9.52e−2	5.16e−2
NEE++	1.92e−1	1.02e−1	6.37e−2	3.70e−2

Table 4: *NEE++ as light sampling strategy (Fig.5 and Fig.7). Statistics for rMSE. Top: the* Kitchen *scene; bottom: the* Staircase *scene. Overall, NEE++ has a* $\sim$ *20% improvement with the same number of samples.*

PBRT [PJH17] and the exact mechanism behind this phenomenon is not known to us.

5.4. NEE++ Bi-Directional Subpath Sampling

We compare NEE++ using BDPT Subpath Sampling (Sec.4.3) with three unbiased MC estimators: unidirectional path tracing (Path), bi-directional path tracer (BDPT) and *Metropolis light transport* (MLT). We test all methods on three scenes: *Test Corridor*, *Nasty Classroom* and *Bathroom*. All scenes feature complex visibility: *Test Corridor* exhibits a minimum connected path depth of 4 (Fig.8), *Nasty Classroom* has many light sources with blocking geometry (Fig.10). Statistics for rMSE are provided in Tab.5.

Test Corridor This is a typical scenario where traditional MC methods, such as ours, have a strong disadvantage compared to Markov chain MC (i.e., MLT). All traditional MC methods struggle to connect the camera to the light source, given the nature of the scene configuration. In fact, this scene should not exhibit a single black pixel upon convergence but all connected light rays have to undergo several bounces. The render results and the runtime are given in Fig.8, with stats plotted in Fig.9. For the same sample rate, we observe:

- NEE++ outperforms all other methods in terms of zero-radiance pixel ratio (Fig.9 (right));
- MLT has better rMSE, at the expense of uneven convergence (Fig.9 (left));
- At the same sample rate NEE++ has a $\sim$ 15% runtime overhead compared to BDPT (Fig.9) while achieving 20% error reduction;
- Path tracing performs worst in terms of rMSE and zero-radiance pixel ratio per sample (Fig.9).

The fact that NEE++ has less zero-radiance pixels illustrates the benefit of our solution and the effectiveness of finding even complex paths. Further, the runtime overhead with respect to BDPT is low, while achieving a stronger error reduction in this scene, despite it being a very disadvantageous scene for our approach. While MLT does outperform all competitors in terms of error, MLT is less often employed in practice due to its irregular convergence, which becomes more visible in the following scenes.

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

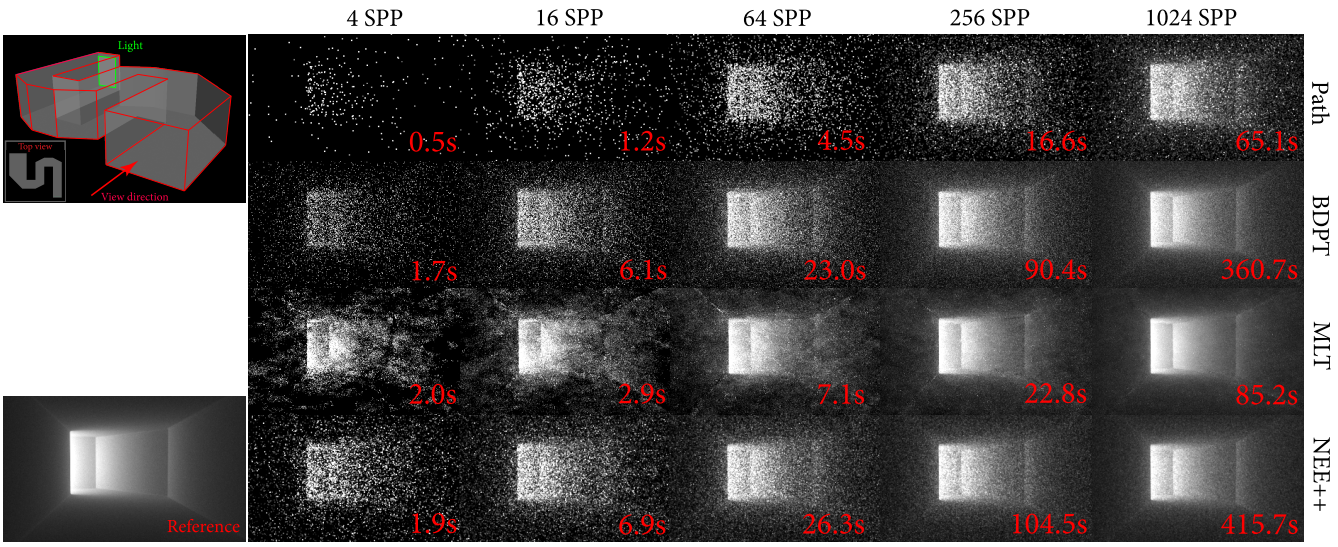


Figure 8: *Left top:* Test Corridor *scene. The whole corridor geometry is only open at the right end and closed elsewhere. Left bottom: reference, rendered using unidirectional path tracing with* 102,400 *samples per pixel. Notice that none of the pixels is black. Right: Results for various methods. Notice the difficult visibility configuration, which poses a challenge for all methods. Path tracing struggles to reach the light source, causing many occluded shadow-ray queries. BDPT manages to connect the light source to the sensor but a large portion of the pixels remain black. MLT successfully exploits the vicinity in path space, once a connected path is constructed, resulting in much less zero-radiance pixels. The uneven convergence is an inherent problem. Our NEE++ manages to establish more connected paths than BDPT, while achieving more even convergence than MLT with the same number of samples. Render times are indicated in the bottom-right corner.*

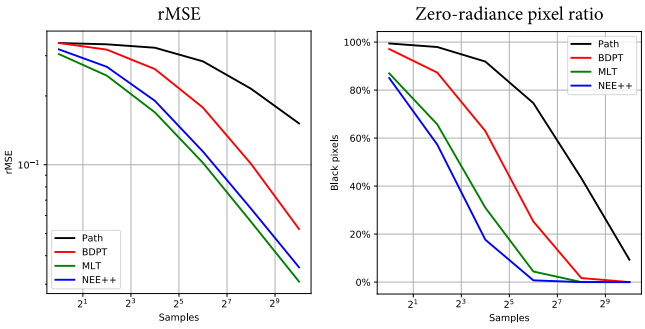


Figure 9: *Statistics of the* Test Corridor *scene (Fig.8): rMSE and zero-radiance pixel ratio. MLT performs slightly better in terms of rMSE than NEE++ but it suffers from uneven convergence and more black pixels. To get more out of Fig.8 and the plots here, we see that at a roughly equal error rate* (10^{-1})*: Path tracing takes* 2048 *samples and* 123 *seconds; BDPT* 256 *samples and* 90 *seconds; NEE++* 84 *samples and* 36 *seconds: a* 67% *sample-rate reduction and* 60% *in render time.*

Nasty Classroom and Bathroom We use a modified version of the *Classroom* scene, using a night scenario with light emitting geometry on the ceiling instead of the original virtual sun and sky light. Challenging parts of the scene include classroom furniture with glossy materials, the cabinet of desks that are hidden away from direct light sources, and the area below the desks, where visibility towards light sources is complex. Render results, rMSE plot, and runtime analysis are given in Fig.10, Fig.12 and Tab.5 . Equal quality comparisons are given in Fig.11. When comparing at the same sample rate, we observe:

- NEE++ outperforms all other methods given the same render time;
- NEE++ improves over BDPT by 15% $\sim$ 20% variance reduction with $\sim$ 7% runtime overhead
- MLT shows very irregular convergence behaviour, largely due to the complex visibility and glossy materials

When comparing equal quality, we observe that NEE++ typically takes a fraction of the sample rate ($50-1.5\%$) and runtime ($49.9-14\%$).

6. Discussions

Throughout all our experiments, we see the effectiveness of NEE++. We tested multiple application scenarios and showed improvements over existing methods: in Sec.5.2, we see the reduction in visibility tests; in Sec.5.3, we see how applying our solution to the visibility term in the rendering equation, we are able to perform better direct illumination sampling; in Sec.5.4, we see a similar but extended version of direct sampling, for which we are able to connect camera subpaths to better candidate light subpaths given the visibility information.

The improvements are achieved at the cost of a small computa-

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

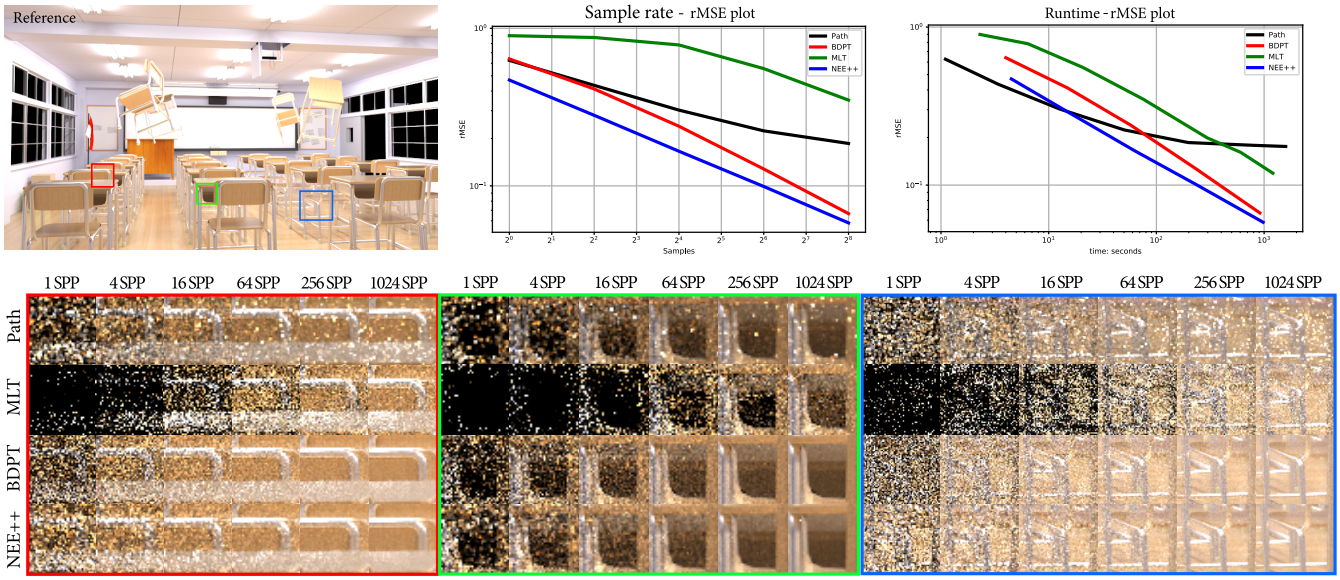


Figure 10: *Top left:* Nasty Classroom *reference. Top centre: rMSE plot. Top right: experiment timings.*

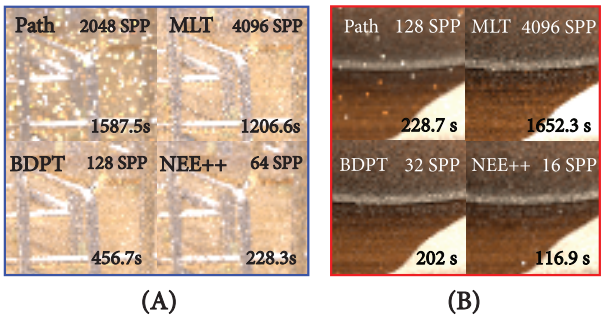


Figure 11: *Equal quality comparison (A) for scene* Nasty Classroom *in Fig.10 and (B) for scene* Bathroom *in Fig.12.*

Method	4 SPP	16 SPP	64 SPP	256 SPP
Path	3.36e−01	3.25e−01	2.83e−01	2.15e−01
MLT	2.45e−01	1.69e−01	1.02e−01	5.63e−02
BDPT	3.19e−01	2.62e−01	1.78e−01	1.01e−01
NEE++	2.68e−01	1.90e−01	1.14e−01	6.42e−02
Path	4.35e−01	3.03e−01	2.23e−01	1.86e−01
MLT	8.75e−01	7.84e−01	5.55e−01	3.50e−01
BDPT	4.12e−01	2.39e−01	1.28e−01	6.67e−02
NEE++	2.81e−01	1.66e−01	9.91e−02	5.82e−02
Path	2.03e−01	1.22e−01	8.78e−02	6.52e−02
MLT	7.26e−01	6.39e−01	4.34e−01	3.04e−01
BDPT	1.87e−01	9.91e−02	5.55e−02	3.71e−02
NEE++	1.25e−01	7.62e−02	4.68e−02	3.35e−02

Table 5: *NEE++ for Bi-Directional subpath sampling (Fig.8, Fig.10 and Fig.12). Statistics for rMSE. Top:* Test Corridor*; Middle:* Nasty Classroom*; Bottom:* Bathroom*. Overall, our method manages to improve* $\sim$ 20% *upon BDPT at the same sample rate. The poor robustness of MLT could also be clearly derived by comparing the different performance in three test scenes.*

tional overhead. Per sample costs occur when accessing the visibility map queries and its building process, which could be further optimized, adds an additional cost. Nevertheless, with respect to the overall gain or total computation times, these costs are negligible. Further, we could envision reducing these costs by caching or reusing information, which we do not do in our current implementation - all examples generated the visibility map from scratch.

In most cases, we manage to achieve better convergence per sample, mostly by constant improvement over existing MC methods, i.e., roughly at the same convergence rate but better by a constant offset. Even with the computational overhead and without further optimization, we can conclude that our technique outperforms the other techniques when it comes to fix-runtime performance given the nature of the MC convergence rate being $1/\sqrt{N}$.

6.1. Grid resolution

While we are working on a relatively low grid resolution, one might argue that a visibility map with higher resolution gives higher precision and therefore better results. However, in our experiments, we find that increasing the resolution increases memory requirements and preprocessing time but quickly shows little benefit (Fig.13 shows an example in the *Kitchen* scene).

Higher resolution poses other challenges such as high memory

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

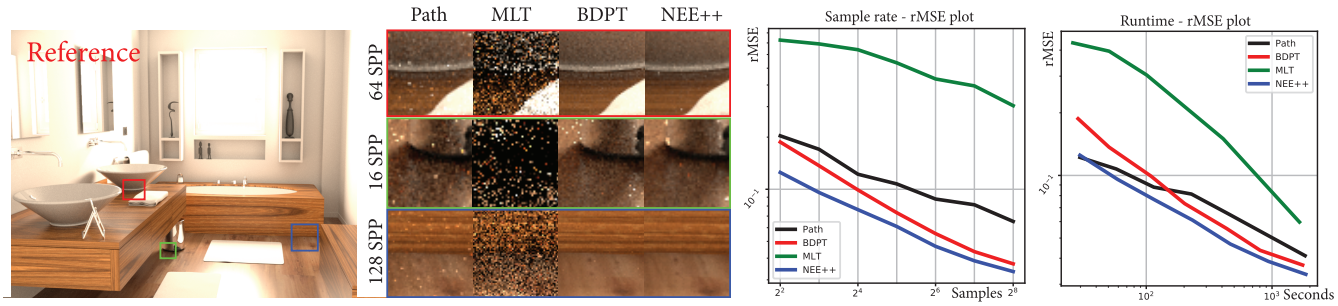


Figure 12: *Results for the* Bathroom *scene. rMSE is plotted against sample rate and runtime.*

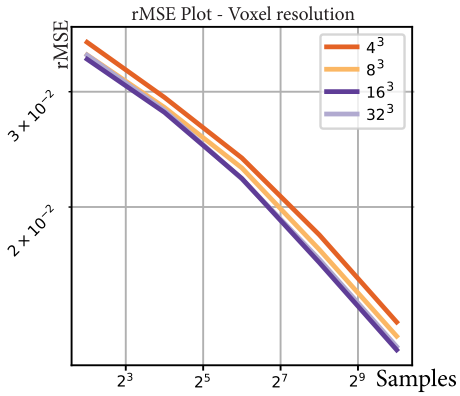


Figure 13: *Comparison of different voxelgrid resolutions for the* Kitchen *scene (Fig.5). The overall error is reduced with increasing visibility-map resolution but a sweetpoint exists, depending on the scene complexity (compare* 16^3 *and* 32^3*). Legend: grid resolution.*

footage and overfitting. Our primary experiments for different grid resolution at high sample rate show no conclusive evidence about the improvement. Our way of stochastically using the values might contribute to this phenomenon.

Given the cost-benefit at current low computational overhead, we leave further investigation for future work.

6.2. Compatibility with Path Guiding

As mentioned in Sec.2, our visibility mapping could be used along with various path guiding techniques. Most path guiding methods rely on crafting distributions for importance sampling during path construction. Such distributions usually target one or more terms in the rendering equation. In the same spirit as in Sec.4.2, we could further incorporate our visibility mapping in assisting processes where rays are guided by, e.g., photons, since visibility is usually not taken care of. Additionally, our visibility Russian roulette is always an option for pruning shadow ray tests.

6.3. Comparison with PCBPT

Our technique resembles PCBPT [PRDD15] in some aspects. As both methods target better connections during BDPT. Yet, our method works on subpath level while PCBPT works on path vertex level. This induces three main differences.

- First of all, most of the cached information for our technique comes for free in a previous and independent pass, while for PCBPT a dedicated procedure to test visibility between a subset of camera vertices and all light vertices in current pass is needed repeatedly. This difference not only affects runtime, but the domain. Our visibility cache is a spatial-spatial relation while PCBPT caches only spatial-index relations. This difference makes it possible for our technique to be used in various scenarios while PCBPT is only applicable to BDPT;

- Secondly, we select a whole light subpath based on the visibility map while for PCBPT at each camera vertex a completely different light subpath could be chosen. This further complicates the calculation for multiple importance sampling. In this sense PCBPT is a derivation of *Combinatorial BDPT* [PBPP11] while our technique is an extension for standard BDPT;

- Thirdly, PCBPT runs iteratively and as a result the computational overhead is much higher than ours ($\sim$ 20% as reported in [PRDD15]).

6.4. Limitations

During our experiments, we noticed that while our method mostly improves the result given a fixed sample budget or runtime, our method will be less efficient in scenes where visibility is not the major source of complexity (e.g., scenes with large open spaces). Furthermore, we currently englobe the entire scene in a voxel grid, which does not cover directional light or environment maps.

6.5. Future work

Using a uniform voxelization wastes some memory with empty voxels and the coarse resolution does not always well represent the underlying geometry. This could be addressed with adaptive voxelization but it comes at the expense of further pre-processing

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

and query time. Additionally, a geometry-based adaptation alone will not be successful, as geometry density is not necessarily the cause of complex visibility. Therefore it is not clear if this would improve the technique. Currently, our solution does already avoid calculations on empty voxels and we have seen that a higher resolution does not yield much additional benefit in our test scenes. Also, a ray traversal in the acceleration tree structure would have to be replaced with a query in a tree structure, although one might be able to make use of hashing schemes.

One potential solution for adaptive structure could be to make use of hierarchical division where submaps are associated to spatial divisions of the scene. I.e., creating several level of visibility map and only querrying at the correct level in the hierarchy. One other potential solution is to make use of an existing acceleration structure: e.g., *bounding volume hierarchy* (BVH). This volume information comes for free but requires further indexing before it can be compatible with our pipeline.

In this paper, we investigate the possibilities of unbiased usage of NEE++, but unbiasedness might not always be of highest priority. For example, in the context of *many light rendering*, sharing one visibility map across *virtual point light* (VPL) sources instead of rendering one shadow map for each VPL could be useful. In realtime applications, visibility tests could become the bottleneck as such shading operations are typically cheap. Our technique could serve as an approximation in situations where inaccurate visibility won't pose fatal error. It could also be an option for secondary and onward bounces in path tracing to use visibility-map values directly, since for interactive purposes primary bounces make the most visual impact.

7. Conclusions

We presented NEE++, a visibility mapping technique with applications to several rendering problems. It efficiently caches and reuses visibility information to improve light transport simulations. By collecting visibility in a matrix-like form during a low-cost preprocess, we can reduce visibility tests by up to 80% and prefer tests that are likely to succeed instead. Variance at the same time is also reduced by ~20%. Our solution is unbiased and general. It is easy to implement and integrate into existing pipelines. The exemplary cases show that it can be widely applied in several scenarios but many more potential applications exist.

8. Acknowledgments

This work was partially supported by the NWO Vernieuwingsimpuls VIDI grant *Next View*. The authors would like to thank the reviewers for the valuable comments and suggestions. Special thanks for Dr. Billeter for fruitful discussions. The majority of the test scenes used in this work are based on rendering resources released by Bitterli [Bit16].

References

[Bit16] BITTERLI B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 12

[CAM08] CLARBERG P., AKENINE-MÖLLER T.: Exploiting visibility correlation in direct illumination. *Computer Graphics Forum 27*, 4 (2008), 1125–1136. doi:10.1111/j.1467-8659.2008.01250.x. 2

[CBH*18] CHAITANYA C. R. A., BELCOUR L., HACHISUKA T., PREMOZE S., PANTALEONI J., NOWROUZEZAHRAI D.: Matrix bidirectional path tracing. In *Proceedings of the EGSR: EI&I* (Goslar, DEU, 2018), Eurographics Association, p. 23–32. doi:10.2312/sre.20181169. 2

[FBG02] FERNANDEZ S., BALA K., GREENBERG D. P.: Local illumination environments for direct lighting acceleration. *Rendering Techniques 2002* (2002), 13th. doi:10.5555/581896.581898. 2

[GBBE18] GUO J. J., BAUSZAT P., BIKKER J., EISEMANN E.: Primary sample space path guiding. In *Proceedings of the Eurographics Symposium on Rendering: EI&I* (2018), SR '18, Eurographics Association, p. 73–82. doi:10.2312/sre.20181174. 2

[HZE*19] HERHOLZ S., ZHAO Y., ELEK O., NOWROUZEZAHRAI D., LENSCH H. P. A., KŘIVÁNEK J.: Volume path guiding based on zero-variance random walk theory. *ACM Trans. Graph. 38*, 3 (2019), 25:1–25:19. doi:10.1145/3230635. 2

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (1986), SIGGRAPH '86, ACM, pp. 143–150. doi:10.1145/15922.15902. 1, 2

[LBBS08] LACEWELL D., BURLEY B., BOULOS S., SHIRLEY P.: Raytracing prefiltered occlusion for aggregate geometry. In *2008 IEEE Symposium on Interactive Ray Tracing* (2008), IEEE, pp. 19–26. doi:10.1109/RT.2008.4634616. 2

[MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum 36*, 4 (2017), 91–100. doi:10.1111/cgf.13227. 2

[OHHD18] OTSU H., HANIKA J., HACHISUKA T., DACHSBACHER C.: Geometry-aware metropolis light transport. *ACM Trans. Graph. 37*, 6 (Dec. 2018). doi:10.1145/3272127.3275106. 2

[PBPP11] PAJOT A., BARTHE L., PAULIN M., POULIN P.: Combinatorial bidirectional path-tracing for efficient hybrid cpu/gpu rendering. *Computer Graphics Forum 30*, 2 (2011), 315–324. doi:10.1111/j.1467-8659.2011.01863.x. 11

[PGSD13] POPOV S., GEORGIEV I., SLUSALLEK P., DACHSBACHER C.: Adaptive quantization visibility caching. *Computer Graphics Forum 32* (2013), 399–408. doi:10.1111/cgf.12060. 2

[PJH17] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering, Third Edition: From Theory To Implementation*, 3rd ed. Morgan Kaufmann Publishers Inc., 2017. 3, 5, 6, 8

[PRDD15] POPOV S., RAMAMOORTHI R., DURAND F., DRETTAKIS G.: Probabilistic connections for bidirectional path tracing. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 34*, 4 (2015). doi:10.1111/cgf.12680. 2, 11

[SJHD18] SIMON F., JUNG A., HANIKA J., DACHSBACHER C.: Selective guided sampling with complete light transport paths. *Transactions on Graphics (Proceedings of SIGGRAPH Asia) 37*, 6 (2018). doi:10.1145/3272127.3275030. 2

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Trans. Graph. 15*, 1 (Jan. 1996), 1–36. doi:10.1145/226150.226151. 2

[TB94] TELLIER P., BOUATOUCH K.: Physics-based lighting models: Implementation issues. In *Photorealistic Rendering in Computer Graphics*. Springer, 1994, pp. 112–121. doi:10.1007/978-3-642-57963-9_12. 2

[UNRD13] ULBRICH J., NOVÁK J., REHFELD H., DACHSBACHER C.: Progressive visibility caching for fast indirect illumination. In *Proceedings of the Vision, Modeling, and Visualization Workshop* (2013), Bronstein M. M., Favre J., Hormann K., (Eds.), Eurographics Association, pp. 203–210. URL: https://cg.ivd.kit.edu/english/PVCFID.php. 2

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1998. URL: http://graphics.stanford.edu/papers/veach_thesis/. 2

[VHH*19] VORBA J., HANIKA J., HERHOLZ S., MÜLLER T., KŘIVÁNEK J., KELLER A.: Path guiding in production. In *ACM SIGGRAPH 2019 Courses* (New York, NY, USA, 2019), SIGGRAPH '19, ACM, pp. 18:1–18:77. doi:10.1145/3305366.3328091. 2

[VK16] VORBA J., KŘIVÁNEK J.: Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Trans. Graph. 35*, 4 (July 2016). doi:10.1145/2897824.2925912. 2

[VKK18] VÉVODA P., KONDAPANENI I., KŘIVÁNEK J.: Bayesian online regression for adaptive direct illumination sampling. doi:10.1145/3197517.3201340. 2

[VKŠ*14] VORBA J., KARLÍK O., ŠIK M., RITSCHEL T., KŘIVÁNEK J.: On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014) 33*, 4 (2014). doi:10.1145/2601097.2601203. 2

[vRKEE16] VON RADZIEWSKY P., KROES T., EISEMANN M., EISEMANN E.: Efficient stochastic rendering of static and animated volumes using visibility sweeps. *Transactions on Visualization and Computer Graphics 23*, 9 (2016), 2069 – 2081. doi:10.1109/TVCG.2016.2606498. 2

[War94] WARD G. J.: Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics*. Springer, 1994, pp. 11–20. doi:10.1007/978-3-642-57963-9_2. 2

[ZS95] ZIMMERMAN K., SHIRLEY P.: A two-pass solution to the rendering equation with a source visibility preprocess. In *Eurographics Workshop on Rendering Techniques* (1995), Springer, pp. 284–295. doi:10.1007/978-3-7091-9430-0_27. 2

© 2020 The Author(s)
Computer Graphics Forum © 2020 The Eurographics Association and John Wiley & Sons Ltd.