Journal of Computer Graphics Techniques Vol. 14, No. 2, 2025 http://jcgt.org

Neural Visibility Cache for Real-Time Light Sampling

Jakub Bokšanský
Advanced Micro Devices, Inc.

Daniel Meister
Advanced Micro Devices, Inc.

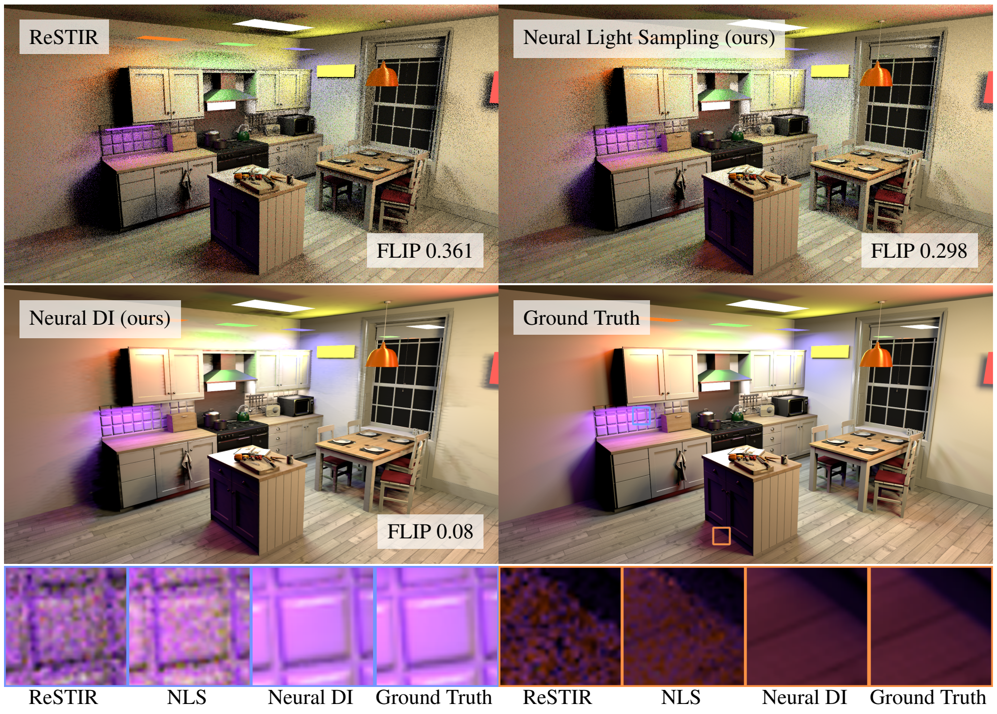

Figure 1. Rendering of the kitchen scene with 32 lights using screen-space ReSTIR (top left) at 3.61 ms per frame, compared to our neural light sampling method (NLS) (top right) running at 3.95 ms, our neural direct illumination (Neural DI) (bottom left) at 4.41 ms, and ground truth (bottom right). Compared to screen-space ReSTIR, our neural light sampling produces less noise (especially in occluded areas), reducing the FLIP error metric [Andersson et al. 2021]. Additionally, we can use the neural network to compute approximate direct lighting without the necessity to cast shadow rays, except for the rays needed for neural network training.

Abstract

Direct illumination with many lights is an inherent component of physically-based rendering that remains challenging, especially in real-time scenarios. We propose an online-trained

neural cache that stores visibility between lights and 3D positions. We feed light visibility to weighted reservoir sampling (WRS) [Chao 1982; Wyman 2021] to sample a light source. The cache is implemented as a fully-fused multilayer perceptron (MLP) [Müller et al. 2021] with multi-resolution hash-grid encoding [Müller et al. 2022], enabling online training and efficient inference on modern GPUs in real-time frame rates. The cache can be seamlessly integrated into existing rendering frameworks and can be used in combination with other real-time techniques such as spatiotemporal reservoir sampling (ReSTIR) [Bitterli et al. 2020].

1. Introduction

Direct illumination has a significant impact on both the quality of rendered images and the rendering performance. The reflected radiance due to direct illumination at point $\mathbf{x}$ in direction ω_o can be described as an integral over all light emitting surfaces A:

$$L(\mathbf{x},\omega_o) = \int_A f_r(\mathbf{x},\omega_{\mathbf{x}\to\mathbf{y}},\omega_o) L_e(\mathbf{x},\omega_{\mathbf{y}\to\mathbf{x}}) G(\mathbf{x},\mathbf{y}) V(\mathbf{x},\mathbf{y}) \mathrm{d}A(\mathbf{y}), \tag{1}$$

where f_r is the bidirectional reflectance function (BRDF), L_e is the emitted radiance, $\omega_{\mathbf{x}\to\mathbf{y}}$ is a unit direction pointing from point $\mathbf{x}$ to $\mathbf{y}$, G is the geometry term including cosine terms and the squared distance, and V is the visibility function indicating binary visibility between two points.

We solve Equation (1) by means of Monte Carlo integration as there is no general analytic solution. As with any method based on Monte Carlo integration, the challenging part is to find a probability density function that closely matches the desired distribution. To tackle this problem, we train a neural network to provide estimates of visibility between light sources and 3D positions that we use to guide the sampling process.

We utilize weighted reservoir sampling (WRS) [Chao 1982; Wyman 2021] to sample a light source based on the light visibility estimated by the neural network and BRDF contribution to the shaded point, providing an unbiased sampling mechanism (see Section 3). The network architecture is based on a fully-fused multilayer perceptron (MLP) [Müller et al. 2021], which allows for efficient online training and inference on contemporary GPUs in real-time frame rates. We employ multi-resolution hash-grid encoding [Müller et al. 2022] to learn high-frequency details in a lower-dimensional space. The proposed method can be easily integrated into existing real-time rendering pipelines. For instance, it can be used in next event estimation for reflected bounces in path tracing, or for spatiotemporal reservoir sampling (ReSTIR) [Bitterli et al. 2020] to sample initial candidates or to recover after abrupt visibility changes that might otherwise cause significant noise. Our neural representation of visibility works with either individual lights directly or their clusters, to support scenes with an arbitrary number of lights.

2. Previous Work

Global illumination algorithms are notoriously known for their high computational demands. Therefore, a vast body of acceleration techniques has been proposed. Among these, various caching strategies play an important role, including irradiance caching [Ward 1994], which was later generalized to radiance caching [Krivanek et al. 2005]. These approaches are generally biased due to interpolation. To avoid bias, the idea is to reuse cached information only to guide the sampling process to better match the target distribution. Thus, numerous path guiding algorithms have been proposed, employing algorithms such as hierarchical data structures [Müller et al. 2017; Tokuyoshi et al. 2024], online-trained parametric mixture models [Vorba et al. 2014], neural networks [Müller et al. 2019], or a combination of these [Huang et al. 2024].

With the advent of many-light rendering [Dachsbacher et al. 2014], the problem of global illumination reduces to direct lighting with a large number of virtual point lights, necessitating efficient light sampling techniques. Traditional solutions are either based on arranging light sources into a hierarchical data structure [Walter et al. 2005; Conty Estevez and Kulla 2018; Moreau et al. 2022], sampling the light transport matrix [Hašan et al. 2007], visibility-aware reinforcement learning [Pantaleoni 2019], or Bayesian inference [Vévoda et al. 2018]. Guo et al. [2020] presented a method for caching visibility between voxels, significantly reducing the number of precise visibility tests required. Most of the aforementioned methods are designed for offline rendering; the overhead is prohibitively expensive for real-time applications. Li et al. [2024] presented an online-trained hierarchical light cache for sampling a very large number of lights in an unbiased manner in a production renderer.

With hardware acceleration of deep learning and ray tracing, physically-based rendering and neural-based approaches have become more compelling for real-time applications. Spatiotemporal reservoir sampling (ReSTIR) [Bitterli et al. 2020] became the de facto standard for sampling direct lighting in real-time ray tracing, forgoing building any complex data structures and exploiting spatial and temporal correlation to efficiently process a very large number of lights. Several neural-based approaches have been recently proposed for real-time scenarios: a neural radiance cache [Müller et al. 2021], neural shadow mapping [Datta et al. 2022], a neural light grid for precomputed indirect lighting [Iwanicki et al. 2024], and a neural-based rendering framework employing an attention mechanism to solve the many-light problem [Ren et al. 2024]. Concurrently to our work, Dereviannykh et al. [2025] suggest to use neural incidence radiance caching in combination with two-level Monte Carlo integration to achieve unbiased estimates. The authors also proposed to cache visibility of the environment map lighting as a special case. A neural importance sampling of many lights [Figueiredo et al. 2025] combines a light hierarchy with a neural network to predict light selection distribution directly, based on reflected radiance.

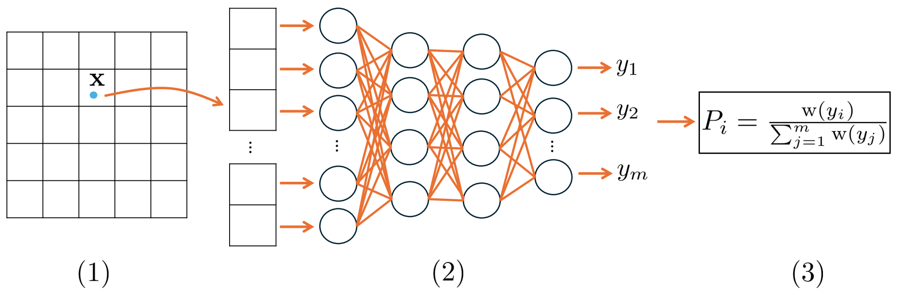

Figure 2. Overview of our method. We use a multi-resolution hash-grid encoding [Müller et al. 2022] to encode a 3D position (1), which is fed to a neural network (2). The output of the network is plugged into weighted reservoir sampling (WRS) [Chao 1982; Wyman 2021] (3).

3. Neural Visibility Cache

3.1. Algorithm Outline

In this section, we describe our *neural visibility cache* (NVC) and how to use it for light sampling. We train a multilayer perceptron (MLP) to predict (nonbinary) visibility between any point and any light source (or a cluster of light sources) in the scene. The nonbinary visibility accounts for soft shadows cast by area lights and semitransparent surfaces. Given a 3D position in the scene, we first use the multi-resolution hash-grid encoding [Müller et al. 2022] to encode the position, which we subsequently feed into the MLP, which outputs an estimated visibility for each light source; each neuron of the output layer corresponds to a single light. Finally, we use weighted reservoir sampling (WRS) [Chao 1982; Wyman 2021] to sample a light source using the visibility estimates provided by the MLP (see Figure 2). To ensure our method remains unbiased, we clamp zero and possibly negative values to a small positive constant (see Figure 3). This introduces a slight amount of noise while avoiding bias.

We approximate the product of BRDF and the cosine term (see Equation (1)) by linearly transformed cosines (LTC) [Heitz et al. 2016]; we then multiply this by the light radiance and visibility predicted by the neural network to calculate the weights of the lights contributing to the shaded points for WRS. Thus, the probabilities of selecting each sample account for both visibility and BRDF, reducing noise in shadowed areas and penumbras as well. In contrast to methods based on estimating radiance [Figueiredo et al. 2025], we only estimate visibility and calculate exact reflected radiance analytically. This leads to faster training of the network, requiring less training iterations to converge.

Our algorithm can be used as a standalone or as a generator of initial candidates for ReSTIR (see Section 4.2). This increases convergence speed and reduces noise in disoccluded pixels where ReSTIR can struggle to find meaningful initial light sam-

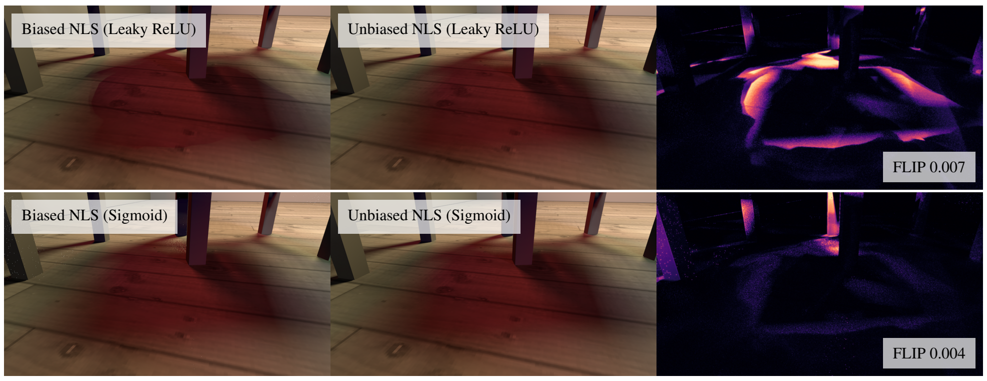

Figure 3. Our method can produce a biased result (left column) that typically exhibits as a hard boundary around heavily shadowed areas or as fireflies. The top row shows neural light sampling (NLS) with leaky rectified linear unit (ReLU) as the activation function for the output layer, which accentuates hard shadow boundary artifacts. The bottom row uses sigmoid for output activation, which has more fireflies instead. Clamping the visibility to 0.001 (center column) yields an unbiased result at the cost of slightly higher variance, alleviating these artifacts and converging to the ground truth. Images have been accumulated for 16K frames.

ples. ReSTIR implementations typically cast a shadow ray for a selected initial candidate and invalidate it if it is occluded. When generating initial candidates using our method, this is unnecessary as we already take visibility into account for all candidates.

To solve a many-lights problem, we employ a clustered approach, where each output neuron represents the average visibility of a light cluster, instead of an individual light source. This way, we can support an arbitrary number of lights, sorted into a fixed number of clusters. Sampling then becomes a two-step process, where a cluster of lights is selected first, and then a light sample within the cluster.

3.2. Neural Direct Illumination

Since the light weights used for WRS are based on LTC shading and visibility, we can also use them directly as an approximate direct illumination (see Figure 4). This yields illumination with approximate shadows, which is biased but very fast and noise-free, without casting any shadow rays. We call this *neural direct illumination* (Neural DI), which can be used as an approximation of direct illumination for deeper bounces in path tracing or for a fast preview. Note that this is only applicable to the case when output neurons represent individual lights, not clusters.

3.3. Neural Network Architecture

We use a multilayer perceptron (MLP) with two hidden layers, each containing 32 neurons. The activation function for hidden layers is the leaky rectified linear unit

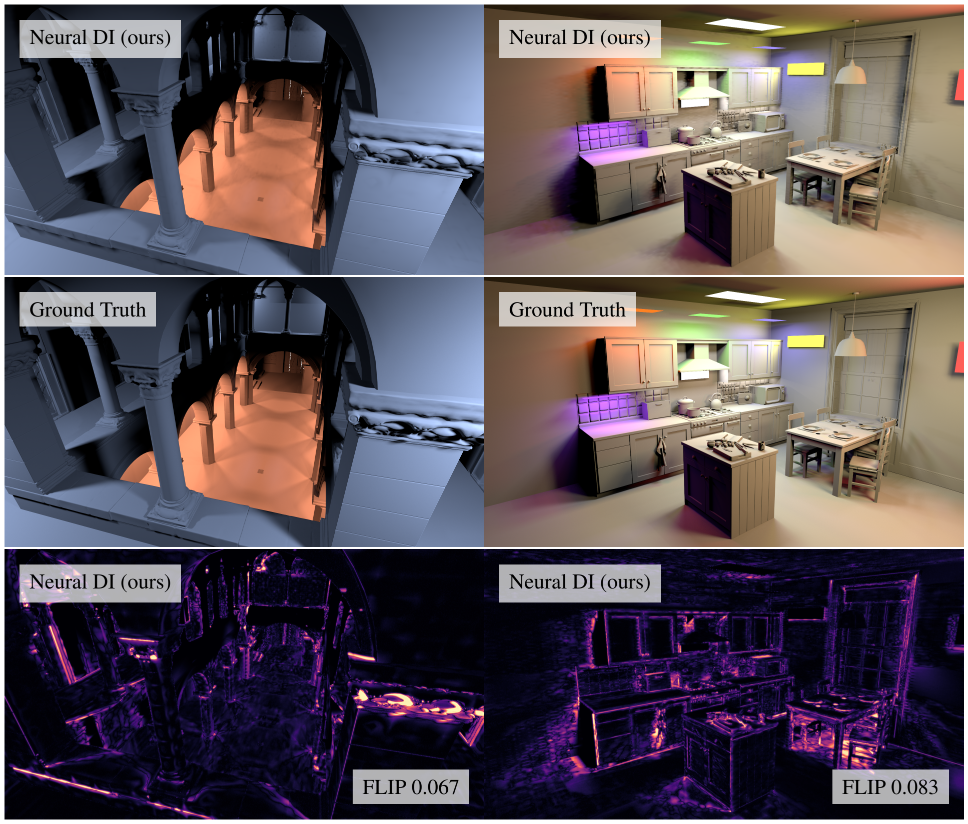

Figure 4. Neural DI produces noise-free images using only one sample per pixel: Sponza (left) and kitchen (right) scenes using 32 lights and one sample per pixel. To accentuate shadows, we replaced textured materials with gray diffuse material. Notice that our method can learn penumbras from noisy training data.

(ReLU) with a slope of 0.01. We have tested several activation functions and found that leaky ReLU achieves the lowest training loss. For the output layer, we use the sigmoid activation function. Sigmoid not only maps the output to the range of valid visibility values $[0,1]$, but it also reduces the training loss faster than leaky ReLU. The neural network training with backpropagation uses the L2 loss function. For the multi-resolution hash-grid encoding, we use ten levels with the base resolution 16 and four features per level. This setup leads to approximately 562K learned parameters represented using 32-bit floats. Notice that the majority of these parameters correspond to the hash-grid weights, with only a marginal number dedicated to the MLP. The hash encoding is a critical component in achieving high quality and fast training with our method. For scenes with a low light count, the number of features per level can be decreased.

3.4. Training

We use the He initialization strategy [He et al. 2015] to initialize the MLP. For training, we use the Adam optimizer [Kingma and Ba 2014] with a variable base learning rate. We start with a learning rate of 0.05 and we linearly lower it for the first 200 training steps down to 0.001. This significantly speeds up training at the beginning, which converges to a stable state faster. We perform one training step (one epoch with one batch) per frame. Our training examples consist of a random position in the scene as the input to the network and the corresponding visibility of each light as the target for the network output.

We have also tried training the network on the shadowed radiance of lights (light intensity attenuated by the squared distance to the light multiplied by visibility). For complex scenes, most of the training samples were close to zero due to strong distance attenuation, and the network had a tendency to predict extremely low values everywhere. Therefore, we settled on training the network only on visibility values. Each training sample comprises mutual visibility for one random sample on a light and a given point. The visibility in this case is binary, but for area lights, the neural network will eventually learn the average visibility over the whole area of the light (penumbra), which is not binary.

There are several options for generating these training examples. Random points on surfaces visible from the camera (screen space) achieve best result for a given camera view, but the network adapts slowly for new views. Using random points within scene bounds (world space) needs no adaptation for new views, but introduces dark blob artifacts (see Figure 5). A third option, generating samples on the geometry, does not train the network to predict visibility in empty areas, which might become occupied in future frames due to animation. Additionally, it makes it impossible to use our method for light sampling within volumes of participating media, so we do not recommend it. We found that using a combination of world-space data and screen-space data works best as it fixes the artifacts but also adapts rapidly to camera

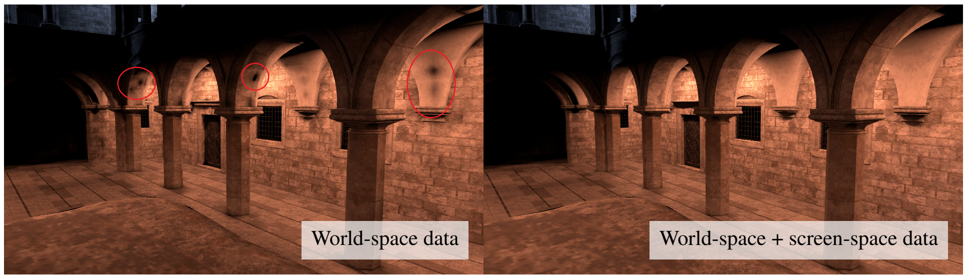

Figure 5. Using world-space data for training causes artifacts that manifest as dark blobs, highlighted in red circles (left). Introducing screen-space data for training fixes the problem (right). Images show our biased Neural DI method.

movement. Our solution uses a combination of 4096 world-space samples and another 4096 screen-space samples.

We cast a shadow ray toward each of the light sources from each training point to produce the training example. This is very fast in practice, since we only use 8196 training points per frame, and the number of lights is limited to the number of output neurons (32 in our tests).

3.5. Dynamic Scenes

Our method is online-trained, therefore it supports dynamic scenes including animated geometry, lights, and camera. When the scene changes, the network state might not approximate the visibility well until it adapts to the new situation. As our method is unbiased, this exhibits as an increase in variance (noise) but not bias. When training from scratch (see Figure 6), the training needs only about 16 frames to reduce the FLIP error to almost a half. Smooth changes to the scene can be expected to adapt faster than 16 frames, but the application may need to increase the learning rate temporarily, or perform more than one training step per frame. Abrupt changes, such as teleportation of camera, or dynamic geometry are handled by using world-space samples that pre-train the network to be used on unseen views and previously unoccupied space.

3.6. Clustering

The method described so far, representing visibility of each light in the scene with a dedicated output neuron, limits the number of supported lights to the size of the output layer allowed by the selected network architecture and target hardware (32 in our implementation). In this section, we introduce a *clustered neural visibility cache* approach (Clustered NVC), where instead of representing individual lights, the output neurons represent the visibility of a cluster, which can consist of an arbitrary number of lights.

We use the k-means algorithm [MacQueen 1967; Lloyd 1982] to cluster the lights into k clusters (we use $k = 32$). During training, we randomly select a light source within each cluster to train the network to predict average visibility of the cluster for any point in the scene. This approach works especially well for interior scenes with many rooms, as we can quickly cull light clusters not contributing to the room with the camera. Alternatively, lights can be clustered based on the mesh they belong to (e.g., clustering emissive triangles of a complex mesh representing a lamp or a neon sign).

To sample a light using our neural visibility cache with clusters, we employ a two-step process based on WRS and resampled importance sampling (RIS) [Talbot et al. 2005] implemented with reservoirs [Bitterli et al. 2020]. The first step uses WRS to sample a cluster y out of m clusters, based on the inferred average cluster visibility at that point, resulting in a reservoir with the selected sample y, its sampling weight

$\mathrm{w}(y)$, and the sum of their sampling weights $\mathrm{w}_{\mathrm{sum}} = \sum_{j=1}^{m} \mathrm{w}(y_j)$. The second step uses a streaming RIS [Bitterli et al. 2020] to generate a final light sample x selected from the pool of m_y lights within the cluster y. The source probability density function $p(x)$ for RIS is defined as $p(x) = m\frac{\mathrm{w}(y)}{\mathrm{w}_{\mathrm{sum}}}\frac{1}{m_y}$, where $m\frac{\mathrm{w}(y)}{\mathrm{w}_{\mathrm{sum}}}$ is a reciprocal weight of the reservoir from the first step and $\frac{1}{m_y}$ is the probability of sampling a light uniformly within the cluster. The target probability density function $\hat{p}(x)$ for RIS is selected based on the BRDF contribution and LTC lighting, same as before.

We found out that more training data is needed to provide plausible results for scenes with many lights. Therefore, we use 49,152 training examples for the Subway scene with 59K lights. We have also found out that the optimal hash-grid settings are different for the clustered approach, since the approximation we are trying to learn is lower frequency in nature, and we use eight levels with the base resolution 2 for this case.

4. Results and Discussion

We implemented the proposed method in DirectX 12 and HLSL. Rendering each frame consists of the following five passes: (1) G-buffer generation, (2) training data generation, (3) network training, (4) inference and light sampling, and (5) shading. The neural network is implemented as a fully fused MLP [Müller et al. 2021], allowing us to train and infer the neural network entirely in the on-chip shared memory. Our implementation allows to execute inference inline in the scope of DirectX Raytracing (DXR) ray tracing shaders, enabling to use our method on every bounce of light while tracing paths. All tests were executed on AMD Radeon RX 7900 XT.

4.1. Comparison to Screen-Space ReSTIR

As a reference method, we implemented screen-space ReSTIR using eight initial candidates, casting a shadow ray for the selected initial candidate. We use temporal resampling where we clamp the contribution of the previous reservoir to $20\times$ the contribution of the new reservoir. Spatial resampling uses a radius of 32 pixels. Unless stated otherwise, we use one sample per pixel for all tests, and both ReSTIR and the neural network are converged for 1024 frames. Note that the FLIP error of both methods is already significantly reduced after 20 to 30 frames (see Figure 6).

Compared to screen-space ReSTIR, our neural light sampling achieves a lower FLIP error [Andersson et al. 2021] at a similar time budget, especially in the occluded regions (see Figures 1 and 7). FLIP error is reduced by about 20% for the Kitchen scene and about 45% for the Sponza scene. We can make the method unbiased by clamping the output of the neural network (discussed in Section 3.1) at the cost of slightly increased variance (see Figure 3). Neural DI that directly uses the visibility estimates provided by the neural network (see Section 3.2) is biased by definition,

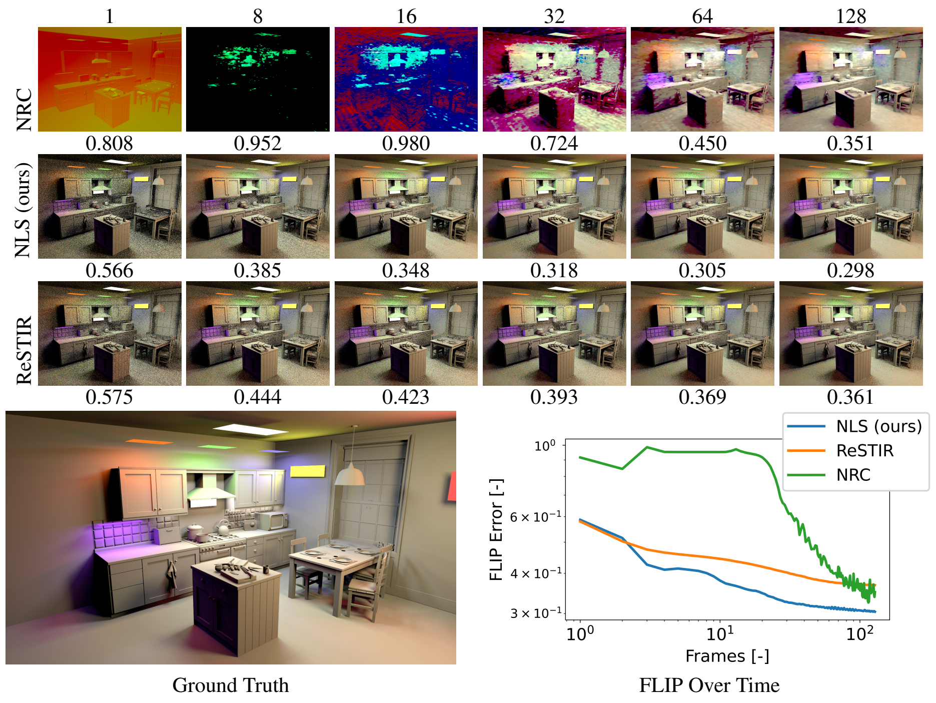

Figure 6. A comparison of NRC (top), our neural light sampling (NLS) (middle) and screen space ReSTIR (bottom) on a progression of 128 frames. The number under each image represents its FLIP error, which decreases over time for all methods, but NRC suffers from color artifacts, especially at the beginning of training. Our method remains unbiased even at the beginning while the network is not sufficiently trained, exhibiting as increased variance, similarly to ReSTIR. Our method reduces the FLIP error faster than other methods, achieving lower error than ReSTIR even only after three training steps.

yet it provides significantly lower error than ReSTIR and neural light sampling (see Figure 1). The bias exhibits at the edges and in heavily shadowed areas (see Figure 4).

4.2. Combining ReSTIR with Clustered NVC

To generate initial candidates, standard ReSTIR uses a RIS loop with a user-selected number of candidates (we use eight), which generates a reservoir that can participate in the streaming RIS and for spatial and temporal reuse. We combine our neural approach with ReSTIR by replacing the initial candidates generation routine with our clustered NVC. Our method also generates a reservoir, using a two-step process described in Section 3.6, therefore it can be directly used to generate initial candidates.

Though using NVC for initial candidates improves the quality of ReSTIR overall, the biggest benefit is for boosting the convergence rate for disocclusions. Figure 8 shows a significant noise reduction when NVC is used for initial candidates after

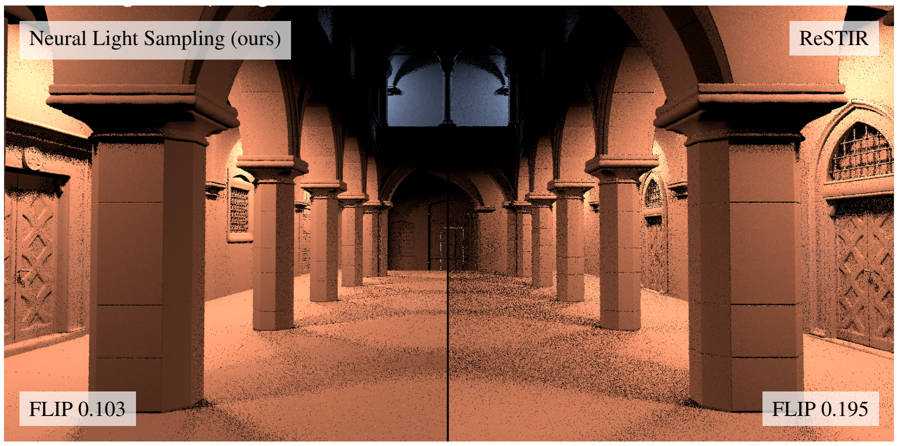

Figure 7. Comparison of neural light sampling (left) to screen-space ReSTIR (right) on the Sponza scene with 32 lights. Neural light sampling produces lower error at the same performance cost.

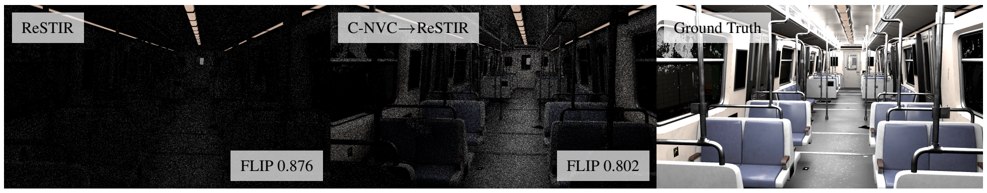

Figure 8. A comparison of standard ReSTIR (left), ReSTIR with initial candidates generated by our clustered NVC (center), and the ground truth (right) in an idealized case where disocclusions happen in every pixel. The Subway scene contains 59K lights. We simulate disocclusions by invalidating motion vectors for all pixels, using one sample per pixel. This demonstrates how clustered NVC can help ReSTIR to recover after disocclusion.

a disocclusion event. Running NVC for ReSTIR once it converges only improves ReSTIR quality insignificantly; the average FLIP reduction is about 5% (see Figure 9, while introducing an overhead of running the inference for every pixel. Therefore, we recommend running NVC only for disoccluded pixels to boost the quality of initial candidates. This implementation leads to runtime performance being only 5% lower compared to standard ReSTIR, on a test using a dynamic camera flying around the scene for 1024 frames. Results of clustered NVC in comparison to other methods are summarized in Figure 9.

4.3. Comparison to Neural Radiance Cache

For a direct comparison of our method to the neural radiance cache (NRC) [Müller et al. 2021], we have implemented a version of NRC that caches only direct illumination from all lights on a primary hit (NRC DI). We use the same network architecture and training data procedure as we do for our method, except the output layer has three

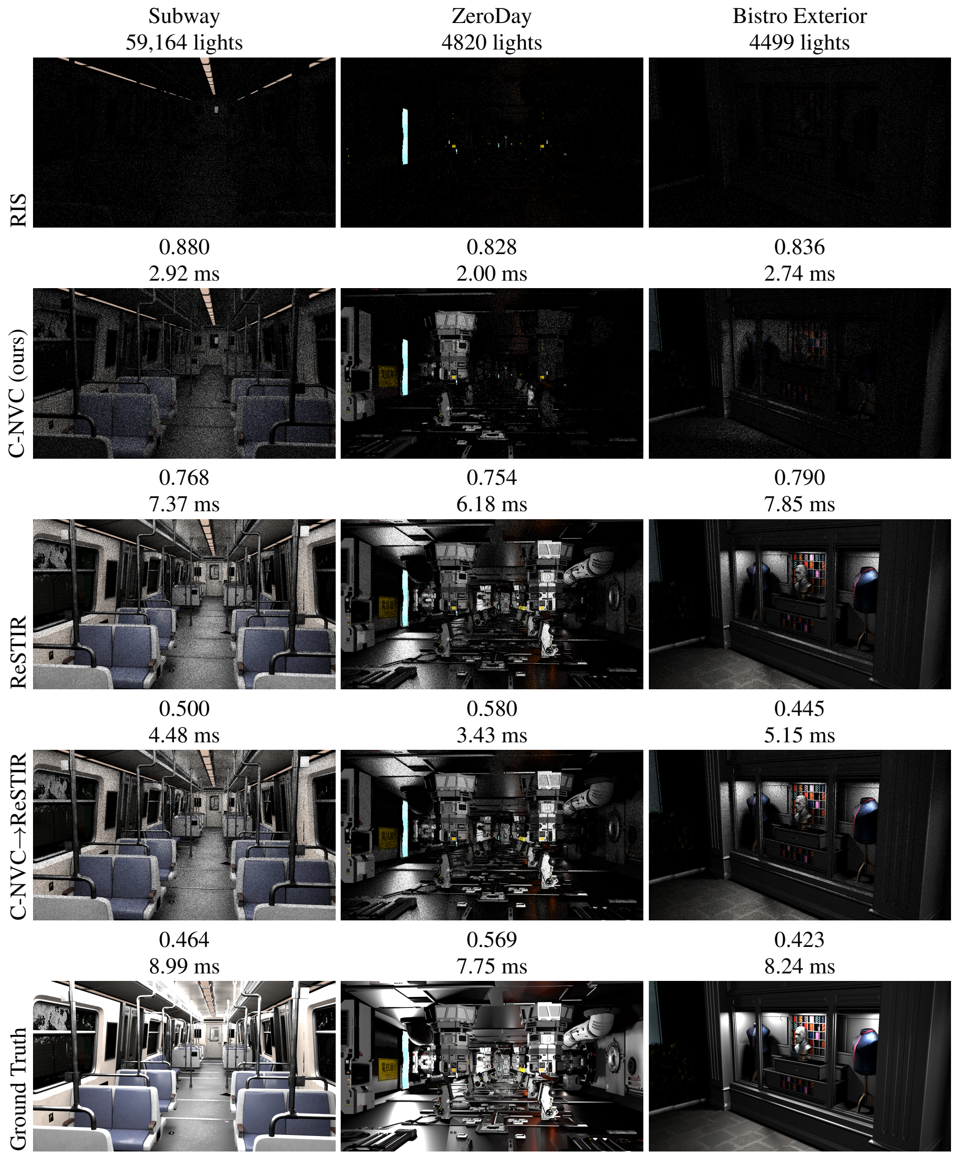

Figure 9. Comparison of tested methods on scenes with many lights, captured after 1024 frames. The numbers under each image correspond to the FLIP error against the ground truth and a total frame time. Note that Cluster NVC is more expensive than NVC compared to ReSTIR, due to 6$\times$ more training data needed and a more complex sampling process. For scenes with up to 32 lights, the performance of NVC is comparable to ReSTIR. In these tests, we ran Clustered NVC (C-NVC) to generate initial ReSTIR candidates for every pixel, instead of only on disocclusions as we recommend.

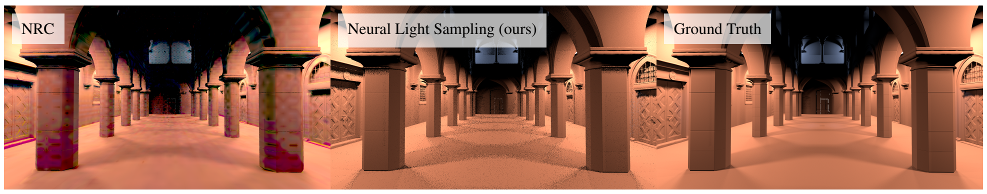

Figure 10. NRC approximating a product of radiance and visibility exhibits blurry artifacts and color shifts (left). Our method computes exact radiance analytically and samples the light sources stochastically for unbiased rendering (middle). The ground truth was accumulated with many frames (right).

neurons for RGB radiance values, and we use the leaky ReLU activation function to allow for unbounded values. NRC is not limited by the number of lights, but it has several drawbacks compared to our method. For optimal results, NRC implementation requires a larger MLP, which increases the training and inference time. Another disadvantage is that NRC requires more training steps to achieve usable results (see Figure 6). At the beginning of training, NRC produces random colors, introducing significant error to the resulting image. NRC predicts a product of the wavelength-dependent radiance and visibility, manifesting blurry artifacts and color shifts (see Figure 10). Our method instead calculates the radiance analytically with LTC. The predicted visibility is used to guide the light sampling process, which remains unbiased, even when the network training has not yet converged. In this case, we get an increase in variance instead of bias. Compared to NRC, our method is limited to direct illumination and a smaller number of lights, but for this purpose it achieves unbiased results with a smaller network and does not suffer from artifacts due to under-training and radiance approximation.

4.4. Performance

The performance of all tested methods is summarized in Table 1. Training data generation with our default configuration (8196 training examples and 32 lights) takes 0.34 ms for the Kitchen scene and 0.43 ms for the Sponza scene. The neural network training step (backpropagation and optimization) takes ~ 0.75 ms for both scenes. When training data generation and the training itself are implemented as separate steps, where the first step writes out the data into memory for the second step to consume, it is possible to perform multiple training iterations over the same data. As we only perform one step per frame, we can fuse data generation and training together, achieving about 5% speedup. We run inference once per pixel at ~ 1.8 ms per frame for 1920×1080 resolution with 32 lights. The inference itself takes 1.32 ms, while the remaining time is spent on the WRS algorithm. For comparison, replacing the inference call of our NVC by casting a shadow ray toward each of the light sources for every pixel on screen takes about 7 ms for the Kitchen scene with 32 lights (more than $3\times$ higher).

	G-Buffer [ms]	Training (fused) [ms]	Light Sampling [ms]	Shading [ms]	Frame Total [ms]
RIS					
Subway	1.09	-	0.62	1.16	2.92
ZeroDay	0.61	-	0.36	0.99	2.00
Bistro	0.78	-	0.31	1.61	2.74
Kitchen	0.58	-	0.47	0.57	1.65
Sponza	1.00	-	0.27	0.67	1.97
ReSTIR					
Subway	1.17	-	2.01	1.11	4.48
ZeroDay	0.64	-	1.67	0.92	3.43
Bistro	0.83	-	2.30	1.83	5.15
Kitchen	0.58	-	2.45	0.53	3.61
Sponza	1.00	-	2.23	0.64	3.95
NVC (ours)					
Kitchen	0.60	0.99	1.74	0.52	3.95
Sponza	0.99	1.18	1.85	0.61	4.69
Clustered NVC (ours)					
Subway	1.11	3.01	1.87	1.11	7.37
ZeroDay	0.63	2.63	1.70	0.95	6.18
Bistro	0.80	3.20	1.71	1.87	7.85
Clustered NVC→ReSTIR (ours)					
Subway	1.12	3.00	3.61	1.05	8.99
ZeroDay	0.64	2.58	3.43	0.90	7.75
Bistro	0.80	3.38	2.24	1.61	8.24

Table 1. Performance breakdown of tested methods. Scenes with 32 lights are tested on NVC and many-light scenes on Clustered NVC. Training data generation and network training has been fused into a single pass for better performance. The light sampling column contains time spent in either the resampling loop (RIS), inference and light selection process (NVC and Clustered NVC), all ReSTIR passes (ReSTIR), or ReSTIR passes with Clustered NVC as the initial samples generator (Clustered NVC→ReSTIR).

5. Conclusion and Future Work

We proposed a lightweight neural-based sampling method for real-time direct illumination based on caching the nonbinary visibility. With a minor modification, our method provides unbiased estimates with lower error than screen-space ReSTIR at a similar cost. Compared to ReSTIR, our method operates in the world space, making it more robust to visibility changes with which ReSTIR struggles. In fact, our neural light sampling could be used in combination with ReSTIR to sample the initial candidates. As a world-space method, it can also be used for direct illumination of volumes of participating media. We used a clustered approach to support an arbitrary number of lights with a fixed-size neural network. We also proposed a biased variant that directly uses the visibility estimates, decreasing variance even further at the cost of some bias. Thanks to continuous online training, our method adapts to the dynamic content including animated lights, geometry, and camera. Compared to neural importance sampling of many lights [Figueiredo et al. 2025], we are only caching visibility with a smaller MLP, achieving faster training and simpler implementation (we do not require other inputs, e.g., the surface normal to the MLP, only the position).

The limitation of our clustered NVC approach is that efficiency is highly dependent on the quality of the clusters created and the total number of lights. For future work, an interesting direction would be to explore other methods for light clustering, e.g., the ones based on hierarchies [Vévoda et al. 2018; Figueiredo et al. 2025] and light culling [Tokuyoshi and Harada 2016]. An early approach of Shirley et al. [1996] of sorting lights into sets of important and unimportant lights could be adapted to our approach by creating one cluster of unimportant lights to ensure unbiasedness and using the remaining available clusters to represent the important lights.

Another interesting research direction would be to cache mutual visibility between any two points in the scene. This would enable us to query the visibility of any light, not just the ones represented by output neurons. To make this practical, such a query would either have to be faster than a ray cast or have to return visibility for a batch of queries in a single inference call, to amortize the cost.

Finally, we need to overcome the limitation of the count of lights (or light clusters) due to the restrictions imposed on the size of the neural network. We can achieve linear scaling naïvely by having multiple networks and possibly time-slicing their training (only train one network per frame) to maintain a fixed training budget. Overcoming this limitation in a scalable way is a nontrivial task for the future.

Acknowledgements

We would like to thank Holger Grün and Carsten Benthin for useful discussions and their support. The models are courtesy of Crytek (Sponza), Sketchfab/Alex Murias (Subway), Blendswap/Jay-Artist (Country Kitchen), Mike Winkelmann (ZeroDay), and Amazon Lumberyard (Bistro).

References

ANDERSSON, P., NILSSON, J., AND AKENINE-MÖLLER, T. Visualizing and communicating errors in rendered images. In MARRS, A., SHIRLEY, P., AND WALD, I., editors, *Ray Tracing Gems II*, pages 301–320. APress, 2021. URL: https://doi.org/10.1007/978-1-4842-7185-8_19. 1, 9

BITTERLI, B., WYMAN, C., PHARR, M., SHIRLEY, P., LEFOHN, A., AND JAROSZ, W. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics*, 39(4):148:1–148:17, August 2020. URL: https://doi.org/10.1145/3386569.3392481. 2, 3, 8, 9

CHAO, M. T. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, December 1982. URL: https://doi.org/10.1093/biomet/69.3.653. 2, 4

CONTY ESTEVEZ, A. AND KULLA, C. Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(2):25:1–25:17, August 2018. URL: https://doi.org/10.1145/3233305. 3

DACHSBACHER, C., KŘIVÁNEK, J., HAŠAN, M., ARBREE, A., WALTER, B., AND NOVÁK, J. Scalable realistic rendering with many-light methods. *Computer Graphics Forum*, 33 (1):88–104, February 2014. URL: https://doi.org/10.1111/cgf.12256. 3

DATTA, S., NOWROUZEZAHRAI, D., SCHIED, C., AND DONG, Z. Neural shadow mapping. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, pages 8:1–8:9. Association for Computing Machinery, 2022. URL: https://doi.org/10.1145/3528233.3530700. 3

DEREVIANNYKH, M., KLEPIKOV, D., HANIKA, J., AND DACHSBACHER, C. Neural two-level Monte Carlo real-time rendering. *Computer Graphics Forum*, 44(2): e70050, 2025. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.70050. 3

FIGUEIREDO, P., HE, Q., BAKO, S., AND KALANTARI, N. K. Neural importance sampling of many lights. arXiv preprint arXiv:2505.11729, 2025. URL: https://arxiv.org/abs/2505.11729. 3, 4, 15

GUO, J. J., EISEMANN, M., AND EISEMANN, E. Next event estimation++: Visibility mapping for efficient light transport simulation. *Computer Graphics Forum*, 39(7):205–217, 2020. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14138. 3

HAŠAN, M., PELLACINI, F., AND BALA, K. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics*, 26(3):26–es, July 2007. URL: https://doi.org/10.1145/1276377.1276410. 3

HE, K., ZHANG, X., REN, S., AND SUN, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034. IEEE, 2015. URL: https://doi.org/10.1109/ICCV.2015.123. 7

HEITZ, E., DUPUY, J., HILL, S., AND NEUBELT, D. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics*, 35(4):41:1–41:8, July 2016. URL: https://doi.org/10.1145/2897824.2925895. 4

HUANG, J., IIZUKA, A., TANAKA, H., KOMURA, T., AND KITAMURA, Y. Online neural path guiding with normalized anisotropic spherical gaussians. *ACM Transactions on Graphics*, 43(3):26:1–26:18, April 2024. URL: https://doi.org/10.1145/3649310. 3

IWANICKI, M., SLOAN, P.-P., SILVENNOINEN, A., AND SHIRLEY, P. The neural light grid: A scalable production-ready learned irradiance volume. Presented at SIGGRAPH, Advances in Real-Time Rendering in Games, 2024. URL: https://research.activision.com/publications/2024/08/Neural_Light_Grid. 3

KINGMA, D. P. AND BA, J. Adam: A method for stochastic optimization, 2014. URL: https://api.semanticscholar.org/CorpusID:6628106. 7

KRIVANEK, J., GAUTRON, P., PATTANAIK, S., AND BOUATOUCH, K. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):550–561, 2005. URL: https://doi.org/10.1109/TVCG.2005.83. 3

LI, Y. K., ZHU, C., NICHOLS, G., KUTZ, P., HUANG, W.-F. W., ADLER, D., BURLEY, B., AND TEECE, D. Cache points for production-scale occlusion-aware many-lights sampling and volumetric. In *DigiPro '24: Proceedings of the Digital Production Symposium 2024*, pages 6:1–6:19. Association for Computing Machinery, July 2024. URL: https://doi.org/10.1145/3665320.3670993. 3

LLOYD, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. URL: https://doi.org/10.1109/TIT.1982.1056489. 8

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297. University of California Press, 1967. URL: https://api.semanticscholar.org/CorpusID:6278891. 8

MOREAU, P., PHARR, M., AND CLARBERG, P. Dynamic many-light sampling for real-time ray tracing. In *Proceedings of the Conference on High-Performance Graphics*, HPG '19, page 21–26. Eurographics Association, 2022. URL: https://doi.org/10.2312/hpg.20191191. 3

MÜLLER, T., MCWILLIAMS, B., ROUSSELLE, F., GROSS, M., AND NOVÁK, J. Neural importance sampling. *ACM Transactions on Graphics*, 38(5):145:1–145:19, October 2019. URL: https://doi.org/10.1145/3341156. 3

MÜLLER, T., ROUSSELLE, F., NOVÁK, J., AND KELLER, A. Real-time neural radiance caching for path tracing. *ACM Transactions on Graphics*, 40(4), July 2021. URL: https://doi.org/10.1145/3450626.3459812. 2, 3, 9, 11

MÜLLER, T., EVANS, A., SCHIED, C., AND KELLER, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, July 2022. URL: https://doi.org/10.1145/3528223.3530127. 2, 4

MÜLLER, T., GROSS, M., AND NOVÁK, J. Practical path guiding for efficient light-transport simulation. *Computer Graphics Forum*, 36(4):91–100, 2017. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13227. 3

PANTALEONI, J. Importance Sampling of Many Lights with Reinforcement Lightcuts Learning. arXiv preprint arXiv:1911.10217, 2019. URL: https://arxiv.org/abs/1911.10217. 3

REN, H., HUO, Y., PENG, Y., SHENG, H., XUE, W., HUANG, H., LAN, J., WANG, R., AND BAO, H. LightFormer: Light-oriented global neural rendering in dynamic scene. *ACM Transactions on Graphics*, 43(4):75:1–75:14, July 2024. URL: https://doi.org/10.1145/3658229. 3

SHIRLEY, P., WANG, C., AND ZIMMERMAN, K. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, January 1996. URL: https://doi.org/10.1145/226150.226151. 15

TALBOT, J. F., CLINE, D., AND EGBERT, P. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, page 139–146. Eurographics Association, 2005. URL: https://doi.org/10.2312/EGWR/EGSR05/139-146. 8

TOKUYOSHI, Y. AND HARADA, T. Stochastic light culling. *Journal of Computer Graphics Techniques (JCGT)*, 5(1):35–60, March 2016. URL: http://jcgt.org/published/0005/01/02/. 15

TOKUYOSHI, Y., IKEDA, S., KULKARNI, P., AND HARADA, T. Hierarchical light sampling with accurate spherical gaussian lighting. In *SIGGRAPH Asia 2024 Conference Papers*, pages 82:1–82:11. Association for Computing Machinery, 2024. URL: https://doi.org/10.1145/3680528.3687647. 3

VÉVODA, P., KONDAPANENI, I., AND KŘIVÁNEK, J. Bayesian online regression for adaptive direct illumination sampling. *ACM Transactions on Graphics*, 37(4):125:1–125:12, July 2018. URL: https://doi.org/10.1145/3197517.3201340. 3, 15

VORBA, J., KARLÍK, O., ŠIK, M., RITSCHEL, T., AND KŘIVÁNEK, J. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics*, 33(4):101:1–101:11, July 2014. URL: https://doi.org/10.1145/2601097.2601203. 3

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. Lightcuts: A scalable approach to illumination. *ACM Transactions on Graphics*, 24(3):1098–1107, July 2005. URL: https://doi.org/10.1145/1073204.1073318. 3

WARD, G. J. Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics: Proceedings of the Second Eurographics Workshop on Rendering*, pages 11–20. Springer, 1994. URL: https://doi.org/10.1007/978-3-642-57963-9_2. 3

WYMAN, C. Weighted reservoir sampling: Randomly sampling streams. In MARRS, A., SHIRLEY, P., AND WALD, I., editors, *Ray Tracing Gems II*, pages 345–349. APress, 2021. URL: https://doi.org/10.1007/978-1-4842-7185-8_22. 2, 4

Index of Supplemental Materials

- Video of Sponza walkthrough
  jcgt.org/published/0014/02/01/NVC_Sponza_Walkthrough.mp4
- Video of Sponza with dynamic lighting
  jcgt.org/published/0014/02/01/NVC_Sponza_Dynamic_Light.avi

Author Contact Information

Jakub Bokšanský
Advanced Micro Devices GmbH
Einsteinring 22-28
85609 Aschheim
Munich, Germany
jakub.boksansky@amd.com
https://boksajak.github.io

Daniel Meister
AMD Japan Co. Ltd.
Marunouchi Trust Tower
1-8-3 Marunouchi, Chiyoda-ku
Tokyo, Japan
daniel.meister@amd.com
https://meistdan.github.io

Jakub Bokšanský and Daniel Meister, Neural Visibility Cache for Real-Time Light Sampling, *Journal of Computer Graphics Techniques (JCGT)*, vol. 14, no. 2, 1–19, 2025
http://jcgt.org/published/0014/02/01/

Received:	2025-06-04		
Recommended:	2025-07-02	Corresponding Editor:	Brent Burley
Published:	2025-08-18	Editor-in-Chief:	Alexander Wilkie