Neural Visibility Cache for Real-Time Light Sampling

Jakub Bokšanský and Daniel Meister

Advanced Micro Devices, Inc.

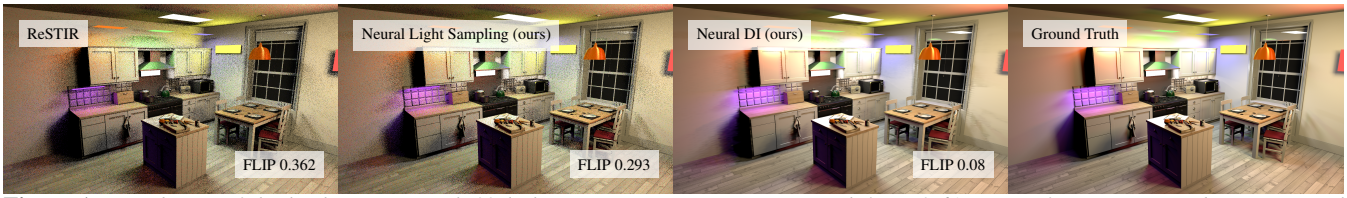

Figure 1: *Rendering of the kitchen scene with 32 lights using screen-space ReSTIR (left) at 3.61 ms per frame, compared to our neural light sampling method (center left) running at 3.95 ms, our neural direct illumination (Neural DI) (center right) at 4.41 ms, and ground truth (right). Compared to screen-space ReSTIR, our neural light sampling produces less noise (especially in occluded areas), reducing the FLIP error metric [ANA21]. 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, remaining 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) [Cha82, Wym21] to sample a light source. The cache is implemented as a fully-fused multilayer perceptron (MLP) [MRNK21] with multi-resolution hash-grid encoding [MESK22], 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) [BWP*20].*

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 bidirectional reflectance function (BRDF), L_e is 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 the equation above 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) [Cha82, Wym21] 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) [MRNK21], which allows for efficient online training and inference on contemporary GPUs in real-time frame rates. We employ multi-resolution hash-grid encoding [MESK22] 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, the proposed method can be used in next event estimation for reflected bounces in path tracing, or for spatiotemporal reservoir sampling (ReSTIR) [BWP*20] to sample initial candidates or to recover after abrupt visibility changes that might otherwise cause significant noise. Our neural representation of visibility works either with 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 [War94], which was later generalized to radiance caching [KGPB05]. 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 [MGN17, TIKH24], parametric mixture models [VKŠ*14], neural networks [MMR*19], or a combination of these [HIT*24].

With the advent of many-light rendering [DKH*14], 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 [WFA*05, CEK18, MPC19], sampling the light transport matrix [HPB07], or Bayesian inference [VKK18]. Guo et al. [GEE20] presented a method for caching visibility between voxels significantly reducing 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.

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) [BWP*20] 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 [MRNK21], neural shadow mapping [DNSD22], neural light grid for precomputed indirect lighting [ISSS24], and a neural-based rendering framework employing an attention mechanism to solve the many-light problem [RHP*24]. Concurrently to our work, Dereviannykh et al. [DKHD24] proposed 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 [FHBK25] combined a light hierarchy with neural network to predict light selection distribution directly, based on reflected radiance.

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 (non-binary) visibility between any point and any light source (or a cluster of light sources) in the scene. The non-binary visibility accounts for soft shadows cast by area lights and semi-transparent surfaces. Given a 3D position in the scene, we first use the multi-resolution hash-grid encoding [MESK22] to encode the position, that 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) [Cha82, Wym21] 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 5). This introduces a slight amount of noise while avoiding bias.

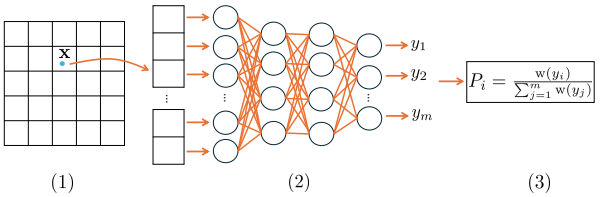

Figure 2: *Overview of our method. We use the multi-resolution hash-grid encoding [MESK22] to encode a 3D position (1), which is fed to a neural network (2). The output of the network is plugged into the weighted reservoir sampling (WRS) [Cha82, Wym21] (3).*

We approximate the product of BRDF and the cosine term (see Equation 1) by linearly transformed cosines (LTC) [HDHN16]; we then multiply this by the light radiance and visibility predicted by the neural network to calculate the weights of lights contributing to 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 [FHBK25], 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 4.1.1). This increases convergence speed and reduces noise in disoccluded pixels where ReSTIR can struggle to find meaningful initial light samples. 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 arbitrary number of lights, sorted into 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.

3.1.1. Neural Direct Illumination

Since 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 3). 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.

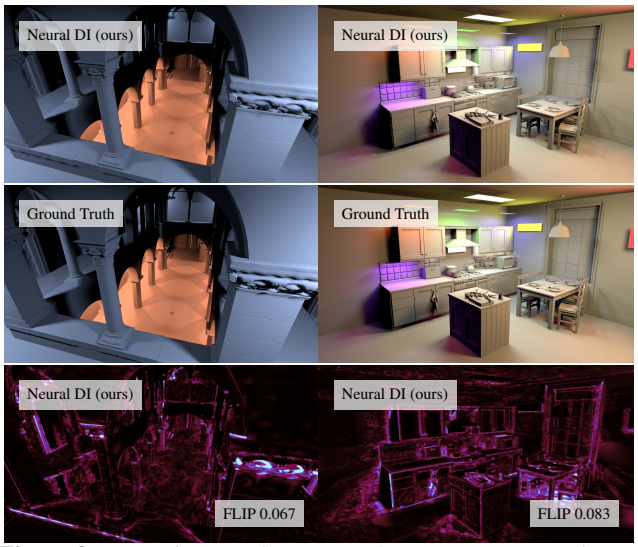

Figure 3: *Neural DI produces noise-free images using only one sample per pixel: Sponza and kitchen scenes using 32 lights and one sample per pixel. To accentuate shadows, we replaced textured materials with gray diffuse material. Notice that our method can learn penumbras from noisy training data.*

Note that this is only applicable to the case when output neurons represent individual lights, not clusters.

3.2. Neural Network Architecture

We use a multilayer perceptron (MLP) with 2 hidden layers, each containing 32 neurons. The activation function for hidden layers is the leaky rectified linear unit (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 10 levels with the base resolution 16 and 4 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 of our method. For scenes with a low light count, the number of features per level can be decreased.

3.3. Training

We use the He initialization strategy [HZRS15] to initialize the MLP. For training, we use the Adam optimizer [KB14] 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 per frame. Our training examples consist of a random position in the scene as the input to the network and the visibility of each light at that input point as the target for the network output.

Figure 4: *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).*

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). This only worked for simple scenes, while for scenes with complex occlusion, most of the training samples were close to zero due to strong distance attenuation. As a result, the network had a tendency to predict extremely low values everywhere. Therefore, we settled on training the network on visibility values between 0 and 1. 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 within the scene in world-space, random points on surfaces visible from the camera (screen-space) and random points distributed over the scene geometry. Using screen-space data achieves the best result for a given camera view, but the network adapts slower for new views. Using only world-space data needs no adaptation for new views, but introduces dark blob artifacts (see Figure 4). Generating data on the geometry does not train the network to predict visibility in empty areas, which might become occupied in the future frames due to animation. Additionally, it makes it impossible to use our method for light sampling within volumes of participating media. We found that using a combination of world-space data and screen-space data works best as it fixes the artifacts but also adapts very fast to camera movement. Our solution uses a combination of 4096 world-space samples distributed randomly across the scene and another 4096 samples taken randomly on surfaces visible from the camera.

We cast a shadow ray towards 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 for our test scenes).

3.4. Clustering

A 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

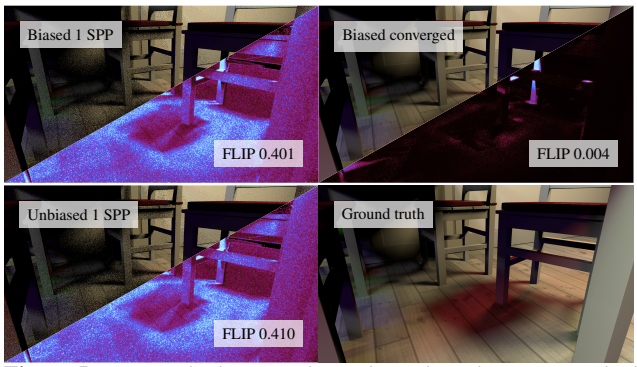

Figure 5: *Our method can produce a biased result (top row) which typically exhibits as a hard boundary around heavily shadowed areas. Clamping the visibility to 0.001 (bottom row) yields an unbiased result at the cost of slightly higher variance, converging to the ground truth. The left column shows results for one sample per pixel, while the right column shows converged results.*

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 [Mac67, Llo82] 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 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) [TCE05] implemented with reservoirs [BWP*20]. 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 $\mathrm{w}_{sum} = \sum_{j=1}^{m} \mathrm{w}(y_j)$. The second step uses a streaming RIS [TCE05] 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 $p(x) = m\frac{\mathrm{w}(y)}{\mathrm{w}_{sum}}\frac{1}{m_y}$, where $m\frac{\mathrm{w}(y)}{\mathrm{w}_{sum}}$ is a reciprocal weight of the reservoir from the first step and $\frac{1}{m_y}$ is a 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 49152 training examples for the Subway scene with 60k 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 8 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 & light sampling, and (5) shading. The neural network is implemented as a fully-fused MLP [MRNK21], 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 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 8 initial candidates, casting a shadow ray for the selected initial candidate. We use temporal resampling where we clamp contribution of the previous reservoir to 20× the contribution of 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 training steps (see Figure 6).

Compared to screen-space ReSTIR, our neural light sampling achieves a lower FLIP error [ANA21] at a similar time budget, especially in the occluded regions (see Figures 1 and 9). 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 5). Neural DI that directly uses the visibility estimates provided by the neural network (see Section 3.1.1) is biased by definition, yet it provides significantly lower error than ReSTIR and neural light sampling (see Figure 1). The bias exhibits at the edges and heavily shadowed areas (see Figure 3).

4.1.1. Combining ReSTIR with Clustered NVC

To generate initial candidates, standard ReSTIR uses a RIS loop with user-selected number of candidates (we use 8), 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 3.4, therefore it can be directly used to generate initial candidates.

While using NVC for initial candidates improves the quality of ReSTIR overall, the biggest benefit is for boosting convergence rate for disocclusions. Figure 7 shows a significant noise reduction when NVC is used for initial candidates after a disocclusion event. Running NVC for ReSTIR once it converges only improves ReSTIR quality insignificantly, while introducing an overhead of running the inference for every pixel. Therefore, we recommend running NVC only for disoccluded pixels to boost quality of initial candidates. This implementation leads to runtime performance only 5% lower compared to standard ReSTIR, on a test using a dynamic camera flying around the scene for 1024 frames.

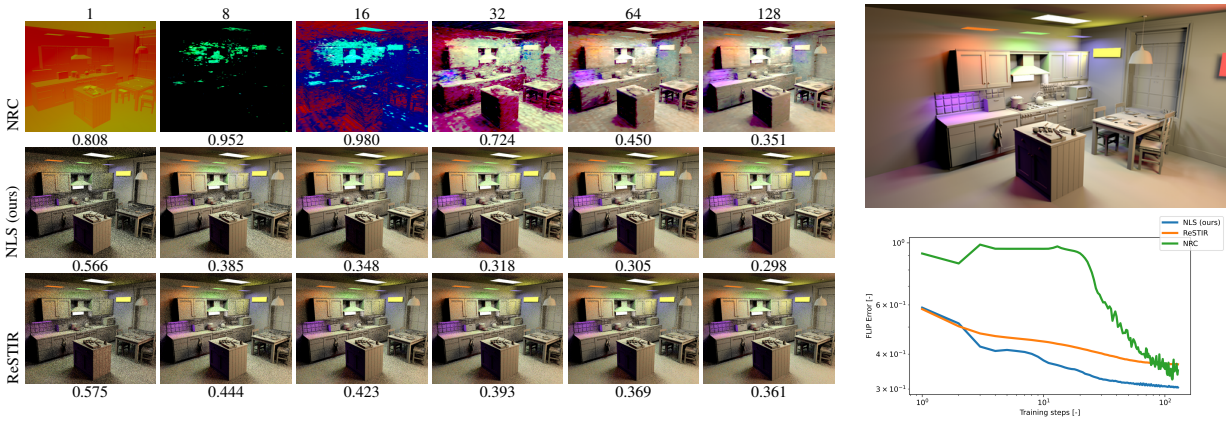

Figure 6: *A comparison of NRC (top), our neural light sampling (NLS) (center) and screen space ReSTIR (bottom) on a progression of 128 frames. Number under each image represents its FLIP error, which decreases over time for all methods, but NRC suffers from color artifacts, especially at the beginning 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*

Figure 7: *A comparison of standard ReSTIR (left), ReSTIR with initial candidates generated by our clustered NVC (center) and ground truth (right) in an idealized case when disocclusions happen in every pixel. The Subway scene contains 60k 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.*

4.2. Comparison to Neural Radiance Cache

For a direct comparison of our method to the neural radiance cache (NRC) [MRNK21], we have implemented a version of NRC which caches only direct illumination from all lights on a primary hit (NRC DI). We use the exactly same network architecture (except the output layer which consists of three neurons as radiance is RGB value, and we use leaky ReLU activation function to allow for unbounded values) and training data procedure as we do for our method. The NRC is not limited by number of lights, but it has several drawbacks compared to our method. For optimal results, NRC implementation requires a larger MLP, which lowers the training and inference performance. Another disadvantage is that NRC requires more training steps to achieve usable results (see Figure 6). At the beginning of training, the 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 8). Our method instead calculates the radiance analytically with LTCs. 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 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.

4.3. Performance

Training data generation with our default configuration (8196 training examples and 32 lights) takes 0.34 ms for the Kitchen scene and 0.50 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 training itself is implemented as separate steps, where first step writes out the data into memory for second step to consume, it is possible to perform multiple training iterations over 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 1.87 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 towards 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).

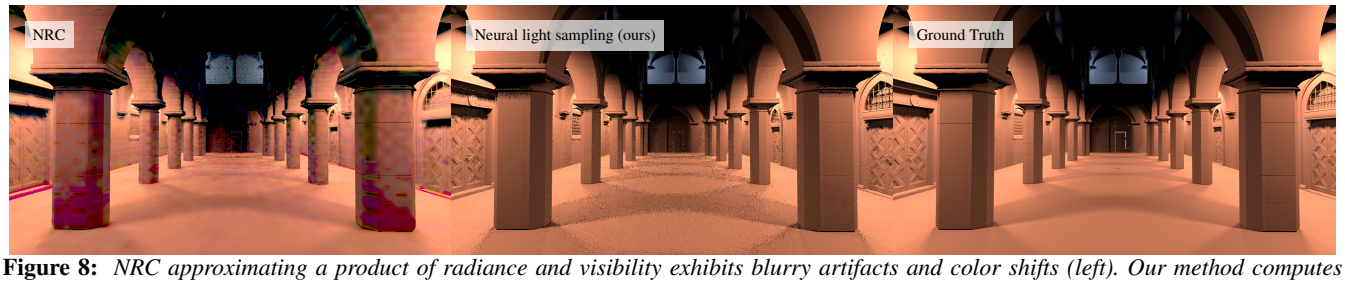

Figure 8: *NRC approximating a product of radiance and visibility exhibits blurry artifacts and color shifts (left). Our method computes exact radiance analytically and samples the light sources stochastically for unbiased rendering (middle). The ground truth accumulated with many frames (right).*

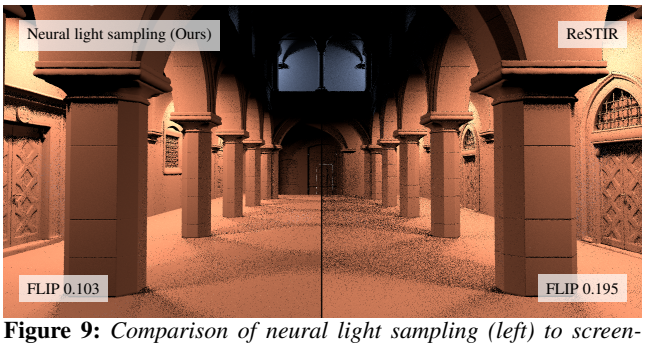

Figure 9: *Comparison of neural light sampling (left) to screen-space ReSTIR (right) on the Sponza scene with 32 lights. Neural light sampling produces lower error at the same performance cost.*

5. Conclusion and Future Work

We proposed a lightweight neural-based sampling method for real-time direct illumination based on caching the non-binary 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 that ReSTIR struggles with. 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 [FHBK25], we are only caching visibility with a smaller MLP, achieving faster training and simpler implementation (except from the position, we do not require other inputs, e.g. the surface normal to the MLP).

The limitation of our clustered NVC approach is that efficiency is highly dependent on the quality of the clusters created, and 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 [VKK18, FHBK25] and light culling [TH16]. An early approach of Shirley et al. [SWZ96] 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 use 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 visibility of any light, not just the ones represented by output neurons. To make this practical, such query would either have to be faster than a ray cast, or it would 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 fixed training budget. Overcoming this limitation in a scalable way is a non-trivial task for the future work.

Acknowledgements

We would like to thank Holger Grün and Carsten Benthin for useful discussions and their support. Model courtesy: Sponza (Crytek), Subway (Sketchfab/Alex Murias), Country Kitchen (Blendswap/Jay-Artist).

References

[ANA21] Andersson P., Nilsson J., Akenine-Möller T.: Visualizing and Communicating Errors in Rendered Images. In *Ray Tracing Gems II*, Marrs A., Shirley P., Wald I., (Eds.). 2021, ch. 19, pp. 301–320. 1, 4

[BWP*20] Bitterli B., Wyman C., Pharr M., Shirley P., Lefohn A., Jarosz W.: Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (July 2020). 1, 2, 4

[CEK18] Conty Estevez A., Kulla C.: Importance sampling of many lights with adaptive tree splitting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 1*, 2 (2018), 1–17. 2

[Cha82] Chao M.-T.: A general purpose unequal probability sampling plan. *Biometrika 69*, 3 (1982), 653–656. 1, 2

[DKH*14] Dachsbacher C., Křivánek J., Hašan M., Arbree A., Walter B., Novák J.: Scalable realistic rendering with many-light methods. *Computer Graphics Forum 33*, 1 (2014), 88–104. 2

[DKHD24] Dereviannykh M., Klepikov D., Hanika J., Dachsbacher C.: Neural two-level monte carlo real-time rendering. In *Computer Graphics Forum* (2024), Wiley Online Library, p. e70050. 2

[DNSD22] DATTA S., NOWROUZEZAHRAI D., SCHIED C., DONG Z.: Neural shadow mapping. In *ACM SIGGRAPH 2022 Conference Proceedings* (2022), pp. 1–9. 2

[FHBK25] FIGUEIREDO P., HE Q., BAKO S., KALANTARI N. K.: Neural importance sampling of many lights. *arXiv preprint arXiv:2505.11729* (2025). 2, 6

[GEE20] GUO J. J., EISEMANN M., EISEMANN E.: Next event estimation++: Visibility mapping for efficient light transport simulation. *Computer Graphics Forum 39*, 7 (2020), 205–217. 2

[HDHN16] HEITZ E., DUPUY J., HILL S., NEUBELT D.: Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–8. 2

[HIT*24] HUANG J., IIZUKA A., TANAKA H., KOMURA T., KITAMURA Y.: Online neural path guiding with normalized anisotropic spherical gaussians. *ACM Trans. Graph. 43*, 3 (Apr. 2024). 2

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26*, 3 (July 2007), 26–es. 2

[HZRS15] HE K., ZHANG X., REN S., SUN J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (2015), pp. 1026–1034. 3

[ISSS24] IWANICKI M., SLOAN P.-P., SILVENNOINEN A., SHIRLEY P.: The neural light grid: A scalable production-ready learned irradiance volume. *SIGGRAPH Advances in Real-Time Rendering in Games* (2024). 2

[KB14] KINGMA D., BA J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (12 2014). 3

[KGPB05] KŘIVÁNEK J., GAUTRON P., PATTANAIK S., BOUATOUCH K.: Radiance caching for efficient global illumination computation. *Visualization and Computer Graphics, IEEE Transactions on 11*, 5 (sept.-oct. 2005), 550 –561. 2

[Llo82] LLOYD S.: Least squares quantization in pcm. *IEEE transactions on information theory 28*, 2 (1982), 129–137. 4

[Mac67] 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* (1967), vol. 5, University of California press, pp. 281–298. 4

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG) 41*, 4 (2022), 1–15. 1, 2

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

[MMR*19] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVÁK J.: Neural importance sampling. *ACM Transactions on Graphics (ToG) 38*, 5 (2019), 1–19. 2

[MPC19] MOREAU P., PHARR M., CLARBERG P.: Dynamic many-light sampling for real-time ray tracing. In *High Performance Graphics (Short Papers)* (2019), pp. 21–26. 2

[MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021). 1, 2, 4, 5

[RHP*24] REN H., HUO Y., PENG Y., SHENG H., XUE W., HUANG H., LAN J., WANG R., BAO H.: Lightformer: Light-oriented global neural rendering in dynamic scene. *ACM Trans. Graph 1*, 1 (2024). 2

[SWZ96] SHIRLEY P., WANG C., ZIMMERMAN K.: Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics (TOG) 15*, 1 (1996), 1–36. 6

[TCE05] TALBOT J., CLINE D., EGBERT P.: Importance Resampling for Global Illumination. In *Eurographics Symposium on Rendering (2005)* (2005), Bala K., Dutre P., (Eds.), The Eurographics Association. 4

[TH16] TOKUYOSHI Y., HARADA T.: Stochastic light culling. *J Comput Graph Tech 5*, 1 (2016). 6

[TIKH24] TOKUYOSHI Y., IKEDA S., KULKARNI P., HARADA T.: Hierarchical light sampling with accurate spherical gaussian lighting. In *SIGGRAPH Asia 2024 Conference Papers* (New York, NY, USA, 2024), SA '24, Association for Computing Machinery. 2

[VKK18] VÉVODA P., KONDAPANENI I., KŘIVÁNEK J.: Bayesian online regression for adaptive direct illumination sampling. *ACM Trans. Graph. 37*, 4 (July 2018), 125:1–125:12. 2, 6

[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 (aug 2014). 2

[War94] WARD G. J.: Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics: Proceedings of the Second Eurographics Workshop on Rendering* (1994), Springer, pp. 11–20. 2

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 1098–1107. 2

[Wym21] WYMAN C.: *Weighted Reservoir Sampling: Randomly Sampling Streams*. Apress, Berkeley, CA, 2021, pp. 345–349. 1, 2