Eurographics Symposium on Rendering (2025)
B. Wang and A. Wilkie (Editors)

Neural Resampling with Optimized Candidate Allocation

A. Rath*,[1] M. Manzi*,[1] S. Weiss[1] T. Portenier[1] F. Salehi[1] S. Hadadan[1] M. Papas[1]

* Joint first author
[1]Disney Research | Studios, Switzerland

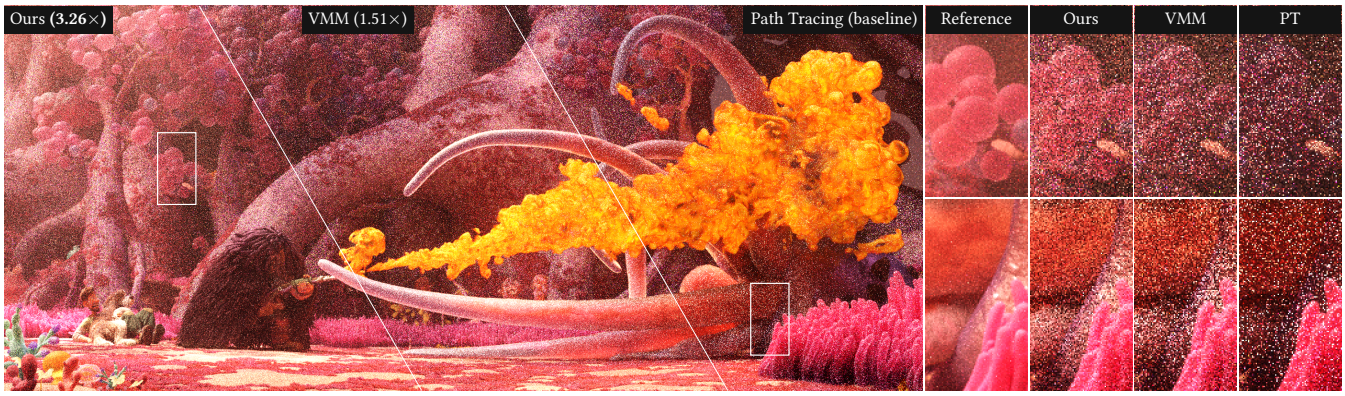


Figure 1: *We demonstrate the benefit of our sampling approach in equal-time renderings of an example frame (*TENTACLES*) from a feature film using our in-house CPU production renderer. Our approach can be used to improve path sampling in CPU production renderers by utilizing otherwise idle GPU resources without intrusive changes to the codebase, and achieves robust efficiency improvements over the existing state-of-the-art in path guiding [RHL20] (VMM) on large-scale production scenes. © Disney*

Abstract

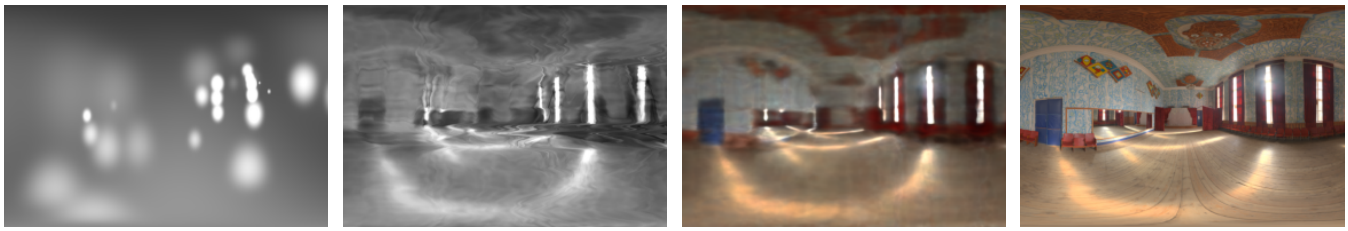
We propose a novel framework that accelerates Monte Carlo rendering with the help of machine learning. Unlike previous works that learn parametric distributions that can be sampled directly, our method learns the 5-dimensional unnormalized incident radiance field and samples its product with the material response (BRDF) through Resampled Importance Sampling. This allows for more flexible network architectures that can be used to improve upon existing path guiding approaches and can also be reused for other tasks such as radiance caching. To reduce the cost of resampling, we derive optimized spatially-varying candidate counts to maximize the efficiency of the render process. We designed our method to accelerate CPU production renders by benefiting from otherwise idle GPU resources without need of intrusive changes to the renderer. We compare our approach against state-of-the-art path guiding methods, both neural and non-neural, and demonstrate significant variance reduction at equal render times on production scenes.

1. Introduction

Path tracing [Kaj86] is the dominant algorithm used in the VFX industry to synthesize high-quality imagery [KFF*15], but for complex production scenes, render times can reach hundreds of core hours per frame. While there is a push in the industry to exploit massive parallelization on GPUs to accelerate path tracing, many contemporary production renderers are still limited to CPU only, predominantly due to their massive code-bases, intricate architectures, and substantial memory requirements, making fully porting them to GPU infeasible without enormous engineering investments. This results in a significant under-utilization of available resources as GPUs become more prevalent on artists' workstations and compute clusters. We thus propose to leverage GPU resources to accelerate existing production CPU renderers with minimal intrusion. Our approach trains a neural network to model the scene's global illumination, which is trained asynchronously on the GPU and evaluated on the CPU to improve sampling decisions, resulting in a net increase of efficiency.

Path tracing stochastically samples light paths in a virtual scene, whereas the quality and cost of those samples affect how fast images

DOI: 10.2312/sr.20251181



(a) *Neural Parametric Mixtures* **(b)** *Neural Importance Sampling* **(c)** *Direct Prediction (Ours)* **(d)** *Reference*

Figure 2: *Existing neural guiding approaches fall into two categories:* ***(a)*** *Implicit approaches (e.g. Neural Parametric Mixtures, NPM) and* ***(b)*** *normalizing flows (e.g. Neural Importance Sampling, NIS). Implicit approaches use a network to predict parameters of an analytical distribution and are thus cheap to sample (1µs per sample) but lack expressiveness. Normalizing flows are expressive (8µs per sample), but their architecture makes them expensive to evaluate. We use direct prediction* ***(c)*** *where we train a network that directly learns incident radiance (1µs per sample). Note that – unlike the other methods – our network prediction is in color and is not normalized. We resort to RIS for sampling from it, and can use the prediction for additional down-stream tasks (radiance caching, adjoint-driven RRS, etc).*

converge. Path guiding, a form of adaptive importance sampling that performs online learning of a scene's light transport, can significantly improve the quality of the samples for path tracing. Its effectiveness depends on the overhead of training the model and inferring from it, as well as its accuracy. CPU-bound path guiding methods typically employ histograms or parametric mixture models which are fast but inaccurate, e.g. they struggle at capturing high frequency content in the signal, whereas neural path guiding methods require GPUs but are significantly more expressive.

We propose a neural path-guiding method that learns the incident radiance field on the fly and uses this information to improve sampling decisions. It differs from related previous neural methods (see Section 2) by directly representing the colored incident radiance field instead of fitting a normalized density function. This representation is highly expressive, outperforming more constrained parametric models in representation quality (see Figure 2) and can be used for other downstream tasks besides importance sampling, *e.g.* for radiance caching. For sampling, we leverage *resampled importance sampling* (RIS) [Tal05] to generate samples according to the product of the learned incident radiance and the material response from a set of candidate samples. Since the approximation quality and cost of RIS directly depends on the number of resampling candidates, we derive a novel mathematical framework to optimize the spatially-varying candidate counts with respect to efficiency with the help of aggregated statistics.

Our method has been designed for use in CPU-based production renderers with highly complex production scenes. Such scenes exhibit high per-ray costs due to complex materials and geometry, making the use of expensive guiding methods that rely on neural networks, such as ours, more affordable. As light transport in production scenes typically differs significantly from that in academic test scenes, we focus our evaluation on the former to ensure our results are predictive of the method's performance in realistic usage scenarios.

In summary, our contributions are as follows:

- We propose directly learning the incident radiance and use it for directional path guiding through resampled importance sampling (RIS).
- We present a general mathematical framework to optimize rendering efficiency with spatially-varying candidate counts for RIS.
- We evaluate our method in a production CPU renderer and demonstrate its effectiveness on large-scale production scenes.

2. Background and Related Work

We briefly outline the theoretical foundations of path tracing and path guiding before discussing prior work and their relation to our proposed method.

Monte Carlo Light Transport The rendering equation (RE) [Kaj86] relates incident (L_i), emitted (L_e), and reflected (L_r) radiance between any two surface points x and $y = x - t\omega_\mathrm{o}$ in a scene as

$$L_\mathrm{i}(x,-\omega_\mathrm{o}) = L_\mathrm{e}(y,\omega_\mathrm{o}) + \underbrace{\int_\Omega L_\mathrm{i}(y,\omega_\mathrm{i})B(y,\omega_\mathrm{o},\omega_\mathrm{i})\mathrm{d}\omega_\mathrm{i}}_{L_\mathrm{r}(y,\omega_\mathrm{o})}, \tag{1}$$

where B is the material response, including the cosine shortening term on surfaces. The color for each pixel px can then be determined by integrating L_i against the sensor response function W_px

$$I_\mathrm{px} = \int_\Omega W_\mathrm{px}(x,\omega_\mathrm{i})L_\mathrm{i}(x,\omega_\mathrm{i})\,\mathrm{d}\omega_\mathrm{i}. \tag{2}$$

The RE can be estimated using Monte Carlo (MC) sampling, for example, through path tracing [Kaj86], which incrementally builds paths from the camera to the light sources. At each intersection, a direction is chosen according to some probability density function (PDF) $p(\omega_\mathrm{i} \mid x,\omega_\mathrm{o})$, thus the one-sample estimator for L_r is

$$\langle L_\mathrm{r}(x,\omega_\mathrm{o})\rangle = \frac{\langle L_\mathrm{r}(\omega_\mathrm{i} \mid x,\omega_\mathrm{o})\rangle}{p(\omega_\mathrm{i} \mid x,\omega_\mathrm{o})} = \frac{\langle L_\mathrm{i}(x,\omega_\mathrm{i})\rangle B(x,\omega_\mathrm{o},\omega_\mathrm{i})}{p(\omega_\mathrm{i} \mid x,\omega_\mathrm{o})}. \tag{3}$$

We omit the volume RE here for brevity, but our method is trivially applicable to volumes as later demonstrated in Section 6.

Importance sampling (IS) reduces the variance of the MC estimator by sampling approximately proportionally to (parts of) the integrand. A common sampling technique in path tracing is to sample proportionally to the material response, i.e., with $p(\omega_\mathrm{i} \mid x,\omega_\mathrm{o}) \propto$

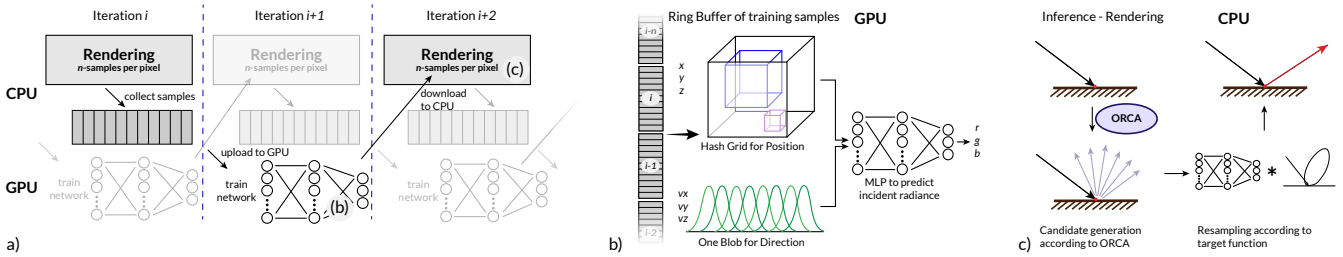


Figure 3: *Overview of our general architecture. Since rendering is performed on the CPU, we can train the network in parallel on the GPU (a), see Section 5. To avoid overfitting, we keep a ring buffer of previous frames to train the small MLP of incident radiance (b). In the renderer (c) on scattering locations, we use ORCA (Section 4) to obtain the desired number of candidate directions. For each direction, we query the network to compute the probability of the direction and use RIS to select the final direction (Section 3).*

$B(x, \omega_o, \omega_i)$, or according to the direct illumination as an approximation of L_i. Multiple such strategies can be (near) optimally combined using Multiple Importance Sampling (MIS) [Vea97]. One limitation of (multiple) importance sampling is that they can only generate samples from tractable distributions. This limits the use of (M)IS to functions for which analytical sampling exists, or when it is computationally affordable to perform inverse CDF sampling.

Resampled Importance Sampling (RIS) Our method heavily relies on RIS [Tal05], a two-step sampling scheme for generating samples that are approximately distributed according to an arbitrary non-negative target function $\hat{p}$. RIS is typically used when direct sampling from $\hat{p}$ is impractical, particularly when it's normalization constant is unknown. First, c candidate samples x_i are generated from a candidate distribution q, and second, a subset of the candidates are resampled with probabilities proportional to $w(x_i) = \hat{p}(x_i)\, q^{-1}(x_i)$. In the case of selecting a single sample from c candidates the RIS estimate is

$$\langle f; c \rangle = f(x_s) W_{\text{RIS}}, \quad \text{with} \quad W_{\text{RIS}} = \frac{1}{c\hat{p}(x_s)} \sum_{i=1}^{c} w(x_i), \qquad (4)$$

where x_s is the selected sample after resampling. $\langle f; c \rangle$ is unbiased if $\hat{p} > 0$ within the support of the integrand f.

Path guiding Path guiding strives to importance sample the incident radiance or the full product from Equation 3, i.e.,

$$p(\omega_i \mid x) \propto L_i(\omega_i \mid x) \quad \text{or} \quad p(\omega_i \mid x, \omega_o) \propto L_r(\omega_i \mid x, \omega_o). \qquad (5)$$

Intuitively, this corresponds to guiding paths with higher probability into regions where more light arrives from. Since L_i is not known beforehand, most guiding methods use samples from earlier iterations to fit distributions that become progressively more accurate throughout rendering. Over the last decade, a plethora of methods have been proposed to learn incident radiance [MGN17, VKv*14, RHL20] or the full product [HEV*16, HZE*19, DGJ*20, DPÖM22, SHJD22].

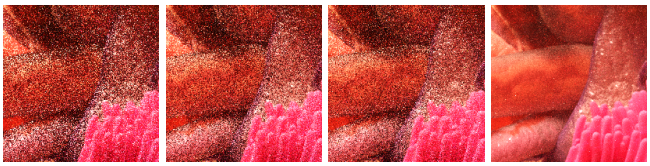
Relevant to us is the work of Deng et al. [DWWH20], that combines incident radiance distributions and the material response via RIS to perform product sampling. Learning the 5D incident radiance allows for models that are simple to fit, unlike methods that directly aim to learn the 7D product distribution [DPÖM22, SHJD22]. At the same time, this approach can be more efficient to sample than methods that compute product distributions on the fly [HEV*16].

Most of the non-neural path guiding techniques mentioned above simplify the problem of learning a global spatio-directional (5D) function by subdividing the scene space into regions where radiance is assumed to be constant, for each of which a directional (2D) distribution is then learned. This assumption, however, is often violated in complex scenes and currently no robust method to circumvent this problem is known. Ruppert et al. [RHL20] mitigates some of the above mentioned issues by compensating for parallax, and Dodik et al. [DPÖM22] performs implicit parallax-compensation using local spatio-directional mixture models to relax the constraint. While these approaches mitigate the problem to some degree, discontinuities still arise due to the explicit subdivision.

Neural Sampling Neural path guiding methods learn global incident or product distributions through neural networks and thus circumvent problems related to spatio-directional subdivision due to their superior interpolation capability. For instance, Neural Parametric Mixtures (NPM) [DWL23] uses a multi-layer perceptron (MLP) to predict parametric model parameters as a function of spatial location to describe the angular distribution. This approach is fast to evaluate and sample, but lacks expressiveness, see Figure 2a. The method from Huang et al. [HIT*24] improves upon NPM by using more expressive basis-functions, but fundamentally still suffers from difficulties in modeling discontinuities. Alternatively, Neural Importance Sampling (NIS) [MMR*19] uses normalizing flows to model a global, sampleable, and invertible distribution (see Figure 2b). This approach has subsequently also been extended to Neural Control Variates [MRKN20]. Techniques based on normalizing flows are more expressive than non-neural techniques and NPM, but more expensive to train and query due to their complex architecture. In contrast, our proposed method is both highly expressive, efficient to sample from and learns a RGB incident radiance function directly. The latter allows for the design of more powerful target functions (Equation 6) and the use in additional down-stream task (e.g. radiance caching).

3. Neural Resampling

In this work, we propose a novel path guiding method that samples directions ω_i proportional to the reflected radiance integrand (Equation 5). To make such importance sampling more tractable in presence of complex, high-frequency materials as encountered in



source = p_B 8cand / 34min MSE 2.47 | source = p_B 32cand / 48min MSE 1.79 | source = Eq.7 8cand / 33min MSE 1.83 | Reference

Figure 4: *We compare our resampling approach using different source distributions and (fixed) candidate counts in insets of* TENTACLES *at 64 samples per pixel. Using the mixed source distribution (Equation 7) yields superior results over using the material sampling distribution p_B. The MSE is with respect to the entire image.* © *Disney*

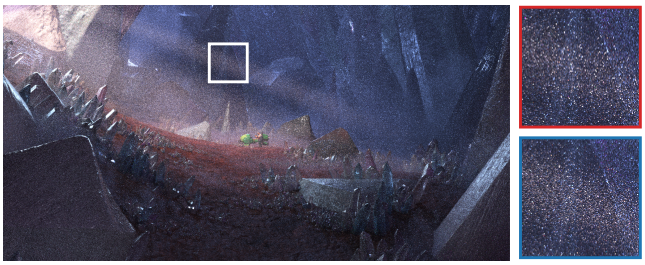


Figure 5: *We improve the stability of our method by proposing a novel regularization method tailored to our model, see Section 3.1. With regularization (blue box), outliers in* CRYSTAL CAVE *at 64 samples per pixel are reduced significantly compared to without regularization (red box).* © *Disney*

production scenes, we do not directly learn the 7D reflected radiance function, but instead learn the 5D incident radiance term and use its product with the material response as target function for RIS.

We use a lightweight MLP to learn the incident radiance field, a 5D function of position and direction to color. In contrast to existing methods that are either cheap to sample but limited in expressiveness (NPM) or more expressive but expensive to sample (NIS), see Section 2, our MLP representation is faster to evaluate than NIS and more expressive than NPM, see Figure 2c. The main drawback of such a model is that the represented function is neither normalized nor directly sampleable. We circumvent this issue by employing RIS [Tal05], which enables the predicted incident radiance to be used for path guiding. Since our model explicitly learns the incident radiance and not, *e.g.*, a gray-scale distribution of luminance, it can be used for additional tasks beyond sampling, such as Neural Radiance Caching (see Section 7).

An overview of our pipeline is depicted in Figure 3. Since many production renderers perform path tracing on the CPU, we can train the MLP in parallel on the GPU without impacting rendering performance (see Figure 3a). Training details and the network architecture (Figure 3b) are discussed in Section 5. Below we discuss how to sample new directions using the learned incident radiance field (see Figure 3c) followed by a description of our novel framework to automatically optimize the number of RIS candidates in Section 4.

Target Function Our model predicts the RGB incident radiance which enables the evaluation of the product with the RGB material response and even the throughput of the prefix path $\bar{x}$

$$\hat{p}(\omega_i \mid \omega_o, \bar{x}) = \frac{1}{3} \sum_{c \in \{R,G,B\}} T^c(\bar{x}) B^c(\omega_i \mid x, \omega_o) \mathcal{N}^c_\Theta(\omega_i, x). \quad (6)$$

Here, $\mathcal{N}_\Theta$ with trainable parameters Θ is the MLP that predicts the incident radiance, B is the material response (including the cosine term on surfaces), and $T(\bar{x})$ is the throughput of the prefix path $\bar{x}$ up to the current scattering event. The reasoning for the inclusion of $T(\bar{x})$ is similar to the one in Rath et al. [RGH*22], and takes into account how much light per color channel is expected to reach the camera. This target function estimates the expected contribution of the path to the image if we sample towards ω_i. E.g., if the throughput heavily attenuates the green color channel, sampling into directions of green incident radiance will be disincentivized. Note that this would not be possible if our network predicted a scalar instead of an RGB value. Since all quantities are in RGB we use the average to compute resampling weights.

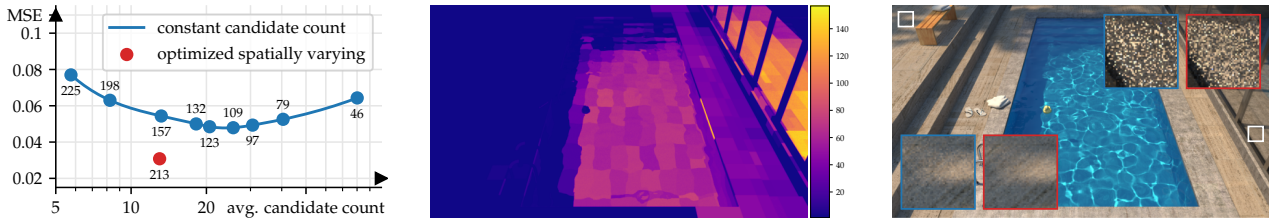
Source Distribution A good source distribution should be cheap to generate candidates from, and as similar to the target function as possible to reduce the number of required candidates. The latter is especially important in our case, since our target requires to evaluate an expensive MLP per candidate. While one could use material sampling to generate candidates, a large number of candidates would be needed to approximate the target function well, decreasing the efficiency of resampling. We instead decided to use a two-level guiding approach; In a first step, we generate candidates from a guiding model that can be sampled from directly. The choice of the source distribution does not matter too much as long as it can be trained cheaply alongside the more powerful neural model and the resulting guiding PDF roughly approximates the incident radiance. In a second step, we resample from these guided candidates according to the more expressive neural target function Equation 6. For robustness, we combine sampling from the candidate guiding distribution with material sampling through MIS with the balance heuristic and equal strategy selection probability. The source distribution q is hence

$$q(\omega_i \mid x) = \frac{1}{2} p_G(\omega_i \mid x) + \frac{1}{2} p_B(\omega_i \mid x), \quad (7)$$

where p_G is the candidate guiding PDF and p_B the material sampling PDF. This choice of source distribution comes at the cost of training an additional (cheap, but less accurate) guiding model along-side our neural model, but decreases the number of required candidates, and hence overhead of our method significantly as shown in Figure 4.

3.1. Defensive Resampling

Importance sampling can greatly reduce variance by sampling more rays towards bright sources of light while sending fewer rays in directions that are expected to contribute little. The effectiveness of neural resampling depends on the accuracy of our model $\mathcal{N}_\Theta$. If our model fits the incident radiance poorly, our approach can lead to high variance. Such issues are commonly addressed using one-sample MIS [Vea97], which combines *exploitation* of the predicted distribution with *exploration* through more defensive sampling strategies. In



(a) *Image error vs candidate count for equal render time (2mins)*

(b) *Our candidate allocation*

(c) *Best constant vs. our spatially varying candidate allocation*

Figure 6: ***(a)*** *We compare equal-time renderings (2mins) with various constant and spatially varying candidate counts, and overlay each datapoint with the affordable number of samples per pixel that fit the time budget. By balancing computational cost and variance reduction across pixels and bounces* ***(b)****, our spatially varying candidate counts outperform the best possible constant allocation* ***(c)****.*

our case, however, one-sample MIS is not directly applicable since the PDF of resampling is unknown [Tal05].

One solution is to add a small regularization constant to the radiance prediction to ensure that all directions can be sampled. However, this approach lacks robustness for our use-case since the scale of our unnormalized radiance estimate varies greatly within the scene. Instead, we propose a similar concept to one-sample MIS directly in the RIS estimator, which we detail in the following.

For each candidate, we propose to compute two targets: $\hat{p}_1$ (as described in (6)), as well as a defensive target $\hat{p}_2$, which we choose to be the source density q as it is known to explore all directions. Since $\hat{p}_1$ and $\hat{p}_2$ do not match in scale ($\hat{p}_1$ is unnormalized, $\hat{p}_2$ is normalized) it is not possible to blend the targets directly. We instead propose to alter the target function of each candidate i:

$$\hat{p}'_i = \alpha \frac{c\,\hat{p}_1(x_i)}{\sum_{j=1}^{c} \hat{p}_1(x_j)\,q^{-1}(x_j)} + (1-\alpha)\frac{c\,\hat{p}_2(x_i)}{\sum_{j=1}^{c} \hat{p}_2(x_j)\,q^{-1}(x_j)}. \quad (8)$$

Similar to how one-sample MIS randomly picks between strategies, this target ensures that we have a chance of α to pick from strategy $\hat{p}_1$, and a chance of $1-\alpha$ to pick from strategy $\hat{p}_2$. Note that this modified target does not only depend on the current candidate i, but also incorporates estimates from all other candidates. The proof for unbiasedness of RIS as shown by Talbot [Tal05] in their appendix A.1 is unaffected by this additional dependency, and hence our defensive scheme remains unbiased. In Figure 5 we demonstrate the benefit of this approach in the CRYSTAL CAVE scene. We have found that $\alpha = 0.9$ strikes a good balance between exploration and exploitation, hence we use it throughout our experiments.

4. Optimized Resampling Candidate Allocation

A crucial parameter of resampled importance sampling is the candidate count as it allows us to trade off sampling quality against computational cost. In Figure 6a we demonstrate its impact in an academic path tracer [Jak10] as proof-of-concept. Here we use BSDF sampling as source distribution and an incident radiance estimate based on histograms [MGN17] to determine the product target. Note that by carefully choosing the candidate count, the efficiency of the render can be significantly improved. Unfortunately, the optimal value is not known beforehand and depends on the scene at hand. Ideally, we would also like the candidate count to depend on the path prefix at hand, which makes it even more challenging to find optimal values.

While previous approaches have derived optimal candidate counts for single RIS estimators in isolation [Tal05], we aim to optimize them for all pixel estimators simultaneously. These estimators are also recursive: Each path vertex employs resampling to perform its own reflected radiance estimate. In this section, we propose a practical approach to optimize candidate counts that automatically adapts to the scene at hand. By letting the candidate count vary spatially (Figure 6b), we increase efficiency even further compared to what a constant candidate counts can achieve (Figure 6c).

4.1. Optimization goal

At each vertex of our path, we determine a candidate count $c(\bar{x})$ based on all the information we have seen along the path prefix $\bar{x}$ so far. Our goal is to arrive at an optimal such assignment $c^\star(\bar{x})$ that maximizes the efficiency of our rendering

$$c^\star = \arg\max_c \Bigg(\underbrace{\left(\sum_{\text{px}} \mathbb{C}\left[\langle I_{\text{px}}; c\rangle\right]\right)}_{\mathbb{C}_I} \underbrace{\left(\sum_{\text{px}} \mathbb{V}\left[\langle I_{\text{px}}; c\rangle\right]\right)}_{\mathbb{V}_I} \Bigg)^{-1}, \quad (9)$$

where $\langle I_{\text{px}}; c\rangle$ is the estimator for a given pixel px employing the candidate allocation function c. $\mathbb{C}[\cdot]$ denotes the expected cost of an estimator, and $\mathbb{V}[\cdot]$ the variance thereof. Put simply, a good assignment of candidate counts must strive for a low total cost $\mathbb{C}_I$ and low image variance $\mathbb{V}_I$ across the entire image.

4.2. Fixed point scheme

To tackle this optimization problem, we can draw inspiration from a related problem setting: Russian roulette and splitting (RRS), in which splitting factors are optimized at each path vertex. Similar to our candidate counts, splitting factors are the parameter that determine the trade-off between sampling quality and computational cost in RRS. In the following, we extend the theory of [RGH*22] to the problem of resampling.

Sufficient criterion As shown by Ruppert et al. [RGH*22] in their equation (20), optimizing an efficiency functional such as ours (Eq.

(9)) boils down to solving the following for each prefix $\bar{x}$

$$\mathbb{V}_I \frac{\partial}{\partial c(\bar{x})} \mathbb{C}\left[\langle L_\mathrm{r}(\bar{x}); c(\bar{x})\rangle\right] + \mathbb{C}_I T^2(\bar{x}) \frac{\partial}{\partial c(\bar{x})} \mathbb{V}\left[\langle L_\mathrm{r}(\bar{x}); c(\bar{x})\rangle\right] = 0. \tag{10}$$

Here, $\langle L_\mathrm{r}(\bar{x})\rangle$ denotes the estimator for the radiance at the last point of the prefix $\bar{x}$ which is reflected towards the preceding vertex. By $\langle L_\mathrm{r}(\bar{x}); c(\bar{x})\rangle$ we express that this estimator performs RIS with $c(\bar{x})$ candidates. Finally, $T(\bar{x})$ denotes the throughput weight of the path prefix $\bar{x}$. To solve this equation, we need to take a closer look at the variance and expected cost of the RIS estimator.

Variance of RIS Talbot [Tal05] has shown that the variance of an RIS estimator is a blend between the variances of estimating the integral using the source distribution $\mathbb{V}_s\left[\langle L_\mathrm{r}\rangle\right]$ and using the target distribution $\mathbb{V}_t\left[\langle L_\mathrm{r}\rangle\right]$

$$\mathbb{V}\left[\langle L_\mathrm{r}; c\rangle\right] = \frac{1}{c}\mathbb{V}_s\left[\langle L_\mathrm{r}\rangle\right] + \left(1 - \frac{1}{c}\right)\mathbb{V}_t\left[\langle L_\mathrm{r}\rangle\right]. \tag{11}$$

Cost of RIS We model the cost of RIS as a linear function

$$\mathbb{C}\left[\langle L_\mathrm{r}; c\rangle\right] = cK_0 + K_1 + \mathbb{C}\left[\langle L_\mathrm{r}\rangle\right]. \tag{12}$$

Here K_0 denotes all costs that scale linearly with the number of candidates (e.g., generating a candidate, or evaluating its target). Constant costs are captured by K_1, which includes any pre-computations needed that are executed only once (e.g., computing spatial encodings). The final step of RIS is evaluating the chosen candidate, the expected cost of which we denote $\mathbb{C}\left[\langle L_\mathrm{r}\rangle\right]$. Note that the average cost of evaluating the final sample can vary as the distribution of samples drawn by RIS varies under c. Since this shift in distribution cannot be expressed in closed form [Tal05], we make the simplifying assumption that the cost of evaluating the sample remains constant under c.

Fixed point Combining the derivatives of (11) and (12) into (10) and solving for $c(\bar{x})$, we arrive at the fixed point

$$c(\bar{x}) = \sqrt{\frac{T^2(\bar{x})\left(\mathbb{V}_s\left[\langle L_\mathrm{r}(\bar{x})\rangle\right] - \mathbb{V}_t\left[\langle L_\mathrm{r}(\bar{x})\rangle\right]\right)}{\mathbb{V}_I} \frac{\mathbb{C}_I}{K_0}}. \tag{13}$$

A full derivation can be found in the supplementary document. Intuitively, this function relates the reduction in variance we expect in the image (i.e., the local reduction in variance of L_r times the throughput squared) against the total amount of variance already present in the image. This ratio is counteracted by the expected increase in cost in relation to the total cost of rendering the image. A negative value in the square root indicates that the RIS target performs worse than the source distribution, in which case one should resort to using source sampling instead. To turn the fractional candidate count $c(\bar{x})$ into an integer one, stochastic rounding can be employed [VK16a, MRYS24].

Required estimates To compute our fixed point scheme, we need to know the variance $\mathbb{V}_I$ and total cost $\mathbb{C}_I$ of the image, as well as local estimates for the difference in variance when sampling according to the target distribution versus the source distribution. The variance of the image can be computed by tracking the second moments of each pixel [MGN17], and a simple but effective proxy for the cost $\mathbb{C}_I$ is to count the number of rays that have been traced [BM97]. Note that unlike the work of Rath et al. [RGH*22], we do not require an estimate of the cost of the primary estimator–in RRS these are needed as the cost of sample evaluation scales with the splitting factor, but in RIS we only evaluate a single final sample and hence the local cost vanishes in the variance derivative. The key challenge is the variance delta, which relates the variances of source and target distribution, for which we derive an estimator based on the samples collected during rendering in the following.

The variance delta relates the variance of estimating L_r using the source distribution $\mathbb{V}_s\left[\langle L_\mathrm{r}(\bar{x})\rangle\right]$ and the variance under the target distribution $\mathbb{V}_t\left[\langle L_\mathrm{r}(\bar{x})\rangle\right]$. As both variances share the same first moment, their delta equals the difference of their second moments

$$\mathbb{V}_s\left[\langle L_\mathrm{r}\rangle\right] - \mathbb{V}_t\left[\langle L_\mathrm{r}\rangle\right] = \mathbb{M}_s\left[\langle L_\mathrm{r}\rangle\right] - \mathbb{M}_t\left[\langle L_\mathrm{r}\rangle\right]. \tag{14}$$

Given an RIS estimate for reflected radiance

$$\langle L_\mathrm{r}(\bar{x}); c\rangle = W_\mathrm{RIS}\,\langle L_\mathrm{r}(\bar{x}, \omega_\mathrm{i})\rangle, \tag{15}$$

we know that the RIS weight W_RIS is an unbiased estimate of the reciprocal probability density p_RIS^{-1} of having sampled the incident direction ω_i and hence we can use it to estimate the second moment of source sampling

$$\mathop{\mathbb{E}}_{\omega_\mathrm{i}\sim p_\mathrm{RIS}}\left[W_\mathrm{RIS}\frac{\langle L_\mathrm{r}^2(\bar{x}, \omega_\mathrm{i})\rangle}{q(\omega_\mathrm{i} \mid \bar{x})}\right] = \mathbb{M}_s\left[\langle L_\mathrm{r}\rangle\right]. \tag{16}$$

To estimate the second moment of target sampling, we can unfortunately not rely on the same trick as the ground truth density $p(\omega_\mathrm{i} \mid \bar{x})$ is unknown. We can, however, estimate the second moment of the RIS estimator itself

$$\mathop{\mathbb{E}}_{\omega_\mathrm{i}\sim p_\mathrm{RIS}}\left[W_\mathrm{RIS}^2\,\langle L_\mathrm{r}^2(\bar{x}, \omega_\mathrm{i})\rangle\right] = \mathbb{M}\left[\langle L_\mathrm{r}; c\rangle\right]. \tag{17}$$

Using (11) we rewrite the second moment of the RIS target

$$\mathbb{M}_t\left[\langle L_\mathrm{r}\rangle\right] = \frac{c\mathbb{M}\left[\langle L_\mathrm{r}; c\rangle\right] - \mathbb{M}_s\left[\langle L_\mathrm{r}\rangle\right]}{c-1}, \tag{18}$$

which allows us to estimate the variance delta from our sample

$$\left\langle \mathbb{V}_s\left[\langle L_\mathrm{r}\rangle\right] - \mathbb{V}_t\left[\langle L_\mathrm{r}\rangle\right]\right\rangle = \frac{c}{c-1} W_\mathrm{RIS}\,\langle L_\mathrm{r}^2(\bar{x}, \omega_\mathrm{i})\rangle\left(\frac{1}{q} - W_\mathrm{RIS}\right). \tag{19}$$

These estimates from individual samples can then be accumulated in different regions of the scene by employing a spatial data structure.

Switching between RIS and source sampling So far, our derivation assumed that we will always employ the RIS estimator. In certain cases–for example when the expected reduction in variance is insignificant or the RIS target performs worse than the source strategy–, the overhead might not be justified and we would like to fallback to plain source sampling. We will now derive the necessary equations to detect these cases.

We only want to perform RIS at the current path vertex if it is expected to improve the efficiency of rendering the image compared to using source sampling at the current vertex

$$\left(\mathbb{V}_I + \mathbb{V}\left[\langle L_\mathrm{r}; c\rangle\right]\right)\left(\mathbb{C}_I + \mathbb{C}\left[\langle L_\mathrm{r}; c\rangle\right]\right) < \left(\mathbb{V}_I + \mathbb{V}_s\left[L_\mathrm{r}\right]\right)\left(\mathbb{C}_I + \mathbb{C}_s\left[L_\mathrm{r}\right]\right). \tag{20}$$

Since $\mathbb{V}_I$ and $\mathbb{C}_I$ are typically large compared to the local variances

(unless the cost and variance of the image are dominated by very few pixels), we simplify this constraint to

$$\mathbb{V}_I\mathbb{C}\left[\langle L_\mathrm{r};c\rangle\right]+\mathbb{C}_I T^2\mathbb{V}\left[\langle L_\mathrm{r};c\rangle\right]<\mathbb{V}_I\mathbb{C}_s\left[\langle L_\mathrm{r}\rangle\right]+\mathbb{C}_I T^2\mathbb{V}_s\left[\langle L_\mathrm{r}\rangle\right]. \tag{21}$$

For further details on the derivation we refer the reader to our supplementary document. Testing for this constraint can be conveniently incorporated into the computation of the candidate count from (13): RIS should only be performed if

$$c(\bar{x})>\sqrt{\frac{K_1}{K_0}}, \tag{22}$$

which relates the constant overhead K_1 to the cost of each candidate K_0 to determine the minimum candidate count $c(\bar{x})$ for which it is worth to perform RIS. If this constraint is not met, it is more efficient for the render to fall back to source sampling at the current vertex.

Continuing to collect estimates When disabling RIS we lose the possibility to estimate the variance delta. This becomes a problem when the delta is expected to improve over time–in our setup we expect the MLP to become more accurate during training. To mitigate this issue, one can clamp the minimum candidate count to a value between one and two before stochastic rounding (we choose 1.125), which ensures that some fraction of samples (one eighth in our case) will perform RIS and statistics can continue to be gathered.

5. Implementation

We integrated our approach into an experimental branch of Walt Disney Animation Studio's CPU production renderer Hyperion [BAC*18] to perform tests on full-scale production scenes. Even though our method is not limited to a specific hardware architecture, we train the model on the GPU, but query it on the CPU. This avoids the need to batch inference queries for the GPU, which significantly simplifies integration in the renderer, especially since the number of queries changes per ray and intersection due to ORCA. Since we designed our network to be small and fast to query, this strategy is viable in production renders where the render-time is dominated by expensive shading and tracing operations.

5.1. Pipeline Overview

In our implementation, we render images progressively starting with one path per pixel and increasing geometrically until 16 paths per pixel.

Data Aggregation During rendering we collect the current accumulated radiance, position, and direction at every path vertex, corresponding to 9 floats per sample. To control the amount of storage needed to aggregate the samples, we only store training data for a dynamically determined fraction of paths. This fraction is automatically adjusted across iterations to correspond to roughly 4 samples per path, or 3.5 GiB of data for a render-resolution of 1920×804.

Network Details For maximal inference speed, we keep the neural network as lightweight as possible, see Figure 3b for an overview. For the purpose of this experiment, we use a hash grid encoding [MESK22] for the position with 8 levels of 4 channels each, coarse resolution of 8^3 and fine resolution of 2048^3 and a capacity of 2^{16} voxels per level in the hash map. The direction is encoded using one-blob [MMR*19] with 8 bins per channel. For the MLP we use 3 hidden layers with 64 channels and ReLU activation. We found that a bias in the MLP is not necessary and omitting it saves a bit of performance.

Training Training happens asynchronously on the GPU using the PyTorch C++ API [PGC*17] while the renderer continues rendering the next iteration (see Figure 3a). We use the Adam optimizer [KB14] since it works nicely for loss functions of unknown normalization. We set the learning rate to $1e{-}3$ and weight decay to $1e{-}5$, and optimize for a relative MSE loss to fit our model to the observed incident radiance data. To mitigate overfitting to samples from the most recent rendering iteration, we store samples on the GPU in a fixed-sized ring buffer holding the samples of the last 24 spp using a FIFO retention policy. Training stops either after a fixed number of epochs or when the main render process signals that the next rendering iteration has completed. This allows us to automatically adjust the number of training epochs based on the scene complexity for rendering.

Querying and Resampling For querying, the network weights get copied to CPU at the end of each iteration and are used in the renderer through optimized Eigen [GJ*10] inference code. We improve efficiency of the inference by caching the latent from the spatial hash grid across candidates when doing RIS. As candidate source distribution (Equation 7) we chose the approach from Ruppert et al. [RHL20], and use the implementation in Intel's Open Path Guiding Library [HD22]. The spatial KD-tree employed in that approach to store the local guiding models is also used to store the local variance delta estimates (Equation 19) used by our optimized sample allocation (Section 4).

6. Results

We compare our method against state-of-the-art neural and non-neural guiding methods on a variety of production scenes.

Experiment setup We render all test scenes with 32 threads on an AMD EPYC™ 7763 server CPU using an Nvidia RTX 4090 GPU. Russian roulette uses albedo estimates as survival probabilities, which, as demonstrated by Rath et al. [RGH*22], outperforms throughput-based RR for guiding methods. Direct light is down-weighted in the learned models by the MIS weight between directional sampling and direct light sampling, as described by Ruppert et al. [RHL20]. This prevents path guiding from wasting too many samples on direct light when it is well sampled by our light sampling routines already. This is especially important since our production CPU renderer employs a sophisticated light sampling strategy [LZN*24]. For our comparisons with Neural Importance Sampling (NIS [MRKN20]) and Neural Parametric Mixtures (NPM [DWL23]), we use the same training pipeline as for our method (see Section 5), i.e. we also train them on the GPU in parallel to rendering. While we have observed their respective product sampling variants to perform well in smaller scenes, they were consistently outperformed by their non-product variants in our test scenes. Hence, we omit them for brevity. We have also found them

to perform worse than phase function sampling for volume scattering, and accordingly only use them on surfaces. For our comparison to Robust Parallax-Aware Path Guiding (VMM [RHL20]), we use the implementation from Intel's Open Path Guiding Library [HD22]. We evaluate two flavors of the method; VMM using one-sample MIS to combine guiding with material sampling as described by the original authors (VMM), and VMM using RIS to sample the product. For VMM+RIS we use the same source distribution as our method and always resample from 8 candidates. As target it uses Equation 6 with the scalar-valued learned VMM density replacing $\mathcal{N}_\Theta$. Both flavors of VMM [RHL20] and our method are fully applied to surface and volume scattering. For optimizing the resample candidate count in our method (Section 4), we used empirically measured costs factors, $K_0 = 0.01$ and $K_1 = 0.04$, for Equation 12.

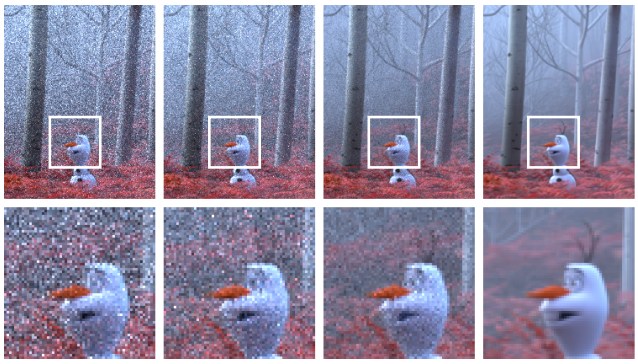
Results Our results are summarized in Figure 8. Note that production scenes often feature much simpler light transport than academic guiding test scenes; caustics are rare, and invisible light sources are used to fake indirect light, which are well explored through sophisticated next event estimation [LZN*24]. At the same time, these scenes feature higher geometric complexity and intricate, highly detailed materials, which can make it more challenging to fit guiding models to them. We focus our evaluation on such production scenes to make it more predictive for realistic use-cases.

We find that VMM consistently outperforms prior neural methods mainly due to its low overhead and more robust fits. Similarly, NPM outperforms the more expensive NIS in our experiments. An example of the distributions learned by each method is given in Figure 11. Combining VMM with resampling (similar to Deng et al. [DWWH20]) can help on glossy surfaces (e.g., CRYSTAL CAVE), but has little impact in volumes (e.g., LANDSCAPE), in which the product is already approximated in the source distribution [HD22]. Our method consistently outperforms both variants of VMM in equal time rendering, with noticeable improvements on materials requiring accurate product sampling, such as the vegetation in the TENTACLES scene. Across our representative set of test scenes, VMM+RIS achieves an average speed-up of **+25%** over path tracing, whereas our method achieves **+77%** average speed-up over path tracing.

Optimized Resampling Candidate Allocation Averaged over all tested production scenes, applying ORCA on top of our neural resampling accounts for roughly 1/4 of the total improvement of our full method over the existing state-of-the-art. More importantly, however, we have found that ORCA increases robustness in the presence of mispredictions, an example of which is showcased in Figure 10. Interestingly, the experiment in Figure 6 suggests that in setups with large discrepancy between the source and target distribution, *e.g.* when candidates are generated from the material distribution only, improvements due to ORCA can be more significant.

7. Discussion, Limitations and Future Work

Combination with Radiance Caching Our work focuses on path guiding, but other uses for the learned incident radiance field are feasible. As proof of concept, we use our model as incident radiance cache at the first scattering event. This simple approach works well for scenes with mostly diffuse materials (Figure 7), but more sophisticated heuristics for cache lookups are desirable for complex materials [MRNK21]. Concurrent work [DKHD24] implies that incident radiance caches are powerful tools to cut down rendering costs further. Exploring synergies between path guiding and other sampling decisions that can utilize adjoint estimates while sharing the same incident radiance model, such as radiance caching or Russian roulette [VK16b], is an exciting avenue for future work.



(a) *Path Tracing* **(b)** *Ours* **(c)** *Ours + RC* **(d)** *Reference*

Figure 7: *While our work focuses on path guiding* ***(b)****, the learnt incident radiance field can also be used for other downstream tasks, such as radiance caching* ***(c)****. All images except for the reference took roughly 30mins to render. © Disney*

Other Uses for Optimized Resample Candidate Allocation RIS can be used in rendering beyond path guiding, for instance to improve direct illumination sampling [BWP*20], or to sample distances in volumes [XHM*24]. Our proposed optimized candidate allocation scheme could potentially be adapted for such use cases.

Low-Discrepancy Sampling It is unclear how to effectively combine resampling and low-discrepancy (LD) sampling. While we observed minor benefits from distributing candidates according to LD sequences, likely because the re-sampling weight W_{RIS} is of lower variance, further analysis seems warranted.

Path Prefix Conditioning Path space regularization [KD13] is often used to reduce variance from higher order bounces at minimal bias, but makes the incident radiance depend of the path prefix. Similarly, limiting the number of bounces affects the incident radiance as a function of remaining bounces. Conditioning the incident radiance models on the relevant properties of the path prefix could improve the accuracy of the model.

8. Conclusion

We presented a path-guiding approach for Monte Carlo path tracing that learns the incident radiance using an unconstrained MLP and samples according to it using resampled importance sampling. We further presented a novel approach to optimize the RIS candidate allocation for efficiency, providing additional improvements in terms of render time and robustness. Finally, we demonstrated how this

approach can be integrated into CPU production renderers as a non-intrusive way to use readily available GPU resources and accelerate rendering.

Acknowledgment We thank Charlotte Zhu and Daniel Teece for providing the production scenes, Brian Green for helping with the implementation of the data aggregation in Hyperion, Karl Li and Lea Reichardt for the valuable discussions, feedback and integration support, Brent Burley and David Adler for code reviewing, and Lento Manickathan for compilation and integration support. We also thank Sergej Majboroda for the environment map used in Figure 2, and Ondrej Karlik for the Pool scene in Figure 6. Finally, we thank Sebastian Herholz for the support with the Open Path Guiding Library.

References

[BAC*18] BURLEY B., ADLER D., CHIANG M. J.-Y., DRISKILL H., HABEL R., KELLY P., KUTZ P., LI Y. K., TEECE D.: The design and evolution of disney's hyperion renderer. URL: https://doi.org/10.1145/3182159, doi:10.1145/3182159. 7

[BM97] BOLIN M. R., MEYER G. W.: An error metric for monte carlo ray tracing. In *Rendering Techniques' 97: Proceedings of the Eurographics Workshop in St. Etienne, France, June 16–18, 1997 8* (1997), Springer, pp. 57–68. 6

[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). doi:10/gg8xc7. 8

[DGJ*20] DIOLATZIS S., GRUSON A., JAKOB W., NOWROUZEZAHRAI D., DRETTAKIS G.: Practical product path guiding using linearly transformed cosines. In *Computer graphics forum* (2020), vol. 39, Wiley Online Library, pp. 23–33. 3

[DKHD24] DEREVIANNYKH M., KLEPIKOV D., HANIKA J., DACHSBACHER C.: Neural two-level monte carlo real-time rendering, 2024. arXiv:2412.04634. 8

[DPÖM22] DODIK A., PAPAS M., ÖZTIRELI C., MÜLLER T.: Path guiding using spatio-directional mixture models. *Computer Graphics Forum 41*, 1 (2022), 172–189. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14428, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14428, doi:https://doi.org/10.1111/cgf.14428. 3

[DWL23] DONG H., WANG G., LI S.: Neural parametric mixtures for path guiding. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3588432.3591533, doi:10.1145/3588432.3591533. 3, 7, 11

[DWWH20] DENG H., WANG B., WANG R., HOLZSCHUCH N.: A practical path guiding method for participating media. *Computational Visual Media 6*, 1 (Mar 2020), 37–51. URL: https://doi.org/10.1007/s41095-020-0160-1, doi:10.1007/s41095-020-0160-1. 3, 8

[GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen v3. http://eigen.tuxfamily.org, 2010. 7

[HD22] HERHOLZ S., DITTEBRANDT A.: Intel® open path guiding library, 2022. http://www.openpgl.org. 7, 8

[HEV*16] HERHOLZ S., ELEK O., VORBA J., LENSCH H., KŘIVÁNEK J.: Product importance sampling for light transport path guiding. *Computer Graphics Forum 35*, 4 (2016), 67–77. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12950, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12950, doi:https://doi.org/10.1111/cgf.12950. 3

[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). URL: https://doi.org/10.1145/3649310, doi:10.1145/3649310. 3

[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 (June 2019). URL: https://doi.org/10.1145/3230635, doi:10.1145/3230635. 3

[Jak10] JAKOB W.: Mitsuba Renderer, 2010. URL: https://mitsuba-renderer.org/. 5

[Kaj86] KAJIYA J. T.: The rendering equation. *SIGGRAPH Comput. Graph. 20* (1986). 1, 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv [cs.LG]* (Dec. 2014). arXiv:1412.6980. 7

[KD13] KAPLANYAN A. S., DACHSBACHER C.: Path Space Regularization for Holistic and Robust Light Transport . *Computer Graphics Forum* (2013). doi:10.1111/cgf.12026. 8

[KFF*15] KELLER A., FASCIONE L., FAJARDO M., GEORGIEV I., CHRISTENSEN P., HANIKA J., EISENACHER C., NICHOLS G.: The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (New York, NY, USA, 2015), SIGGRAPH '15, Association for Computing Machinery. URL: https://doi.org/10.1145/2776880.2792699, doi:10.1145/2776880.2792699. 1

[LZN*24] LI Y. K., ZHU C., NICHOLS G., KUTZ P., HUANG W.-F. W., ADLER D., BURLEY B., TEECE D.: Cache points for production-scale occlusion-aware many-lights sampling and volumetric scattering. In *Proceedings of the 2024 Digital Production Symposium* (New York, NY, USA, 2024), DigiPro '24, Association for Computing Machinery. URL: https://doi.org/10.1145/3665320.3670993, doi:10.1145/3665320.3670993. 7, 8

[MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph. 41*, 4 (July 2022), 102:1–102:15. URL: https://doi.org/10.1145/3528223.3530127, doi:10.1145/3528223.3530127. 7

[MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum 36*, 4 (jul 2017), 91–100. URL: https://doi.org/10.1111/cgf.13227, doi:10.1111/cgf.13227. 3, 5, 6

[MMR*19] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVÁK J.: Neural importance sampling. *ACM Trans. Graph. 38*, 5 (Oct. 2019), 145:1–145:19. URL: http://doi.acm.org/10.1145/3341156, doi:10.1145/3341156. 3, 7

[MRKN20] MÜLLER T., ROUSSELLE F., KELLER A., NOVÁK J.: Neural control variates. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–19. 3, 7, 11

[MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *ACM Trans. Graph. 40*, 4 (July 2021). URL: https://doi.org/10.1145/3450626.3459812, doi:10.1145/3450626.3459812. 8

[MRYS24] MEYER J., RATH A., YAZICI Ö., SLUSALLEK P.: Mars: Multi-sample allocation through russian roulette and splitting. In *SIGGRAPH Asia 2024 Conference Papers* (2024), pp. 1–10. 6

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in pytorch. 7

[RGH*22] RATH A., GRITTMANN P., HERHOLZ S., WEIER P., SLUSALLEK P.: Ears: Efficiency-aware russian roulette and splitting. *ACM Transactions on Graphics (TOG) 41*, 4 (2022), 1–14. 4, 5, 6, 7

[RHL20] RUPPERT L., HERHOLZ S., LENSCH H. P. A.: Robust fitting of parallax-aware mixtures for path guiding. *ACM Trans. Graph. 39*, 4 (aug 2020). URL: https://doi.org/10.1145/3386569.3392421, doi:10.1145/3386569.3392421. 1, 3, 7, 8, 11

[SHJD22] SCHÜSSLER V., HANIKA J., JUNG A., DACHSBACHER C.: Path guiding with vertex triplet distributions. *Computer Graphics Forum 41*, 4 (2022), 1–15. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14582, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14582, doi:https://doi.org/10.1111/cgf.14582. 3

[Tal05] TALBOT J. F.: *Importance resampling for global illumination*. Brigham Young University, 2005. 2, 3, 4, 5, 6

[Vea97] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997. 3, 4

[VK16a] VORBA J., KŘIVÁNEK J.: Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 1–11. 6

[VK16b] VORBA J., KŘIVÁNEK J.: Adjoint-driven russian roulette and splitting in light transport simulation. *ACM Trans. Graph. 35*, 4 (July 2016). URL: https://doi.org/10.1145/2897824.2925912, doi:10.1145/2897824.2925912. 8

[VKv*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). 3

[XHM*24] XU K., HERHOLZ S., MANZI M., PAPAS M., GROSS M.: Volume scattering probability guiding. *ACM Trans. Graph. 43*, 6 (Nov. 2024). URL: https://doi.org/10.1145/3687982, doi:10.1145/3687982. 8

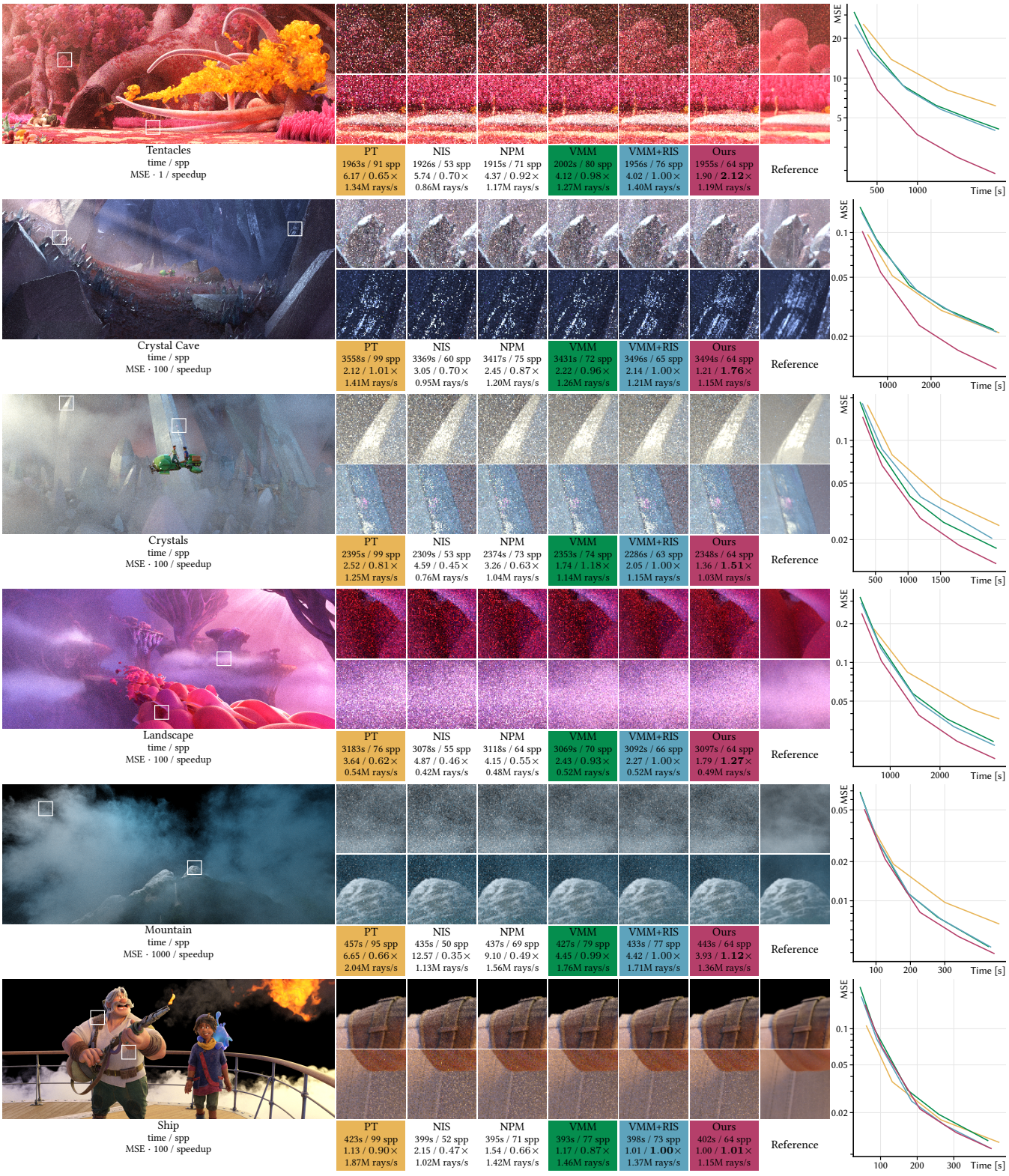


Figure 8: *Equal-time renders of production scenes with path tracing (PT), neural guiding (NIS [MRKN20] and NPM [DWL23]), state-of-the-art non-neural guiding (VMM [RHL20], both with and without RIS), and our proposed method.* (continues in Figure 9) © *Disney*

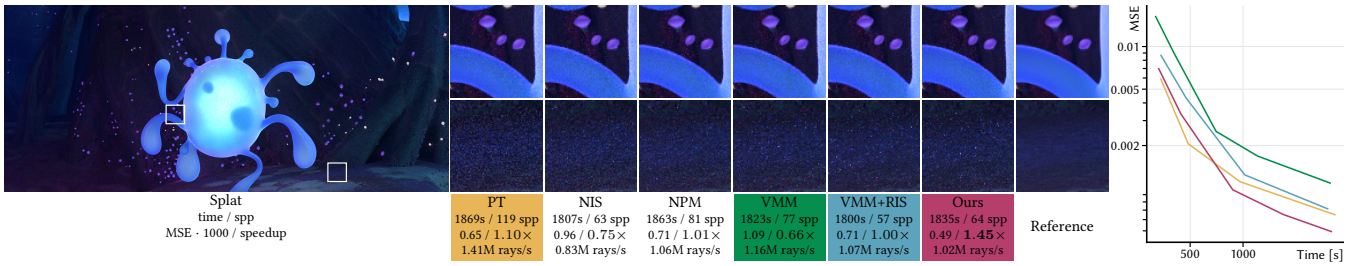


Figure 9: (continued from above) © *Disney*

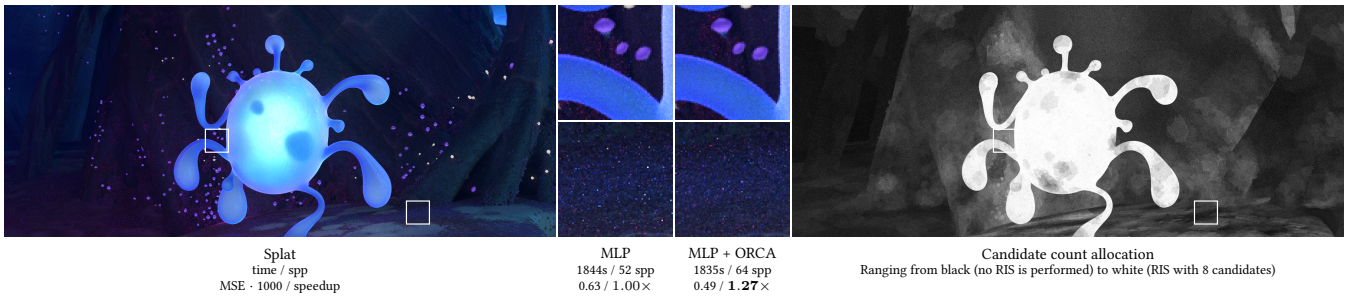
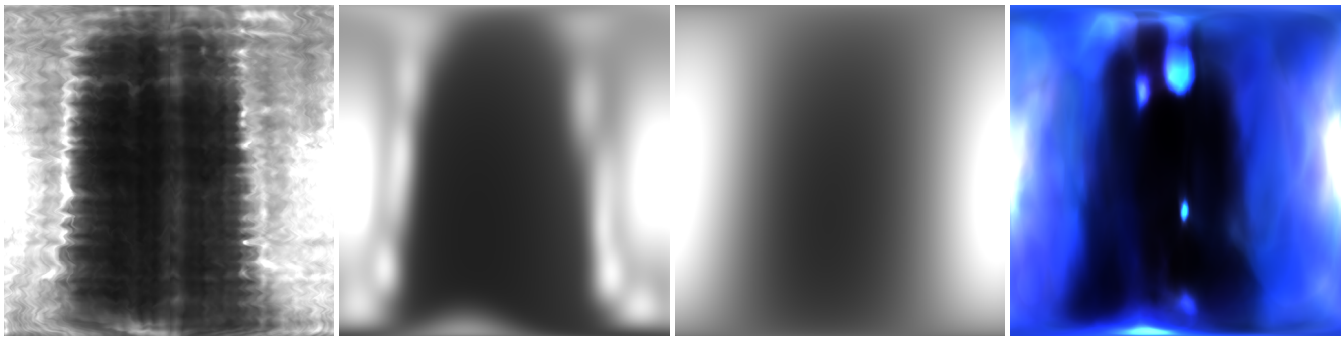


Figure 10: *Demonstrating the benefit of optimized candidate count allocation (ORCA) in the* SPLAT *scene. In this scene, the MLP performs poorly on the floor, which is remedied by ORCA disabling RIS there. On splat itself, sampling according to the product is important due to splat's translucent skin, which ORCA detects and accordingly assigns higher candidate counts to. © Disney*



(a) *NIS* **(b)** *NPM* **(c)** *VMM* **(d)** *Our MLP*

Figure 11: *Comparing the distributions learned for a point on splat's skin.* ***(a)*** *Neural importance sampling reconstructs some of the finer details (such as one of splat's arms in the top center), but suffers from wave artefacts due to noise present in the training samples.* ***(b)*** *Neural parametric mixture models are less sensitive to training noise and learn a smooth representation that can be sampled efficiently, but thereby blur finer structures.* ***(c)*** *Despite using the same underlying parametric mixture model as NPM, non-neural VMM distributions are less effective at spatial information sharing and thus blur out even more structures.* ***(d)*** *Our small MLP reconstructs most structures (such as multiple arms), and can also predict the color of both the outside and splat's interior, making our method useful for other tasks that depend on radiance estimates (such as radiance caching or Russian roulette). © Disney*