EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Neural Progressive Photon Mapping

Justin Benoist[1] Joey Litalien[2] Adrien Gruson[1]

[1]École de Technologie Supérieure, Canada
[2]Independent Researcher

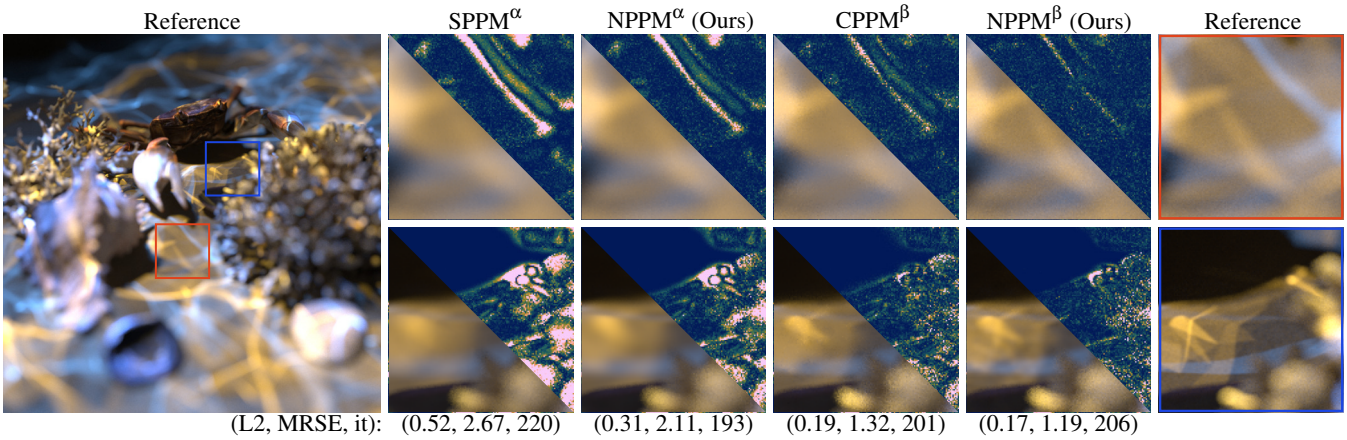

Figure 1: *Equal-time comparison (15 seconds) between our neural progressive photon mapping (NPPM), SPPM [HJ09] and CPPM [LLZ*20] on the* CRAB DOF *scene with depth of field. The superscripts* α *and* β *respectively refer to the different radius reduction policy used by the two baseline methods, which we incorporate atop NPPM. Our technique reduces the overall bias compared to its nonneural counterparts, capturing sharper caustics on most of the scene. False color error shows the MRSE metric.*

Abstract

Photon density estimation is a robust solution for estimating complex light transport, such as those involving caustics and pure specular interactions. The shape and bandwidth of the density kernel are both crucial in achieving optimal performance. Recently, density kernels directly predicted by neural networks from local photon statistics have shown improved reconstruction results for small numbers of photons. The direct weight prediction approach of these methods, however, is fundamentally incompatible with consistent estimators as it does not allow for direct control over bias and variance. We address this problem by relying on a simpler yet effective analytical kernel, also inferred by a neural network. Unlike prior work, our technique supports progressive schemes by design, hence unlocking a large variety of applications such as stochastic photon mapping. Our method is fast, trivial to train and demonstrates state-of-the-art caustics reconstruction at equal-time over other photon mapping techniques.

CCS Concepts

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

1. Introduction

Light transport simulation aims to recreate the complex interactions between light and matter in virtual environments. These interactions, especially *caustics*—like light focusing through a whisky glass or sunrays slowly dancing on coral reefs—are crucial to photorealism, especially in architectural visualization and visual effects. Caustic light paths formed from long specular chains present unique challenges to classical Monte Carlo algorithms: not only are they difficult to discover but they are also hard to explore efficiently once found. Consequently, specialized rendering techniques are required to achieve acceptable noise levels in production. Such methods focus, for instance, on modeling and solving the intricate geometric constraints of such paths [ZGJ20, FGW*24, FWW*25].

An alternative approach to traditional methods like path tracing

is to employ spatial relaxation [Jen96], where the physical constraints of specular chains are softened to enable path reuse through nearby clustering. Density estimation represents one such family of techniques and offers an attractive balance between implementation simplicity and robustness. A significant practical challenge, however, lies in determining the appropriate spatial relaxation kernel parameters. This in turn introduces an inherent trade-off between kernel shape and bandwidth, where one must maximize path reuse while also minimizing the bias introduced in the underlying estimator. As a result, the process of selecting optimal kernel settings tends to be laborious and context-dependent as it must account for local and arbitrarily complex radiometric properties such as photon distribution patterns in a given scene.

To mitigate this problem, prior works have proposed adaptive and progressive density estimation methods based on statistics gathered during the rendering pass [KD13, SFES07, KWX*16, PAMM19]. In particular, deep photon mapping (DPM) [ZXJ*20] employs K-nearest neighbor (K-NN) and a small neural network to predict weights for each photon, demonstrating impressive results over standard photon mapping approaches. Intuitively, DPM learns to adapt to the local context by exploiting statistics from collected nearby photons. Unfortunately, this method cannot be combined with consistent estimators as the kernel bandwidth and bias cannot be directly controlled. This lack of control prevents applications to a wider range of rendering methods [MBGJ22, WGH22, HPJ12, GKDS12].

In light of these observations, we introduce neural progressive photon mapping (NPPM), a learning-based framework that predicts spatially adaptive kernels with progressive rendering in mind. Our network generates *parametric and normalized* kernels that can be restricted at will during the iterative rendering process. Since we do not predict per-photon weights, our architecture eliminates the need to compute and store deep context features for each photon. Consequently, our method is not only simpler than DPM but also fully controllable. Moreover, our design allows the use of a 3D grid to average latent photon representations over time, enabling further optimizations. Finally, we show how to effectively train our model while ensuring compatibility with radius range queries, hence introducing the first learning-based *progressive* photon mapping method to the rendering community.

To summarize, our contributions are:

- a simple and lightweight parametric neural model that can efficiently adapts to the local photon distribution;
- a controllable density kernel scheme compatible with progressive rendering methods; and
- an expressive latent representation that can be averaged and stored on the scene surfaces.

2. Related Work

Photon mapping. Jensen's seminal work on density estimation [Jen96] has shown great promise in rendering complex light phenomena like caustics. Photon mapping (PM) is now standard in light transport simulation and has been adopted by production renderers like the Corona renderer [ŠK19]. PM has been successfully adapted to BSSRDF rendering [JMLH01], volume photon mapping [JC98], and beam radiance estimation [JZJ08], and can also be combined with other path construction techniques [GKDS12, HPJ12, KGH*14] via multiple importance sampling [Vea98] for improved robustness.

A key consideration in practice is the bias-variance trade-off of photon mappers. The amount of bias can be reduced by incorporating ray differential information [SFES07], local radiance gradient estimates [KWX*16], more flexible (but generally more costly) kernels [PAMM19] or progressive kernel bandwidth reduction schemes [HOJ08, HJ09, KZ11]. Since these kernel reduction techniques can be sensitive to a nonoptimal initial radii, the overall convergence of the estimator may suffer dramatically, leading to either excessive bias or variance in the final image. A mitigation measure is to derive improved progressively shrinking strategies based on derivative information [KD13] or χ^2-statistics [LLZ*20]. We focus on enabling progressive estimation through large, controllable kernels while simultaneously minimizing bias. Our method is compatible with different radius reduction policies and surpasses the performance of prior works.

Note that bias in density estimation can be further removed based on Bernoulli trials [QSH*15] or telescopic series [MBGJ22], at the cost of a slight increase in variance. We see these techniques as orthogonal to ours and we expect our method to be compatible due to our controllable kernel support. This control potentially allows our method to be applied to path filtering algorithms, which perform local gathering with range queries [KDB16, WGGH20] or local clustering [DHC*21], to reduce variance. Alternatively, methods for guiding photon's emission [GPGSK18, EK20] or their entire path [HJ11, CWY11, GRŠ*16, vOHK16] can similarly be used in our method to improve the overall photon distribution.

Neural methods for density estimation. Learning-based approaches have proven successful in many areas of rendering, including Monte Carlo image denoising [CKS*17, IMF*21, HY21]. Such reconstruction techniques can be adapted to density estimation [CM21] by extracting and combining correlated images with different biases. We can incorporate these techniques into NPPM, as we can bound the bias we introduce.

Photon mapping has also been applied to path guiding by binning photon contributions [Jen95]. Recently, neural methods have been used to denoise these distributions [ZXS*21b], even when they are stored in a quadtree [ZXS*21a]. Such methods are generally more memory intensive and require complex spatial partitioning strategies. Alternatively, neural networks can directly predict distribution in path guiding [MMR*19] based on the framework of normalizing flows [KPB21]. However, due to their complex architecture, these approaches are often too expensive except when targeting lower-dimensional distributions [LHL*24]. Simpler, online neural guiding methods can also optimize for mixtures of von Mises–Fischer [DWL23] or anisotropic Gaussian [HIT*24] distributions. Our work is inspired by these parametric models as they are both more computationally efficient and expressive enough for density estimation.

To the best of our knowledge, only Zhu et al. [ZXJ*20] have employed neural networks for density estimation in rendering. Their

proposed DPM approach leverages a PointNet-like architecture [QSMG17] to directly regress photon weights, resulting in neural kernels that outperform previous non-neural density estimators at different photon sampling rates. DPM does not support a varying number of nearest neighbors and, importantly, cannot be directly used in a progressive estimator. NPPM addresses both these problems by providing an estimator that is consistent by construction.

3. Background

Density estimation. In surface rendering, we are interested in computing the exitant radiance L_o at a location $\mathbf{x}$ in the outgoing direction $\boldsymbol{\omega}_\mathrm{o}$:

$$L_\mathrm{o}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o}) = \int_\Omega f_\mathrm{r}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o},\boldsymbol{\omega})\, L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega})\,(\mathbf{n}\cdot\boldsymbol{\omega})^+ \,\mathrm{d}\boldsymbol{\omega}\,, \tag{1}$$

where f_r is the BRDF, L_i the incoming radiance in direction $\boldsymbol{\omega}\in\Omega$, and $\mathbf{n}$ is the surface normal at $\mathbf{x}$. As proposed in classical photon mapping [Jen96], this exitant radiance can be formulated as a kernel density estimation (KDE) problem on the photon map:

$$\langle L_\mathrm{o}\rangle_N^\mathrm{PM} = \frac{1}{N}\sum_{n=1}^{N} Q_r(\|\mathbf{x}_n-\mathbf{x}\|)\, f_\mathrm{r}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o},\boldsymbol{\omega}_n)\,\Phi_k\,, \tag{2}$$

where N is the total number of emitted photons, r is the search radius, $\mathbf{x}_n$ is the position of the n-th photon with flux Φ_n and incoming direction $\boldsymbol{\omega}_n$. The flux includes the throughput of the photon. The normalized kernel Q has support within radius $r>0$ only, allowing fast spatial query through specialized data structure or by using ray tracing hardware routines [KBG23]. In general, this spatial kernel is chosen to be simple and radial; for example $Q_r(d) = (\pi r^2)^{-1}$ if $d<r$ otherwise 0.

Deep density estimation. Zhu et al. [ZXJ*20] replace parametric kernels with a neural network that directly regresses a weight for each collected photon. Under this framework, a deep context vector $\mathbf{c}\in\mathbb{R}^c$ is inferred from the nearby collected photons using K-nearest neighbors at the gather point location $\mathbf{x}$:

$$\begin{aligned}\boldsymbol{\rho}_k(\mathbf{x}) &\coloneqq \left[\mathbf{x}'_k,\boldsymbol{\omega}_k,\Phi'_k\right],\\ \mathbf{c}(\mathbf{x};\boldsymbol{\theta}) &= \mathrm{Pool}_K\left(\left\{F(\boldsymbol{\rho}_k(\mathbf{x});\boldsymbol{\theta})\right\}_{k=1}^K\right).\end{aligned} \tag{3}$$

Here, $\boldsymbol{\rho}_k$ represents the concatenated photons' input features including the normalized position within the kernel support tranformed in tangent space $\mathbf{x}'_k$, the photon's incoming directions $\boldsymbol{\omega}_k$ and the associated tonemapped flux Φ'_k. Each photon is encoded independently by a feature extractor network F with trainable parameters $\boldsymbol{\theta}$. To ensure input order invariance, the deep context vector is computed by average/max pooling across all K photons, following the design principles of PointNet [QSMG17]. The resulting deep context vector (DCV) is then decoded by a separate network D with trainable parameters $\boldsymbol{\varphi}$, substituting the kernel function in Eq. (2) to yield the estimator:

$$\begin{aligned}\langle L_\mathrm{o}\rangle_K^\mathrm{DPM} &= \frac{1}{N\pi r^2}\sum_{k=1}^{K} w_k\, f_\mathrm{r}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o},\boldsymbol{\omega}_k)\,\Phi_k,\\ w_k &= D\big([\boldsymbol{\rho}_k(\mathbf{x}),\mathbf{c}(\mathbf{x},\boldsymbol{\theta})];\boldsymbol{\varphi}\big).\end{aligned} \tag{4}$$

A per-photon weighting scalar $w_k>0$ is predicted and the normalization factor $(\pi r^2)^{-1}$ ensures that it remains scale-invariant with respect to the K-NN radius.

Both networks are small multilayer perceptrons (MLPs) that are trained jointly to minimize the L^2-norm between the estimated radiance from Eq. (4) and the target radiance $L_\mathrm{o}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o})$ obtained from converged references. The error can stem from two sources: a poor kernel (e.g., geometric discontinuities within its support), which DPM is designed to handle, or too few photons to suppress variance. In the latter case, the incompatibility of DPM with the progressive frameworks blocks further variance reduction.

Progressive density estimation. Progressive photon mapping (PPM) [HOJ08, HJ09, KZ11] introduces the concept of averaging density estimates through multiple iterations. At each iteration, the radius r is reduced to form a consistent estimator. Knaus and Zwicker [KZ11] propose with APA to average the different density estimates at each pixel and update the kernel's radii as

$$r_{i+1}^2 = \frac{i+\alpha}{i+1}\, r_i^2, \qquad i\in\mathbb{Z}_{\geq 0}, \tag{5}$$

where $\alpha \coloneqq 2/3$ is a user-defined parameter to balance bias reduction and variance increase. The initial radius r_0 can be set to a constant user input, optionally scaled by the scene's extents, camera ray differential or K-NN estimates on an independent photon map. Alternate radius reduction schemes also exist [KD13, LLZ*20]. We also utilize CPPM's radius reduction policy which applies a threshold on the minimum number of gathered photons ($N_\mathrm{min}=10$) before reducing the radius by a constant factor ($k=\sqrt{0.8}$). To ensure a consistent estimator, the minimum photon count is then increased at each radius reduction step by a constant factor $\beta = 1.2$. We add subscripts NPPM^α and NPPM^β to denote which radius reduction strategy (APA's or CPPM's, respectively) is used in our results.

3.1. Motivation

The application of photon density estimation often involves a delicate balance between minimizing both variance and bias. Reducing variance can be achieved by averaging more photons per gather point by increasing the photon density via more aggressive photon tracing or by using larger density kernels. These solutions can be impractical and inefficient when implemented without extra care:

- **Larger kernels** effectively increase the number of reused paths at each gathering point, amortizing the cost of the photon tracing pass. However, these kernels can introduce additional bias if their shapes do not adapt to the underlying photons' density profiles. DPM address this by directly learning to predict the photon's weights. This relies on K-NN for collecting photons and does not offer a mechanism to compute the kernel normalization factor. This is problematic when trying to control the bias introduced.
- **Tracing more photons** naturally leads to memory problems. To solve this, iterative methods can be employed to average photon passes. This iterative strategy can be combined with radius reduction to make the estimator consistent.

Our main objective is thus to design a deep adaptive kernel that can be combined with iterative radius reduction. To this end, we

need our adaptive kernel to be *controllable*, implying we can express it at *any* bandwidth value. This in turn requires our method to support range queries rather than K-NN. Our method must also be lightweight as a massive amount of kernel density estimates will be performed throughout the rendering process. Finally, our method must retain information across iterations to ensure stable and precise predictions over time.

4. Neural progressive photon mapping

We now introduce our neural progressive photon mapping (NPPM) algorithm and detail how consistency is enforced. Similar to DPM [ZXJ*20], our approach modifies only the kernel evaluation, leaving the rest of the photon mapping architecture unchanged. We first describe the model architecture, including how our adaptive kernel is normalized (Section 4.1). Next, we present a practical application of our framework using 2D anisotropic Gaussian kernels (Section 4.2). Finally, we discuss practical considerations (Section 4.3).

4.1. Model architecture

Figure 2 illustrates our framework. Similarly to DPM, we first preprocess photon features to construct the input to our model. We choose $\boldsymbol{\rho}_k(\mathbf{x}) = [\mathbf{x}'_k]$ only. We express $\mathbf{x}'_k$ in tangent space (with $z = 0$ for the surface plane) and use only this feature, as adding photon directions or flux did not improve convergence in our experiments. For instance, applying Russian roulette, photon flux remains nearly constant across photons, rendering this feature uninformative. We also discard the z component of $\mathbf{x}'$, since retaining it provided no measurable benefit. We encode each photon's statistics using a lightweight feature extractor network F. These latent features are subsequently aggregated through both average/max pooling layers to construct the deep context vector $\mathbf{c}$, effectively representing the gathered photon statistics in a shared latent space. We augment this representation with a global statistic ν (described in Section 4.3), which encodes the overall photon density within the kernel. A second network D then predicts our parametric kernel. While this applies to arbitrary kernel families, we show that iso/anisotropic Gaussian densities are sufficient candidates.

Kernel normalization estimator. Based on the predicted kernel parameters, we compute a normalization factor to control the bias in our method. We restrict the kernel domain to maintain this control by using the 2D disk $\mathcal{B}_r$ of radius $r > 0$. However, using a given world-space radius makes our method scale-sensitive and hinders its generalization across different scenes. We address this by reformulating our adaptive kernel Q in a normalized space where $r = 1$. This reparametrization yields the following normalization factor

$$Z := \left(\int_{\mathcal{B}} Q(\mathbf{x}; \mathbf{c}, \boldsymbol{\varphi}) \, \mathrm{d}\mathbf{x} \right)^{-1}, \tag{6}$$

which *reciprocal* can be estimated via Monte Carlo integration as

$$\langle Z^{-1} \rangle_M = \frac{1}{M} \sum_{m=1}^{M} \frac{Q(\mathbf{x}_m; \mathbf{c}, \boldsymbol{\varphi})}{p(\mathbf{x}_m)}, \qquad \mathbf{x}_m \sim p, \tag{7}$$

where p denotes the probability density function. Note that, because we estimate the reciprocal, Eq. (7) can introduce bias in the true normalization factor's estimate. This bias can be reduced to an negligible level with sufficient samples or by applying debiasing techniques [QSH*15, MBGJ22]. Since both methods incur significant overhead, a more practical solution is to precompute and tabulate the normalization factor when it is smooth and the kernel has few parameters.

Parametric density estimator. We use our normalized adaptive kernel to perform density estimation. Writing $\boldsymbol{d}_k := \mathbf{x}_k - \mathbf{x}$ as the 2D displacement with respect to the gather point, our estimator is

$$\langle L_\mathrm{o} \rangle_K^{\mathrm{NPPM}} = \frac{Z}{N \pi r^2} \sum_{k=1}^{K} Q(\boldsymbol{d}_k / r) \, f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{\omega}_k) \, \Phi_k, \tag{8}$$

where the factor πr^2 accounts for the radius rescaling. Note that our estimator is similar to the KDE formulation on the photon map (Eq. (2)). More importantly, it differs from DPM's estimator (Eq. (4)) in two key aspects:

- We decode the parametric kernel values using the DCV *only once*. This approach is significantly more efficient than performing an MLP inference for each photon. This also allows our method to amortize the cost of computing the DCV by exploiting its spatial redundancy in 3D.
- Our kernel's bias is controllable by restricting its domain. In addition, the kernel can be easily normalized through precomputation. This eliminates the need to jointly learn the kernel's shape and its normalization, enabling faster and more stable training.

4.2. Application: 2D truncated anisotropic Gaussian kernel

Kernel formulation. A suitable kernel candidate for our method is a 2D anisotropic Gaussian centered at the gather point. In the normalized space, this Gaussian has zero mean. Inspired by the formulation from 3D Gaussian Splatting [KKLD23], we form a covariance matrix Σ by constructing a 2D scaling matrix $S(\mathbf{c}, \boldsymbol{\varphi})$ and rotation matrix $R(\mathbf{c}, \boldsymbol{\varphi})$:

$$\Sigma(\mathbf{c}, \boldsymbol{\varphi}) = RSS^T R^T, \tag{9}$$

ensuring that Σ is symmetric and positive semi-definite. Our predicted kernel is then defined as

$$Q_\Sigma^{\mathrm{2DG}}(\boldsymbol{d}) = \mathbf{1}_{\mathcal{B}}(\boldsymbol{d}) \cdot \mathcal{N}(\boldsymbol{d}; \mathbf{0}, \Sigma) \big/ Z, \tag{10}$$

where $\mathbf{1}_{\mathcal{B}}$ is the indicator function on the unit disk and we have dropped the dependency on the deep context vector and the decoder parameters for simplicity. Intuitively, our kernel resembles an anisotropic Gaussian that gets truncated at unit radius (Fig. 2).

Kernel normalization. Since our predicted covariance matrix Σ is determined by three parameters but the reciprocal normalization factor (Eq. (7)) is invariant under rotation due to the disk-shaped domain $\mathcal{B}$, the look-up table (LUT) only needs to be two-dimensional. We reparametrize our scale in log-space to get more precision for small scale values. The full LUT takes at most 4 MB of GPU memory.

Kernel hyperparameters. We construct our rotation matrix R by predicting an angle ξ and rescaling by 2π. The scale matrix S is built by predicting two scaling factors $s_1, s_2 > 0$. To ensure numerical

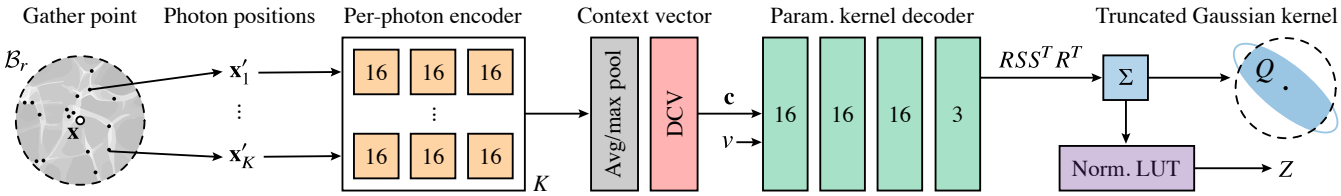

Figure 2: *Our NPPM architecture. We gather nearby photons in a given radius and compute per-photon features using a lightweight MLP encoder F. The resulting features are then pooled into a deep context vector (DCV). This embedding is consumed by a decoder network D, alongside the gathered density statistics, to produce the parameters (rotation and scale) of the 2D covariance matrix of a truncated Gaussian distribution. Thanks to the symmetric nature of this kernel, we can precompute the corresponding normalization factor up to a rotation in a small look-up table, hence enabling exact density estimation.*

stability and prevent degenerate scales, we use an ELU activation with $\alpha = 1$ and $\epsilon = 1.0001$:

$$s_i = \epsilon + \mathrm{ELU}(z_i - 2) = \epsilon + \begin{cases} z_i - 2, & \text{if } z_i \geq 2 \\ \exp(z_i - 2), & \text{if } z_i < 2 \end{cases} \tag{11}$$

with z_i being the i-th output of the network. We clamp the scale value to a maximum value of $\sqrt{3}$, which makes the predicted Gaussian corresponds roughly to a box kernel. We fix these hyperparameters across all experiments.

4.3. Practical considerations

Implementing our algorithm efficiently requires addressing several key considerations. This section examines how to support range queries during both training and inference phases of our method. We also demonstrate how our network design facilitates caching and reuse of latent representations across different rendering iterations and scene locations, hence enhancing computational efficiency.

Range versus nearest neighbor queries. DPM [ZXJ*20] uses K-NN since it does not need to control bias. Additionally, K-NN simplifies implementation by ensuring that a constant number of photons is processed by the network, making both training and inference more parallelizable: since the number of photons per gather point is fixed, the memory can be preallocated efficiently.

Our method is also compatible with nearest neighbors and can be trained using this density estimation strategy. As our domain is restricted to a disk $\mathcal{B}_r$ however, we require range queries instead. This results in a *variable* number of input photons, which complicates batch training. To address this, we impose a fixed upper bound on the number of photons collected. This leads to two scenarios:

I. **Fewer photons than the maximum:** We mask the missing photons by zeroing out their encodings and setting their associated weights in the predicted kernel to zero. This introduces bias in the average pooling. To compensate, we apply a correction factor $\zeta = B/K_{\text{matches}}$ to the mean vector, where B is the fixed batch size (i.e., the maximum number of photons) and K_{matches} is the actual number of photons gathered. Max-pooling remains unaffected by this change.
II. **More photons than the maximum:** We uniformly resample the photons to match the desired count. This can be efficiently implemented using reservoir sampling, avoiding the need to store the full list of gathered photons. We also store ζ as it is an easily accessible and valuable statistic to inform our decoder of the relative photon density at the gather point location.

Computing the deep context vector In addition to the generated deep context vector $\mathbf{c}$, we append a compressed representation of the relative photon density, defined as $\nu = \log(1+\zeta)$, to avoid saturating $\mathbf{c}$. This statistic is inexpensive to obtain, varies smoothly across the scene and can guide the decoder towards more aligned kernels.

Deep kernel evaluation After constructing the deep context vector $\mathbf{c}$ with ν, we decode the parameters of our normalized parametric kernel. Using these parameters, we query the corresponding normalization factor from our LUT and evaluate the KDE estimator (Eq. (8)). This estimator can either be applied to the (resampled) subset of photons used to form the network input or to all photons within the gather point radius via a separate range query. Both approaches have trade-offs: reusing the resampled photons reduces computation, but increases variance, especially when the predicted kernel is narrow. We found that performing an additional range query is more advantageous in practice, as it does not introduce extra variance and its cost can be amortized using a spatial latent averaging strategy, which we describe next.

4.4. Spatial latent averaging

Our approach naturally allows for decoupling the encoding and decoding of the deep context vector. To this end, we maintain a latent representation at each pixel and average these representations over multiple iterations. This approach allows the kernel's adaptive properties to gradually decay as rendering proceeds. While the encoder network is most computationally expensive (as it is applied independently to each of the B photons selected per gather point), the decoding network is invoked only once per gather point. As such, an opportunity arises to amortize the DCV encoding operation across gather points and iterations.

By exploiting the smoothness of the photon distribution over the scene surfaces, we propose to combine our method with a 3D grid that store and access these latent representations. The grid size is

Figure 3: *Examples of our procedurally generated training dataset. Gather points travelling through dielectric surfaces are masked out to avoid excessive noise during training.*

initialized according to the largest axis of the scene's bounding box. At each iteration, we perform the following two steps:

1. Collect photons at a subset of gather points, encode them, and average their resulting DCVs along with the density statistic ν inside the 3D grid. To avoid aliasing, we apply stochastic jittering to the gather point locations before splatting [BFK20].
2. Retrieve the averaged deep context vector from the grid, decode the corresponding adaptive parametric kernel, and perform kernel density estimation using a range query (Eq. (8)).

Averaging the spatial DCV can be done using a small exponential smoothing factor ($\gamma = 0.5$) or by averaging over all iterations. We found that averaging over all iterations gives slightly better results than exponential smoothing. We leave the investigation on the best policy to update our spatial grid as future work. This grid approach achieves better performance compared to predicting the DCV at every iteration for all gather points, as it reduces noise in both the produced DCV and the density statistic ν, as well reduce our overall cost. Sharp discontinuities in the photon density can introduce bias; however, this bias remains bounded due to the underlying properties of our kernel and will vanish during the progressive rendering process.

Early stopping To further reduce overhead, we stop updating the latent representations after 50 iterations. This strategy is similar to that used in neural path guiding [DWL23,HIT*24]; however, unlike neural path guiding—which relies on online training and maintaining computation graphs—our method only performs MLP evaluations per photon for encoding.

Surface normal test During photon gathering (both for producing the DCV or performing KDE (Eq. (8)), we reject photons whose surface normals significantly deviate from that of the gather point. This filtering strategy is common in traditional photon mapping to reduce bias. In our setting, we found that applying this normal-based filtering helps the network focus on more relevant regions of the scene, rather than over-emphasizing geometric discontinuities such as edges, creases and corners. Our adaptive kernels account for these regions by predicting better suited kernels.

4.5. Training

Dataset generation. Our training set consists of 50 procedurally generated scenes rendered at a resolution of 256×256. Figure 3 shows a few representative examples. In all scenes, we isolate and compute only the caustic component of light transport to let the network focus on complex and high-frequency photon distributions typical of caustics. Reference images are generated using progressive photon mapping [HOJ08]. Range queries are performed at runtime to augment our data with random photon maps. We uniformly pick radii in fixed intervals to add variability in the density of photons. We set the maximum number of photon to $K_{\max} = 1000$ and apply our masking and resampling strategy from Section 4.3. We use a high value here to mitigate the noise on training gradients caused by the resampling step. In total, our dataset is composed of more than 3.2 million supervised training samples.

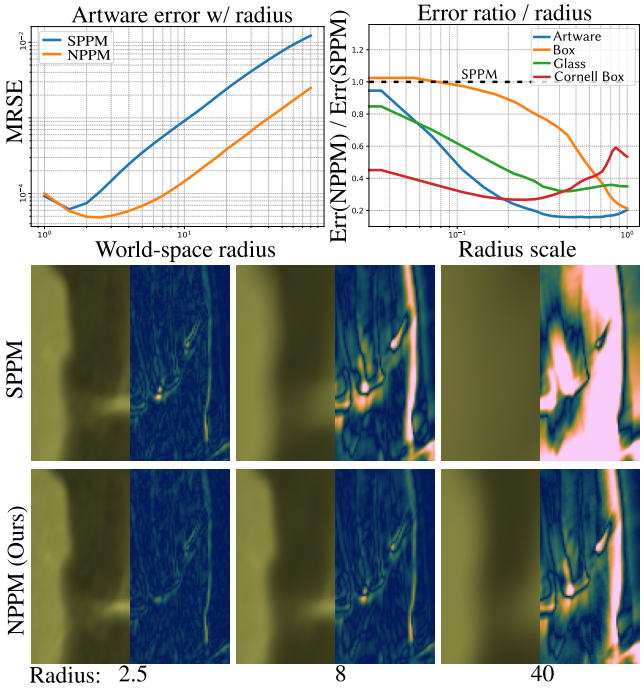

Figure 4: *We plot and visualize the MRSE error for SPPM and our method across various radius scales. Our method consistently outperforms the box kernel, with diminishing returns at smaller radii.*

The scenes themselves are randomly generated from an interior cube. We assign random albedo colors to each diffuse wall and place a rectangular area light source with random orientation, scale, and position. We then insert multiple dielectric for caustics and diffuse objects for shadows and non-flat surfaces shading scenarios. We apply random bump mapping to all dielectrics, producing a wide variety of intricate light patterns. We further randomize the position and look-at direction of the camera.

Encoder pre-training. To stabilize and accelerate training, we pretrain our encoder network $F(\boldsymbol{\theta})$ within a variational autoencoder (VAE) [KW14]. The VAE is trained to capture and reconstruct the input photon distribution, thereby promoting an expressive deep context vector (DCV) that generalizes across a wide range of photon distributions. To reuse the pretrained encoder in our kernel adaptation network (Fig. 2), we predict the mean and variance of $K = 32$ Gaussians through a single linear projection from the DCV.

New photons are then generated by sampling from these Gaussians and decoding their positions with a small MLP. The VAE is optimized for 300 training steps on small subsets of the dataset by minimizing the Chamfer distance between the generated and input photons. In practice, we found it more effective to freeze the encoder and use it directly to produce the DCV fed into our kernel decoder $D(\boldsymbol{\varphi})$, rather than retraining it from scratch.

Model hyperparameters and optimization. Our feature extractor network $F(\boldsymbol{\theta})$ is a 3-layer MLP with Tanh activations, while our decoder $D(\boldsymbol{\varphi})$ shares the same characteristics. We use $c = 32$ dimensions for our deep context vectors.

We use the Adam optimizer [KB15] with a learning rate of 10^{-4}. To manage memory usage during training, we subsample 50% of the image pixels, as backpropagation over all gathered photons exceeds our GPU memory capacity. We minimize the L^2-loss computed between the predicted and target radiance values, both passed through the compression function

$$L \mapsto \log(1+\mu L) \big/ \log(1+\mu), \tag{12}$$

where $\mu = 5000$, following Zhu et al. [ZXJ*20]. We initialize the network to produce small kernels at the start of training, as we observed that large initial kernels often cause the optimization to converge to poor local minima. We optimize our model for only 10 epochs on an NVIDIA RTX 3080 GPU. This takes roughly 90 minutes, a stark contrast with DPM reporting two days of training.

5. Results

Implementation details. We use Mitsuba 3 [JSR*22] to perform the eye pass and the photon tracing pass. We then execute our radius search on the GPU using a custom CUDA implementation of fixed-radius nearest neighbors search by Rama [Ram14]. We use PyTorch [PGM*19] and fully-fused MLPs [M21] for acceleration during training. All our test references are computed with stochastic progressive photon mapping (SPPM) [HJ09] to have fresh gather points at every iterations. We reimplemented DPM [ZXJ*20] as no open source codebase is available. Similarly, we reimplemented CPPM [LLZ*20] in Mitsuba 3. All learning methods are trained on our dataset described in Section 4.5. We will release our implementation and our generated dataset upon publication.

To evaluate our approach we use Mean Squared Error (L^2) and Mean Relative Squared Error (MRSE) with $\varepsilon = 10^{-2}$. We also used the Symmetric Mean Absolute Percentage Error (SMAPE) to evaluate fireflies in the unbiased application. All results were obtained under equal iteration counts or equal runtime conditions. For the photon encoding, we use range query with 50 maximum photons. SPPM, CPPM, and our NPPM use all photons. We trace 2M photons / iteration for the CRAB DOF and CRAB scenes (Figs. 1, 6 and 11); all other scenes use 400K photons / iteration. For glossy materials, we use 0.1 as minimum roughness to deposit photons and gather points. No guiding is performed during the tracing. DPM contains a total of 6.2K parameters, whereas ours only has 2.5K parameters.

Sensitivity to range query size. Figure 4 shows how our method is more robust to the scale of the range query compared to the non-adaptive box kernel from SPPM. All closeup images are produced by averaging multiple density estimates to make the bias more apparent. The improvement of our method diminishes with smaller kernels, unsurprisingly, as the assumption of uniform density holds in this regime. When the radius falls below a certain threshold (1.0 ray differential radius), we switch to a box kernel to maintain numerical stability and reduce computational cost.

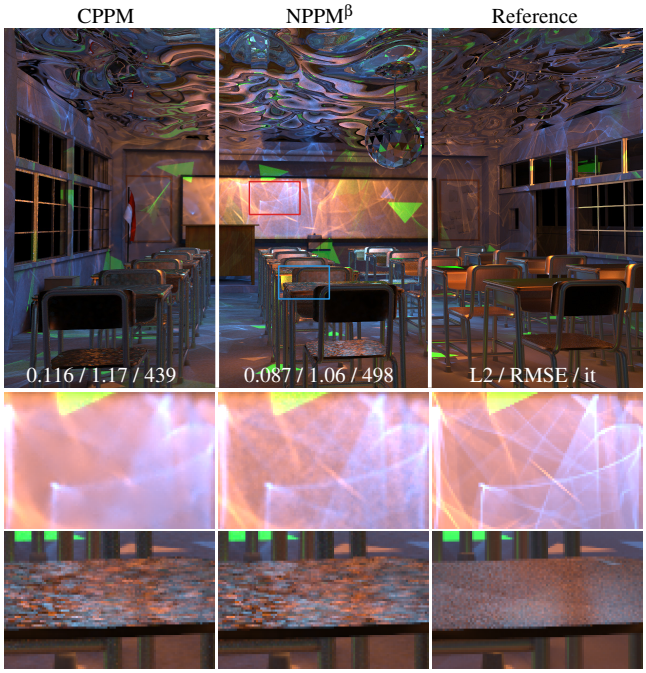

Figure 5: *Equal-time comparison over five minutes of CPPM [LLZ*20] and NPPM$^\beta$ in the* CLASSROOM *scene. The scene contains several glossy materials, including the whiteboard, chairs, and desks. Glossy surfaces are challenging for photon mapping methods, particularly at grazing angles such as those observed on the chair seats and desks.*

Consistent method comparions. Figure 6 shows the comparison between our methods with anisotropic 2D kernels, SPPM [HJ09] and CPPM [LLZ*20]. We omit DPM [ZXJ*20] since it cannot be made consistent. CPPM has a different reduction policy based on the χ^2-test, which often reduces the radius faster than SPPM, hence reducing the overall bias. Our method NPPM$^\alpha$ uses the same reduction policy as APA, so that each iteration gather the same set of photons. Due to a more expressive kernel, our method can achieve improved convergence with significantly lower bias. We also include results for our method NPPM$^\beta$. This more aggressive reduction strategy reduces bias by employing smaller bandwidths. Unlike CPPM, this strategy does not rely on a statistical test; instead, the radius is reduced geometrically once a minimum photon count is reached. The rendered images in Fig. 6 show results comparable to CPPM. In scenes with depth of field effects, such as GLASS DOF, CPPM's per-pixel statistical assumption breaks down, producing blurrier caustics compared to our method. However, in uniformly lit regions, our method performs slightly worse than CPPM.

Figure 6: *Comparison for NPPM with SPPM [HJ09] and CPPM [LLZ*20] at equal-time (30 seconds) with different radius reduction strategies. Our method produces sharper caustic details than SPPM. In some cases, CPPM tends to blur more complex caustic patterns (e.g.,* ARTWARE, GLASS DOF*); however, its radius reduction strategy better preserves uniform, well-lit regions, which can lead to lower overall error.*

Table 1: *Performance breakdown between SPPM and our method (NPPM$^{\alpha}$) for two representative scenes at different iteration counts (5 / 50 / 500 it.). All values are in milliseconds. Tracing time includes both eye and photon passes. Overhead is computed relative to the total SPPM time (Tracing + SPPM Eval.) compared to our method (Tracing + Enc. + Dec. + NPPM Eval.).*

Scene	Tracing	SPPM Eval.	Enc. + Dec.	NPPM Eval.	Overhead (%)
ARTWARE	60.0	70.3 / 42.1 / 22.5	34.3 / 27.1 / 1.5	107.1 / 69.5 / 33.6	54.5% / 53.3% / 15.2%
CRAB	53.2	7.9 / 4.9 / 3.7	12.8 / 11.5 / 1.5	21.9 / 17.6 / 5.9	44.3% / 46.0% / 6.5%

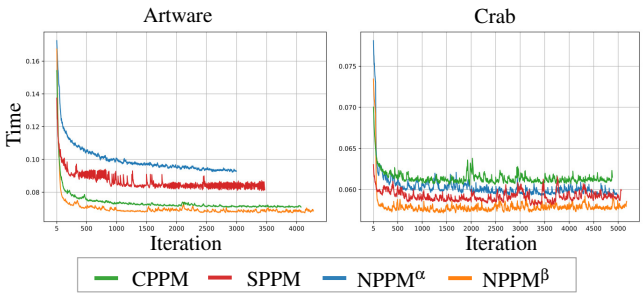

Figure 7: *Per-iteration runtime for all methods on* ARTWARE *and* CRAB. *Runtimes are smoothed with an exponential moving average, and timings before the 5th iteration are excluded for improved readability.*

Support for glossy materials. We support a microfacet conductor BSDF (GGX) in our GPU implementation. Figure 5 demonstrates our method on glossy surfaces in the CLASSROOM scene. While our method exhibits some low-frequency noise on glossy surfaces, it outperforms CPPM, which tends to overblur sharp caustics. Grazing view angles produce noisy results across all our tests. Including roughness information in the DCV or combining our method with vertex connection and merging (VCM) [GKDS12] may be promising directions for further improving results on such surfaces.

Timings. Table 1 quantifies the inference time of our NPPM approach compared to SPPM on the ARTWARE and CRAB scenes. ARTWARE exhibits highly heterogeneous photon density, with certain regions containing very high photon counts. In contrast, CRAB features a more homogeneous and sparse photon distribution across the scene. As a result, CRAB yields shorter range query and evaluation times than ARTWARE, as it produces a more uniform and smaller workload on the GPU. During the first phase, in which we encode 10% of the gather points and update the grid, our method represents a 45–55% overhead. This phase lasts for only 50 iterations, after which we simply read the DCV from the grid and decode it. This results in a final overhead ranging from 6–15%. Figure 7 provides a broader view of per-iteration cost during progressive rendering. For both NPPM$^{\beta}$ and CPPM, the range query radius decreases more rapidly, leading to faster evaluation and further reducing overhead. However, the highly heterogeneous query sizes introduced by CPPM's radius reduction scheme can cause GPU performance slowdowns. In contrast, our method makes more efficient use of GPU memory, enabling higher iteration counts under the same time budget. Further optimizations are possible; for instance, using a hash grid similar to Binder et al. [BFK20] or other spatially accelerated data structures would likely reduce our overhead further.

Unbiased estimators Explicit control over the kernel bandwidth is required for unbiased density estimation via telescopic series [MBGJ22], which is naturally supported by our adaptive kernel approach. For this application, we adapt our parametric kernel with

$$Q_{\tau}^{\text{Cone}}(\boldsymbol{d}) = \mathbf{1}_{\mathcal{B}}(\boldsymbol{d}) \cdot (1-\boldsymbol{d})^{\tau} \big/ Z_{\text{cone}}\,, \tag{13}$$

so that it mimics a parametric cone kernel, yielding smaller asymptotic variance compared to constant kernels [MBGJ22]. We obtain the normalization term $Z_{\text{cone}} = \frac{2\pi}{(\tau+1)(\tau+2)}$ by analytically integrating the kernel function. Our neural architecture is optimized to produce a suitable $\tau > 0.1$ for adaptive density estimation, in the same way as Q_{Σ}^{2DG}. This experiment also illustrates how our framework can be extended to other parametric kernels, highlighting its potential for generalization.

Figure 8 compares progressive estimators obtained with our method and with standard density estimation, along with their unbiased variants. We use the same random numbers to correlate the stochastic radius selections and the photon map generations, since unbiased variants often exhibit instability in convergence curves due to rare unlikely very small radii sampled. The comparison demonstrates that our method benefits both progressive and unbiased estimators by reducing overall error. While unbiased estimators show larger error than their biased counterparts, the resulting images remain compatible with existing denoising methods. It is important to note that, in theory, correlations between kernel estimations and local statistics during the grid construction phase may introduce bias. However, we did not observe clear bias both visually and quantitatively.

Convergence analysis Figure 9 shows the evolution of L2 and MRSE for our methods, SPPM, and CPPM over rendering time. These results reinforce the trends observed in Figure 6: the introduction of depth of field reduces the performance gap between our approach and CPPM. Notably, our method consistently achieves lower error rates at shorter durations, which is advantageous for preview settings. At longer durations, our method delivers rendering quality that is on par with, or slightly below, that of CPPM on average.

5.1. Ablation study

Range-query versus K-NN Figure 10 shows a comparison between photon mapping, DPM [ZXJ*20] and our method with

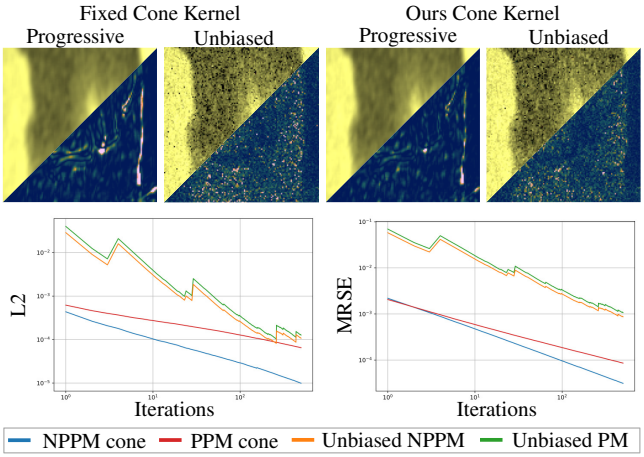

Figure 8: *Comparison between our method using an adaptive cone kernel, SPPM [HJ09], and Unbiased PM [MBGJ22], both using fixed cone kernels. Our method reduces bias in sharp caustic regions and decreases fireflies in the unbiased case. Error maps show MRSE for progressive methods and SMAPE for unbiased counterparts.*

range-query and K-NN. We can see that adopting a progressive approach is important in order to reduce bias. In this scenario, DPM diverges since nothing in its design enforces the normalization of the kernel; consequently its consistency suffers. Our approach strongly reduces bias on sharp caustics while preserving smooth lighting in uniform density regions. Thus making it effective in range queries and progressive settings thanks to our normalized kernels.

Number of iterations for DCV averaging. While our method does not require online training during inference, DCV formation remains a major bottleneck since its cost grows linearly with the number of gathered photons. To mitigate this, our approach relies on spatial latent averaging (Section 4.4) and gatherpoint subsampling. Table 2 illustrates the diminishing impact of the number of averaging iterations on both error reduction and rendering time. Based on these results, we select 50 iterations as a practical compromise between computational overhead and overall accuracy.

Grid resolution. The resolution of the grid used to store the DCV spatially is an important hyperparameter. Low resolutions tend to over-blur the latent codes, resulting in poorly fitted kernels and visible bias in the final image. Conversely, higher resolutions increase memory consumption and slightly raise the cost of retrieval and updates. Table 3 summarizes the impact of grid resolution on image quality, showing diminishing returns beyond a certain resolution. In all our experiments, we use a 256^3 grid, which fits comfortably within the GPU memory. More optimized data structures, such as hash grids or tree-based layouts, could further improve performance while maintaining a comparable memory footprint.

The resolution of the grid used to store the DCV spatially is an important hyperparameter. Low resolutions tend to over-blur the latent codes, resulting in poorly fitted kernels and visible bias in the final image. Conversely, higher resolutions increase memory consumption and slightly raise the cost of retrieval and updates. Table 3 summarizes the impact of grid resolution on image quality, showing diminishing returns beyond a certain resolution. In all our experiments, we use a 256^3 grid, which fits comfortably within the GPU memory.

Figure 5 and Figure 11 demonstrate that the uniform grid can also handle larger scenes. For even larger environments, we recommend using adaptive data structures, such as sparse grids, octrees, or kD-trees constructed from gather-point statistics during a preprocessing stage. These structures would significantly improve the memory efficiency of DCV storage, although computation speed may be reduced due to the logarithmic traversal complexity. Exploring such data structures and implementing the corresponding DCV interpolation are left for future work.

Table 2: *Effect of number of iterations for the DCV averaging for* 100 *passes in the* ARTWARE *with NPPM*$^{\beta}$.

Iterations	MRSE ($\times 10^{-4}$)	MSE ($\times 10^{-5}$)	Time (s)
1 iter.	8.83	2.86	12.41
10 iter.	2.84	1.40	12.59
100 iter.	2.60	1.11	13.57

Table 3: *Impact of grid resolution on rendering error for the* ARTWARE *scene using NPPM*$^{\alpha}$ *with* 100 *iterations. The* 400^3 *configuration corresponds to the maximum resolution that fits within our hardware.*

Grid Resolution	MRSE ($\times 10^{-3}$)	MSE ($\times 10^{-4}$)
64^3	3.00	1.74
128^3	2.35	1.36
256^3	2.31	1.33
400^3	1.99	1.21

6. Conclusions

We have presented neural progressive photon mapping (NPPM), a learning-based framework that addresses kernel parameter selection challenges in density estimation through spatially adaptive, analytical kernels. Our lightweight neural model efficiently adapts to local photon distributions while maintaining compatibility with progressive rendering workflows and consistent estimators. NPPM represents the first learning-based progressive photon mapping method, providing a practical solution for high-quality caustic rendering that combines neural adaptivity with the controllability required for robust light transport simulation.

Limitations and future work. Although independent truncated Gaussians produce consistent estimators, our kernels actually depend on the photon distribution and are therefore correlated. As such, we cannot claim theoretical consistency for our estimator. In practice, this does not cause issue with NPPM$^{\boldsymbol{\alpha}}$, as shown in the convergence plots in Fig. 9.

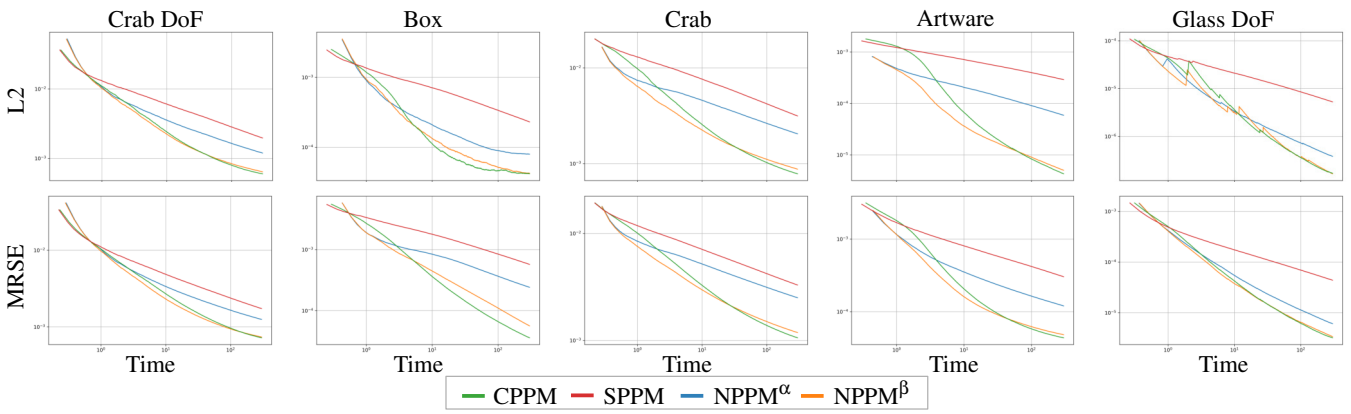

Figure 9: *Convergence curves per time step (seconds) of our methods NPPM, SPPM [HJ09] and CPPM [LLZ*20] over five minutes. Our method with the APA radius reduction scheme NPPM$^{\alpha}$ exhibits consistent convergence and higher quality renders than SPPM. Although NPPM$^{\beta}$ demonstrates superior performance over CPPM in short-duration scenarios, the hypothesis testing scheme of CPPM leads to lower or equivalent long-term error rates.*

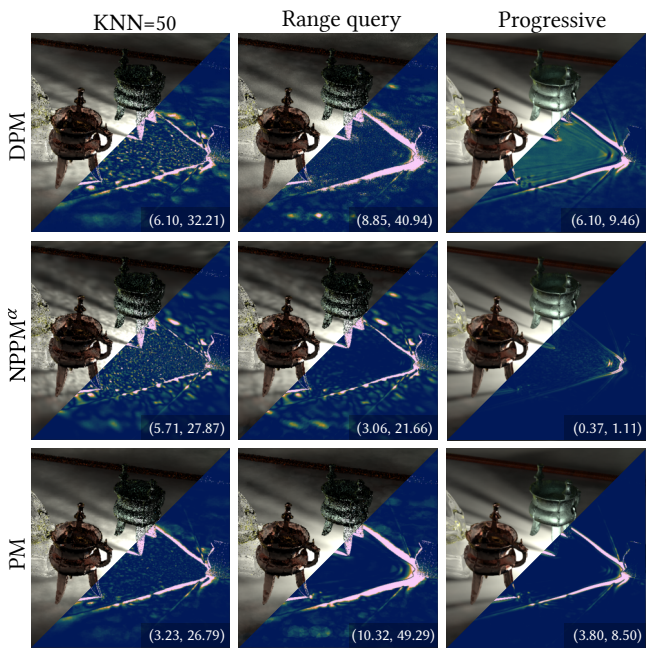

Figure 10: *Comparison between our method, DPM [ZXJ*20] and SPPM [HJ09]. Neural methods outperform box kernel for single queries. However, due to its non-progressive design, DPM diverges when radius reduction is applied. We show the SMAPE in false color; numbers refer to L^2 and MRSE scores, respectively.*

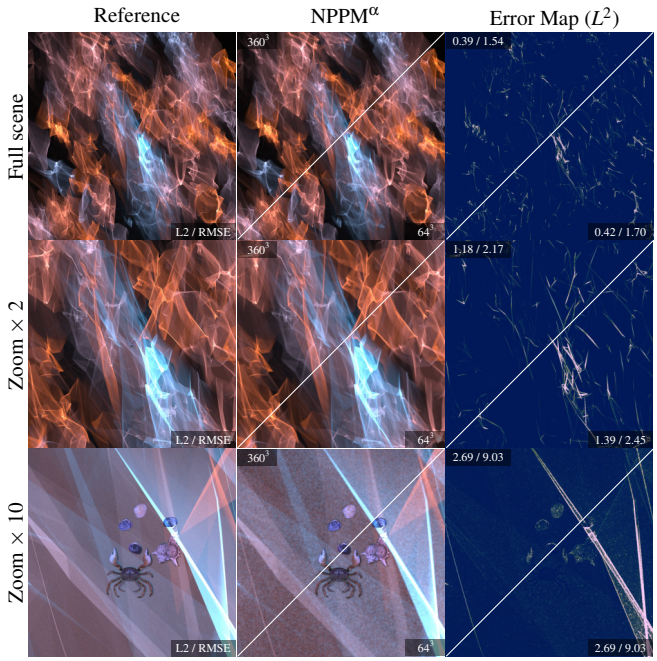

Figure 11: *Equal-iteration comparison of different grid resolutions for a larger version of the* CRAB *scene across multiple zoom levels. The renderings were obtained with 2M photons per iteration and identical initial radii.*

For future work, we posit that incorporating material information into our photon features before encoding could potentially produce better-adjusted kernels. Extending our method to support more expressive parametric kernels, such as multi-modal or piecewise distributions, could also be particularly beneficial for volumetric photon rendering and guiding, where complex scattering patterns require sophisticated kernel representations to be well captured.

Acknowledgment We would like to thank *aXel* (GLASS), *NovaZeeke* (CLASSROOM), the CPPM [LLZ*20] authors (ARTWARE and BOX) and *Benedikt Bitterli* [Bit16] for providing the scenes. We also thank the *Smithsonian Digitization Program Office* for the marine life 3D models (CRAB). This work was supported by the NSERC Discovery Grant RGPIN-2022-03182 and was enabled in part by support provided by Calcul Québec (`calculquebec.ca`) and the Digital Research Alliance of Canada (`alliancecan.ca`).

References

[BFK20] BINDER N., FRICKE S., KELLER A.: Massively parallel path space filtering. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing* (2020), Springer, pp. 149–168. 6, 9

[Bit16] BITTERLI B.: Rendering resources, 2016. https://benedikt-bitterli.me/resources/. 11

[CKS*17] CHAITANYA C. R. A., KAPLANYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. 98:1–98:12. doi:10/gbxhcv. 2

[CM21] CHOI H., MOON B.: Consistent post-reconstruction for progressive photon mapping. *Computer Graphics Forum 40*, 7 (2021), 121–130. doi:10.1111/cgf.14406. 2

[CWY11] CHEN J., WANG B., YONG J.-H.: Improved stochastic progressive photon mapping with metropolis sampling. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1205–1213. doi:10.1111/j.1467-8659.2011.01979.x. 2

[DHC*21] DENG X., HAŠAN M., CARR N., XU Z., MARSCHNER S.: Path graphs: iterative path space filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 40*, 6 (2021). doi:10.1145/3478513.3480547. 2

[DWL23] DONG H., WANG G., LI S.: Neural parametric mixtures for path guiding. In *ACM SIGGRAPH Conference Papers* (2023), pp. 1–10. doi:10.1145/3588432.3591533. 2, 6

[EK20] ESTEVEZ A. C., KULLA C.: Practical caustics rendering with adaptive photon guiding. In *ACM SIGGRAPH Talks* (New York, NY, USA, 2020), Association for Computing Machinery. doi:10.1145/3388767.3407370. 2

[FGW*24] FAN Z., GUO J., WANG Y., XIAO T., ZHANG H., ZHOU C., CHEN Z., HONG P., GUO Y., YAN L.-Q.: Specular polynomials. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 43*, 4 (2024). doi:10.1145/3658132. 1

[FWW*25] FAN Z., WANG C., WANG Y., LI B., GUO Y., YAN L.-Q., GUO Y., GUO J.: Bernstein bounds for caustics. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2025). doi:10.1145/3731145. 1

[GKDS12] GEORGIEV I., KŘIVÁNEK J., DAVIDOVIČ T., SLUSALLEK P.: Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 31*, 6 (2012). doi:10.1145/2366145.2366211. 2, 9

[GPGSK18] GRITTMANN P., PÉRARD-GAYOT A., SLUSALLEK P., KŘIVÁNEK J.: Efficient caustic rendering with lightweight photon mapping. In *Computer graphics forum* (2018), vol. 37, Wiley Online Library. doi:10.1111/cgf.13481. 2

[GRŠ*16] GRUSON A., RIBARDIÈRE M., ŠIK M., VORBA J., COZOT R., BOUATOUCH K., KŘIVÁNEK J.: A spatial target function for metropolis photon tracing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 36*, 4 (2016). doi:10.1145/2963097. 2

[HIT*24] HUANG J., IIZUKA A., TANAKA H., KOMURA T., KITAMURA Y.: Online neural path guiding with normalized anisotropic spherical gaussians. *ACM Transactions on Graphics 43*, 3 (2024). doi:10.1145/3649310. 2, 6

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*. ACM New York, NY, USA, 2009. doi:10.1145/1618452.1618487. 1, 2, 3, 7, 8, 10, 11

[HJ11] HACHISUKA T., JENSEN H. W.: Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics 30*, 5 (2011). doi:10.1145/2019627.2019633. 2

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 27*, 5 (2008). doi:10.1145/1409060.1409083. 2, 3, 6

[HPJ12] HACHISUKA T., PANTALEONI J., JENSEN H. W.: A path space extension for robust light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 31*, 6 (2012). doi:10.1145/2366145.2366210. 2

[HY21] HUO Y., YOON S.-E.: A survey on deep learning-based monte carlo denoising. *Computational Visual Media 7* (2021). doi:10.1007/s41095-021-0209-9. 2

[IMF*21] IŞIK M., MULLIA K., FISHER M., EISENMANN J., GHARBI M.: Interactive monte carlo denoising using affinity of neural features. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 40*, 4 (2021). doi:10.1145/3450626.3459793. 2

[JC98] JENSEN H. W., CHRISTENSEN P. H.: Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the annual conference on Computer graphics and interactive techniques* (1998). doi:10.1145/280814.280925. 2

[Jen95] JENSEN H. W.: Importance driven path tracing using the photon map. In *Rendering Techniques: Proceedings of the Eurographics Workshop* (1995), Springer, pp. 326–335. doi:10.1007/978-3-7091-9430-0_31. 2

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Rendering Techniques* (Vienna, 1996), Springer, pp. 21–30. doi:10.1007/978-3-7091-7484-5_3. 2, 3

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), Association for Computing Machinery, p. 511–518. doi:10.1145/383259.383319. 2

[JSR*22] JAKOB W., SPEIERER S., ROUSSEL N., NIMIER-DAVID M., VICINI D., ZELTNER T., NICOLET B., CRESPO M., LEROY V., ZHANG Z.: Mitsuba 3 renderer, 2022. URL: https://mitsuba-renderer.org. 7

[JZJ08] JAROSZ W., ZWICKER M., JENSEN H. W.: The beam radiance estimate for volumetric photon mapping. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* (2008). doi:10.1145/1401132.1401137. 2

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015), Bengio Y., LeCun Y., (Eds.). 7

[KBG23] KERN R., BRÜLL F., GROSCH T.: Accelerating photon mapping for hardware-based ray tracing. *Journal of Computer Graphics Techniques (JCGT) 12*, 1 (2023). URL: http://jcgt.org/published/0012/01/01/. 3

[KD13] KAPLANYAN A. S., DACHSBACHER C.: Adaptive progressive photon mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 32*, 2 (2013). doi:10.1145/2451236.2451242. 2, 3

[KDB16] KELLER A., DAHM K., BINDER N.: Path space filtering. In *Monte Carlo and Quasi-Monte Carlo Methods*. Springer, 2016, pp. 423–436. doi:10.1007/978-3-319-33507-0_21. 2

[KGH*14] KŘIVÁNEK J., GEORGIEV I., HACHISUKA T., VÉVODA P., ŠIK M., NOWROUZEZAHRAI D., JAROSZ W.: Unifying points, beams, and paths in volumetric light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 33*, 4 (2014). doi:10.1145/2601097.2601219. 2

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 42*, 4 (July 2023). doi:10.1145/3592433. 4

[KPB21] KOBYZEV I., PRINCE S. J., BRUBAKER M. A.: Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence 43*, 11 (2021), 3964–3979. doi:10.1109/TPAMI.2020.2992934. 2

[KW14] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)* (2014). 6

[KWX*16] KANG C.-M., WANG L., XU Y.-N., MENG X.-X., SONG Y.-J.: Adaptive photon mapping based on gradient. *Journal of Computer Science and Technology 31* (2016), 217–224. doi:10.1007/s11390-016-1622-x. 2

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph. 30*, 3 (2011). doi:10.1145/1966394.1966404. 2, 3

[LHL*24] LITALIEN J., HAŠAN M., LUAN F., MULLIA K., GEORGIEV I.: Neural product importance sampling via warp composition. In *ACM SIGGRAPH Asia Conference Papers* (2024). doi:10.1145/3680528.3687566. 2

[LLZ*20] LIN Z., LI S., ZENG X., ZHANG C., JIA J., WANG G., MANOCHA D.: Cppm: chi-squared progressive photon mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 39*, 6 (Nov. 2020). doi:10.1145/3414685.3417822. 1, 2, 3, 7, 8, 11

[M21] MÜLLER T.: tiny-cuda-nn, 2021. URL: https://github.com/NVlabs/tiny-cuda-nn. 7

[MBGJ22] MISSO Z., BITTERLI B., GEORGIEV I., JAROSZ W.: Unbiased and consistent rendering using biased estimators. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 41*, 4 (2022). doi:10.1145/3528223.3530160. 2, 4, 9, 10

[MMR*19] MÜLLER T., MCWILLIAMS B., ROUSSELLE F., GROSS M., NOVÁK J.: Neural importance sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 38*, 5 (2019). doi:10.1145/3341156. 2

[PAMM19] PERROT R., AVENEAU L., MORA F., MENEVEAUX D.: Photon mapping with visible kernel domains. *The Visual Computer 35* (2019), 707–720. doi:10.1007/s00371-018-1505-y. 2

[PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., DESMAISON A., KOPF A., YANG E., DEVITO Z., RAISON M., TEJANI A., CHILAMKURTHY S., STEINER B., FANG L., BAI J., CHINTALA S.: *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019, pp. 8024–8035. doi:10.5555/3454287.3455008. 7

[QSH*15] QIN H., SUN X., HOU Q., GUO B., ZHOU K.: Unbiased photon gathering for light transport simulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 34*, 6 (2015). doi:10.1145/2816795.281811. 2, 4

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017. URL: https://arxiv.org/abs/1612.00593, arXiv:1612.00593. 3

[Ram14] RAMA H.: Fast fixed-radius nearest neighbors: Interactive million-particle fluids. *GPU Technology Conference* (2014). 7

[SFES07] SCHJØTH L., FRISVAD J. R., ERLEBEN K., SPORRING J.: Photon differentials. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia* (2007), pp. 179–186. doi:10.1145/1321261.1321293. 2

[ŠK19] ŠIK M., KRIVÁNEK J.: Implementing one-click caustics in corona renderer. In *Proceedings of EGSR (Industry Track)* (2019), pp. 61–67. doi:10.2312/sr.20191221. 2

[Vea98] VEACH E.: *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998. 2

[vOHK16] ŠIK M., OTSU H., HACHISUKA T., KŘIVÁNEK J.: Robust light transport simulation via metropolised bidirectional estimators. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 35*, 6 (2016). doi:10.1145/2980179.2982411. 2

[WGGH20] WEST R., GEORGIEV I., GRUSON A., HACHISUKA T.: Continuous multiple importance sampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (2020). doi:10.1145/3386569.3392436. 2

[WGH22] WEST R., GEORGIEV I., HACHISUKA T.: Marginal multiple importance sampling. In *ACM SIGGRAPH Asia Conference Papers* (2022). doi:10.1145/3550469.3555388. 2

[ZGJ20] ZELTNER T., GEORGIEV I., JAKOB W.: Specular manifold sampling for rendering high-frequency caustics and glints. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 39*, 4 (2020), 149–1. doi:10.1145/3386569.3392408. 1

[ZXJ*20] ZHU S., XU Z., JENSEN H. W., SU H., RAMAMOORTHI R.: Deep kernel density estimation for photon mapping. *Computer Graphics Forum 39*, 4 (2020), 35–45. doi:10.1111/cgf.14052. 2, 3, 4, 5, 7, 9, 11

[ZXS*21a] ZHU S., XU Z., SUN T., KUZNETSOV A., MEYER M., JENSEN H. W., SU H., RAMAMOORTHI R.: Hierarchical neural reconstruction for path guiding using hybrid path and photon samples. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 40*, 4 (2021). doi:10.1145/3450626.3459810. 2

[ZXS*21b] ZHU S., XU Z., SUN T., KUZNETSOV A., MEYER M., JENSEN H. W., SU H., RAMAMOORTHI R.: Photon-driven neural reconstruction for path guiding. *ACM Transactions on Graphics 41*, 1 (2021). doi:10.1145/3476828. 2