Adaptive Neural Kernels for Gradient-domain Rendering

MATTHIEU JOSSE, École Polytechnique, France
JOEY LITALIEN, Independent, Canada
ADRIEN GRUSON, École de technologie supérieure, Canada

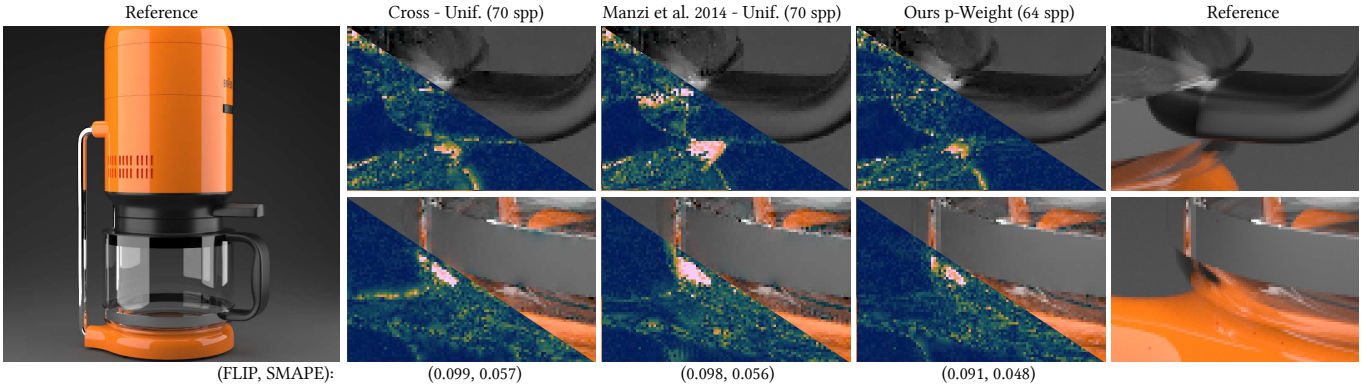

Fig. 1. Equal-time comparison between our adaptive kernel, the structure-adaptive kernel [Manzi et al. 2014], and the standard cross-shaped kernel. All methods use gradients produced by gradient-domain path tracing with path reconnection.

Monte Carlo methods are a cornerstone of physics-based light transport simulations, valued for their ability to produce high-quality photorealistic images. These stochastic methods often suffer from variance, resulting in undesirable noise in the rendered images. Gradient-domain rendering (GDR) techniques mitigate this problem by estimating unbiased image-space gradients via so-called shift-mapping operators. While these mappings are computationally efficient, they can yield high-variance gradients—and thus poor reconstruction quality—when applied to pixels with wildly different integrals. We tackle this challenge by dynamically selecting the optimal set of neighboring pixels for applying shift-mapping under random sequence replay. Key to our approach is a differentiable sorting network that softly ranks the output of a convolutional neural network conditioned on input sample features for weighted reconstruction. This module is carefully rigidified over time to converge to a hard top-k selection, allowing end-to-end optimization with respect to the reconstruction error. Our method is versatile and can be jointly optimized with other adaptive sampling strategies. We demonstrate variance reduction over other traditional adaptive gradient-domain methods across scenes of varying radiometric complexity.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Monte Carlo light transport, photorealistic rendering, gradient-domain rendering, neural rendering

Authors' addresses: Matthieu Josse, matthieu.josse@polytechnique.edu, École Polytechnique, Paris, France; Joey Litalien, joey.litalien@gmail.com, Independent, Montréal, Canada; Adrien Gruson, adrien.gruson@etsmtl.ca, École de technologie supérieure, Montréal, Canada.

SA Conference Papers '25, December 15–18, 2025, Hong Kong, Hong Kong

ACM ISBN 979-8-4007-2137-3/2025/12
https://doi.org/10.1145/3757377.3763920

ACM Reference Format:
Matthieu Josse, Joey Litalien, and Adrien Gruson. 2025. Adaptive Neural Kernels for Gradient-domain Rendering. In *SIGGRAPH Asia 2025 Conference Papers (SA Conference Papers '25), December 15–18, 2025, Hong Kong, Hong Kong*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3757377.3763920

1 INTRODUCTION

Photorealistic rendering aims to simulate the complex interactions between light and matter in virtual environments. Achieving noise-free results with widely-adopted Monte Carlo methods often requires an impractical number of samples, especially in radiometrically challenging scenes with complex visibility and indirect lighting. Gradient-domain rendering (GDR) [Hua et al. 2019; Lehtinen et al. 2013] offers a compelling alternative for reducing variance by exploiting correlation in image-space. In essence, GDR use local similarities between neighboring pixels and computes image gradients to recover the final image through a Poisson solver.

Core to gradient-domain light transport is the *shift-mapping operator*, a mechanism that builds offset, correlated paths from primal light paths. This is generally achieved via a cross-shaped, finite-difference shifting kernel that uses immediate pixel neighbors to compute gradients. While GDR methods can achieve higher quality results compared to more traditional rendering methods under such kernels, they tend to produce undesirable structural and singularity artifacts in the output. Reconstruction quality may then suffer when neighboring pixels do not correlate well with the base pixel. As a result, adaptive shift-mapping operators based on cross-bilateral filtering have been developed [Manzi et al. 2014, 2016b] to modify the kernel shape based on additional salient per-pixel features (e.g., albedo, normal). Their use in practice, however, remains fairly limited due to their heuristic nature that can fail to properly inform

the reconstruction. Adaptive sampling techniques [Back et al. 2018; Liang et al. 2024] can help mitigate the drawbacks of poorly conditioned kernels but their integration with gradient-domain renderers remains relatively unexplored. As such, the need for robust adaptive shift-mapping kernels for robust GDR still persists.

By design, shift-mapping operators are sensitive to the choice of neighbors, and well-correlated neighbors are essential to their effectiveness. Inspired by the recent progress of neural denoisers [Huo and Yoon 2021; Işık et al. 2021], we introduce a learning-based approach that adaptively and directly optimizes this neighbor selection. Noting that k-nearest neighbor is not differentiable and is not directly amenable to learning pipelines, we propose to employ differentiable sorting networks [Petersen et al. 2021] to learn a soft-top-k approximation that we harden over time to converge to the true top-k operation. This natural design allows our reconstruction pipeline to be efficiently trained end-to-end.

We demonstrate improved neighbor selection for random sequence replay shift-mappings on direct illumination on a custom indoor scene dataset as well with path reconnection (Fig. 1). We show the flexibility of how our method by combining it with neural adaptive sampling for additional gains. Our model—which we implement atop Mitsuba 3 [Jakob et al. 2022b]—is remarkably simple and produces adaptive neural kernels that consistently outperform other gradient-domain rendering techniques at equal sample count. Both our model and dataset will be made publicly available upon publication.

To summarize, our main contributions are:

- a per-sample flexible neural module that automatically determines how to select neighboring pixels,
- an integration with adaptive neural sampling, and
- an indoor scene dataset for training reconstruction methods.

Code and datataset for this paper are available at: https://github.com/MattJosse/adaptive-neural-kernel.

2 RELATED WORK

Gradient-domain rendering. The seminal work of Lehtinen et al. [2013] on gradient-domain Metropolis light transport demonstrates the benefits of estimating image-space gradients [Bhat et al. [n. d.]] for variance reduction. In its original formulation, a screened Poisson solver is used to reconstruct the final image, which can also be expressed as an image-space control variate [Rousselle et al. 2016]. Multiple shift-mapping operators have been developed [Hua et al. 2019] and adapted to path-tracing [Kettunen et al. 2015], bidirectional path-tracing [Manzi et al. 2015], density estimation [Hua et al. 2017], vertex-connection and merging [Sun et al. 2017] and volume rendering [Gruson et al. 2018]. Specialized shift-mappings have also been introduced to extend gradient-domain rendering to the temporal [Manzi et al. 2016a] or spectral [Petitjean et al. 2018] domains. Gradient outlier clipping [Ha et al. 2019] and generalized albedo demodulation [Fang and Hachisuka 2024] can further reduce visual artifacts. Recently, these operators have been extended to real-time rendering with ReSTIR [Lin et al. 2022; Wyman et al. 2023] and multi-view rendering [Fraboni et al. 2022]. In this work, we focus on gradient-domain path-tracing [Kettunen et al. 2015] as a case study for our adaptive neighboring selection algorithm and leave the exploration of the other integrators for future work.

In general, shift-mappings compute gradients of one-off neighboring pixels in a cross-like pattern. Bauszat et al. [2017] show that larger kernels can further reduce noise. Our work draws inspiration from Manzi et al. [2014] which employs guiding features and a simple heuristic to choose neighbors and estimate screen-space gradients. We propose a learning-based approach that circumvents the need for handcrafted strategies, hence allowing for more flexibility.

Neural methods for Monte Carlo reconstruction. Neural image denoising has been extensively applied to Monte Carlo rendering, where neural networks are optimized to predict noise-free images from input-target pairs [Bako et al. 2017; Chaitanya et al. 2017; Gharbi et al. 2019; Huo and Yoon 2021; Işık et al. 2021; Meng et al. 2020; Munkberg and Hasselgren 2020; Wong and Wong 2019]. To improve guidance, auxiliary per-pixel or per-sample features can be supplied to the network. We adopt a hybrid approach similar to Işık et al. [2021] where we process individual samples but summarize their latent contexts into per-pixel statistics. We note that guiding features can be repurposed to regularize the Poisson reconstruction process [Manzi et al. 2016b] in gradient-domain rendering. Additionally, gradient estimates can directly be used as input [Kettunen et al. 2019a] or as a first-order loss term [Guo et al. 2019; Xu et al. 2020]. Yan et al. [2024] instead propose to use filtered gradient estimates for image smoothing. These works are orthogonal to our method and do not support arbitrary image-space gradient estimates by default. We note, however, that our reconstructed images can be combined with other denoisers [Back et al. 2020, 2018].

Adaptive sampling. Nonuniform sampling over the image space is an efficient method to redistribute the error in image-space [Zwicker et al. 2015]. Adaptive techniques often rely on error estimates and multiple sampling rounds to refine them. Manzi et al. [2016a] apply this idea to GDR and introduce a variance-based adaptive sampling approach based on prior frames. Back et al. [2018] perform adaptive sampling using the polynomial rendering framework of Moon et al. [2016], where a reconstructed image from the gradient domain is used for guiding. These works cannot be easily combined or extended when adaptive kernels are employed. In contrast, our method directly supports adaptive sampling by design.

Most relevant to our adaptive scheme is DASR [Kuznetsov et al. 2018] which computes the gradient of a renderer with respect to the number of samples. Such gradients allow the joint training of all components of the model. Due to its ease of implementation, DASR was adapted to real-time, temporal scenarios [Hasselgren et al. 2020] and volume rendering [Hofmann et al. 2021]. Salehi et al. [2022] compute similar gradients by expressing the pixel errors via an analytical distribution, thereby circumventing the need to construct a large training dataset. To the best of our knowledge, the most recent adaptive method tailored to gradient-domain rendering is that of Liang et al. [2024] which combines DASR and a neural reconstruction module [Kettunen et al. 2019a]. Here, different sampling densities are allocated on primal, vertical and horizontal gradients. This technique differs from ours in the shape of the proposed kernels since we train adaptive neural kernels hand-in-hand with our adaptive sampling prediction network.

3 BACKGROUND

3.1 Gradient-domain rendering

We start by briefly reviewing Monte Carlo rendering and its gradient-domain extension.

Path integral formulation. We are interested in solving the light transport equation [Veach 1998]: given a light path $\bar{\mathbf{x}} \in \mathcal{P}$, we seek a radiance measurement I_i at pixel i by computing the integral

$$I_i = \int_{\mathcal{P}} r_i(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \, \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{1}$$

where r_i is the reconstruction filter, f is the path contribution function, and μ is the area-product (Lebesgue) measure over the space of path $\bar{\mathbf{x}} \in \mathcal{P}$. Equation (1) can be approximated using Monte Carlo (MC) integration.

Shift-mapping operators. The goal of shift-mapping [Lehtinen et al. 2013] is to construct offset paths from a base path and leverage correlation between them. At their core, these operators transform an input path $\bar{\mathbf{x}}$ passing through pixel i into a *shifted* path $\bar{\mathbf{y}}$ passing through a *different* neighboring pixel j. Denoting this mapping as $T : \bar{\mathbf{x}} \to \bar{\mathbf{y}}$ and covering the full domain of j, we can formulate the new measurement at pixel j as

$$I_j = \int_{\mathcal{P}} r_j\big(T(\bar{\mathbf{x}})\big) f\big(T(\bar{\mathbf{x}})\big) \left|\det \mathbf{J}_T(\bar{\mathbf{x}})\right| \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{2}$$

where $\mathbf{J}_T(\bar{\mathbf{x}}) \coloneqq \partial T(\bar{\mathbf{x}}) / \partial \bar{\mathbf{x}}$ is the Jacobian matrix which determinant accounts for the change in integration domain. Since the reconstruction filter (box) remains unchanged under shifting, we have that $r_j\big(T(\bar{\mathbf{x}})\big) = r_i(\bar{\mathbf{x}})$. Taking the difference $\Delta_{i\to j} \coloneqq I_j - I_i$ between the two pixels shifting from i to j, we obtain

$$\Delta_{ij} = \int_{\mathcal{P}} r_i(\bar{\mathbf{x}}) \Big(f\big(T(\bar{\mathbf{x}})\big) \left|\det \mathbf{J}_T(\bar{\mathbf{x}})\right| - f(\bar{\mathbf{x}}) \Big) \, \mathrm{d}\mu(\bar{\mathbf{x}}). \tag{3}$$

Since it is generally the case that $T(\bar{\mathbf{x}}) \approx \bar{\mathbf{x}}$ (making $|\det \mathbf{J}_T(\bar{\mathbf{x}})| \approx 1$), Eq. (3) becomes zero on most pixels. Consequently, a MC estimator $\langle \Delta_{i\to j} \rangle_N$ tends to have lower variance in image-space than its primal. Note that the domain coverage property for deriving Eq. (2) can be achieved with *random replay* shift-mapping [Hua et al. 2019].

Image reconstruction. After estimating gradients, we perform a screened Poisson reconstruction [Bhat et al. 2008] :

$$I^* = \arg\min_I \Big(\lambda \big\| I - \langle I \rangle \big\|_n^n + (1-\lambda) \big\| \delta I - \langle \Delta_{ij} \rangle \big\|_n^n \Big), \tag{4}$$

where δ is the finite-difference convolution operator and $\lambda > 0$ is a hyperparameter controlling the importance of the primal image. Solving Eq. (4) with $n = 2$ yields unbiased reconstructions, while $n = 1$ is typically more robust [Lehtinen et al. 2013]. Albeit many solvers exist, we use the simpler iterative reconstruction scheme from Rousselle et al. [2016] which allows arbitrary weighting strategies. Here, we solve iteratively as

$$\begin{aligned} I_i^{(0)} &= \langle I_i \rangle \\ I_i^{(t+1)} &= w_i^{\mathrm{p}} I^{(t)} + \sum_j w_{ij}^{\mathrm{g}} \Big(I_j^{(t)} - \langle \Delta_{ij} \rangle \Big), \qquad t \in \mathbb{N}, \end{aligned} \tag{5}$$

Fig. 2. Example of a simple sorting network on $n = 4$ integer inputs. *Left*: Each solid line represents a wire/lane and the dashed lines indicate pairwise comparators for potential swaps. *Right*: After a series of conditional swap operations, the final output vector is sorted as $(3, 2, 4, 1) \mapsto (1, 2, 3, 4)$.

where $w^{\mathrm{p}}, w^{\mathrm{g}} \geq 0$ are the primal and gradient respective weights satisfying $w^{\mathrm{p}} + \sum_j w_j^{\mathrm{g}} = 1$, and $t \leq 50$ iteration steps generally suffice.

Structure-adaptive gradient kernels. The performance of GDR algorithms highly depends on the choice of pixel neighbors. To mitigate this issue, Manzi et al. [2014] propose to quantify pixel similarity using auxiliary feature guides $\mathbf{g}$ and formulate a similarity score between pixels i and j via a Gaussian bilateral filter [Dammertz et al. 2010] defined as

$$s_{ij} = \exp\Big(-\max\big(0, \|\mathbf{g}_i - \mathbf{g}_j\|_2^2 - d\big) / b^2 \Big), \tag{6}$$

where $b, d > 0$ are user-defined bandwidth and threshold parameters conveying the importance of a feature. The final affinity score is then computed as the minimum over all features. While fast to compute, this technique has some limitations. First, the choice of global parameters can produce locally suboptimal results in different scenes. Second, these coefficients cannot be optimized as standard gradient-domain reconstruction is not differentiable with respect to the kernel selection weights. Finally, Eq. (6) is a heuristic and cannot easily scale to non-geometric path features such as material and scattering types. We address these shortcomings in Section 4.3.

3.2 Differentiable sorting networks

Definition. Sorting networks [Knuth 1998] are a class of computational models used to perform sorting through a fixed sequence of comparisons and swaps.[1] They can be visualized as a series of *wires* (or *lanes*) representing the data, and conditional pairwise *swapping operators*. Intuitively, each operator swaps two connected values if they are out of order. Conditional swap operators can be implemented with min and max operations only. Given an input $\mathbf{x} = (x, y) \in \mathbb{R}^2$, the *odd-even sorting network* [Habermann 1972] updates the elements as

$$\mathbf{x}' = (x', y') = \big(\min(x, y), \max(x, y)\big), \tag{7}$$

which yields the output satisfying $x' \leq y'$. This process can be trivially generalized to sort any length-n vector by repeating swaps across n successive wires. We provide a visualization of this in Fig. 2. The more efficient *bitonic sorting network* [Batcher 1968] repeatedly merges and sorts bitonic sequences via a divide-and-conquer approach [Kipfer and Westermann 2005].

[1] Note that despite their name, sorting networks are *not* neural networks that sort.

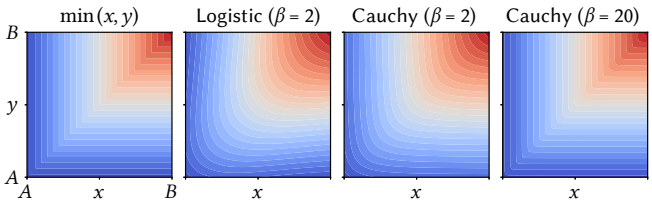

Fig. 3. We visualize different smooth relaxations of the hard minimum operator $\min(x, y)$ for pairs $(x, y) \in [A, B]^2$ with different stiffness $\beta = 1/\tau$. The logistic relaxation approximates the minimum but is nonmonotonic, which is undesirable. The Cauchy relaxation offers a monotonic alternative with bounded error, which can be "hardened" by progressively decreasing the temperature parameter τ to improve the approximation.

Differentiable swap operators. The above min/max operators are nondifferentiable but can be made differentiable via relaxation [Petersen et al. 2021], e.g. via the logistic relaxation:

$$\begin{aligned} \min_F(x, y) &= x \cdot F(y - x) + y \cdot F(x - y), \\ \max_F(x, y) &= x \cdot F(x - y) + y \cdot F(y - x), \end{aligned} \tag{8}$$

where $F(z; \tau) = 1/(1 + \exp(-\tau z))$ is the cumulative distribution function (CDF) of the tempered logistic distribution with (inverse) temperature $\tau > 0$. Petersen et al. [2022a] showed that any sigmoid-like relaxation should be monotonic and have bounded error for the resulting differentiable sorting network to match the result of the hard sorting function. To this end, the logistic function is replaced with the CDF of the Cauchy distribution as an improved relaxation:

$$F_{\text{Cauchy}}(z; \tau) = \frac{1}{\pi} \arctan(\tau z) + \frac{1}{2}. \tag{9}$$

We show a comparison in Fig. 3 to provide some intuition.

Discussion. Like any differentiable functions, *differentiable sorting networks* (DSNs) can be used in learning pipelines and trained with gradient-based optimization. Some applications include differentiable k-nearest-neighbors [Grover et al. 2019], differentiable patch selection for image recognition [Cordonnier et al. 2021] and top-k attention for machine translation [Xie et al. 2020]. In Section 4.3, we show how to apply this technique to gradient-domain rendering for top-k neighbor selection under shift-mapping.

4 NEURAL SHIFT-MAPPING KERNELS

4.1 Overview

We propose an adaptive shift-mapping kernel operators for gradient-domain rendering using neural networks. Core to our technique are (I) a sample embedder that reduces samples to pixel embeddings, (II) a convolutional neural network (CNN) that propagates information to neighboring pixels, (III) a differentiable sorting network that performs neighbor selection, and (IV) a weighted iterative image reconstruction scheme (Eq. (5)). More precisely, the sample reducer is a small multilayer perceptron (MLP) that summarizes per-sample features to per-pixel statistics. A CNN then processes these embeddings by sharing information spatially and outputs a vector of normalized neighbor scores for each pixel. We employ a differentiable top-k network [Petersen et al. 2022b] to rank and select the best neighbors based on these predicted scores. Selected candidates are traced to compute image-space gradients under random sequence replay and an iterative weighted reconstruction loop is ultimately performed to generate the final image. Our neural pipeline is summarized in Fig. 4 (left). Additionally, we illustrate how our adaptive sampling network uses data from our neural pipeline to construct a sampling map in Fig. 4 (right).

4.2 Per-pixel feature encoder

To guide the reconstruction process, we supply several per-sample features to our model, denoted as $\mathbf{g}_{is}$ for pixel i and sample s. First, we use normalized *depth* (1), *surface normal* expressed in camera space (3), *diffuse albedo* (3), and *surface roughness* (1), which we linearize with $r \mapsto 1 - \exp(-r)$. We also store the *scattering type* (6) encoding the surface interaction as a binary vector (i.e, diffuse, glossy, pure-specular, reflection, transmission, emissive). Since shift-mapping operators are sensitive to different base and offset path surface interactions, we explicitly provide these to our network. We also supply the *primal radiance* (3) per sample, tonemapped with $x \mapsto \log(1 + x)$. We found this noisy feature to be useful to reason about radiance variation, such as visibility in direct illumination. In total, our per-sample auxiliary features have 17 dimensions.

Inspired by prior work on hybrid sample-pixel denoising techniques [Işık et al. 2021], we independently encode N samples via a small, 3-layer MLP $E(\boldsymbol{\theta})$ with leaky ReLU activations, then compute the average and maximum statistics of the embeddings. We further concatenate positionally-encoded *image-space positions* (8) [Tancik et al. 2020] to obtain per-pixel summary features $\mathbf{e}_i$. We show in our ablation study (Section 5.3) that this encoding is important in preventing correlation in the output, which may occur across surfaces with very little variation in features (e.g., a flat white plane).

4.3 Spatial neighbor selector

Once the sample features are reduced and encoded, we employ a U-Net [Ronneberger et al. 2015], denoted as $U(\boldsymbol{\phi})$, to transform per-pixel contexts to neighbor scores. Recall that our goal is to use the output of this network to select $k \leq h^2 - 1 := H - 1$ neighboring candidates in a square window $\mathcal{N}$ of size h centered at pixel i. This process should be *differentiable* so that we can learn directly learn a selection strategy end-to-end with image reconstruction.

Naive neighbor selection. A simple approach is to directly regress neighboring scores and pick the k highest ones. To do so, we can predict weights $\mathbf{w}^{\text{g}} \in \mathbb{R}^{H-1}$ and trivially apply a τ-relaxed softmax:

$$U(\mathbf{e}; \boldsymbol{\phi}) = \sigma(\mathbf{w}^{\text{g}}), \quad \text{where} \quad \sigma(\mathbf{w})_l = \frac{\exp(\tau w_l)}{\sum_j \exp(\tau w_j)}, \tag{10}$$

where $\tau > 0$ is a temperature parameter controlling the shape of the output distribution over the window and we have dropped the pixel subscript i for clarity. Intuitively, $\tau \to +\infty$ produces a sharper distribution converging to a single entry (i.e., a one-hot vector), whereas $\tau \to 0$ converges to a uniform distribution.

While this annealing scheme can correctly identify *the* best neighboring candidate, it cannot generalize to multiple candidates. Computing top-k scores inherently requires ranking, which typically involves index swapping—a nondifferentiable operation—making it challenging to explicitly define the training objective. Additionally, propagating gradients to selected neighbors does not truly emulate

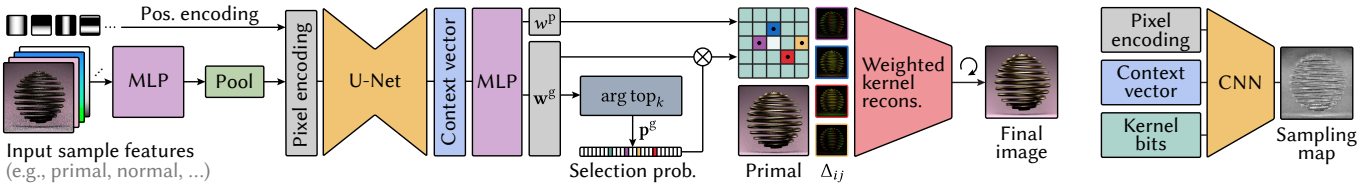

Fig. 4. **Pipeline overview.** *Left*: A small per-sample MLP first encodes and aggregates sample features into per-pixel summary statistics. These pixel encodings are then processed by a U-Net to produce a context vector that gets decoded into a weighted neighbor kernel $\mathbf{w}^g$ and a primal weight w^p. A differentiable sorting network uses these weights to rank and pick appropriate neighbors for the weighted reconstruction. *Right*: Our adaptive sampling network uses the encoded inputs, the deep context and the predicted kernel bits to build the sample map that is used to generate I and Δ_{ij}. The model is trained end-to-end.

a top-k operation since we cannot capture the gradients with respect to the selection process. One alternative is to use multiple rounds of a relaxed Gumbel selection process without replacement [Kool et al. 2019]. However, we empirically found that this prevents the optimization from converging entirely. As such, a more principled neighbor ranking module is required.

Differentiable top-k selection. Our key insight is to replace the softmax layer from Eq. (10) with a differentiable arg top-k layer [Petersen et al. 2022b], which outputs a deterministic selection probability for each weight. The inferred probabilities are then multiplied with the predicted gradient weights $\mathbf{w}^g$, as we discuss in Section 4.4. When the temperature τ is sufficiently low, the output of this operation becomes indistinguishable from the (hard) top-k. In practice, we observe that switching to the (hard) top-k becomes necessary when τ drops below 10^{-3}, as gradient computation tends to become unstable at low temperatures.

Score predictor network. Our differentiable sorting network acts on normalized neighbor scores, similar in spirit to the output of kernel prediction network. We learn these vectors using a simple 5-scale U-Net [Ronneberger et al. 2015] with concatenation for skip-connections. Our neural architecture reads

$$(\mathbf{c}64)_2 \blacktriangleright (\mathbf{c}64)_2 \blacktriangleright (\mathbf{c}80)_2 \blacktriangleright (\mathbf{c}96)_2$$
$$\blacktriangleleft (\mathbf{c}80)_2 \blacktriangleleft (\mathbf{c}64)_2 \blacktriangleleft (\mathbf{c}64)_2 \blacktriangleleft (\mathbf{c}H)_2,$$

where $(\mathbf{c}\ell)_r$ denotes r blocks of 3×3 convolution with ℓ channels, $\blacktriangleright$ denotes 2×2 max-pooling, and $\blacktriangleleft$ is 2×2 bilinear upsampling. We use leaky ReLU activations for all convolutions except the last one. The output of the U-Net goes through a small MLP to reduce the dimension. The weight $\mathbf{w}^g$ is normalized and w^p is activated via a softplus function. Our convolutional and sorting network respectively output

$$U(\mathbf{e}; \boldsymbol{\phi}) = (w^p, \mathbf{w}^g), \tag{11}$$
$$\arg\text{top}_k(\mathbf{w}^g; \tau) = \mathbf{p}^g, \tag{12}$$

where $\mathbf{p}^g \geq 0$ is the probability for each neighbor $\{j_1, \ldots, j_k \mid j \in \mathcal{N}\}$ to be selected at stiffness τ. Note that this selection probability is made binary at inference time since we use a hard top-k selection.

4.4 Image reconstruction

Reconstruction process. We use the iterative solver from Eq. (5) using the weights w^p and $\mathbf{w}^g$ predicted by our model, where we ensure normalized weights for correct reconstruction [Rousselle et al. 2016]. Adding weights to the reconstruction step has two benefits. First, it improves reconstruction quality by encoding the importance of neighbors, aiding both stability and diversity. The network can also choose to balance primal and gradient information, akin to the parameter λ in Eq. (4). Second, it establishes a direct link between predicted weights and the final image during optimization, which makes hard top-k selection differentiable. We show in Table 2 that soft top-k with temperature scheduling leads to improved stability during training.

Since both gradient directions from $i \rightarrow j$ (and vice versa) are sampled in Δ_{ij}, note that we update both i and j with Δ_{ij}'s information, even in cases where j is i's neighbor.

Control variates weighted reconstruction. Our technique can be extended to support weighted reconstruction based on variance information [Rousselle et al. 2016]. In this scheme, we still use our predicted weights w^p and $\mathbf{w}^g$ and multiply our weights to the variance-based weights during reconstruction. We then update the variance of the primal accordingly.

4.5 Kernel-aware adaptive sampling

Our method can be easily combined with DASR [Kuznetsov et al. 2018] for additional variance reduction. We use their finite-difference approach to calculate the gradient with respect to the number of sample s for both the primal and the gradient images:

$$\frac{\partial I_{i,s}}{\partial s} = \frac{I_{i,\infty} - I_{i,s}}{s}, \qquad \frac{\partial \Delta_{ij,s}}{\partial s} = \frac{\Delta_{ij,\infty} - \Delta_{ij,s}}{s}. \tag{13}$$

Here, $I_{i,s}$ and $\Delta_{ij,s}$ are computed on-the-fly by combining independent estimates using power-of-two sampling counts, and $I_{i,\infty}$ and $\Delta_{ij,\infty}$ denote groundtruth images.

Adaptive network. To predict our sampling map, we use a simple 3-layer convolutional network with leaky ReLUs, taking as input the encoded per-pixel features, the context vector computed by the U-Net, and the binary kernel. The output is projected to a positive scalar, activated with a square function, and normalized across the image. We then convert the resulting *sampling weights* $\mathbf{M} \in \mathbb{R}^{H \times W}$ to a *sampling map* via the mapping $\mathbf{M} \mapsto \mathbf{M}(HW(S-1)) + 1$, where S is our sample budget, ensuring every pixel receives at least one sample. Contrary to Liang et al. [2024], we use the same number of samples for the gradient and primal information. We leave for future research the exploration of using different samples for primal and individual gradients in adaptive kernel frameworks.

4.6 Loss function

We use a composite loss to train to regularize our model. Define $\boldsymbol{\Theta} := (\boldsymbol{\theta}, \boldsymbol{\phi})$ and let x, y be the Reinhard-tonemapped predicted and

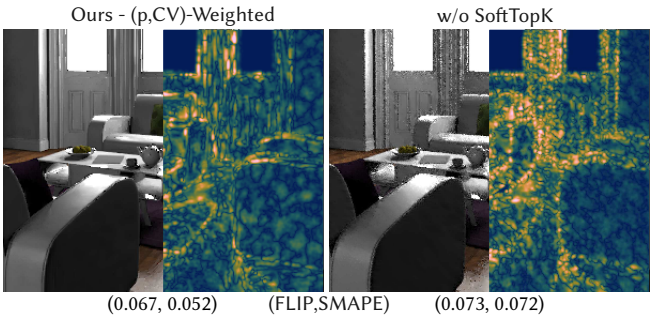

Fig. 5. Comparison with and without differentiable sorting networks (DSNs) for our method. Without DSN, optimization get trap inside a miminma which translate to artefacts inside the reconstructed image.

target pixels, respectively. We define our objective function as

$$\mathcal{L}(\boldsymbol{\Theta}) = \mathcal{L}_{\text{perc}}(\boldsymbol{\Theta}) + \lambda_1 \mathcal{L}_{\text{recons}}(\boldsymbol{\Theta}) + \lambda_2 \mathcal{L}_{\text{reg}}(\boldsymbol{\Theta}), \tag{14}$$

where

$$\begin{aligned} \mathcal{L}_{\text{perc}}(\boldsymbol{\Theta}) &= \mathbb{E}_{x,y}\, d_{\text{LPIPS}}\big(\text{sRGB}(x), \text{sRGB}(y)\big), \\ \mathcal{L}_{\text{recons}}(\boldsymbol{\Theta}) &= \mathbb{E}_{x,y}\left[\|x - y\|_1 + \|\delta x - \delta y\|_1\right], \\ \mathcal{L}_{\text{reg}}(\boldsymbol{\Theta}) &= \mathbb{E}_{x,y}\left[\|\mathbf{w}^{\text{g}}(1 - \mathbf{p}^{\text{g}})\|_1\right], \end{aligned} \tag{15}$$

and δ is the finite difference operator between a pixel and its neighbors. We add a regularization in the form of a L^1-loss on the non-selected neighbors; in essence, $\mathcal{L}_{\text{reg}}(\boldsymbol{\Theta})$ weakens unselected weights and smooths out the stiffness during annealing. Following previous works on neural gradient-domain rendering [Kettunen et al. 2019a; Liang et al. 2024], we set $\lambda_1 = \lambda_2 = 0.01$ for all experiments.

5 RESULTS AND DISCUSSION

5.1 Experimental setup

Dataset. To generate our dataset, we randomly sample scenes with photorealistic appearances and lighting profiles. We manually select 10 indoor scene templates across 5 categories (Bedroom, Kitchen, Living Room, Bathroom, Office) from the SceneNet dataset [Handa et al. 2016]. Each template is closed and contains a large amount of labelled furniture and everyday objects. We then randomly assign materials and sample camera extrinsics/intrinsics similar to Gharbi et al. [2019], for a total of 1000 unique frames. We compute image-space gradients using random replay shift-mapping at all pixels for a window size of $h = 5$ (i.e., 24 candidate neighbors). For adaptive sampling, we compute all gradient and primal buffers [Kuznetsov et al. 2018] along with their associated variance for the weighted reconstruction. All scenes are rendered at resolution 256×256 and we use $\{2, \ldots, 10\}$ spp for the input, 1024 spp for the gradients, and 4096 spp for the reference. More details about the dataset generation can be found in our supplemental material. Our dataset will be made publicly available upon publication.

Training and implementation. Our neighbor selection network is first trained with uniform sampling using the Adam optimizer [Kingma and Ba 2015]. We use the schedule-free approach of Defazio et al. [2024], starting at 10^{-4} and we simultaneously increase the stiffness parameter from $\tau_0 = 10^0$ to $\tau_1 = 10^4$. This final temperature

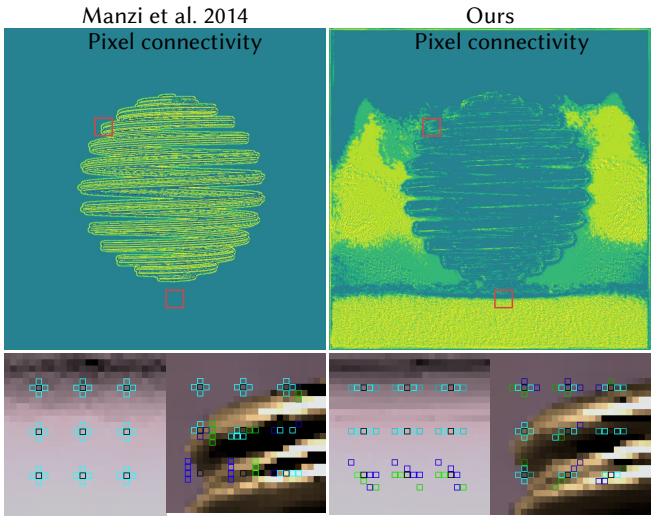

Fig. 6. Comparison between the number of pixels connected and the shape of the kernel between Manzi et al. [2014] and ours. Our method favors unstructured kernels in smooth areas and adapts to local features when required. Cyan indicates bidirectional selection, whereas blue and green mean forward and backward selections.

is chosen so that the soft-top-k operation converges to a pseudo-hard-top-k operation. We perform 50 000 iterations of training with a batch size of one, which saturates GPU utilization. In the last 15 000 iterations, we swap this soft approximation with a hard one.

We bootstrap the training of our adaptive sampling network by initially freezing the weights of the neighbor selection network for 15 000 iterations. After this phase, we jointly train both networks as we found that this boosts the overall performance of the model. In total, we perform 100 000 training iterations, which takes approximately 12 hours on a NVIDIA RTX 3080.

We implemented our method in PyTorch and integrated it atop Mitsuba 3's [Jakob et al. 2022b,a] rendering system. For the differentiable sorting network, we use the sparse implementation of DiffTopK [Petersen et al. 2022a,b] with bitonic sort and a Cauchy relaxation (Eq. (9)).

Baselines. We compare our method to the standard cross-shaped kernel shift-mapping and the structure-adaptive gradient kernel approach of Manzi et al. [2014], henceforth denoted as CSK and SAK, respectively. We also briefly compare with NGPT [Kettunen et al. 2019b] which, while more akin to a neural denoiser, uses gradient-domain information to perform a similar task. We do not compare against Manzi et al. [2016b] or Ha et al. [2019] as these works mainly focus on improving the Poisson reconstruction process.

As described in Section 4.4, if j is selected as a neighbor of i then j also receives information from Δ_{ij} at each reconstruction iteration even if i is not selected as a neighbor. We empirically observed that each pixel is linked to an average of 6 other pixels even though it has 4 selected neighbors. A similar phenomenon appears with the adaptive strategy of Manzi et al. [2014] with over 4.6 neighbors (on average) are selected and concentrated in regions of sharp discontinuities. Even if our adaptive kernel chooses more

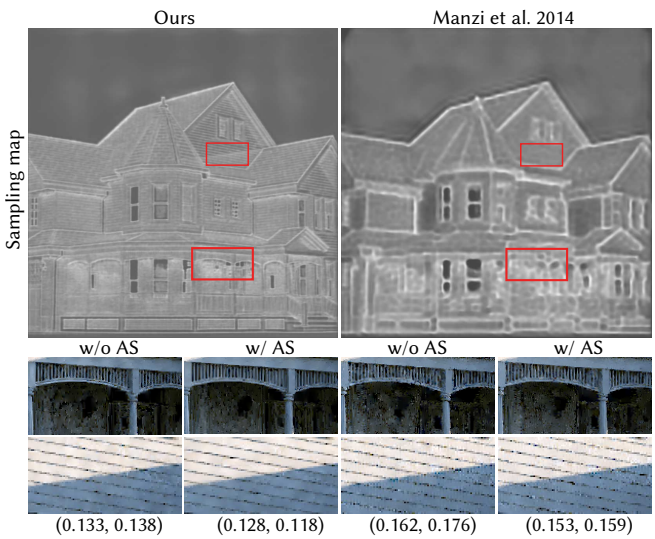

Fig. 7. Comparison of adaptive sampling with and without it for SAK [Manzi et al. 2014] and our method. We see comparable sampling patterns, but our method more effectively distributes samples across the image due to our end-to-end approach.

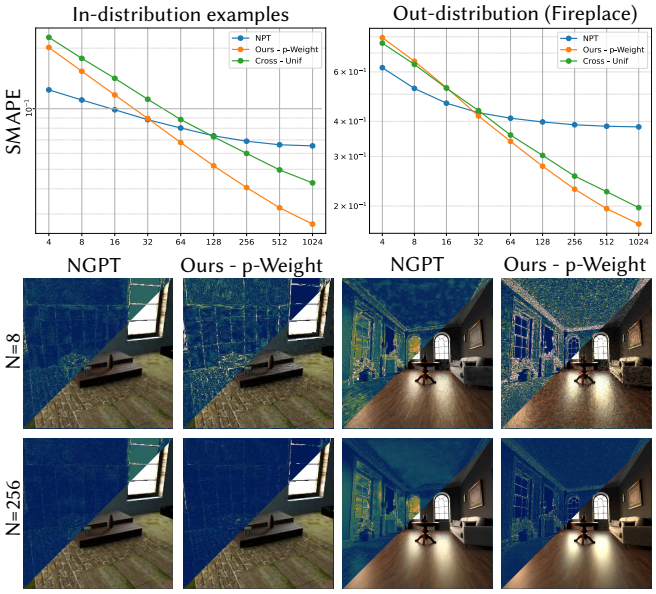

Fig. 8. Comparison between NGPT [Kettunen et al. 2019b] and our method across varying input sample counts. Our method demonstrates improved generalization and is unbiased compared to NGPT.

neighbors, the number of paths for estimating the image-space gradients and primal is the same for all techniques. For our analysis, we use the same number of samples across all different techniques.

5.2 Analysis

We report symmetric mean absolute percentage error (SMAPE) and ꟻLIP [Andersson et al. 2020] for our quantitative evaluation. We provide an interactive web viewer in our supplemental material to show full images as well as error maps for additional metrics. We refer to *the test dataset*, which includes the six scenes shown in Figures 10 and 11. All images are rendered at 800×800 resolution, while training was performed on 128×128 patches.

Kernel shape and number of neighbors. We show in Fig. 5 that using a Soft top-k relaxation during training is necessary for our model to produce optimal neighbors selection. We compare the shapes of our learned kernels and the distribution of selected neighbors at different pixel locations in Fig. 6. Our method produces strided kernels in smoother regions, facilitating the removal of lower frequency noise during the reconstruction step. In contrast, SAK strongly favors a cross-shaped kernel due its thresholding mechanism that reduces the risk of overfitting to specific auxiliary features. In challenging regions, our method generates shift-mapping kernels that better adapt to the object structures and properties, whereas SAK produces suboptimal kernels due to the hand-crafting nature of bilateral filtering.

Comparison with uniform reconstruction. Figure 10 compares the performance of our method against baseline approaches using *unweighted* reconstruction [Rousselle et al. 2016]. Our technique stands out as the only approach capable of effectively reducing dipole artifacts. In contrast, SAK demonstrates uneven reconstruction across the image, likely due to the impact of an uninformed neighbor selection policy on the reconstruction process. In smoother regions (e.g., ceilings), our method correctly removes low frequency artifacts at the same number of reconstruction steps due to our aligned kernels. We note that the inclusion of the predicted weights w^{p} and $\mathbf{w}^{\mathrm{g}}$ further improve the overall reconstruction quality. Compared to variance-based weights Rousselle et al. [2016], our weighted method does not particularly suffer at low sample counts (i.e., 1 spp).

Adaptive sampling. Figure 7 compares our adaptive sampling method with SAK augmented using deep adaptive sampling [Kuznetsov et al. 2018]. Our adaptive network is a lightweight MLP with 4 layers and 32 hidden dimensions each, as we found that the features extracted by our neighbor selection network provide already rich information for this task. The adaptive sampling network used in SAK employs the same per-sample encoder and U-Net architecture as our neighbor selection network. While both methods benefit from adaptive sampling, our fully trainable approach yields superior rendering results, thanks to the sampling map generator's access to salient contextual information from gradient-domain neighbors.

Weighted reconstruction. We showcase results using a weighted reconstruction scheme based on variance statistics [Rousselle et al. 2016] in Fig. 11. In particular, we employ our method with predicted weights w^{p} and $\mathbf{w}^{\mathrm{g}}$ as well a combination of these weights and control variates (see Section 4.4). The inclusion of variance statistics improves our method further and maintains an advantage for low-frequency noise reduction in smooth regions.

Neural reconstruction. Figure 8 compares our method against NGPT [Kettunen et al. 2019b] on both in-distribution and out-of-distribution scenes, since dense residual prediction networks tend to be more sensitive to train-test mismatches than kernel methods.

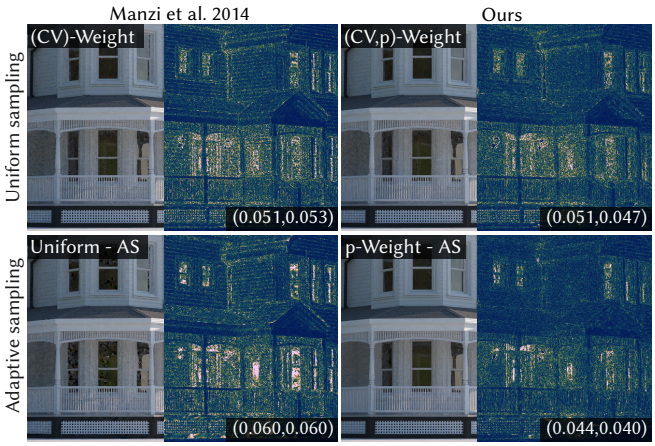

Fig. 9. Comparison of adaptive sampling with and without it for SAK [Manzi et al. 2014] and our method for path tracing integrator. We can observe that our adaptive sampling and kernel adaptation generalize to this integrator. SMAPE Error is visualize here.

We reimplemented NGPT in PyTorch but used the same U-Net backend as our method (i.e., without dense connections) to fairly assess the benefits of dense predictions over adaptive kernels. The input features include primal and image-space gradients, surface normals, albedo, and depth. We used the LPIPS [Zhang et al. 2018] loss instead of E-LPIPS [Kettunen et al. 2019c], as no PyTorch implementation of the latter is publicly available. The in-distribution inputs are generated from our synthetic scene generator used for training. While NGPT exhibits good reconstruction capabilities, results are always biased even at high sample counts and quality plateaus at around 32 spp. In contrast, our framework does not suffer from these issues and remains unbiased, hence showing better generalization.

Indirect lighting. We show how our methods perform with path tracing, where image-space gradients are estimated by random replay (Fig. 9) and path reconnection (Fig. 1). Our network was trained on the same dataset, regenerated using path tracing with random replay. We found that the network trained on direct lighting also generalizes well to the path-traced estimator. We maintain that one of the benefits of our *p-Weight* approach is its simplicity and ease of integration into a deep adaptive sampling procedure. CV Weighted reconstruction requires more complex gradient computation and it is left to future work.

Table 1. Equal-time comparison of our method and prior work across the test dataset, showing error metrics under uniform and weighted reconstruction.

	Method	spp	ꟻLIP	LPIPS	SMAPE
Unif.	CSK [Lehtinen et al. 2013]	74	0.0242	0.117	0.121
	SAK [Manzi et al. 2014]	74	0.0256	0.124	0.110
	Ours	64	**0.0242**	**0.107**	**0.080**
Weight.	CSK [Lehtinen et al. 2013]	74	**0.0234**	0.110	0.086
	SAK [Manzi et al. 2014]	74	0.0246	0.118	0.085
	Ours	64	0.0236	**0.108**	**0.076**

Table 2. Comparison of the different error metrics of different variations of our methods and prior works across the test dataset.

Method	ꟻLIP	LPIPS	SMAPE
CSK [Lehtinen et al. 2013]	0.040	0.185	0.168
SAK [Manzi et al. 2014]	0.042	0.194	0.158
CSK + Reconstruction weights	0.041	0.180	0.128
+ Hard top-k	0.057	0.271	0.181
+ Soft top-k	0.046	0.258	0.178
+ Fourier and material features	0.040	0.178	0.122
+ Primal image	0.039	0.154	0.117
+ Per-sample features	**0.037**	**0.147**	**0.117**
CSK + Full Input Features	0.040	0.173	0.125

Performance. We profile our method on a NVIDIA RTX 4080 Super GPU on the Kitchen scene (800×800). Gathering features other than primal takes a flat 2 ms, and inference takes 12 ms. The former accounts for 6 spp of direct illumination and 2 spp in path tracing mode. Additionally, the 10 samples of input primal cost an equivalent of 3 spp, since image-space gradients are not required. Our total overhead is thus 9 spp in direct illumination and 5 spp in path tracing. Figure 1 shows an equal-time comparison that accounts for this overhead and demonstrates that, even under these conditions, our method consistently outperforms alternative image-space kernels. Moreover, Table 1 reports equal-time comparison results on the test dataset. With uniform reconstruction, our method achieves better metrics than CSK and SAK, particularly in SMAPE. This metric heavily penalizes dipole artifacts, which are not effectively reduced by simply adding more samples. Our framework also generalizes well to weighted reconstruction, achieving comparable performance in FLIP and LPIPS, while obtaining superior scores in SMAPE.

5.3 Ablation study

Table 2 quantifies the contribution of each design decision to average test error, with corresponding CSK and SAK scores. The results confirm that all components are essential for achieving maximum performance. Initially, we train using forced CSK selection with only reconstruction weight optimization. Following Manzi et al. [2014], we use basic depth, albedo, and normal buffers as input. This approach produces only moderate baseline improvements. Hard top-k neighbor selection degrades output quality by restricting exploration. Swapping for a soft top-k selection, on the other hand, improves quality but remains limited due to simplified input data. Enriching the input data further improves results. Pre-sampled primal data provides essential lighting information, while per-sample buffer inputs reveal feature variance that helps identify geometric and illumination discontinuities. Training reconstruction weights without our adaptive kernel selection (last row of Table 2), but using all the input features described above, results in only marginal improvement. This demonstrates that a more flexible neighbor selection is necessary to improve the performance of our method.

6 CONCLUSIONS

We introduced a robust, data-driven method for adaptive shift-mapping kernels in gradient-domain rendering. Our approach is integrated with adaptive sampling and enables flexible reconstruction schemes. We demonstrated that our neural kernels adaptively generate neighbor selection policies that leverages surface interaction properties, achieving superior performance compared to traditional and other adaptive kernel methods.

Limitations and future work. Our method has a few limitations. First, we have trained our network mostly on direct lighting which can lead to degradation in quality for other integrators. Different shift-mappings have different variance profiles that may result in different sets of optimal neighbors. Second, we only trained our network on uniform reconstruction [Rousselle et al. 2016] with predicted weights. We expect that different reconstruction will change the neighbors selection significantly, especially if the reconstruction is learning-based.

For future work, a natural avenue is the exploration of deep reconstruction techniques that support unconnected kernels. Enforcing reversibility directly during neighbor selection—rather than relying on the post-processing currently employed—remains a key area of improvement. Another avenue is the prediction of an optimal number of neighbors to consider dynamically. Incorporating efficiency metrics based on the expected cost of primal versus gradient contributions could also help balance quality and performance. Finally, we believe extending our approach to other types of integrators, such as temporal or volumetric integrators, would broaden its applicability.

Acknowledgments. We thank the anonymous reviewers for their insightful feedback and constructive comments that helped improve this paper. We acknowledge the use of third-party resources from Benedikt Bitterli's dataset [Bitterli 2016], with the following image credits: Coffee scene by cekuhnen, House scene by MrChimp2313, Livingroom and Kitchen scenes by Jay-Artist, and Fireplace and Staircase scenes by Wig42. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), Discovery Grant GPIN/3182-2022. We gratefully acknowledge the Digital Research Alliance of Canada for providing the computational resources that enabled this work.

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 2 (2020), 15–1. https://doi.org/10.1145/3406183

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Transactions on Graphics* 39, 6 (2020), 242–1. https://doi.org/10.1145/3414685.3417847

Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2018. Feature Generation for Adaptive Gradient-Domain Path Tracing. *Computer Graphics Forum* 37, 7 (2018), 65–74. https://doi.org/10.1111/cgf.13548

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics* 36, 4 (2017). https://doi.org/10.1145/3072959.3073708

Kenneth E. Batcher. 1968. Sorting networks and their applications. In *Proceedings of Spring Joint Computer Conference*. 307–314. https://doi.org/10.1145/1468075.1468121

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain path reusing. *ACM Transactions on Graphics* 36, 6 (2017), 1–9. https://doi.org/10.1145/3130800.3130886

Pravin Bhat, Brian Curless, Michael Cohen, and C Lawrence Zitnick. 2008. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *European Conference on Computer Vision*. 114–128. https://doi.org/10.1007/978-3-540-88688-4_9

Pravin Bhat, C. Lawrence Zitnick, Michael Cohen, and Brian Curless. [n. d.]. GradientShop: A gradient-domain optimization framework for image and video filtering. *ACM Transactions on Graphics* 29, 2, Article 10 ([n. d.]), 14 pages. https://doi.org/10.1145/1731047.1731048

Benedikt Bitterli. 2016. Rendering resources. https://benedikt-bitterli.me/resources/.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. 36, 4 (2017), 98:1–98:12. https://doi.org/10/gbxhcv

Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. 2021. Differentiable patch selection for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2351–2360. https://doi.org/10.1109/CVPR46437.2021.00238

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik P. A. Lensch. 2010. Edge-avoiding À-Trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics*. 67–75. https://doi.org/10.5555/1921479.1921491

Aaron Defazio, Xingyu (Alice) Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. 2024. The Road Less Scheduled. *Advances in Neural Information Processing Systems (NeurIPS)* 37, 9974–10007.

Q. Fang and T. Hachisuka. 2024. Lossless Basis Expansion for Gradient-Domain Rendering. *Computer Graphics Forum* 43, 4 (2024), e15153. https://doi.org/10.1111/cgf.15153

Basile Fraboni, Antoine Webanck, Nicolas Bonneel, and Jean-Claude Iehl. 2022. Volumetric Multi-View Rendering. In *Computer Graphics Forum*, Vol. 41. 379–392. https://doi.org/10.1111/cgf.14481

Michaël Gharbi, Tzu-Mao Li, Miika Aittala, Jaakko Lehtinen, and Frédo Durand. 2019. Sample-based Monte Carlo denoising using a kernel-splatting network. *ACM Transactions on Graphics* 38, 4, Article 125. https://doi.org/10.1145/3306346.3322954

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. 2019. Stochastic Optimization of Sorting Networks via Continuous Relaxations. (2019). arXiv:1903.08850 https://arxiv.org/abs/1903.08850

Adrien Gruson, Binh-Son Hua, Nicolas Vibert, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2018. Gradient-domain volumetric photon density estimation. *ACM Transactions on Graphics* 37 (2018). https://doi.org/10.1145/3197517.3201363

Jie Guo, Mengtian Li, Quewei Li, Yuting Qiang, Bingyang Hu, Yanwen Guo, and Ling-Qi Yan. 2019. GradNet: unsupervised deep screened poisson reconstruction for gradient-domain rendering. *ACM Transactions on Graphics* 38, 6 (2019), 1–13. https://doi.org/10.1145/3355089.3356538

Saerom Ha, Sojin Oh, Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2019. Gradient Outlier Removal for Gradient-Domain Path Tracing. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 245–253. https://doi.org/10.1111/cgf.13634

A Nico Habermann. 1972. Parallel neighbor-sort (or the glory of the induction principle). (1972).

Ankur Handa, Viorica Pătrăucean, Simon Stent, and Roberto Cipolla. 2016. SceneNet: an Annotated Model Generator for Indoor Scene Understanding. In *IEEE International Conference on Robotics and Automation (ICRA)*. https://doi.org/10.1109/ICRA.2016.7487797

Jon Hasselgren, Jacob Munkberg, Marco Salvi, Anjul Patney, and Aaron Lefohn. 2020. Neural temporal adaptive sampling and denoising. In *Computer Graphics Forum*, Vol. 39. 147–155. https://doi.org/10.1111/cgf.13919

Nikolai Hofmann, Jon Hasselgren, Petrik Clarberg, and Jacob Munkberg. 2021. Interactive path tracing and reconstruction of sparse volumes. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 4, 1 (2021), 1–19. https://doi.org/10.1145/3451256

Binh-Son Hua, Adrien Gruson, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2017. Gradient-Domain Photon Density Estimation. In *Computer Graphics Forum*, Vol. 36. https://doi.org/10.1111/cgf.13104

Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. 2019. A Survey on Gradient-Domain Rendering. *Computer Graphics Forum* 38, 2, 455–472. https://doi.org/10.1111/cgf.13652

Yuchi Huo and Sung-eui Yoon. 2021. A survey on deep learning-based Monte Carlo denoising. *Computational Visual Media* 7 (2021), 169–185. https://doi.org/10.1007/s41095-021-0209-9

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Transactions on Graphics* 40, 4, Article 37 (2021). https://doi.org/10.1145/3450626.3459793

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022b. *Mitsuba 3 Renderer*. https://mitsuba-renderer.org

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022a. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *ACM Transactions on Graphics* 41, 4 (2022). https://doi.org/10.1145/3528223.3530099

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019a. Deep convolutional reconstruction for gradient-domain rendering. *ACM Transactions on Graphics* 38, 4, Article 126 (2019). https://doi.org/10.1145/3306346.3323038

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019b. Deep convolutional reconstruction for gradient-domain rendering. *ACM Transactions on Graphics* 38, 4, Article 126 (2019). https://doi.org/10.1145/3306346.3323038

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019c. E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles. *CoRR* abs/1906.03973 (2019). arXiv:1906.03973 http://arxiv.org/abs/1906.03973

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain path tracing. *ACM Transactions on Graphics* 34, 4 (2015), 1–13. https://doi.org/10.1145/2766997

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

Peter Kipfer and Rüdiger Westermann. 2005. Improved GPU sorting. *GPU gems* 2 (2005), 733–746.

Donald E. Knuth. 1998. *The Art of Computer Programming, Sorting and Searching* (2 ed.). Vol. 3. Addison Wesley Longman Publishing Co., Inc., USA.

Wouter Kool, Herke Van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning (ICML)*. 3499–3508. http://proceedings.mlr.press/v97/kool19a.html

Alexandr Kuznetsov, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2018. Deep Adaptive Sampling for Low Sample Count Rendering. *Computer Graphics Forum* 37, 4 (2018), 35–44. https://doi.org/10.1111/cgf.13473

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-Domain Metropolis Light Transport. *ACM Transactions on Graphics* 32, 4 (2013). https://doi.org/10.1145/2461912.2461943

Yuzhi Liang, Tao Liu, Yuchi Huo, Rui Wang, and Hujun Bao. 2024. Adaptive sampling and reconstruction for gradient-domain rendering. *Computational Visual Media* 10 (2024), 885–902. https://doi.org/10.1007/s41095-023-0361-5

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized resampled importance sampling: Foundations of restir. *ACM Transactions on Graphics* 41, 4 (2022), 1–23. https://doi.org/10.1145/3528223.3530158

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing. In *Proceedings of EGSR (Experimental Ideas & Implementations)*. https://doi.org/10.2312/sre.20151168

Marco Manzi, Markus Kettunen, Frédo Durand, Matthias Zwicker, and Jaakko Lehtinen. 2016a. Temporal gradient-domain path tracing. *ACM Transactions on Graphics* 35, 6, Article 246 (2016). https://doi.org/10.1145/2980179.2980256

Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. 2014. Improved sampling for gradient-domain Metropolis light transport. *ACM Transactions on Graphics* 33, 6 (2014), 1–12. https://doi.org/10.1145/2661229.2661291

Marco Manzi, Delio Vicini, and Matthias Zwicker. 2016b. Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches. *Computer Graphics Forum* 35, 2 (2016), 263–273. https://doi.org/10.1111/cgf.12829

Xiaoxu Meng, Quan Zheng, Amitabh Varshney, Gurprit Singh, and Matthias Zwicker. 2020. Real-time Monte Carlo Denoising with the Neural Bilateral Grid. In *Proceedings of the Eurographics Symposium on Rendering (EGSR)*. https://doi.org/10.2312/sr.20201133

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive polynomial rendering. *ACM Transactions on Graphics* 35, 4 (2016), 1–10. https://doi.org/10.1145/2897824.2925936

Jacob Munkberg and Jon Hasselgren. 2020. Neural Denoising with Layer Embeddings. *Computer Graphics Forum* 39, 4 (2020), 1–12. https://doi.org/10.1111/cgf.14049

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. 2021. Learning with algorithmic supervision via continuous relaxations. *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), 16520–16531.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. 2022a. Monotonic Differentiable Sorting Networks. In *International Conference on Learning Representations (ICLR)*.

Felix Petersen, Hilde Kuehne, Christian Borgelt, and Oliver Deussen. 2022b. Differentiable Top-k Classification Learning. arXiv:2206.07290 [cs.LG] https://arxiv.org/abs/2206.07290

Victor Petitjean, Pablo Bauszat, and Elmar Eisemann. 2018. Spectral Gradient Sampling for Path Tracing. *Computer Graphics Forum* 37, 4 (2018), 45–53. https://doi.org/10.1111/cgf.13474

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention (MICCAI)*. Springer, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space control variates for rendering. *ACM Transactions on Graphics* 35, 6, Article 169 (2016). https://doi.org/10.1145/2980179.2982443

Farnood Salehi, Marco Manzi, Gerhard Roethlin, Romann Weber, Christopher Schroers, and Marios Papas. 2022. Deep adaptive sampling and reconstruction using analytic distributions. *ACM Transactions on Graphics* 41, 6 (2022), 1–16. https://doi.org/10.1145/3550454.3555515

Weilun Sun, Xin Sun, Nathan A. Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-domain vertex connection and merging. In *Proceedings of EGSR (Experimental Ideas & Implementations)*. 83–92. https://doi.org/10.2312/sre.20171197

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), 7537–7547.

Eric Veach. 1998. *Robust Monte Carlo methods for light transport simulation*. Stanford University.

Kin-Ming Wong and Tien-Tsin Wong. 2019. Deep residual learning for denoising Monte Carlo renderings. *Computational Visual Media* 5 (09 2019). https://doi.org/10.1007/s41095-019-0142-3

Chris Wyman, Markus Kettunen, Daqi Lin, Benedikt Bitterli, Cem Yuksel, Wojciech Jarosz, and Pawel Kozlowski. 2023. A gentle introduction to restir path reuse in real-time. In *ACM SIGGRAPH Courses*. 1–38. https://doi.org/10.1145/3587423.3595511

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. 2020. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), 20520–20531.

Zilin Xu, Qiang Sun, Lu Wang, Yanning Xu, and Beibei Wang. 2020. Unsupervised Image Reconstruction for Gradient-Domain Volumetric Rendering. In *Computer Graphics Forum*, Vol. 39. 193–203. https://doi.org/10.1111/cgf.14137

Difei Yan, Shaokun Zheng, Ling-Qi Yan, and Kun Xu. 2024. Filtering-Based Reconstruction for Gradient-Domain Rendering. In *ACM SIGGRAPH Asia Conference Papers*. 1–10. https://doi.org/10.1145/3680528.3687568

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. arXiv:1801.03924 [cs.CV] https://arxiv.org/abs/1801.03924

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and S-E Yoon. 2015. Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. In *Computer Graphics Forum*, Vol. 34. 667–681. https://doi.org/10.1111/cgf.12592

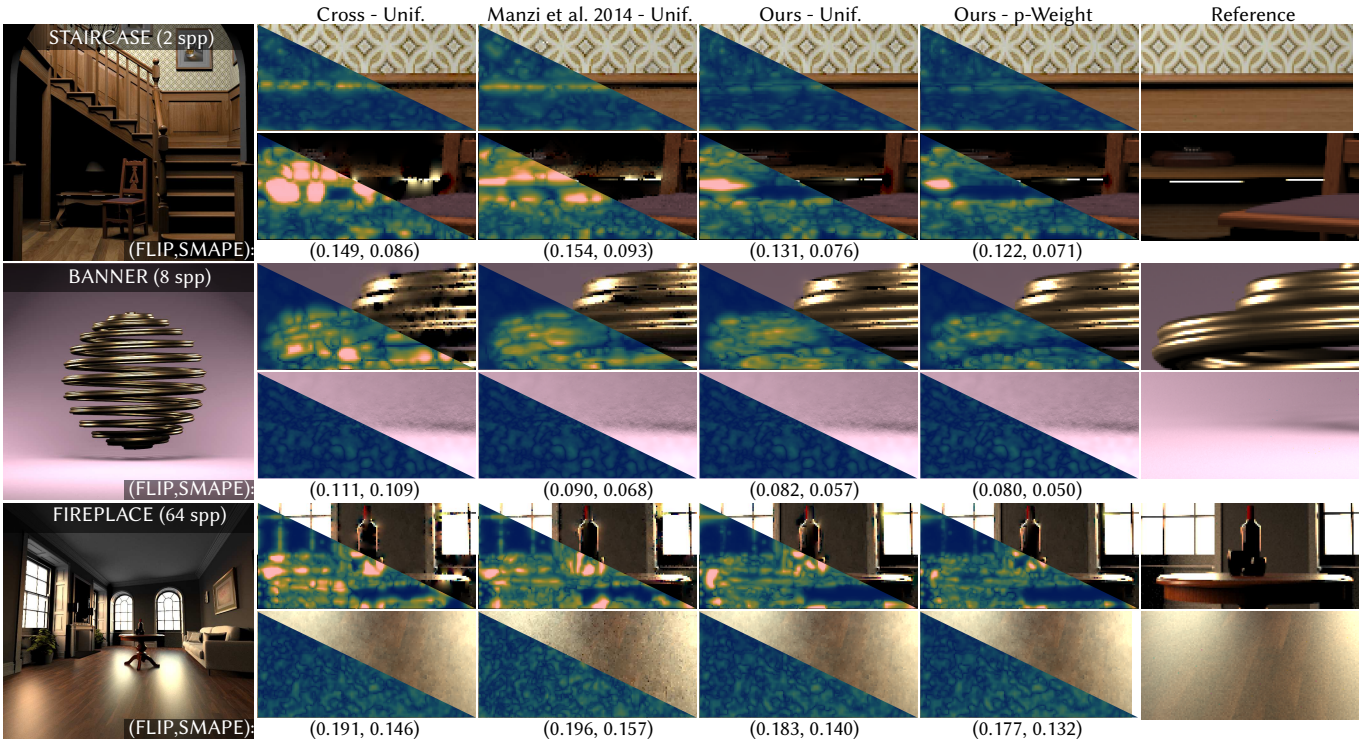

Fig. 10. Comparison between our method and baselines with *uniform* reconstruction [Rousselle et al. 2016] and our methods of reconstruction using our predicted weights (*p-Weight*). We can see that in all regions, our method reduces dipole artefacts and accurately captures geometrical and shading changes.

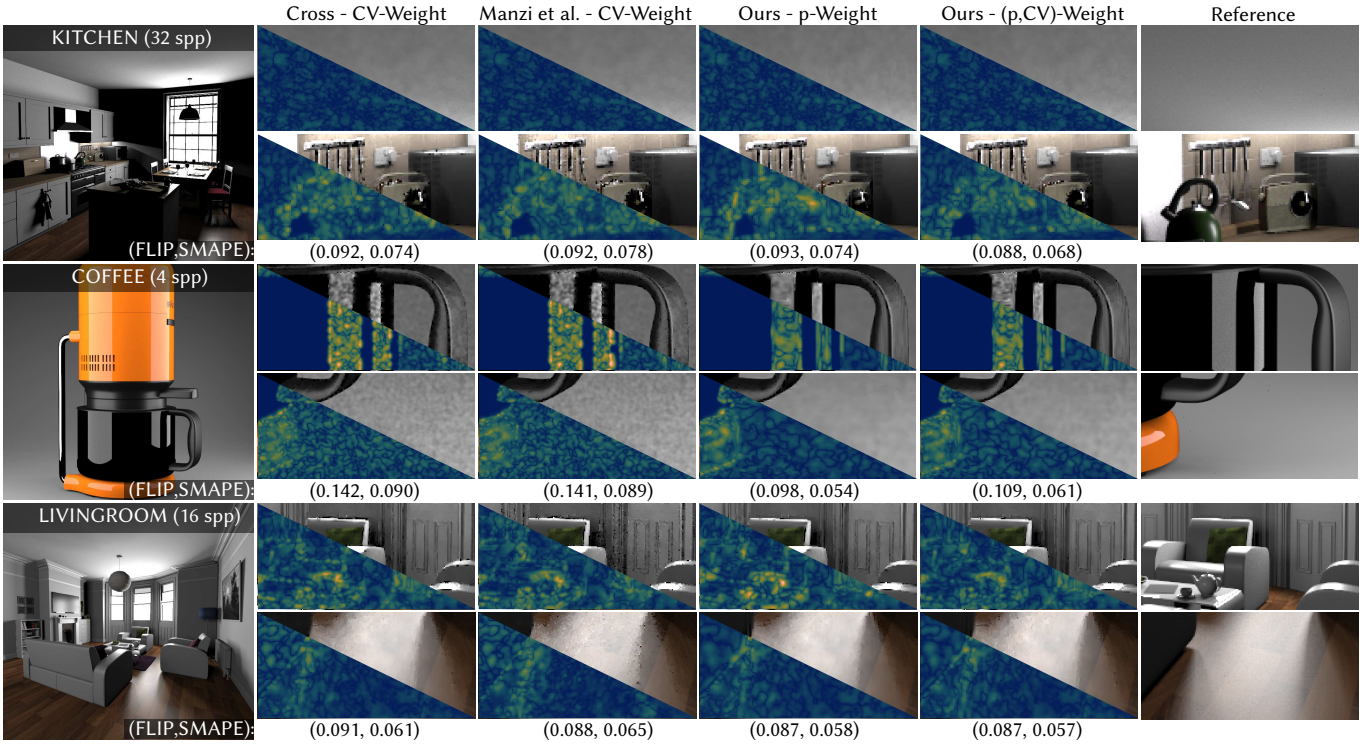

Fig. 11. Comparison between our method and baselines with *CV weighted reconstruction* [Rousselle et al. 2016] and our method combining our predicted weights and variance statistics (*(p,CV)-Weight*). All regions see reduced reconstruction artifacts when variance statistics are included.