DSCombiner: Double Shrinkage for Combining Biased and Unbiased Monte Carlo Renderings

CHENXI ZHOU, KEHENG XU, MUFAN GUO, XIANHAO YU, ZHIMIN FAN, GUIHUAN FENG, YANWEN GUO, and JIE GUO*, State Key Lab for Novel Software Technology, Nanjing University, China

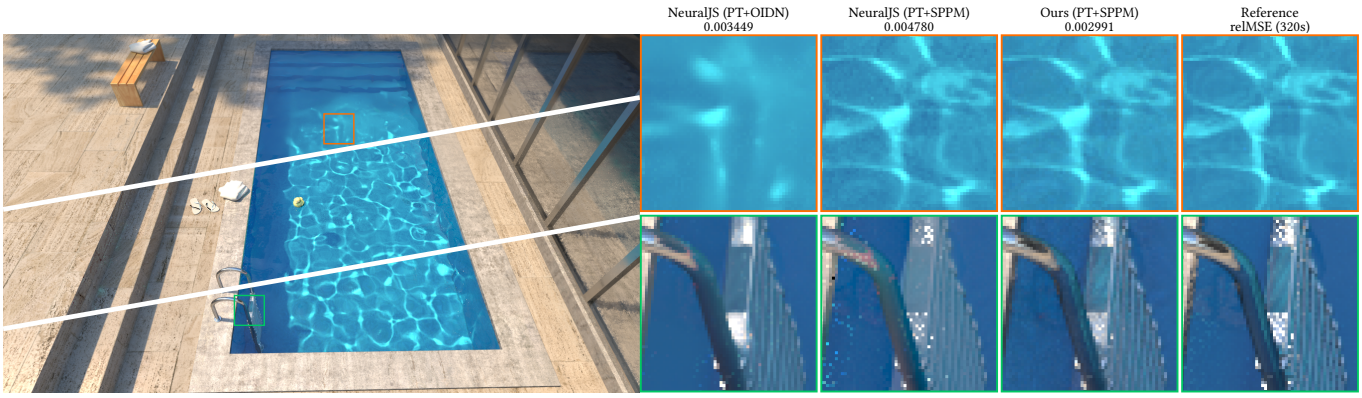

Fig. 1. Equal-time comparison between NeuralJS [Gu et al. 2022] and our *DSCombiner* in the Swimming Pool scene. DSCombiner achieves a superior combination of biased and unbiased renderings compared to the state-of-the-art techniques, especially for renderings generated by different integrators. Left, from top to down: results of NeuralJS (PT+OIDN), NeuralJS (PT+SPPM), and DSCombiner (PT+SPPM). Right: close-up comparisons with error metrics.

Monte Carlo rendering often faces a dilemma, namely, whether to choose an unbiased estimator or a biased one. Although different integrators have been developed to address various scenarios, no single method can effectively manage all situations. Thus, finding a good approach to combine different integrators has always been a topic that warrants exploration.

This work proposes DSCombiner, a new shrinkage estimator that flexibly combines unbiased and biased estimators (typically generated by different integrators) in image space into a single estimating procedure, strategically utilizing the strengths of different integrators while minimizing their weaknesses. DSCombiner overcomes the limitation of single shrinkage combiners by introducing a two-step shrinkage towards a noise-free radiance prior. We derive optimal shrinkage factors for the two steps within a hierarchical Bayesian framework, and provide a deep learning-based method to improve the results. Comprehensive qualitative and quantitative validations across diverse scenes demonstrate visible improvements in image quality, as compared with previous image-space and path-space combiners.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Rendering

ACM Reference Format:
Chenxi Zhou, Keheng Xu, Mufan Guo, Xianhao Yu, Zhimin Fan, Guihuan Feng, Yanwen Guo, and Jie Guo. 2025. DSCombiner: Double Shrinkage for Combining Biased and Unbiased Monte Carlo Renderings. *ACM Trans. Graph.* 44, 6, Article 203 (December 2025), 11 pages. https://doi.org/10.1145/3763315

*Corresponding author.

Authors' Contact Information: Chenxi Zhou, chenxizhou@smail.nju.edu.cn; Keheng Xu, kehengxu@smail.nju.edu.cn; Mufan Guo, orlando128@163.com; Xianhao Yu, 2943003@qq.com; Zhimin Fan, zhiminfan2002@gmail.com; Guihuan Feng, fenggh@nju.edu.cn; Yanwen Guo, ywguo@nju.edu.cn; Jie Guo, guojie@nju.edu.cn, State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China.

1 INTRODUCTION

Over the years of development, Monte Carlo (MC) rendering has remained the de facto standard for offline physically-based light transport simulation [Christensen and Jarosz 2016; Novák et al. 2018; Pharr et al. 2023] and is steadily gaining popularity in real-time scenarios [Bitterli et al. 2020; Cascioli and Reznikov 2024; Wyman et al. 2023]. Despite their simplicity of implementation, ability to handle complex lighting effects, and scalability, existing MC rendering algorithms are always facing the bias-variance dilemma. Specifically, unbiased algorithms (e.g., path tracing [Kajiya 1986]) estimate the pixel's intensity with no statistical bias but suffer from displeasing noise (i.e., variance), which makes them require a huge number of samples to converge. Biased algorithms (e.g., progressive photon mapping [Hachisuka et al. 2008] or path tracing with a denoiser [Bako et al. 2017; Qiao et al. 2024; Vogels et al. 2018; Zheng et al. 2021]), on the other hand, excel in suppressing the influence of noise, but at the cost of introducing bias in their estimators.

An appealing and increasingly active trend to accelerate the convergence rate of MC rendering is by employing a post-correction framework that combines both unbiased and biased rendering pairs. For instance, recent work has demonstrated that the biased denoising results can be blended with their noisy counterparts generated by an unbiased estimator to improve the quality of denoisers [Firmino et al. 2022, 2024; Gu et al. 2022]. Typically, screen-space blending is achieved by the James-Stein (JS) estimator [James and Stein 1961], which is a well-known shrinkage estimator using a single shrinkage

factor. The key insight of the JS estimator is that it can outperform the maximum likelihood estimator (e.g., the sample mean) by shrinking the individual estimates towards the overall mean.

Stein-like shrinkage estimators with a single shrinkage factor are particularly promising for lowering the systematic bias for learning-based denoisers [Gu et al. 2022]. In this context, the shrinker aims to refine the output of a denoiser by shrinking the estimated values towards the raw noisy data. This minimizes the risk of estimation in terms of the mean squared error (MSE) between the final denoised image and the noise-free ground truth, under the assumption that variance is homogeneous in each local patch. However, when combining two estimators with different MC integrators (e.g., one is path tracing and the other is stochastic progressive photon mapping), the single shrinkage method is usually sub-optimal since both estimators may have high variance that varies spatially and wildly.

To address the aforementioned problem, we propose *DSCombiner*, a *double shrinkage estimator* for MC renderings. This estimator could effectively combine a wide range of unbiased and biased MC integrators, providing flexibility in selecting the most appropriate methods for various rendering scenarios. DSCombiner operates in two passes: first, it computes a convex combination of the unbiased and biased estimators using the shrinkage factors derived from the variance and bias of both estimators. Then, augmented by a well-designed *radiance prior*, an additional shrinkage pass is applied to the output of the first pass, resulting in reduced variance. We derive the theoretically optimal shrinkage factors governing the two passes within a hierarchical Bayesian framework, and provide a deep learning-based practical solution to refine the results.

In summary, our main contributions are as follows:

- We introduce the first double shrinkage framework, dubbed *DSCombiner*, to flexibly and efficiently combine a wide range of unbiased and biased MC integrators.
- We obtain the shrinkage factors using a hierarchical Bayesian framework equipped with a learnable convolutional neural network (CNN). Our pixel-level shrinkage factors make the combination robust in regions with details.
- We design a radiance prior to further accelerate the convergence of the shrinkage estimator.

Through extensive experiments on multiple challenging scenes, we demonstrate that our proposed double shrinkage estimator visibly outperforms its single shrinkage counterparts and improves the convergence rate of MC renderings.

2 RELATED WORK

Path-space combination of MC renderings. The quality of Monte Carlo integration is highly affected by its sampling procedure, namely, how the integrator constructs the light path connecting the camera and the light source stochastically in Monte Carlo-based ray tracing. Numerous integrators [Hachisuka et al. 2008; Jensen 2001; Veach and Guibas 1997] have been developed to tackle various challenges in light transport simulation. Due to their different focuses, no single integrator can effectively handle all scenarios. To utilize different sampling techniques simultaneously, some prior studies [Georgiev et al. 2012; Hachisuka et al. 2012; Křivánek et al. 2014] combine different integrators in path space using multiple importance sampling (MIS) [Veach and Guibas 1995]. While MIS is considered the de facto standard solution for combining estimators generated by different integrators, its proper application is not that easy, and combining n techniques incurs a runtime complexity of $O(n^2)$. Additionally, some integrators (e.g., manifold sampling [Fan et al. 2023; Pediredla et al. 2020; Zeltner et al. 2020]) do not provide analytical PDFs that are essential for MIS.

Screen-space combination of MC renderings. Another line of research focuses on combining MC estimates in screen space, drawing inspiration from the *shrinkage estimator* used in other fields [Green and Strawderman 1991; Lavancier and Rochet 2016]. In MC rendering, some recent studies have demonstrated that a combination of rendering results generated by an unbiased integrator with its denoised version can outperform either of the two inputs individually. For example, Back et al. [2020] emphasize the strong correlation in denoised images (e.g., gradient-domain path tracing [Kettunen et al. 2015] with L2 reconstruction) and develop a combination kernel to reduce residual errors. Similarly, Gu et al. [2022] and Firmino et al. [2022] focus on improving the consistency of learning-based denoisers, mitigating errors introduced by learning-based denoisers for nearly converged inputs. Besides, Zheng et al. [2021] propose an ensemble learning-based framework to combine estimates generated by different denoisers. Otsu et al. [2018] explore a new scheme of estimator combination through machine learning, which greatly inspires our work. However, their selected feature representation results in prohibitive storage costs. This issue motivates us to adopt an error-based combination framework instead.

MC denoising. To alleviate the noise in images synthesized by MC integrators, numerous studies have focused on directly manipulating noisy pixels to reduce variance. Filtering-based methods smooth the image via a specially designed filtering process, with kernels ranging from traditional Gaussian [Rousselle et al. 2011], handcrafted bilateral filters [Li et al. 2012; Sen and Darabi 2012] to deep learning-based approaches [Bako et al. 2017; Balint et al. 2023; Vogels et al. 2018]. Leveraging noise-free G-buffers (e.g., normal, albedo, and depth), learning-based denoisers can predict high-quality filtering kernels to significantly improve MC rendering. Alternatively, regression-based methods [Bitterli et al. 2016; Choi et al. 2024; Qiao et al. 2024] suppress noise by fitting a smooth function to express pixel values in local image blocks. Notably, some works interpret the denoising process from a statistical perspective. For instance, Boughida and Boubekeur [2017] resolve the denoised color through maximizing a posteriori probability, while Sakai et al. [2024] exclude unsuitable pixels using the Welch t-test to prevent noticeable bias from averaging samples with different statistical properties.

3 BACKGROUND AND MOTIVATION

In this section, we begin with a brief review of single shrinkage methods, with a focus on the prevailing James-Stein shrinkage [James and Stein 1961], for combining two different MC estimators. Subsequently, we elucidate the motivation behind our new double shrinkage method. For a more comprehensive explanation of shrinkage estimators, please refer to [Lavancier and Rochet 2016]. We summarize important symbols in Table 1.

Table 1. List of important symbols.

Symbol	Description
X	Ground truth of the quantity we want to estimate
X_u, X_b	Unbiased and biased estimates of X
σ_u^2	The variance of unbiased estimate
σ_b^2, ξ	The variance and bias of biased estimate
γ^2	The variance of the Gaussian prior distribution for ξ
X_p	The mean of the Gaussian prior distribution for X
η^2	The variance of the Gaussian prior distribution for X

Single Shrinkage. Suppose we have an unbiased estimate X_u (variance: σ_u^2) and a biased estimate X_b (variance: σ_b^2, and bias: ξ) of a real-valued scalar $X \in \mathbb{R}$, such that

$$\begin{aligned} X_u &\sim \mathcal{N}(X, \sigma_u^2), \\ X_b &\sim \mathcal{N}(X + \xi, \sigma_b^2), \end{aligned} \tag{1}$$

where $\mathcal{N}$ denotes a normal distribution. We can construct a single shrinkage estimator by linearly combining them as follows

$$X_{\text{Single}} = \lambda X_b + (1 - \lambda) X_u, \tag{2}$$

where λ is the shrinkage factor. By carefully selecting the shrinkage factor, this new estimator X_{Single} can be shown to improve X_u, in terms of MSE (mean squared error), via shrinking X_u towards X_b.

James-Stein Shrinkage. Suppose $\mathbf{X} \in \mathbb{R}^p$ is a p-dimensional vector. We have an unbiased estimate $\mathbf{X}_u$ and a biased estimate $\mathbf{X}_b$. If

- the dimensionality $p \geq 3$, and
- the variance of $\mathbf{X}_u$ is homogeneous across all dimensions (i.e., the covariance of $\mathbf{X}_u$ is a diagonal matrix $\sigma_u^2 \mathbf{I}$),

the James-Stein's style shrinkage estimator [Green and Strawderman 1991; James and Stein 1961]

$$\mathbf{X}_{\text{JS}} = \lambda \mathbf{X}_b + (1 - \lambda) \mathbf{X}_u \tag{3}$$

with

$$\lambda = \frac{(p-2)\sigma_u^2}{\|\mathbf{X}_u - \mathbf{X}_b\|^2} \tag{4}$$

produces a biased estimator of $\mathbf{X}$ that achieves lower MSE than any least squares estimators of $\mathbf{X}$. More specifically, the MSE of $\mathbf{X}_{\text{JS}}$

$$\mathbb{E}\left[(\mathbf{X}_{\text{JS}} - \mathbf{X})^2\right] = p\sigma_u^2 - \frac{(p-2)^2 \sigma_u^4}{\|\mathbf{X}_u - \mathbf{X}_b\|^2} \tag{5}$$

is always lower than that of $\mathbf{X}_u$ [Green and Strawderman 1991].

Motivation. Gu et al. [2022] recently proposed NeuralJS that applies James-Stein shrinkage to combine a pair of unbiased and biased renderings (typically the input and output of a Monte Carlo denoiser), which improves the error reduction rates of state-of-the-art learning-based denoisers for many scenes. To meet the previously listed requirements, they perform the shrinkage estimation within local image patches with approximated homogeneous variance.

Nevertheless, the variance within an image patch can exhibit great spatial variation. Consequently, an averaged patch-level shrinkage factor is inadequate for addressing this internal variation, as demonstrated in Fig. 2. Since the unbiased renderings (the second column) have spatially-varying variance (the last column), the shrinkage factor λ of pixels with high variance will be underestimated since their neighbor variances are much lower, making the noise remain around the boundaries where the variance changes wildly.

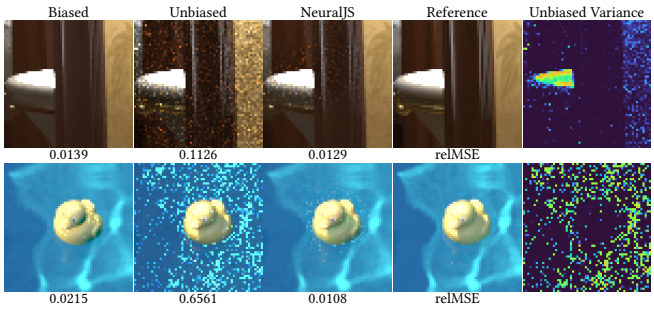

Fig. 2. Influence of the assumption of homogeneous variance in NeuralJS [Gu et al. 2022]. Note that an averaged block-wise shrinkage factor is insufficient to handle the regions with spatially varying variance.

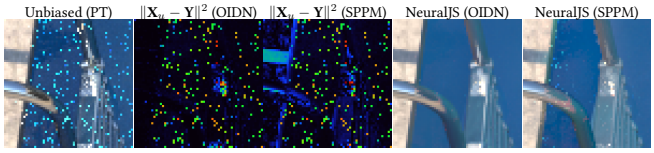

Fig. 3. When $\mathbf{X}_u$ and $\mathbf{X}_b$ are generated by different MC integrators, NeuralJS [Gu et al. 2022] performs suboptimally, since the squared residual between $\mathbf{X}_u$ and optimized biased $\mathbf{Y}$ (i.e., $\|\mathbf{X}_u - \mathbf{Y}\|^2$) remains high within the region (e.g., shadow and pool edge) where the two integrators behave differently. This decreases the shrinkage factor towards $\mathbf{Y}$.

Moreover, when the biased rendering $\mathbf{X}_b$ is generated by an MC integrator (e.g., stochastic progressive photon mapping [Hachisuka and Jensen 2009] (SPPM)) that is significantly different from the MC integrator generating $\mathbf{X}_u$, the high MSE and spatial variation of $\mathbf{X}_b$ can make NeuralJS [Gu et al. 2022] suboptimal, as illustrated in Fig. 3. As seen, NeuralJS favors denoised renderings (e.g., results from open image denoise (OIDN) [Áfra 2025]) serving as $\mathbf{X}_b$.

The above observations inspire us to use pixel-wise rather than block-wise combinations like NeuralJS, for better control over the details. Unfortunately, simply adopting pixel-wise combination in the single shrinkage framework fails to remove residual noises, since it can only reduce the MSE by half compared to its inputs. To address this issue, we propose a double shrinkage method (DSCombiner) that provides flexibility in selecting the most appropriate integrators for various rendering scenarios.

4 DSCOMBINER

We propose our double shrinkage estimator *DSCombiner* that combines a pair of unbiased and biased estimators generated respectively by two different MC integrators within the screen space. For brevity, we simplify the color estimate to a scalar in the following derivation. The entire pipeline of our method and its comparison with NeuralJS is shown in Fig. 4.

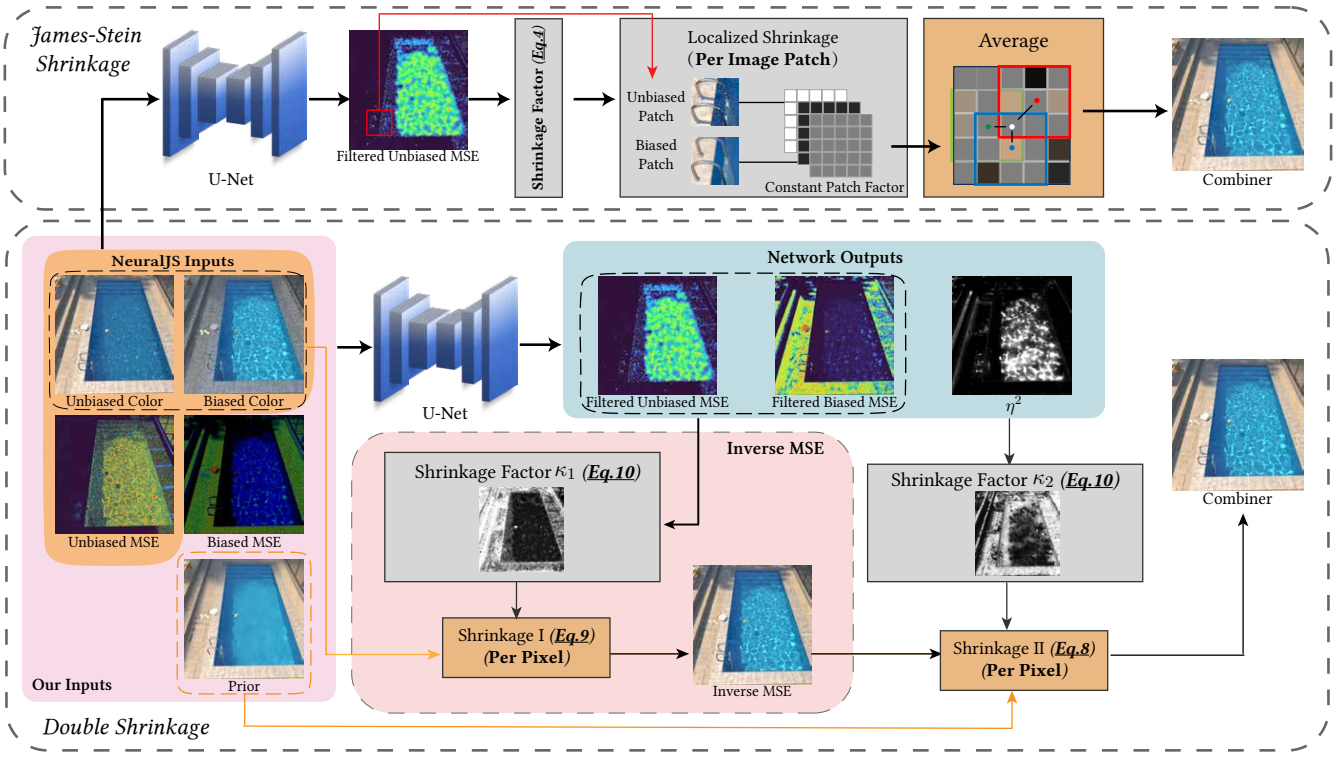

Fig. 4. An overview of our pipeline, we feed the unbiased color, the MSE (i.e., variance) estimate of unbiased color, biased color, the MSE estimate of biased color, and radiance prior to the network. The network infers a scalar η^2 and two kernels used to filter the pair of MSE estimates. We also show the difference between the previous single shrinkage estimator (James-Stein shrinkage) and our double shrinkage estimator for MC renderings.

4.1 Theoretically Optimal Double Shrinkage

Given two estimators X_u and X_b, our goal is to estimate the true value X. We can construct an optimal estimator for X by maximizing the posterior probability in a Bayesian framework, i.e.,

$$\begin{aligned}\hat{X} &= \underset{X}{\operatorname{argmax}}\, p(X|X_u, X_b) \\ &= \underset{X}{\operatorname{argmax}}\, \frac{p(X_u, X_b|X)p(X)}{p(X_u, X_b)} \\ &= \underset{X}{\operatorname{argmax}}\, p(X_u|X)p(X_b|X)p(X).\end{aligned} \tag{6}$$

In this equation, the prior distribution $p(X)$ of X is the key to removing the noise in $\hat{X}$. We model the prior distribution of X and ξ as two normal distributions $\mathcal{N}(X_p, \eta^2)$ and $\mathcal{N}(0, \gamma^2)$, respectively. Given the joint normal prior $p(X, \xi)$ and the joint normal likelihood $p(X_u, X_b|X, \xi)$, the conjugate prior property guarantees that the posterior $p(X, \xi|X_u, X_b)$ is also a joint normal distribution, indicating that the marginal probability distribution $p(X|X_u, X_b)$ is a normal distribution. Then, the marginal distribution of $p(X_b|X)$ can be derived as

$$\begin{aligned}p(X_b|X) &= \int_{-\infty}^{\infty} p(X_b|X, \xi)p(\xi)\mathrm{d}\xi \\ &= \mathcal{N}(X_p, \sigma_b^2 + \gamma^2).\end{aligned} \tag{7}$$

Consequently, the MAP (maximum a posteriori) estimator in Eq. (6) can be solved by setting the derivative of $p(X_u|X)p(X_b|X)p(X)$ to zero. Since the product itself is an exponential function, this is equivalent to finding $\hat{X}$ that makes the exponential component zero:

$$\frac{\hat{X} - X_u}{\sigma_u^2} + \frac{\hat{X} - X_b}{\sigma_b^2 + \gamma^2} + \frac{\hat{X} - X_p}{\eta^2} = 0. \tag{8}$$

Reorganize the above equation, we can write $\hat{X}$ as

$$\hat{X} = \kappa_2 X_p + (1 - \kappa_2)\left(\kappa_1 X_b + (1 - \kappa_1)X_u\right), \tag{9}$$

where

$$\begin{aligned}\kappa_1 &= \frac{\sigma_u^2}{\gamma^2 + \sigma_b^2 + \sigma_u^2}, \\ \kappa_2 &= \frac{\sigma_u^2(\gamma^2 + \sigma_b^2)}{\sigma_u^2(\gamma^2 + \sigma_b^2) + \eta^2(\gamma^2 + \sigma_b^2 + \sigma_u^2)}.\end{aligned} \tag{10}$$

We refer to our combiner (i.e., Eq. (9)) as the *double shrinkage estimator* because it performs shrinkage twice. Specifically, the first shrinkage stage employs an *inverse-MSE*-like mechanism to combine the estimates X_u and X_b, which efficiently mitigates fireflies but may still suffer from low-frequency noise. Then, the second shrinkage shrinks the first one's result towards a prior X_p with a shrinkage factor controlled by η^2. By introducing a smooth prior

X_p, our combiner overcomes the drawback of single shrinkage: its inability to remove noticeable noise when both X_u and X_b are noisy. Simultaneously, the shrinkage factor η^2 controls the proportion of X_p in the final output, which could prevent excessive smoothing.

4.2 Radiance Prior

As we discussed earlier, the prior distribution of X is the key to addressing the limitations of single shrinkage. It has two parameters to be determined, the mean X_p and the variance η^2, where X_p provides the smoothness and η^2 will be discussed later in Sec. 4.3. We emphasize that the design of X_p is different compared with traditional MC denoising. In particular, what we need is more like an auxiliary color buffer rather than a high-quality denoised result. This enables a design of X_p that emphasizes smoothness while relaxing constraints on detail retention.

Here we present our approach to derive the radiance prior X_p. Let X_u^i, X_b^i and $\mathbf{f_i}$ be the unbiased color, biased color and feature vector (i.e., G-Buffers) of pixel i, we approximate the radiance prior with a first-order linear model within a local block Ω_i (with size 51×51) centered on pixel i. This implies a regression:

$$[\alpha_i, \beta_i] = \underset{\alpha_i,\beta_i}{\operatorname{argmin}} \sum_{j\in\Omega_i} w_u^{i,j}\mathcal{L}(X_u^j, \alpha_i + \beta_i(\mathbf{f}_j - \mathbf{f}_i)) + w_b^{i,j}\mathcal{L}(X_b^j, \alpha_i + \beta_i(\mathbf{f}_j - \mathbf{f}_i)), \tag{11}$$

where $\mathcal{L}$ is L2 norm, α_i and β_i are the parameters of a first-order linear function.. $w_u^{i,j}$ and $w_b^{i,j}$ are the weights:

$$w_u^{i,j} = \exp\left(-\frac{\left|X_b^i - X_b^j\right|^2}{\sigma_{b,i}^2 + \sigma_{b,j}^2 + \epsilon_1}\right), w_b^{i,j} = \exp\left(-\frac{\left|X_u^i - X_u^j\right|^2}{\sigma_{u,i}^2 + \sigma_{u,j}^2 + \epsilon_1}\right). \tag{12}$$

The regression of Eq. (11) can be solved through the least squares, which implies

$$\underset{\mathbf{X}}{\operatorname{argmin}} \|\mathbf{A_u X} - \mathbf{Y_u}\|^2 + \|\mathbf{A_b X} - \mathbf{Y_b}\|^2, \tag{13}$$

where each matrix is

$$\mathbf{A}_* = \begin{bmatrix} \sqrt{w_*^{i,1}}\,[1, \mathbf{f}_1 - \mathbf{f}_i] \\ \vdots \\ \sqrt{w_*^{i,n}}\,[1, \mathbf{f}_n - \mathbf{f}_i] \end{bmatrix}, \quad \mathbf{X} = \left[\alpha_i, \beta_i\right]^\top, \quad \mathbf{Y}_* = \begin{bmatrix} \sqrt{w_*^{i,1}}X_*^1 \\ \vdots \\ \sqrt{w_*^{i,n}}X_*^n \end{bmatrix} \tag{14}$$

and n is the number of pixels within Ω_i, $*$ represents u or b.

We can rewrite the residual error as a function of $\mathbf{X}$

$$E(\mathbf{X}) = (\mathbf{A_u X} - \mathbf{Y_u})^\top(\mathbf{A_u X} - \mathbf{Y_u}) + (\mathbf{A_b X} - \mathbf{Y_b})^\top(\mathbf{A_b X} - \mathbf{Y_b}) \tag{15}$$

To minimize $E(\mathbf{X})$, we set the derivative of E with respect to $\mathbf{X}$ equal to zero, that is equivalent to

$$\begin{aligned} 2\mathbf{A_u}^\top\mathbf{A_u X} - 2\mathbf{A_u}^\top\mathbf{Y_u} + 2\mathbf{A_b}^\top\mathbf{A_b X} - 2\mathbf{A_b}^\top\mathbf{Y_b} &= 0 \\ (\mathbf{A_u}^\top\mathbf{A_u} + \mathbf{A_b}^\top\mathbf{A_b})\mathbf{X} &= \mathbf{A_u}^\top\mathbf{Y_u} + \mathbf{A_b}^\top\mathbf{Y_b} \\ \mathbf{X} &= (\mathbf{A_u}^\top\mathbf{A_u} + \mathbf{A_b}^\top\mathbf{A_b})^{-1}(\mathbf{A_u}^\top\mathbf{Y_u} + \mathbf{A_b}^\top\mathbf{Y_b}) \end{aligned} \tag{16}$$

Thus, the inverse matrix of $\mathbf{A_u}^\top\mathbf{A_u} + \mathbf{A_b}^\top\mathbf{A_b}$ is the only thing left unknown. We use Cholesky decomposition to solve a lower triangular matrix $\mathbf{L}$ that satisfies $\mathbf{LL}^\top = \mathbf{A_u}^\top\mathbf{A_u} + \mathbf{A_b}^\top\mathbf{A_b}$. We add an epsilon to each element on the diagonal of the matrix to guarantee its positive definiteness. Then the inverse matrix can be easily obtained through $\mathbf{L}$.

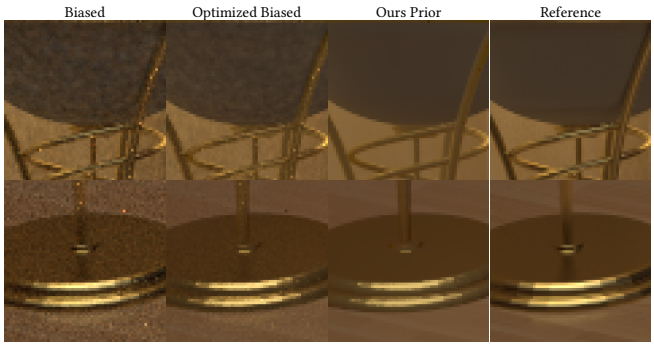

Fig. 5. Comparison between optimized biased rendering in [Gu et al. 2022] with our radiance prior. Our radiance prior effectively eliminates the artifacts introduced by biased rendering methods, such as SPPM, whose results are far from converged.

After solving the above regression, we reconstruct the radiance prior through

$$\begin{aligned} X_p^i &= \sum_{j\in\Omega_i} w_{i,j}(\alpha_j + \beta_j(\mathbf{f}_j - \mathbf{f}_i)), \\ w_{i,j} &= \exp\left(-\frac{\left|X_b^i - X_b^j\right|^2 - (\sigma_{b,i}^2 + \sigma_{b,j}^2)}{\sigma_{b,i}^2 + \sigma_{b,j}^2 + \epsilon_1}\right). \end{aligned} \tag{17}$$

We set $\epsilon_1 = 0.01$ to avoid dividing by zero. It has been shown that our regression produces a smoother color buffer compared to the optimized biased rendering proposed by Gu et al. [2022], as shown in Fig. 5.

By design, X_p is oversmoothed, resulting in detail loss. Hence, a second shrinkage operation is employed to restore missing details while minimizing the resurgence of noise. This is achieved by assigning a confidence level to X_p, denoted as η^2 in our framework. A smaller η^2 indicates a higher confidence of X_p as the designed true value, effectively serving as a regularization term that balances the trade-off between detail preservation and noise suppression.

4.3 Data-driven Shrinkage Factors

To combine the radiance prior buffer with unbiased and biased renderings, we need to determine η^2 for each pixel. Note that η^2 is highly correlated with the regressed X_p. Therefore, rather than analyzing the variance reduction and bias introduction caused by regression, we adopt a Convolutional Neural Network (CNN) to predict η^2. We visualize the predicted η^2 in Fig. 6.

Network architecture. As illustrated in Fig. 7, we employ a U-Net as the backbone, with each layer comprising a 3×3 convolution and a ReLU activation function, followed by a simple projection layer (i.e., 1×1 convolutional layer). The network takes as input the radiance prior buffer and the pair of unbiased and biased renderings, along with their MSE estimates (averaged over RGB channels). For the

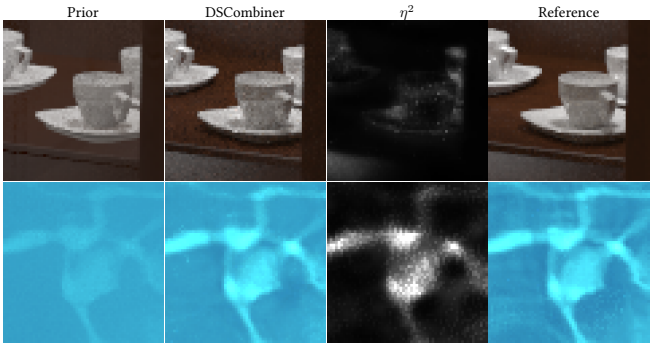

Fig. 6. Results of predicted η^2. Our specially designed neural network identifies regions that are not well handled by the radiance prior (typically indicating an overblurred radiance prior in these areas). It assigns a higher η^2 to these pixels and recovers details based on the original unbiased and biased inputs.

output, our model predicts 339 parameters per pixel: two 13×13 filter kernels (normalized by Softmax) for filtering the biased/unbiased MSE, and the exponential of a predicted scalar parameter as η^2 to ensure positivity. Finally, we use the predicted η^2 and the pair of filtered MSE estimates to compute the shrinkage factors (Eq. 10).

Loss functions. We employ supervised learning to optimize our model, using SMAPE [Vogels et al. 2018] as our loss function:

$$\mathcal{L}(\mathbf{D},\mathbf{R}) = \frac{1}{3N}\sum_i\sum_c \frac{|\mathbf{D}_i^c - \mathbf{R}_i^c|}{|\mathbf{D}_i^c| + |\mathbf{R}_i^c| + \epsilon_2}, \tag{18}$$

where $\mathbf{D}$ is the result image of our *DSCombiner* and $\mathbf{R}$ is the reference image. Here, N is the total number of pixels, i and c represent each pixel and each color channel, respectively. In order to avoid division by zero, we set ϵ_2 to 0.001.

Data generation. We generate a data set containing 2548 pairs of images across 20 different scenes (a selection of which are shown in Fig. 8), each with a resolution of 512×512 rendered by the Mitsuba renderer [Jakob 2010]. In our experiments, unbiased images are rendered by path tracing (PT), and biased images are rendered by SPPM. We use bidirectional path tracing (BDPT) to generate the reference image at high sample rates. The total rendering time for a pair of unbiased and biased images varies from 20 seconds to 10 minutes.

Training. We reserve 10% of the data for validation and use the remaining for training. The model was trained for 2,000 epochs using the Adam optimizer [Kingma and Ba 2017] with a learning rate of 10^{-4} and a batch size of 10. At each iteration, we randomly crop a 128×128 patch from the input images. Consequently, it took about a day for training. The entire framework is implemented in PyTorch [Paszke et al. 2019].

4.4 Variance and Bias Estimation

As aforementioned, we need to estimate the MSE for the pair of biased and unbiased renderings. For an unbiased integrator, the variance σ_u^2 of pixel's intensity X_u can be estimated through sample variance. For biased ones, several works [Hachisuka et al. 2010; Nabata et al. 2016; Schwarzhaupt et al. 2012] have developed error estimation frameworks for some specific integrators, thereby allowing us to obtain the bias and variance for each pixel directly. In particular, since we employ SPPM in our experiments, we use the method in [Hachisuka et al. 2010] for the MSE estimation of biased renderings. Furthermore, we follow the procedure detailed in related works [Gu et al. 2022], which involves filtering the error estimates to refine our MSE calculations.

5 RESULTS AND DISCUSSIONS

We perform experiments on a PC with an Intel i7-13700K CPU and an NVIDIA RTX 3080 Ti GPU. We use Mitsuba renderer [Jakob 2010] to generate all images with a default resolution of 1280×720. We didn't include the execution time of post-processing in our experiments for all methods since we found it is relatively small (less than 5%) compared to the rendering time.

Visualization of the relative weight. We show the relative weight of the radiance prior and inverse-MSE combiner in Fig. 9. In this visualization, red represents a higher proportion of the radiance prior, while blue shows the opposite. The experimental results demonstrate that our method effectively preserves details in the inverse-MSE combiner while successfully suppressing residual noise.

Comparisons with screen-space combination. We compare our method with DeepCombiner [Back et al. 2020] and NeuralJS [Gu et al. 2022], using their released pre-trained models in Fig. 10. We allocate approximately equal runtime to each integrator for generating input images. We analyze the numerical convergence of each method, with rendering time varying from 20 seconds to 2,560 seconds. Our experiments are conducted on four scenes with challenging lighting conditions, each containing distinct areas where different integrators could demonstrate their strengths. As seen, DeepCombiner efficiently mitigates the correlated low-frequency noise (e.g., in Bookshelf) but struggles to retain the details (e.g., caustics in Swimming Pool). While NeuralJS succeeds in preserving high-frequency details, it tends to reintroduce noticeable noise in regions with wildly varied variance. Our DSCombiner surpasses both methods in terms of numerical metrics and visual quality. Further, we also retrained NeuralJS [Gu et al. 2022] using our dataset for a fairer comparison, which takes about a day for training. We find the re-trained NeuralJS behaves sub-optimally compared to the model released by Gu et al. [2022] in low SPP settings. This could be attributed to the stronger noise present in our dataset's images (especially the biased rendering), which hinders the model's convergence during training. As the sampling rate increased, the performance of the re-trained model became essentially consistent with that of the pre-trained model. This demonstrates that our dataset is not the key factor contributing to DSCombiner's improved performance.

Comparisons with path-space combination. We compare DSCombiner (BDPT+SPPM) with the prevailing path-space combination method: vertex connection and merging (VCM) [Georgiev et al. 2012] in Fig. 11. Here, we use the VCM implementation released by [Sun et al. 2017]. Experimental results demonstrate that our method not only provides a more flexible combination framework but also

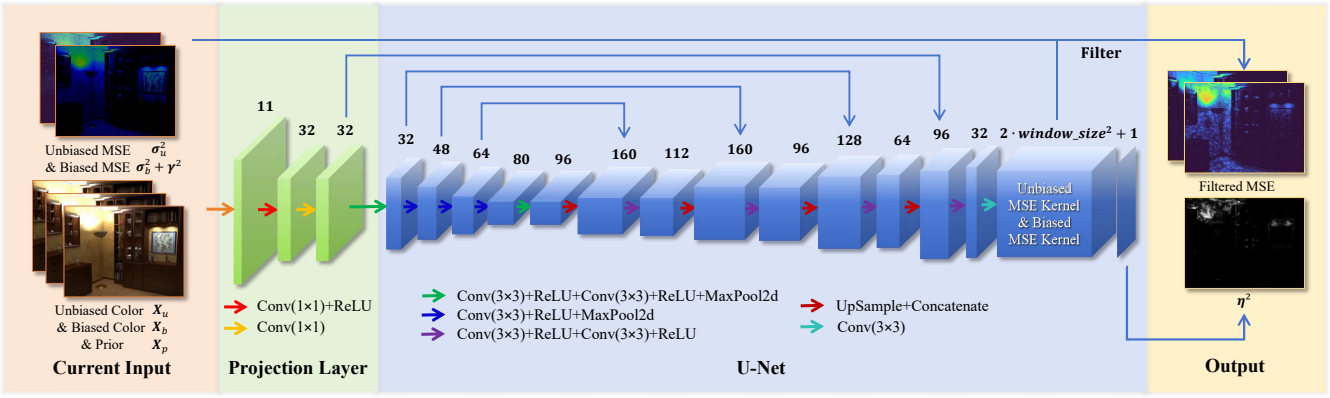

Fig. 7. The architecture of our neural network. We use a U-Net with a projection layer [Thomas et al. 2022] to predict the parameters. The predicted parameters can be split into two parts: two filtering kernels for the pair of MSE estimates of biased and unbiased color, and a scalar serving as η^2. Then, we use the output to perform double shrinkage.

Fig. 8. Some example scenes in our dataset.

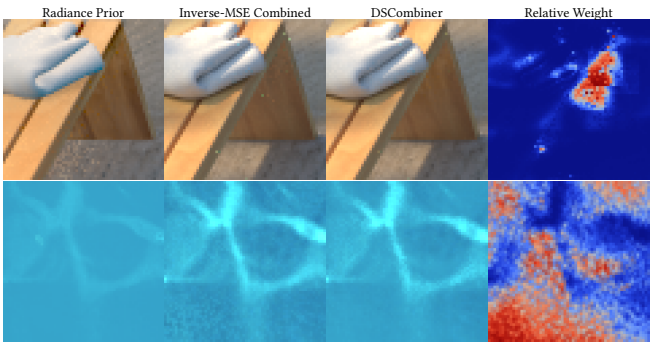

Fig. 9. Relative weight of radiance prior and inverse-MSE combiner in the final result, red represents a higher proportion of the radiance prior and blue shows the opposite.

significantly enhances numerical accuracy by avoiding expensive computations in path space.

Comparisons with NeuralJS with single integrator. Since NeuralJS is originally designed for an unbiased integrator with a denoiser, we compare our method with NeuralJS in a single integrator setting. We select OIDN [Áfra 2025] as the denoiser and provide the auxiliary G-Buffers to generate the biased renderings. Equal-time comparisons in Fig. 12 show that although our method achieves only marginal advantages in numerical metrics, we can better capture the details that are challenging for the denoiser to reconstruct (e.g., caustics in Torus and shadows in Spotlight). Moreover, NeuralJS still struggles in regions where the homogeneous variance assumption does not hold (e.g., line-shaped highlight in Torus), while our approach achieves better noise reduction in these areas.

Comparisons between different biased renderings. We also use OIDN [Áfra 2025] to generate biased renderings for DSCombiner. We use dual-buffer [Bitterli et al. 2016] to estimate the MSE of biased renderings (the time of generating the additional buffer is not included). The comparisons are shown in Fig. 13, which shows that our DSCombiner is more effective in handling biased renderings generated by SPPM. Since DSCombiner primarily acquires smoothness from the radiance prior, we hope that biased rendering can provide some details that unbiased rendering cannot produce, so we suggest using an integrator rather than a denoiser to generate biased renderings.

Ablation studies. To analyze the improvement brought by each of the two designs (i.e., radiance prior and DSCombiner) separately, in Fig. 14, we show the improvement by replacing the optimized bias rendering used by Gu et al. [2022] with our radiance prior. For these two distinct types of priors, we train separate models, ensuring identical training configurations for both. The results validate that our radiance prior significantly reduces the noise pattern introduced by biased rendering (e.g., the correlated noise on the white tissue in Still Life and low-frequency noise on radio in Kitchen). Additionally, in Fig. 15, we compare NeuralJS with our method using optimized biased rendering as the radiance prior. Here, the pixel-level shrinkage control makes our combiner more robust in recovering details, reintroducing considerably less noise compared to NeuralJS.

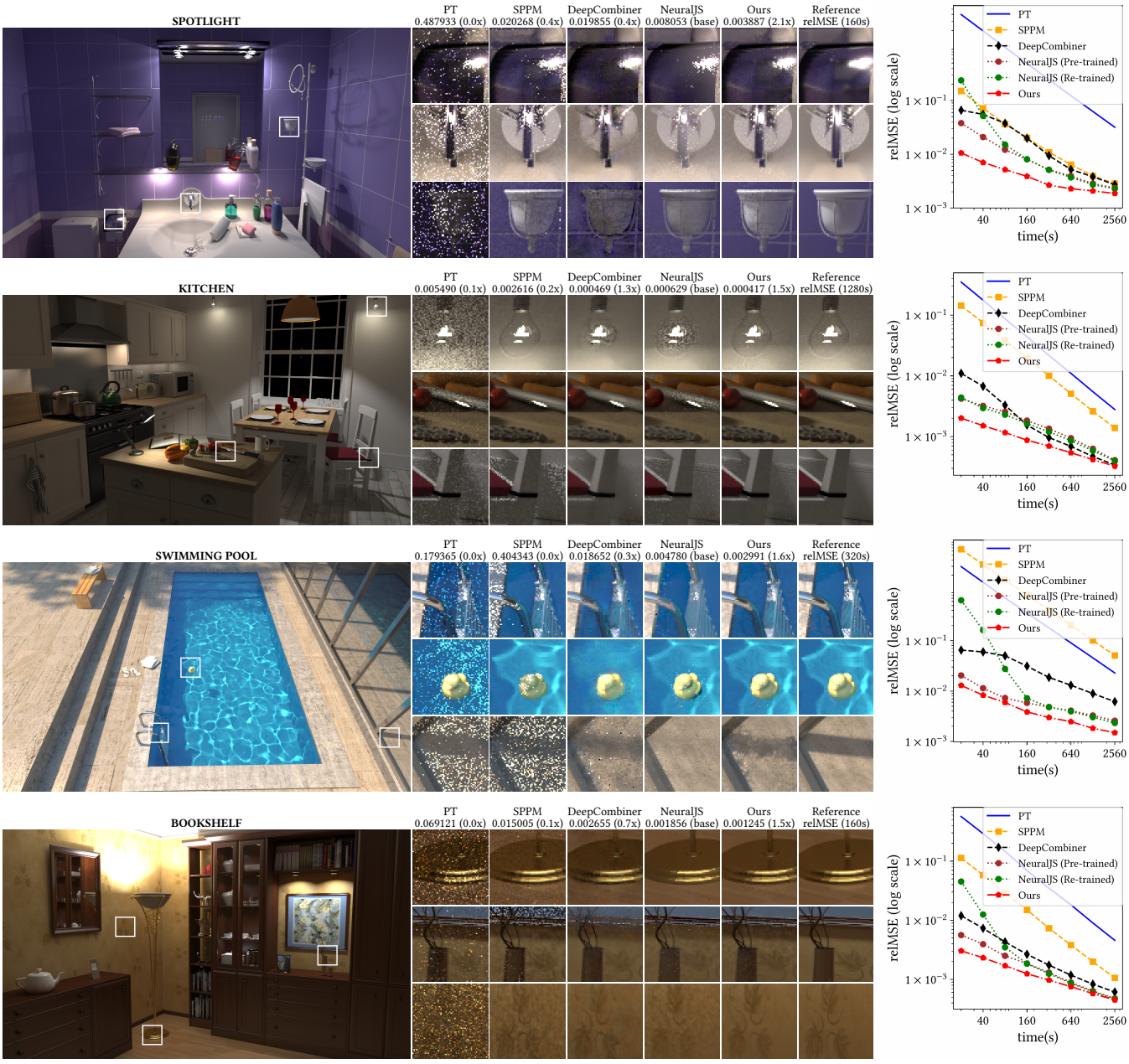

Fig. 10. Equal-time comparisons between our DSCombiner with DeepCombiner [Back et al. 2020] and NeuralJS [Gu et al. 2022]. The unbiased and biased inputs are rendered by PT and SPPM, respectively, with roughly the same time budget. Convergences of re-trained NeuralJS indicate that our dataset provides slight improvements at high sampling rates; however, it's not the key factor why DSCombiner performs better.

6 DISCUSSIONS AND LIMITATIONS

Reintroduced noise. Our method recovers the details by combining the smooth radiance prior with unbiased and biased rendering inputs, which may reintroduce some noise as shown in Fig. 16. Specifically, the reintroduced noise can be either high-frequency or low-frequency. To alleviate such issues, one can enhance the quality by applying a multi-level combination [Balint et al. 2023; Boughida and Boubekeur 2017] for reducing low-frequency noise and a dual-buffer combination [Bitterli et al. 2016; Gu et al. 2022] for reducing high-frequency noise.

Budget allocation between integrators. In our current implementation, we set an equal time budget for two integrators in our experiments. Nonetheless, one can allocate different budgets between

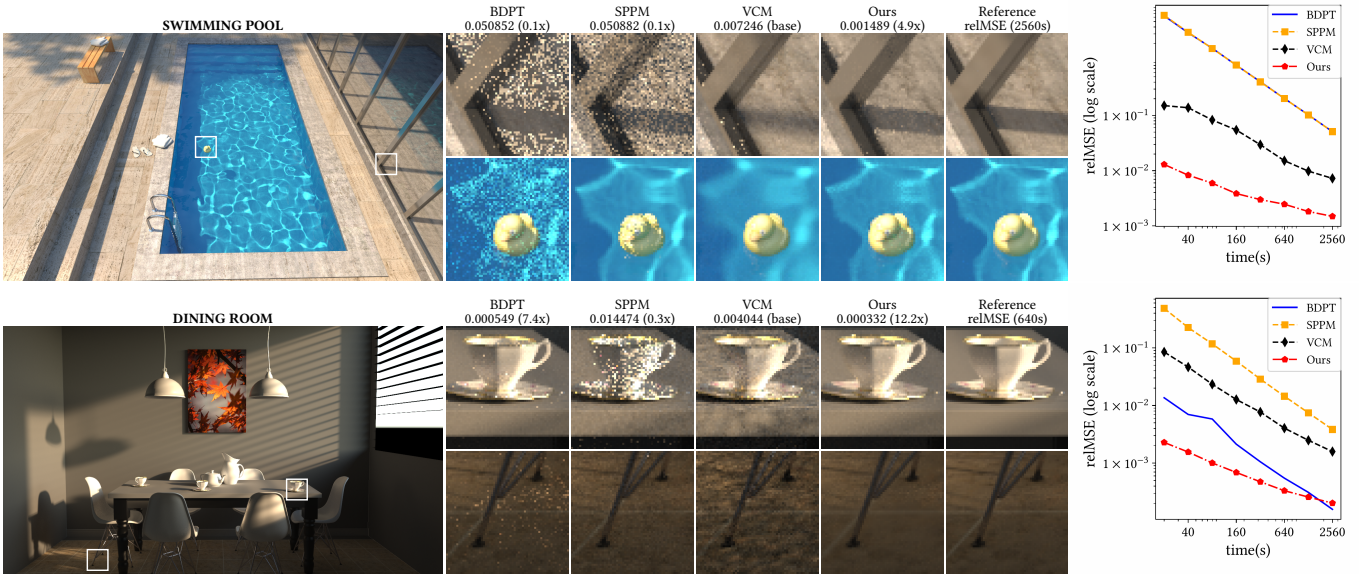

Fig. 11. Comparisons between path-space combination (i.e., VCM [Georgiev et al. 2012]) and our DSCombiner (BDPT+SPPM).

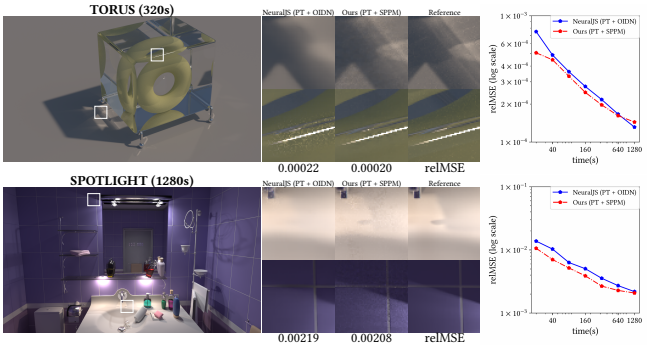

Fig. 12. Equal-time comparisons between NeuralJS (PT+OIDN) and ours (PT+SPPM). For the regions that can't be well handled by unbiased rendering (i.e., caustics and shadows), DSCombiner still outperforms NeuralJS equipped with a powerful denoiser (OIDN).

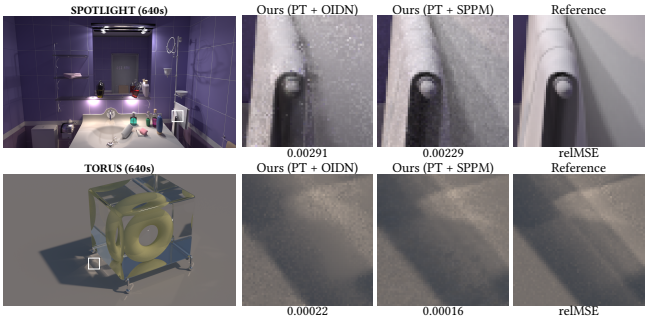

Fig. 13. Equal-time comparisons between using different biased renderings (i.e., OIDN and SPPM) in our DSCombiner. The result shows that a biased integrator can provide more details than a denoiser in some specific regions.

integrators or even pixels. For example, adaptive sample allocation could further enhance our method's performance by assigning more samples to each integrator in its regions of expertise, thereby avoiding computational waste on ineffective sampling.

Prior distribution. We select the normal distribution as the prior, and other distributions [Salehi et al. 2022] can also be considered. Besides, the radiance prior can be obtained through alternative ways. For instance, without requiring reference data, an unsupervised learning approach can be employed to train a neural network for predicting radiance prior. We believe that carefully designing self-supervised loss and enabling the neural network to predict higher-quality radiance prior is a good avenue for further exploration.

7 CONCLUSION

In this paper, we have presented a new combination framework for combining biased and unbiased renderings generated by two different integrators. We formulate the combination task as a Bayesian inference problem and design a simple yet effective radiance prior. The shrinkage factors are inferred using a data-driven approach, leveraging a neural network to evaluate the confidence of the radiance prior. This confidence evaluation controls the proportion of the radiance prior in the final combination results. Experiments show that our method achieves superior error reduction in our test scenes compared to existing methods.

We believe the double shrinkage estimator advances the state of the art in Monte Carlo rendering by providing a more efficient and effective way to balance the trade-off between bias and variance, ultimately leading to higher quality renderings in less time.

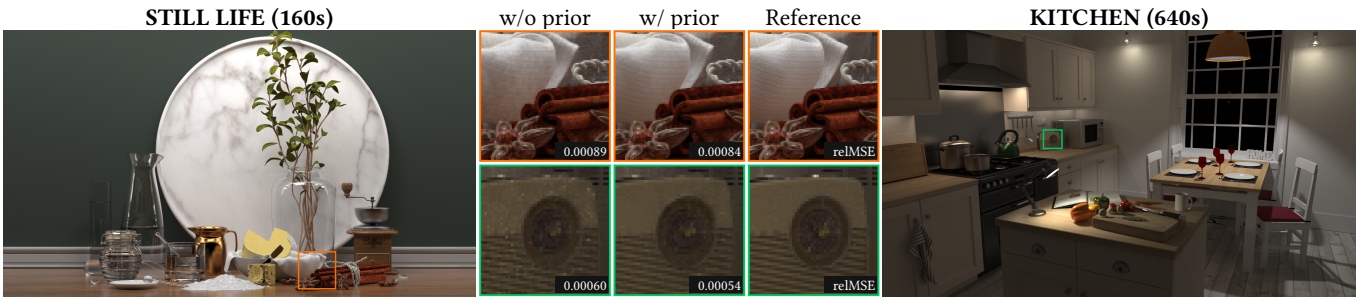

Fig. 14. Ablation studies using different radiance priors (i.e., optimized biased rendering proposed by Gu et al. [2022] and ours) in our pipeline. The unbiased and biased inputs are rendered by PT and SPPM, respectively. Ours radiance prior eliminates the noise pattern introduced by biased rendering.

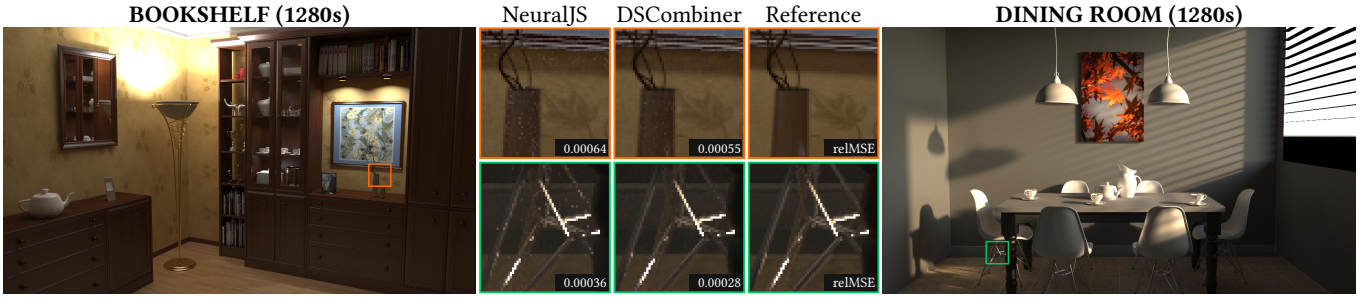

Fig. 15. Ablation studies using different combiners (i.e., NeuralJS and our proposed DSCombiner). We replace our radiance prior with optimized biased rendering used by NeuralJS. The unbiased and biased inputs are rendered by PT and SPPM, respectively. Results show that DSCombiner itself still outperforms NeuralJS.

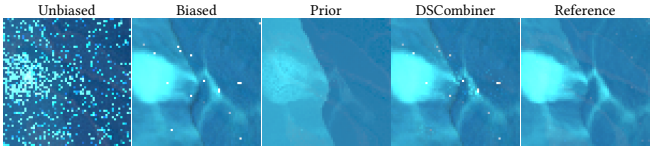

Fig. 16. Examples of our method's limitation. The result shows that our combiner may reintroduce the noise (both low-frequency noise and fireflies).

Acknowledgments

We would like to thank the anonymous reviewers for their valuable suggestions. This work was supported by the National Natural Science Foundation of China (No. 61972194 and No. 62032011) and the Natural Science Foundation of Jiangsu Province (No. BK20211147).

References

Attila T. Áfra. 2025. Intel® Open Image Denoise. https://www.openimagedenoise.org.

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Trans. Graph.* 39, 6, Article 242 (Nov. 2020), 12 pages. https://doi.org/10.1145/3414685.3417847

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 Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073708

Martin Balint, Krzysztof Wolski, Karol Myszkowski, Hans-Peter Seidel, and Rafał Mantiuk. 2023. Neural Partitioning Pyramids for Denoising Monte Carlo Renderings. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 60, 11 pages. https://doi.org/10.1145/3588432.3591562

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Comput. Graph. Forum* 35, 4 (July 2016), 107–117.

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Trans. Graph.* 39, 4, Article 148 (Aug. 2020), 17 pages. https://doi.org/10.1145/3386569.3392481

Malik Boughida and Tamy Boubekeur. 2017. Bayesian Collaborative Denoising for Monte Carlo Rendering. *Computer Graphics Forum* 36, 4 (2017), 137–153. https://doi.org/10.1111/cgf.13231 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13231

Chris Cascioli and Laura Reznikov. 2024. Introduction to Real-Time Ray Tracing. In *ACM SIGGRAPH 2024 Courses* (Denver, CO, USA) *(SIGGRAPH Courses '24)*. Association for Computing Machinery, New York, NY, USA, Article 15, 25 pages.

Hajin Choi, Seokpyo Hong, Inwoo Ha, Nahyup Kang, and Bochang Moon. 2024. Online Neural Denoising with Cross-Regression for Interactive Rendering. *ACM Trans. Graph.* 43, 6, Article 221 (Nov. 2024), 12 pages. https://doi.org/10.1145/3687938

Per H. Christensen and Wojciech Jarosz. 2016. The Path to Path-Traced Movies. *Foundations and Trends in Computer Graphics and Vision* 10, 2 (Oct. 2016), 103–175. https://doi.org/10/gfjwjc

Zhimin Fan, Pengpei Hong, Jie Guo, Changqing Zou, Yanwen Guo, and Ling-Qi Yan. 2023. Manifold Path Guiding for Importance Sampling Specular Chains. *ACM Trans. Graph.* 42, 6, Article 257 (Dec. 2023), 14 pages. https://doi.org/10.1145/3618360

Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2022. Progressive Denoising of Monte Carlo Rendered Images. *Computer Graphics Forum* 41, 2 (2022), 1–11. https://doi.org/10.1111/cgf.14454 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14454

Arthur Firmino, Ravi Ramamoorthi, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2024. Practical Error Estimation for Denoised Monte Carlo Image Synthesis. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 93, 10 pages.

Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. 2012. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.* 31, 6, Article 192 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366211

Edwin J. Green and William E. Strawderman. 1991. A James-Stein Type Estimator for Combining Unbiased and Possibly Biased Estimators. *J. Amer. Statist. Assoc.* 86, 416 (1991), 1001–1006. http://www.jstor.org/stable/2290517

Jeongmin Gu, Jose A. Iglesias-Guitian, and Bochang Moon. 2022. Neural James-Stein Combiner for Unbiased and Biased Renderings. *ACM Trans. Graph.* 41, 6, Article 262 (Nov. 2022), 14 pages. https://doi.org/10.1145/3550454.3555496

Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. 2010. A progressive error estimation framework for photon density estimation. *ACM Trans. Graph.* 29, 6, Article 144 (Dec. 2010), 12 pages. https://doi.org/10.1145/1882261.1866170

Toshiya Hachisuka and Henrik Wann Jensen. 2009. Stochastic progressive photon mapping. In *ACM SIGGRAPH Asia 2009 Papers* (Yokohama, Japan) *(SIGGRAPH Asia '09)*. Association for Computing Machinery, New York, NY, USA, Article 141, 8 pages. https://doi.org/10.1145/1661412.1618487

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive photon mapping. *ACM Trans. Graph.* 27, 5, Article 130 (Dec. 2008), 8 pages. https://doi.org/10.1145/1409060.1409083

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. 2012. A path space extension for robust light transport simulation. *ACM Trans. Graph.* 31, 6, Article 191 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366210

Wenzel Jakob. 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

W. James and Charles Stein. 1961. Estimation with Quadratic Loss. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. University of California Press, Berkeley, Calif., 361–379.

Henrik Wann Jensen. 2001. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., USA.

James T. Kajiya. 1986. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '86)*. Association for Computing Machinery, New York, NY, USA, 143–150. https://doi.org/10.1145/15922.15902

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-domain path tracing. *ACM Trans. Graph.* 34, 4, Article 123 (July 2015), 13 pages. https://doi.org/10.1145/2766997

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980

Jaroslav Křivánek, Iliyan Georgiev, Toshiya Hachisuka, Petr Vévoda, Martin Šik, Derek Nowrouzezahrai, and Wojciech Jarosz. 2014. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.* 33, 4, Article 103 (July 2014), 13 pages. https://doi.org/10.1145/2601097.2601219

F. Lavancier and P. Rochet. 2016. A general procedure to combine estimators. *Comput. Stat. Data Anal.* 94, C (Feb. 2016), 175–192. https://doi.org/10.1016/j.csda.2015.08.001

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph.* 31, 6, Article 194 (Nov. 2012), 9 pages. https://doi.org/10.1145/2366145.2366213

K. Nabata, K. Iwasaki, Y. Dobashi, and T. Nishita. 2016. An Error Estimation Framework for Many-Light Rendering. *Comput. Graph. Forum* 35, 7 (Oct. 2016), 431–439.

Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. 2018. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum (Proceedings of Eurographics - State of the Art Reports)* 37, 2 (May 2018). https://doi.org/10/gd2jqq

Hisanari Otsu, Shinichi Kinuwaki, and Toshiya Hachisuka. 2018. Supervised Learning of How to Blend Light Transport Simulations. In *Monte Carlo and Quasi-Monte Carlo Methods (MCQMC 2016)*. 409–427.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG] https://arxiv.org/abs/1912.01703

Adithya Pediredla, Yasin Karimi Chalmiani, Matteo Giuseppe Scopelliti, Maysamreza Chamanzar, Srinivasa Narasimhan, and Ioannis Gkioulekas. 2020. Path tracing estimators for refractive radiative transfer. *ACM Trans. Graph.* 39, 6, Article 241 (Nov. 2020), 15 pages. https://doi.org/10.1145/3414685.3417793

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically Based Rendering: From Theory to Implementation (4th ed.)* (4th ed.). The MIT Press.

Pengju Qiao, Qi Wang, Yuchi Huo, Shiji Zhai, Zixuan Xie, Wei Hua, Hujun Bao, and Tao Liu. 2024. Neural Kernel Regression for Consistent Monte Carlo Denoising. *ACM Trans. Graph.* 43, 6, Article 222 (Nov. 2024), 14 pages. https://doi.org/10.1145/3687949

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2011. Adaptive sampling and reconstruction using greedy error minimization. *ACM Trans. Graph.* 30, 6 (Dec. 2011), 1–12. https://doi.org/10.1145/2070781.2024193

Hiroyuki Sakai, Christian Freude, Thomas Auzinger, David Hahn, and Michael Wimmer. 2024. A Statistical Approach to Monte Carlo Denoising. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 68, 11 pages. https://doi.org/10.1145/3680528.3687591

Farnood Salehi, Marco Manzi, Gerhard Roethlin, Romann Weber, Christopher Schroers, and Marios Papas. 2022. Deep Adaptive Sampling and Reconstruction Using Analytic Distributions. *ACM Trans. Graph.* 41, 6, Article 259 (Nov. 2022), 16 pages. https://doi.org/10.1145/3550454.3555515

Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. 2012. Practical Hessian-based error control for irradiance caching. *ACM Trans. Graph.* 31, 6, Article 193 (Nov. 2012), 10 pages. https://doi.org/10.1145/2366145.2366212

Pradeep Sen and Soheil Darabi. 2012. On filtering the noise from the random parameters in Monte Carlo rendering. *ACM Trans. Graph.* 31, 3, Article 18 (May 2012), 15 pages. https://doi.org/10.1145/2167076.2167083

Weilun Sun, Xin Sun, Nathan A. Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-Domain Vertex Connection and Merging. In *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*, Matthias Zwicker and Pedro Sander (Eds.). The Eurographics Association. https://doi.org/10.2312/sre.20171197

Manu Mathew Thomas, Gabor Liktor, Christoph Peters, Sungye Kim, Karthik Vaidyanathan, and Angus G. Forbes. 2022. Temporally Stable Real-Time Joint Neural Denoising and Supersampling. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 21 (July 2022), 22 pages. https://doi.org/10.1145/3543870

Eric Veach and Leonidas J. Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 419–428. https://doi.org/10.1145/218380.218498

Eric Veach and Leonidas J. Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 65–76. https://doi.org/10.1145/258734.258775

Thijs Vogels, Fabrice Rousselle, Brian Mcwilliams, Gerhard Röthlin, Alex Harvill, David Adler, Mark Meyer, and Jan Novák. 2018. Denoising with kernel prediction and asymmetric loss functions. *ACM Trans. Graph.* 37, 4, Article 124 (July 2018), 15 pages. https://doi.org/10.1145/3197517.3201388

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* (Los Angeles, California). ACM, New York, NY, USA. https://doi.org/10/knqq

Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. 2020. Specular manifold sampling for rendering high-frequency caustics and glints. *ACM Trans. Graph.* 39, 4, Article 149 (Aug. 2020), 15 pages. https://doi.org/10.1145/3386569.3392408

Shaokun Zheng, Fengshi Zheng, Kun Xu, and Ling-Qi Yan. 2021. Ensemble denoising for Monte Carlo renderings. *ACM Trans. Graph.* 40, 6, Article 274 (Dec. 2021), 17 pages. https://doi.org/10.1145/3478513.3480510