Ensemble Denoising for Monte Carlo Renderings

SHAOKUN ZHENG, BNRist, Department of CS&T, Tsinghua University, China
FENGSHI ZHENG, BNRist, Department of CS&T, Tsinghua University, China
KUN XU*, BNRist, Department of CS&T, Tsinghua University, China
LING-QI YAN, University of California, Santa Barbara, United States of America

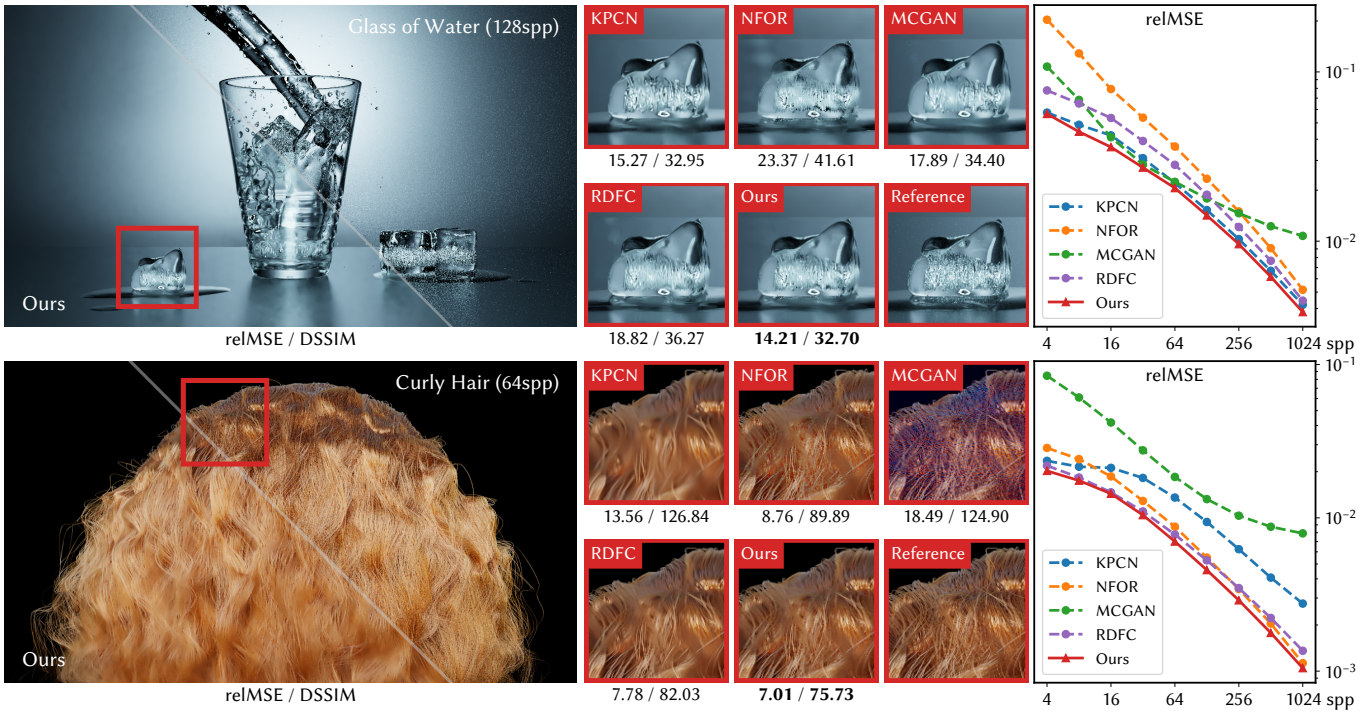

Fig. 1. We propose the ensemble denoising algorithm to combine multiple existing denoisers. Inheriting the excellence of the base denoisers and compensating for their artifacts, our method achieves lower error almost consistently, and robustly improves the perceptual quality across a great diversity of scenes and noise patterns. Left: noisy MC renderings and the denoised images generated by our method combining KPCN, NFOR, MCGAN and RDFC. Middle: close-up views with error numbers (scaled by 1000 for both metrics). Right: relMSE plots for base denoisers and the corresponding ensemble at varying sample rates.

Various denoising methods have been proposed to clean up the noise in Monte Carlo (MC) renderings, each having different advantages, disadvantages, and applicable scenarios. In this paper, we present *Ensemble Denoising*, an optimization-based technique that combines multiple individual MC denoisers. The combined image is modeled as a per-pixel weighted sum of output images from the individual denoisers. Computation of the optimal weights is formulated as a constrained quadratic programming problem, where we apply a dual-buffer strategy to estimate the overall MSE. We further propose an iterative solver to overcome practical issues involved in the optimization. Besides nice theoretical properties, our ensemble denoiser is demonstrated to be effective and robust, and outperforms any individual denoiser across dozens of scenes and different levels of sample rates. We also perform a comprehensive analysis on the selection of individual denoisers to be combined, providing important and practical guides for users.

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

Additional Key Words and Phrases: Monte Carlo, denoising, optimization

ACM Reference Format:
Shaokun Zheng, Fengshi Zheng, Kun Xu, and Ling-Qi Yan. 2021. Ensemble Denoising for Monte Carlo Renderings. *ACM Trans. Graph.* 40, 6, Article 1 (December 2021), 17 pages. https://doi.org/10.1145/3478513.3480510

*Kun Xu is the corresponding author.

Authors' addresses: Shaokun Zheng, BNRist, Department of CS&T, Tsinghua University, Beijing, China, zsk20@mails.tsinghua.edu.cn; Fengshi Zheng, BNRist, Department of CS&T, Tsinghua University, Beijing, China, zhengfs16@mails.tsinghua.edu.cn; Kun Xu, BNRist, Department of CS&T, Tsinghua University, Beijing, China, xukun@tsinghua.edu.cn; Ling-Qi Yan, University of California, Santa Barbara, Santa Barbara, United States of America, lingqi@cs.ucsb.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2021/12-ART1 $15.00
https://doi.org/10.1145/3478513.3480510

1 INTRODUCTION

Monte Carlo (MC) rendering has always been a gold standard in generating photorealistic images. Due to its stochastic nature, it is more and more popular nowadays to perform a denoising pass on an MC rendered image in the post-processing stage, to remove the noise and quickly get a clean result without using a lot of samples and waiting a long time for convergence.

Various MC denoising methods have been developed in recent years. Each of them may have different advantages, disadvantages, and applicable scenarios. For example, traditional filtering- and regression-based denoisers yield high-fidelity results when the input image is rendered at high sample rates, and some are even guaranteed to generate consistent (i.e. asymptotically unbiased) results [Bitterli et al. 2016; Moon et al. 2013; Rousselle et al. 2013; Zwicker et al. 2015]. However, they typically suffer from residual noise at lower sample rates. The emerging learning-based denoisers, on the other hand, greatly outperform at low sample rates by embedding much stronger a priori knowledge, yet the black-box nature of neural networks adds to the uncertainty of their performance across distinct data distributions and breaks the guarantee of consistency. Such observations inspire and motivate us to explore the possibility of combining different denoising techniques, inheriting their advantages while alleviating their disadvantages.

In this paper, we propose *ensemble denoising*, a method to combine the denoised results from two or more individual denoisers, which we refer to as *base denoisers*. The key idea of our method is to compute a set of optimal blending weights for the base denoisers in a per-pixel manner that minimizes the overall mean squared error (MSE), which is formulated as a constrained quadratic programming problem. Since the MSE matrix involved in the optimization cannot be accurately obtained, we apply a dual-buffer strategy to estimate it. To keep our algorithm efficient and robust, we further propose an iterative solver to compute the blending weights. Meanwhile, in practice, we improve the visual smoothness of the combined images with a cross bilateral filtering pass on the weight maps.

From the theoretical aspect, our ensemble denoiser has several nice properties. For example, it is never worse than any of the base denoisers if the accurate MSE matrix is known, and it inherits consistency from base denoisers, i.e., it is guaranteed to be consistent (asymptotically unbiased) if at least one base denoiser is consistent. From the practical aspect, our method is demonstrated to be effective and robust, and in most cases outperforms any of the individual base denoisers, through comprehensive evaluations over dozens of scenes and different levels of sample rates.

Naturally, other practical questions may come to users with our ensemble denoising algorithm on hand. For example, which is the overall optimal combination of the base denoisers and which is the best at a certain sample rate? How to choose the appropriate number of base denoisers? ...and so on. To find the answers, we have a thorough analysis of the selection of base denoisers, providing important and practical guides for users. The byproduct that comes along the analysis is also useful, which provides quantitative benchmarks and comparisons between individual denoisers over a large diversity of different scenes and at varying sample rates.

2 RELATED WORK

Monte Carlo denoising, also referred to as Monte Carlo image reconstruction, has been an important topic in computer graphics. The reconstruction is traditionally handled in the same way as image denoising, using classic filters such as Gaussian filters and bilateral filters to remove the noise. Later, more and more approaches use auxiliary features such as per-pixel normals, albedos and depths that are readily available during the rendering process. Such representative work includes *Anisotropic Diffusion for Monte Carlo Noise Reduction* [McCool 1999], *Joint Bilateral Filtering* (JBF) [Kopf et al. 2007] and *Robust Denoising using Feature and Color Information* (RDFC) [Rousselle et al. 2013]. Improvements are made in various ways. For example, non-local methods [Buades et al. 2005; Gastal and Oliveira 2012] extend the filtering kernel (window) to the entire image plane, and higher-order methods [Bitterli et al. 2016; Moon et al. 2016] use complex polynomial models to utilize the sample values in a filtering kernel. Specific-purpose filtering methods are also developed that exploit more information in rendering, such as the sheared filtering and axis-aligned filtering methods [Egan et al. 2011; Mehta et al. 2013; Yan et al. 2015] that remove noise from distribution effects (depth of field, soft shadows, global illumination, etc.) at low sample rates. And real-time filtering methods [Chaitanya et al. 2017; Schied et al. 2017] exploit spatio-temporal coherence to perform fast approximate reconstruction.

In recent years, deep learning has become successful in Monte Carlo denoising [Huo and Yoon 2021]. For example, *Kernel-Predicting Convolutional Networks* (KPCN) [Bako et al. 2017] proposes the influential kernel-prediction architecture, which is widely adopted and further enhanced by many following-up works [Lin et al. 2020a; Vogels et al. 2018; Zeng et al. 2020]. *Adversarial Monte Carlo Denoising* (MCGAN) [Xu et al. 2019] introduces adversarial learning in the training process to improve perceptual quality. *Deep residual learning for denoising Monte Carlo renderings* [Wong and Wong 2019] leverages residual blocks to denoise the renderings in an end-to-end fashion, eliminating the need of predicting intermediate filter kernels. Reliable MC denoising approaches are also provided by the industry nowadays, such as NVIDIA's *OptiX AI Denoiser* [Chaitanya et al. 2017] and Intel's *Open Image Denoise* (OIDN) [Intel 2019].

Efforts are also made to explore other formulations of the denoising task. Path-space and sample-based techniques [Gharbi et al. 2019; Lin et al. 2020b] perform the reconstruction using individual path samples other than in the image plane. Gradient-domain path tracing [Guo et al. 2019; Kettunen et al. 2015] is essentially a gradient-guided denoising process, where an optimization process involving the Poisson equation is applied.

Our ensemble denoising is different from those individual denoising methods. It takes any number of denoised results produced by these methods as base denoisers, and does not make requirements or constraints on any specific types of them. However, since it is not possible to implement all existing base denoisers, in this paper, we select some representative denoisers as the base denoisers with a wide coverage of different categories, e.g., based on different underlying principles (learning-based or non-learning-based), with different artifact patterns (color bias, over-blurring, or residual noise), and with different theoretical properties (consistent or not).

Combined estimators are effective ways to achieve better results than individual ones. A well-known example in light transport is multiple importance sampling (MIS) [Veach 1997; Veach and Guibas 1995], which combines contributions from different sampling techniques. Originally the combining weights are from simple heuristics and recently various methods are proposed in search of a practically better scheme [Grittmann et al. 2019] or the theoretically optimal one [Kondapaneni et al. 2019]. Also, methods to combine multiple light transport algorithms [Bitterli and Jarosz 2019; Georgiev et al. 2012; Otsu et al. 2018] have gradually aroused researchers' interests.

In statistics and machine learning, model averaging and ensemble learning [Polikar 2006] use multiple models strategically generated and combined to improve classification, prediction, function approximation, etc. Methods to find effective combinations are widely researched. Keller and Olkin [2004] combine correlated unbiased estimators with a maximum-likelihood estimate of the unknown covariance matrix and thoroughly studied its efficiency. Lavanciera and Rochet [2016] propose a general method to combine several (possibly) biased independent estimators of the same quantity, whose approach includes cases of estimating multiple parameters and exploits available information in estimators for different parameters.

We transfer the idea of ensemble learning into the field of MC denoising, combining individual denoisers into a more powerful one. Unlike the general constructs in ensemble learning or model averaging, however, the "estimators" we deal with are sophisticatedly designed modern MC denoisers whose structures are too complicated for an analytical derivation on the statistical properties. Therefore, instead of restricting to a specific denoiser category and simply specialize some general combining technique, we treat base denoisers as black boxes without any assumption imposed on their independence or unbiasedness and make use of the domain-specific knowledge and techniques in MC denoising such as the dual-buffer strategy in the formulation and algorithm.

A close recent work to ours in terms of the idea to combine estimators is *Deep Combiner* [Back et al. 2020], which enhances the results of an existing denoiser (typically, KPCN and NFOR) by combining them with the noisy renders, and the combination kernel is predicted through a neural network. Our method differs from Deep Combiner in several ways. First, we are able to combine multiple denoisers while Deep Combiner only targets a specific one. Second, our ensemble denoiser has a theoretical guarantee on optimality, while they do not have such guarantees since their combination kernel is computed through neural networks. Third, we combine results in a per-pixel way, while they combine the two images with kernels filtering neighboring pixel residuals, which has a close relation to image boosting. Nevertheless, Deep Combiner variants of KPCN and NFOR could still be used as base denoisers for our method, as included in the experiments.

Bandwidth selection is a popular technique in MC denoiser design among previous work [Bauszat et al. 2015; Bitterli et al. 2016; Kalantari and Sen 2013; Li et al. 2012; Rousselle et al. 2013], where a bank of candidate filters with different hyperparameters are applied and the best is selected based on the estimated error for each candidate. Error estimation is typically aided by the dual-buffer strategy [Rousselle et al. 2012], where Monte Carlo samples are split into two equal-sized half-buffers and the MSE is estimated from them. In addition to the usage in MC denoising, local bandwidth selection is also found helpful for deciding kernel supports in photon mapping [Kaplanyan and Dachsbacher 2013].

While our idea shares a similarity with this technique in terms of utilizing multiple filters or denoisers at hand and computing estimates from dual- or multiple buffers, the underlying motivations and high-level ideas substantially distinct. Bandwidth selection serves the primal purpose of "selecting" the best filter parameters for each pixel, thus the candidate filters are usually homogeneous and the selection is done in a one-hot manner, which could lead to poor performance when applied to denoisers of different characteristics and error patterns. Our method, on the other hand, models the combination of heterogeneous denoisers as per-pixel weighted sums, which offers better expressiveness and more room for optimization, making it possible to compensate for artifacts of individual denoisers while inheriting their good properties.

Furthermore, the way we exploit the error estimation techniques is unique: the dual-buffer strategy is extended and formalized in the matrix form for multiple denoisers, capturing and modeling not only the error for individual ones but also their correlation. It is a building block deeply integrated into our theoretical formulation and optimization process, rather than simply used as a post-denoising criterion or heuristic for selecting results as is done in previous work, e.g., RDFC and NFOR. This kind of usage in denoising, to our knowledge, has never been explored before.

3 THEORETICAL FORMULATION

An image-space MC denoiser can be formalized as a function f parameterized by handcrafted or learned parameters Θ, which takes the Monte Carlo rendered noisy image x as input, together with a set of auxiliary feature buffers $\mathcal{G}$ (i.e., normal, albedo, depth, etc.), and generates the denoised image y as output:

$$y = f(x, \mathcal{G}; \Theta). \tag{1}$$

The target is to produce a denoised image y as close as possible to the ground truth $\mu = \mathbb{E}[x]$, where $\mathbb{E}[\cdot]$ is the expectation operator.

The common approach taken by existing methods is to design a suitable function f, and search for the best parameters Θ for a certain image x or a series of training images. We, however, take another path by optimally combining the output images of existing individual denoisers in a per-pixel manner. This allows us to leverage the advantages of different denoising techniques while compensating for their unsatisfactory results.

3.1 Ensemble Denoiser

3.1.1 Definition. We consider a set of N *base denoisers* f_i, where $i \in \{1, 2, \ldots, N\}$. Each base denoiser has a different set of its own parameters Θ_i, but takes the same noisy image x and the same auxiliary feature set $\mathcal{G}$ as input. We use y_i to denote the output of the i^{th} denoiser, which, according to Eq. (1), is computed as

$$y_i = f_i(x, \mathcal{G}; \Theta_i). \tag{2}$$

From a statistical point of view, x and y_i are both estimates of the ground truth image μ.

For each pixel p, the *ensemble* $z(p)$, a.k.a. the combined estimator, is defined as a weighted sum of the output of base denoisers:

$$z(p) = \sum_{i=1}^{N} w_i(p) y_i(p), \quad \text{subject to } \sum_{i=1}^{N} w_i(p) = 1. \tag{3}$$

The ensemble definition can be rewritten into the vector form (for brevity we drop the pixel coordinates here):

$$z = \boldsymbol{w}^\top \boldsymbol{y}, \quad \text{subject to } \sum_{i=1}^{N} w_i = 1, \tag{4}$$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^\top$, and $\boldsymbol{w} = (w_1, w_2, \ldots, w_N)^\top$. We refer to $\boldsymbol{w}$ as the *ensemble weights* or *ensemble weight vector*.

3.1.2 Optimization formulation for ensemble weights. Our goal is to find the optimal ensemble, i.e., to compute the optimal ensemble weights $\boldsymbol{w}$. This is done by minimizing the difference between the ensemble z and the ground truth μ with the MSE metric, which yields the following optimization problem:

$$\underset{\boldsymbol{w}}{\text{minimize}} \ \mathbb{E}\left[\left(\boldsymbol{w}^\top \boldsymbol{y} - \mu\right)^2\right], \quad \text{subject to } \sum_{i=1}^{N} w_i = 1. \tag{5}$$

Using the constraint $\sum_{i=1}^{N} w_i = 1$, we rewrite our objective function in Eq. (5) into the quadratic form:

$$\mathbb{E}\left[\left(\boldsymbol{w}^\top \boldsymbol{y} - \mu\right)^2\right] = \boldsymbol{w}^\top \boldsymbol{M} \boldsymbol{w}, \tag{6}$$

where the $\boldsymbol{M}$ is a Gramian matrix and is defined as

$$\boldsymbol{M} = \mathbb{E}\left[\boldsymbol{\gamma}\boldsymbol{\gamma}^\top\right], \tag{7}$$

with $\boldsymbol{\gamma} = (y_1 - \mu,\ y_2 - \mu,\ \ldots,\ y_N - \mu)^\top$.

We refer to $\boldsymbol{M}$ as the *MSE matrix*. The MSE matrix is positive semidefinite, and the feasible solution space of $\boldsymbol{w}$ is a convex set. Therefore, the optimization problem in Eq. (5) is linearly-constrained quadratic programming and can be solved once $\boldsymbol{M}$ is known. However, it is impossible to obtain the accurate $\boldsymbol{M}$ due to its dependency on the unknown ground truth μ and the existence of the expectation operator. Instead, we apply a dual-buffer strategy to estimate $\boldsymbol{M}$.

3.1.3 Dual-buffer strategy to estimate $\boldsymbol{M}$. Expanding Eq. (7), we obtain the element in the i^{th} row and j^{th} column of $\boldsymbol{M}$:

$$\begin{aligned} M_{ij} &= \mathbb{E}\left[\left(y_i - \mathbb{E}[x]\right)\left(y_j - \mathbb{E}[x]\right)\right] \\ &= \mathbb{E}^2[x] - \mathbb{E}[x]\left(\mathbb{E}[y_i] + \mathbb{E}[y_j]\right) + \mathbb{E}[y_i y_j]. \end{aligned} \tag{8}$$

To keep our ensemble model robust and generalizable, we do not have particular restrictions or assumptions on the base denoisers. Therefore, the base denoisers are black boxes to us, and the only observed information is their output.

However, y_i and y_j being the output of two different base denoisers (i.e., f_i and f_j) given the same inputs x, they are highly correlated. Therefore, it will be difficult to directly estimate the correlated terms involved in Eq. (8), e.g., $\mathbb{E}[x]\mathbb{E}[y_i] \neq \mathbb{E}[x y_i]$. Inspired by previous work [Bitterli et al. 2016; Li et al. 2012; Rousselle et al. 2012, 2013], we instead use a dual-buffer strategy to decorrelate y_i, y_j and x to bypass this obstacle.

Dividing samples equally in half, we obtain two independent and identically distributed (i.i.d.) set of samples of the MC estimator, denoted as $\mathcal{A}$ and $\mathcal{B}$. Then, we obtain the two half-buffer noisy images x^A and x^B using the two set of samples respectively. For example, if the noisy image x is rendered with 64 samples per pixel, x^A and x^B will either contain 32 disjoint samples and be an unbiased estimate of $\mathbb{E}[x]$. We further feed x^A and x^B to each base denoiser f_i, obtaining the denoised half-buffer images y_i^A and y_i^B.

Besides the fact that x^A and x^B are uncorrelated, i.e.,

$$\mathrm{Cov}\left[x^A, x^B\right] = 0, \tag{9}$$

we further assume that the denoised half-buffers generated from them are also uncorrelated, i.e.,

$$\mathrm{Cov}\left[y_i^A, y_j^B\right] = \mathrm{Cov}\left[x^A, y_i^B\right] = \mathrm{Cov}\left[x^B, y_i^A\right] = 0. \tag{10}$$

This immediately enables the estimation of M_{ij} in Eq. (8) w.r.t. half buffers as

$$\begin{aligned} \langle M \rangle_{ij}^A &= x^A x^B - x^B\left(y_i^A + y_j^A\right) + y_i^A y_j^A \quad \text{and} \\ \langle M \rangle_{ij}^B &= x^A x^B - x^A\left(y_i^B + y_j^B\right) + y_i^B y_j^B. \end{aligned} \tag{11}$$

Then, we use the average as the estimation of $\boldsymbol{M}$:

$$\langle \boldsymbol{M} \rangle = \frac{1}{2}\left(\langle \boldsymbol{M} \rangle^A + \langle \boldsymbol{M} \rangle^B\right), \tag{12}$$

where the elements of $\langle \boldsymbol{M} \rangle^A$ and $\langle \boldsymbol{M} \rangle^B$ are computed from Eq. (11).

3.1.4 Modified objective function. After substituting $\boldsymbol{M}$ with the estimated MSE matrix $\langle \boldsymbol{M} \rangle$ obtained using the dual-buffer strategy in Eq. (6) and (7), our optimization problem can be rewritten as

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \mathcal{L}(\boldsymbol{w}) = \boldsymbol{w}^\top \langle \boldsymbol{M} \rangle \boldsymbol{w}, \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i = 1 \ \text{ and } \ w_i \geq 0. (i = 1, 2, \ldots, N) \end{aligned} \tag{13}$$

Note here we impose an extra non-negativity constraint on $\boldsymbol{w}$, which is practically necessary for a stable solution and will be evaluated later in Sec. 5.2.1.

It is also worth noting that the convexity of the problem remains unaffected , since the objective function can be rewritten as

$$\mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^\top\left(\boldsymbol{y}^A \boldsymbol{y}^{A\top} + \boldsymbol{y}^B \boldsymbol{y}^{B\top}\right)\boldsymbol{w} - \left(x^A \boldsymbol{y}^B + x^B \boldsymbol{y}^A\right)^\top \boldsymbol{w} + 2x^A x^B, \tag{14}$$

where the matrix $\left(\boldsymbol{y}^A \boldsymbol{y}^{A\top} + \boldsymbol{y}^B \boldsymbol{y}^{B\top}\right)$ is symmetric and positive semi-definite.

3.2 Theoretical Properties

In this subsection, we give several nice theoretical properties of our ensemble denoiser together with their proof.

3.2.1 $\langle \boldsymbol{M} \rangle$ is an unbiased estimator of $\boldsymbol{M}^{\text{half}}$. We use $\boldsymbol{M}^{\text{half}}$ to denote the MSE matrix estimated w.r.t. half buffers:

$$M_{ij}^{\text{half}} = \mathbb{E}^2\left[x^A\right] - \mathbb{E}\left[x^A\right]\left(\mathbb{E}\left[y_i^A\right] + \mathbb{E}\left[y_j^A\right]\right) + \mathbb{E}\left[y_i^A y_j^A\right]. \tag{15}$$

With the uncorrelation assumptions in Eq. (10) and the fact that x^A and x^B are uncorrelated, we can derive that

$$\begin{aligned}\mathbb{E}\left[\langle M\rangle_{ij}^A\right] &= \mathbb{E}\left[x^A x^B\right] - \mathbb{E}\left[x^B\left(y_i^A + y_j^A\right)\right] + \mathbb{E}\left[y_i^A y_j^A\right] \\ &= \mathbb{E}\left[x^A\right]\mathbb{E}\left[x^B\right] - \mathbb{E}\left[x^B\right]\mathbb{E}\left[y_i^A + y_j^A\right] + \mathbb{E}\left[y_i^A y_j^A\right] \\ &= \mathbb{E}^2\left[x^A\right] - \mathbb{E}\left[x^A\right]\mathbb{E}\left[y_i^A + y_j^A\right] + \mathbb{E}\left[y_i^A y_j^A\right] \\ &= M_{ij}^{\text{half}}.\end{aligned} \tag{16}$$

Therefore, $\langle M\rangle_{ij}^A$ is an unbiased estimator for M_{ij}^{half}. The same holds for $\langle M\rangle_{ij}^B$. Furthermore, since $\langle \boldsymbol{M}\rangle$ is the average of $\langle \boldsymbol{M}\rangle^A$ and $\langle \boldsymbol{M}\rangle^B$ as is defined in Eq. (12), it is easy to know that $\langle \boldsymbol{M}\rangle$ is an unbiased estimator for $\boldsymbol{M}^{\text{half}}$.

Admittedly, there is an inevitable discrepancy between the two MSE matrices $\boldsymbol{M}^{\text{half}}$ and $\boldsymbol{M}$ computed using half and full buffers, especially at low sampling rates. However, we find through experiments that this estimator is practically adequate for capturing the characteristics of the base denoisers and supporting robust and effective computation of weights.

3.2.2 The theoretically optimal ensemble. Hypothetically, if the true MSE matrix as defined in Eq. (7) is known, a closed-form solution to the optimal ensemble weights is possible. Lifting the constraint $\sum_{i=1}^N w_i = 1$ off Eq. (5) with the help of the Lagrange multiplier converts the optimization problem into an unconstrained version:

$$\underset{\boldsymbol{w},\lambda}{\text{minimize}} \quad \boldsymbol{w}^\top \boldsymbol{M}\boldsymbol{w} - \lambda\left(\mathbf{1}^\top \boldsymbol{w} - 1\right), \tag{17}$$

where $\mathbf{1}$ (in boldface) denotes the all-one vector with length N.

Leaving out the unconventional possibility of having a zero-MSE combined estimator, the (always existent) solutions to Eq. (17) satisfy

$$\boldsymbol{M}\boldsymbol{w} = \lambda\mathbf{1} \quad \text{and} \quad \lambda \neq 0. \tag{18}$$

Therefore, one theoretically optimal ensemble (with its weights denoted as $\boldsymbol{w}_*$) could be computed through

$$\lambda_* = \frac{1}{\mathbf{1}^\top \boldsymbol{M}^\dagger \mathbf{1}} \quad \text{and} \quad \boldsymbol{w}_* = \lambda \boldsymbol{M}^\dagger \mathbf{1} = \frac{\boldsymbol{M}^\dagger \mathbf{1}}{\mathbf{1}^\top \boldsymbol{M}^\dagger \mathbf{1}}, \tag{19}$$

where $\boldsymbol{M}^\dagger$ denotes the Moore–Penrose inverse of $\boldsymbol{M}$. The resulting minimized MSE is

$$\boldsymbol{w}_*^\top \boldsymbol{M}\boldsymbol{w}_* = \boldsymbol{w}_*^\top \lambda_* \mathbf{1} = \frac{1}{\mathbf{1}^\top \boldsymbol{M}^\dagger \mathbf{1}}. \tag{20}$$

With a noise-free MSE matrix $\boldsymbol{M}$ and with negative weights allowed, since the optimization process literally finds the optimal solution in the whole feasible region, the resulting minimal MSE in Eq. (20) is guaranteed to be less than or equal to the MSE of any individual base denoiser, i.e.,

$$\frac{1}{\mathbf{1}^\top \boldsymbol{M}^\dagger \mathbf{1}} \leq \boldsymbol{e}_i^\top \boldsymbol{M}\boldsymbol{e}_i = \mathbb{E}\left[(y_i - \mu)^2\right], \ (i = 1, 2, \ldots, N) \tag{21}$$

where selecting an individual base denoiser f_i is equivalent to setting the ensemble weights to the one-hot vector $\boldsymbol{e}_i$ with only the i^{th} element being 1 and others being 0.

Note that we need the accurate $\boldsymbol{M}$ to compute the above optimal ensemble. In practice, however, with the absence of the true $\boldsymbol{M}$, it is intrinsically impossible to find the optimal weights as derived in Eq.(19). That said, Eq. (20) serves as a loose lower bound of MSE for our algorithm which might be practically unreachable. Nevertheless, it theoretically reflects the potential ability of our model, and clearly delivers our conclusion: in theory, our ensemble denoiser will be no worse than any of the base denoisers.

3.2.3 Inherited consistency from base denoisers. One of the important properties of our ensemble denoiser is the ability to inherit consistency: if *any one* of the base denoisers is consistent, we ensure that the ensemble denoiser is also consistent. This property stands even when the accurate $\boldsymbol{M}$ is unknown and the non-negativity constraint on weights is imposed, i.e., when solving for the modified optimization problem Eq. (13) that is practically available.

Without loss of generality, let us suppose the j^{th} base denoiser is consistent. That is, when the spp grows towards infinity, the denoised image converges in probability to the ground truth:

$$y_j \xrightarrow{p} \mu, \tag{22}$$

which as well implies the convergence of the denoised half buffers:

$$y_j^A \xrightarrow{p} \mu \quad \text{and} \quad y_j^B \xrightarrow{p} \mu. \tag{23}$$

For the modified optimization problem defined in Eq. (13), if we select the consistent base denoiser f_j solely, or equivalently, set the weights $\boldsymbol{w}$ to the one-hot vector $\boldsymbol{e}_j$, the objective is reduced to

$$\mathcal{L}\left(\boldsymbol{e}_j\right) = \frac{1}{2}\left[\left(y_j^A - x^B\right)^2 + \left(y_j^B - x^A\right)^2 - 2\left(x^A - x^B\right)^2\right] \xrightarrow{p} 0. \tag{24}$$

Therefore, our ensemble, with the optimal solution $\hat{\boldsymbol{w}}$ minimizing the objective in Eq. (13), satisfies $\mathcal{L}\left(\hat{\boldsymbol{w}}\right) \leq \mathcal{L}\left(\boldsymbol{e}_j\right)$. Hence, we have

$$\begin{aligned}2\mathcal{L}\left(\hat{\boldsymbol{w}}\right) &= \left(\hat{\boldsymbol{w}}^\top \boldsymbol{y}^A - x^B\right)^2 + \left(\hat{\boldsymbol{w}}^\top \boldsymbol{y}^B - x^A\right)^2 - 2\left(x^A - x^B\right)^2 \\ &\leq 2\mathcal{L}\left(\boldsymbol{e}_j\right) \xrightarrow{p} 0,\end{aligned} \tag{25}$$

which indicates the convergence towards μ and thus the consistency:

$$\begin{aligned}&\left(\hat{\boldsymbol{w}}^\top \boldsymbol{y}^A - x^B\right) \xrightarrow{p} 0 \quad \Rightarrow \quad \left(\hat{\boldsymbol{w}}^\top \boldsymbol{y}^A - \mu\right) \xrightarrow{p} 0 \\ \Rightarrow \quad &\left(\hat{\boldsymbol{w}}^\top \boldsymbol{y} - \mu\right) \xrightarrow{p} 0 \quad \Rightarrow \quad \hat{\boldsymbol{w}}^\top \boldsymbol{y} \xrightarrow{p} \mu.\end{aligned} \tag{26}$$

4 THE PRACTICAL ALGORITHM

In this section, we introduce the details of our ensemble denoising algorithm based on the theoretical formulation in Sec. 3. The algorithm consists of the following stages:

(1) **Initialization.** We use path tracing to generate the noisy image x and half-buffer images x^A and x^B. These images are fed into each base denoiser to obtain the denoised images y_i as well as the denoised half-buffer images y_i^A and y_i^B.
(2) **Solving for per-pixel ensemble weights.** For each pixel, we estimate the MSE matrix using the dual-buffer strategy, based upon which we solve for the ensemble weights by optimizing the objective function in Eq. (13). The optimization problem can be solved with a general-purpose optimizer, or preferably, a much faster specialized iterative solver, as will be introduced in Sec. 4.1.
(3) **Filtering ensemble weights.** The ensemble weight maps solved in the previous stage is further filtered with a cross

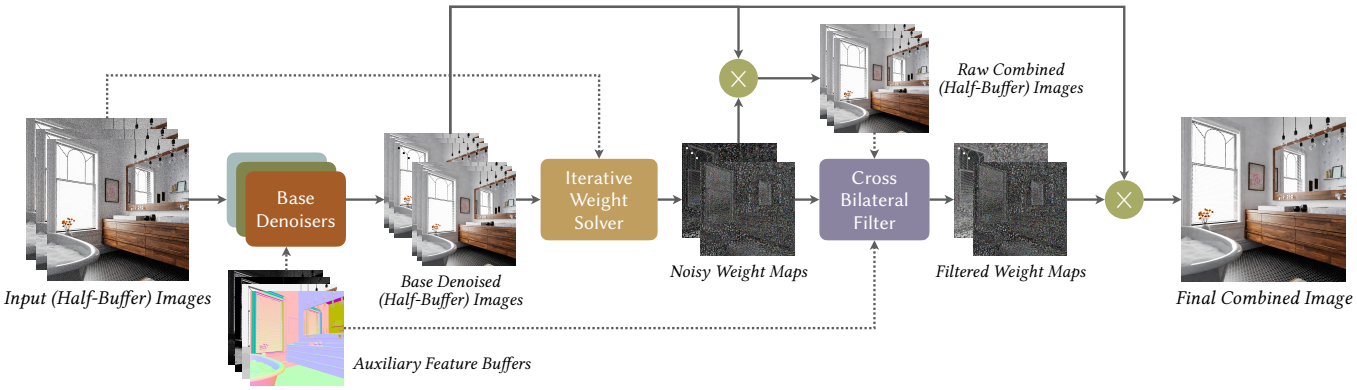

Fig. 2. Pipeline. First, the noisy full and half radiance buffers are fed into base denoisers and denoised in collaboration with auxiliary feature buffers, such as sample variance, albedo, normal, depth, etc. Then, we use the denoised half buffers to empirically estimate combination weight maps using the proposed iterative solver. Later, the noisy weight maps are filtered with a cross bilateral filter guided with the raw combined images and the auxiliary buffers jointly. Finally, the filtered weight maps are applied in a per-pixel manner to the base denoised images to obtain the final combined image.

bilateral filter. After filtering, the combined denoised image could be simply computed in a per-pixel manner as the weighted sums of the base denoised images as is formalized in Eq. (3). See Sec. 4.2.

4.1 Iterative Solver

The ensemble weights $\boldsymbol{w}$ are the solution to the optimization problem in Eq. (13). Since the optimization problem is convex, we can possibly solve it using general-purpose solvers such as OSQP [Stellato et al. 2020], FBstab [Liao-McPherson and Kolmanovsky 2020], etc. While such solvers usually possess heavy optimization for large-scale sparse problems, we found them not well-suited for our application where the number of base denoisers N is relatively small (typically no greater than 10).

Therefore, instead of integrating into our pipeline an existing general-purpose solver, we opt to develop a specialized iterative solver based on the closed-form solution for combining two base denoisers. Below we will first introduce the closed-form solution for combining two denoisers and then describe our iterative solver for combining multiple denoisers.

4.1.1 Closed-form solution for two-denoiser ensembles. An important observation is that, when combining two denoisers, we are able to analytically derive the ensemble weights.

In this case, a rewrite of Eq. (14) with explicitly $\boldsymbol{y} = (y_1, y_2)^\top$ and $\boldsymbol{w} = (w_1, w_2)^\top$ gives the scalar form of the objective function:

$$\begin{aligned}\mathcal{L}(w_1, w_2) = &\left[w_1\left(x^B - y_1^A\right) + w_2\left(x^B - y_2^A\right)\right]^2 \\ &+ \left[w_1\left(x^A - y_1^B\right) + w_2\left(x^A - y_2^B\right)\right]^2 \\ &- \left[\left(w_1 + w_2\right)\left(x^A - x^B\right)\right]^2,\end{aligned} \tag{27}$$

which, considering the constraint that $w_1 + w_2 = 1$, can be further simplified with the substitution $w_1 = 1 - \alpha$ and $w_2 = \alpha$:

$$\begin{aligned}\underset{\alpha}{\text{minimize}} \quad & c_1\alpha^2 - 2c_2\alpha \\ \text{subject to} \quad & 0 \le \alpha \le 1,\end{aligned} \tag{28}$$

where

$$c_1 = \left(y_1^A - y_2^A\right)^2 + \left(y_1^B - y_2^B\right)^2 \quad \text{and}$$
$$c_2 = \left(y_1^A - x^B\right)\left(y_1^A - y_2^A\right) + \left(y_1^B - x^A\right)\left(y_1^B - y_2^B\right).$$

Therefore, the blending factor α for combining two denoisers is

$$\alpha = \text{clip}\left(\frac{\left(y_1^A - x^B\right)\left(y_1^A - y_2^A\right) + \left(y_1^B - x^A\right)\left(y_1^B - y_2^B\right)}{\left(y_1^A - y_2^A\right)^2 + \left(y_1^B - y_2^B\right)^2}\right), \tag{29}$$

where the clip$(\cdot)$ function clamps the value into $[0, 1]$.

4.1.2 Iterative solver for multiple denoisers. We design our iterative solver inspired by the stochastic gradient descent [Saad 1998] and coordinate descent [Wright 2015] techniques in machine learning and non-linear programming. For each pixel in each iteration, we apply the two-denoiser solution to combine the denoised result of our current ensemble and that of a randomly selected base denoiser, leading to a progressive convergence towards the optimum.

First, we initialize $\boldsymbol{w}_{(0)} \in [0, 1]^N$ randomly, and normalize it so that all of its elements sum up to 1.

In the k^{th} iteration, we treat the current ensemble with weights $\boldsymbol{w}_{(k)}$ as a *virtual base denoiser*, and combine it with a randomly selected base denoiser using the two-denoiser solution in Eq. (29). The denoised half-buffer results of the current ensemble are

$$z_{(k)}^A = \boldsymbol{w}_{(k)}^\top \boldsymbol{y}^A \quad \text{and} \quad z_{(k)}^B = \boldsymbol{w}_{(k)}^\top \boldsymbol{y}^B. \tag{30}$$

Assuming the i^{th} base denoiser being selected, we can analytically compute the optimal blending factor $\alpha_{(k)}$ using Eq. (29) based on $z_{(k)}^A$, $z_{(k)}^B$, y_i^A and y_i^B. Then, $\alpha_{(k)}$ is used to update $\boldsymbol{w}$. Specifically,

$$\boldsymbol{w}_{(k+1)} = \left(1 - \alpha_{(k)}\right)\boldsymbol{w}_{(k)} + \alpha_{(k)}\boldsymbol{e}_i. \tag{31}$$

In each iteration, the base denoiser is selected in a way that cycles through all base denoisers, and inside each cycle, the selection is performed in a randomly shuffled order.

The stopping criteria are 1) $\boldsymbol{w}_{(k)}$ not changing for several iterations; or 2) the number of iterations exceeding the user-defined

limit. In our implementation, we set the limit to 1024 and find it more than sufficient for almost all the pixels to converge. Note that since the initial $\boldsymbol{w}_{(0)}$ is normalized and that we always clamp α to $[0, 1]$, the final $\boldsymbol{w}$ naturally satisfies the constraints that all of its elements are in range $[0, 1]$ and sum up to 1.

Theoretically, the convergence and optimality of the iterative solving process is ensured considering the facts: 1) the solution for two base denoisers in Eq. (29) is optimal since the objective is reduced to a simple quadratic function; 2) when following Eq. (31) over iterations, the sequence of the objective function is monotonically bounded and thus convergent to the local minimum; and 3) the convexity of the optimization problem ensures the local minimum being globally optimal. Empirically, we will also demonstrate the effectiveness of our iterative solver in Sec. 5.2.2 by comparison with solvers based on general-purpose quadratic programming.

4.2 Cross Bilateral Filtering on Weight Maps

Till now, the ensemble weights are solved in a per-pixel manner. Since the optimization depends on x^A and x^B that have probably considerable variance especially at low sample rates, the solved weights are usually very noisy, contaminating the final combined image. Nevertheless, errors in effectively denoised images usually exhibit a low-frequency pattern [Bauszat et al. 2015], which also indicates that a filtering pass on the weight maps will potentially lead to a higher fidelity. For clarity, we refer to the weight maps before and after filtering as the raw weight maps and the filtered weight maps, respectively.

Fortunately, we have two sets of images that are relatively clean and are able to guide the filtering of our weight maps. One is the set of raw combined images, i.e. the denoised full and half-buffer images combined by our ensemble denoiser using the raw weight maps, and the other set contains the noise-free auxiliary feature buffers. We use these two sets to cross-guide a bilateral filter [Kopf et al. 2007] on the raw weight maps.

Specifically, for a pixel p in the image plane, the contribution from pixel q in its neighborhood $\mathcal{N}(p)$ is computed as

$$\kappa(p,q) = \exp\left(-\frac{\|p-q\|^2}{2\sigma_s^2}\right)\exp\left(-\frac{D^2\left(z(p),z(q)\right)}{2\sigma_z^2}\right) \prod_{g\in\mathcal{G}}\exp\left(-\frac{\|g(p)-g(q)\|^2}{2\sigma_g^2}\right), \tag{32}$$

where z is the raw combined image and g refers to a feature buffer. σ_s, σ_z and σ_g are user-defined parameters to control the strengths of the filter w.r.t. spatial, color and feature distances. We adopt the idea in [Li et al. 2012] to compute the normalized color distance:

$$D^2\left(z(p),z(q)\right) = \frac{\|z(p)-z(q)\|^2}{\mathrm{Var}[z(p)]+\mathrm{Var}[z(q)]}. \tag{33}$$

There remains the problem of estimating $\mathrm{Var}[z]$, i.e., the per-pixel variance of the raw combined image. We address this by leveraging the dual-buffer strategy again: the raw weights are applied to the denoised half-buffer images $\boldsymbol{y}^A$ and $\boldsymbol{y}^B$, obtaining $z^A = \boldsymbol{w}^\top\boldsymbol{y}^A$ and $z^B = \boldsymbol{w}^\top\boldsymbol{y}^B$, and $\mathrm{Var}[z]$ is then approximated as $\frac{1}{2}\left(z^A - z^B\right)^2$.

Fig. 3. Test scenes.

The filtered weight vector at p is simply the weighted average of neighboring pixels:

$$\boldsymbol{w}^{\mathrm{filter}}(p) = \frac{\sum_{q\in\mathcal{N}(p)}\kappa(p,q)\boldsymbol{w}(q)}{\sum_{q\in\mathcal{N}(p)}\kappa(p,q)}, \tag{34}$$

which will be used for generating the final combined image using our ensemble denoiser in Eq. (3):

$$z^{\mathrm{final}}(p) = \sum_{i=1}^{N} w_i^{\mathrm{filter}}(p)\,y_i(p). \tag{35}$$

Interestingly, the bilateral filtering stage will not affect the inheritance of consistency from base denoisers, which might seem counter-intuitive. This is because when any consistent base denoiser is present, the raw (unfiltered) ensembles z^A and z^B are consistent, thus leading to the extinction of $\mathrm{Var}[z]$ inside the normalized distance metric in Eq. (33). That said, the bilateral filter itself is consistent, so that filtering weight maps should not affect the consistency of the final ensemble in Eq. (35).

In practice, we set the filter radius to 7 pixels and the auxiliary feature set $\mathcal{G} = \{\mathrm{albedo}, \mathrm{normal}\}$. Other parameters are $\sigma_s = 3.0$, $\sigma_z = 2.0$, $\sigma_{\mathrm{albedo}} = 0.25$ and $\sigma_{\mathrm{normal}} = 0.15$. With an optimized implementation on GPU, we typically spend 10 to 30 milliseconds per million pixels, which is negligible as compared to the iterative weight solver, let alone the complete rendering pipeline.

5 EXPERIMENTS

We perform comprehensive experiments on various base denoisers and test scenes to evaluate the performance of our ensemble denoising algorithm. The experiments are performed on a PC equipped with an Intel i9-9900K CPU and an NVIDIA RTX2080-TI GPU.

5.1 Setup

Test scenes. We use the *Tungsten Renderer* [Bitterli 2014] to render a bunch of scenes found on the *Rendering Resources* website [Bitterli

2016]. Totally 30 scenes are selected. The scenes cover a wide range of surface appearances including diffuse, specular, and transparent materials, with coarse and complex geometries including hairs and furs. Fig. 3 shows an overview of them.

For each scene, radiance buffers, feature buffers (e.g. albedo, normal, depth, etc.) and empirical variance buffers (required by some base denoisers), as well as their half-buffer counterparts, are saved. Reference images are rendered with at least 32768 spp.

Base denoisers. We select several state-of-the-art MC denoisers, including #1 KPCN [Bako et al. 2017], #3 MCGAN [Xu et al. 2019], #4 NFOR [Bitterli et al. 2016], #6 OIDN [Intel 2019], #7 the OptiX AI-Accelerated Denoiser [Chaitanya et al. 2017] (referred to as OptiX) and #8 RDFC [Rousselle et al. 2013]. Besides, we also include the Deep Combiner variants [Back et al. 2020] of NFOR and KPCN, which are referred to as #2 NFOR-DC and #5 KPCN-DC, respectively. In total, 8 base denoisers are used in our experiments.

To obtain the denoised half-buffer images y_i^A and y_i^B for each base denoiser, a general solution is to run the denoisers multiple times. Nonetheless, for denoisers like RDFC and NFOR that denoise and exploit half buffers internally for error estimation and bandwidth selection, we make minor modifications to their implementations, saving these intermediate buffers to eliminate the redundancy of denoising them again.

Implementation details. We implement our iterative solver in C++ on CPU aided by the Eigen library [Guennebaud et al. 2010] for vector and matrix data structures and linear algebra operations. Cross bilateral filtering is implemented using CUDA on GPU, where all loops are unrolled and all weight maps are filtered together in a single kernel launch.

Running performance. Our iterative solver takes about 1 second or less, depending on the number of base denoisers. The cross bilateral filtering step takes only negligible times, i.e., tens of milliseconds. The major time overhead lies in running the base denoisers. Generally, each denoiser needs to be run 3 times: once for denoising the full-buffer input image and twice for the half buffers.

Concerns might arise whether running multiple denoisers multiple times is worth the effort. However, any practical denoiser itself is fast enough (typically taking several seconds for a frame) and should not dominate the running time of the hours-long overall rendering pipeline. Besides, as will be demonstrated in the experiments, our algorithm effectively alleviates the artifacts of individual denoisers, delivering higher-quality results, which we believe worthwhile despite the extra overhead.

5.2 Evaluations

To evaluate our design choices, we perform experiments to demonstrate the necessity of imposing the non-negativity constraints, using channel-wise weight maps, and employing a bilateral filtering stage. We also measure the average time for solving weight maps to show the performance gain using our specialized iterative solver over a general-purpose solver.

5.2.1 With vs without restricting to non-negative weights. While looser constraints provide broader solution spaces for potentially better combined estimators [Kondapaneni et al. 2019], they can also lead to overfitting and instability. To decide whether or not we should impose the non-negativity constraints, in Fig. 4 we compare the combined images with and without the constraint $w_i \geq 0$ ($i = 0, 1, \ldots, N$). As the figure shows, the instability of the unconstrained weights leads to numerous fireflies in the combined images, while the constrained weights produce high-fidelity results.

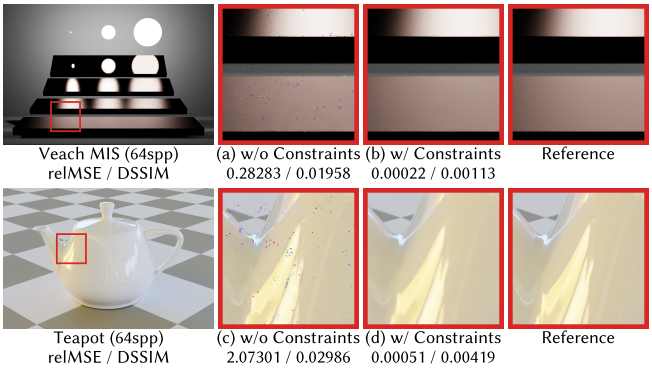

Fig. 4. The combined images generated with and without the non-negativity constraints on weights, respectively. Solving the weights without constraints leads to fireflies in the combined images (a) and (c), and a dramatically increased error, while the images (b) and (d) using constrained weights are of high fidelity in the same regions. Both scenes are rendered with 64 spp, combining 4 base denoisers of RDFC, NFOR, KPCN and OIDN.

5.2.2 Iterative solver vs general-purpose solver. We compare our iterative solver with FBstab [Liao-McPherson and Kolmanovsky 2020] which we consider as the most robust general-purpose solver for our problem compared to e.g. Ceres-Solver [Agarwal and Mierle 2010], OSQP [Stellato et al. 2020], etc. In Fig. 5, we show the weight maps and combined images generated by our iterative solver and the general-purpose solver for the Classroom scene. The scene is rendered with 128 spp and 4 base denoisers (KPCN, NFOR, RDFC and OIDN) are combined. The weight maps and combined images generated from both solvers are visually indistinguishable. Fig. 6 shows the running time required by both solvers for different ensemble sizes. Our iterative solver is about 10 times faster than the general-purpose solver.

5.2.3 Single-channel vs channel-wise weights. We design the ensemble denoising algorithm in a per-pixel combining fashion, but there still remains the open option whether to solve the channel-wise weights or to share weights across channels. Sharing weights across channels reduces computation since we only need to solve them once for a pixel, but allowing channel-wise weights provides more room for optimization, e.g. when there are color bias artifacts in the base denoised images. Fig. 7 compares combined images generated with single-channel and channel-wise weights. We found using channel-wise weights leads to a slightly lower error in both metrics. Considering the efficiency advantage of our iterative solver, we decide to use channel-wise weights for a better denoising quality at the cost of minor computational overhead.

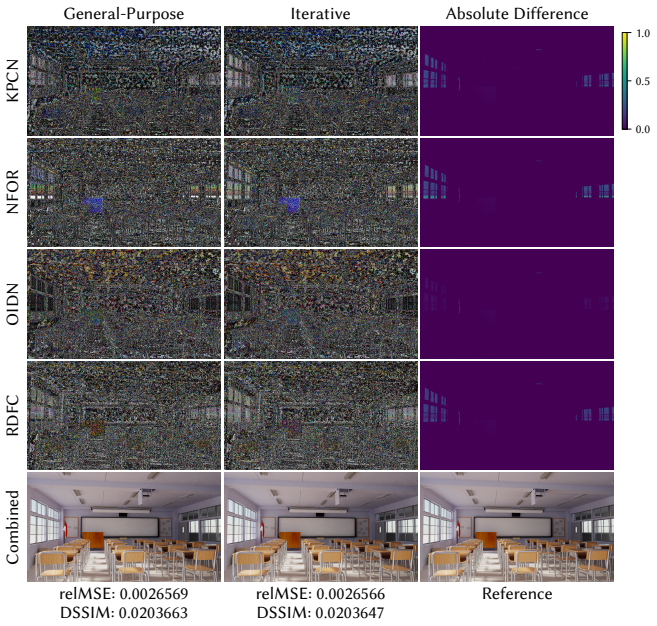

Fig. 5. Weight maps computed using our proposed iterative solver and the general-purpose solver. Left column: noisy weight maps from the general-purpose solver and the raw combined image (without cross bilateral filtering). Middle column: counterparts of our iterative solver. Right column: visualizations of absolute difference between the weight maps. Bottom row: the combined images from the general-purpose solver and from our iterative solver, and the reference render. Weight maps from our iterative solver are very close to that from the general-purpose solver, and the combined images have almost identical relMSE and DSSIM (ours are even lower).

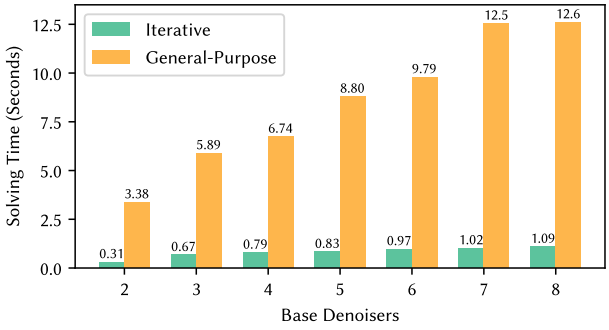

Fig. 6. Average running time of our iterative solver and the general-purpose solver. We test the iterative solver and the general-purpose solver on the Classroom scene rendered with 128 spp, enumerating all combinations. The iterative solver is about 10x faster across all sizes of ensembles.

5.2.4 With vs without cross bilateral filtering. Fig. 8 compares the combined images generated with the raw noisy weights and the filtered weights, respectively. Noisy weight maps can lead to noise in continuous regions and visual artifacts along sharp edges. Nevertheless, such defects disappear after applying cross bilateral filtering to the weight maps.

Fig. 7. Combined images generated with single-channel weights and channel-wise weights, respectively. Using the channel-wise weights offers slightly improved quality and reduced error. Both scenes are rendered with 64 spp, where 3 base denoisers (RDFC, NFOR and KPCN) are used.

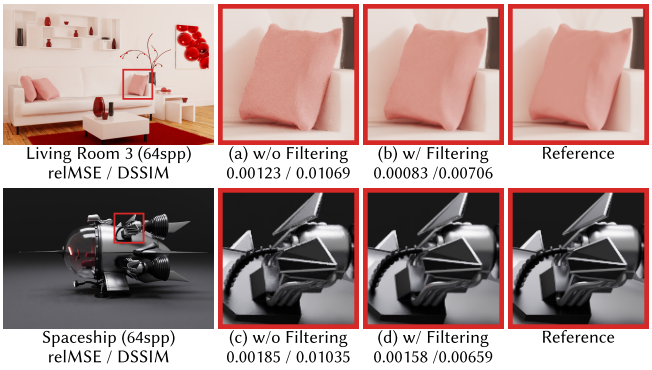

Fig. 8. Combined images generated with and without cross bilateral filtering on weight maps, respectively. Noisy weight maps lead to a less smooth combined image, with noise in the continuous regions (a) or artifacts along sharp edges (c). The images (b) and (d) combined with filtered weights eliminate such defects and achieve significantly improved perceptual quality and reduced error. Both scenes are rendered with 64 spp, where 4 base denoisers (RDFC, NFOR, KPCN and OIDN) are used.

5.2.5 Comparison with naive averaging. In Fig. 9, we compare our algorithm to the naive method which simply averages the output from base denoisers. By solving weights through optimization, our ensemble conducts a more optimized and effective combination of the base denoisers than the naive method, leading to better perceptual quality and significantly reduced quantitative error.

5.2.6 Comparison with one-hot selection. We compare our method to the one-hot selection method originally used for the bandwidth selection technique. Following the formulation in NFOR, MSEs of denoised half buffers from the i^{th} candidate filter are estimated as

$$\begin{aligned} \text{MSE}\left[y_i^A\right] &\approx \left(y_i^A - x^B\right)^2 - \text{Var}[x^B] \text{ and} \\ \text{MSE}\left[y_i^B\right] &\approx \left(y_i^B - x^A\right)^2 - \text{Var}[x^A], \end{aligned} \tag{36}$$

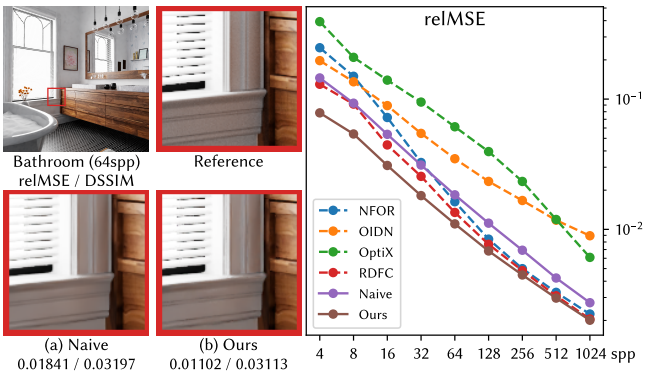

Fig. 9. Comparison with naive averaging, combining the output of NFOR, OIDN, OptiX and RDFC. Left: visual comparison. The naively averaged image (a) has a slightly blurred and faded geometry in the inset region, while our method (b) well matches the reference. Right: error curves. Our method consistently outperforms the base denoisers and the naive averaging method, achieving the lowest relMSE across all sample rates.

	spp	4	8	16	32	64	128	256	Avg.
#Base	2	88.8%	84.2%	80.4%	72.9%	66.7%	62.9%	63.3%	74.2%
	3	84.6%	81.2%	76.7%	74.6%	64.6%	63.3%	60.0%	72.1%
	4	81.0%	79.5%	77.1%	78.6%	74.3%	68.6%	61.0%	74.3%
	5	92.2%	84.4%	84.4%	83.3%	84.4%	81.1%	75.6%	83.7%
	6	81.7%	73.3%	76.7%	79.2%	76.7%	76.7%	65.8%	75.7%
	8	80.0%	76.7%	76.7%	83.3%	83.3%	80.0%	66.7%	78.1%
Avg.		85.1%	80.8%	78.5%	76.8%	71.4%	68.4%	63.5%	74.9%

Table 1. Percentages under the relMSE metric that our ensemble denoising method outperforms both pre-filtering variants (using filters from SBF and NFOR, respectively) of the one-hot selection method. Our method has the lowest error in most cases across varying ensemble sizes and sample rates.

and the MSE for the full buffer is estimated as

$$\mathrm{MSE}\left[y_i\right] = \frac{1}{2}\left(\mathrm{MSE}\left[y_i^A\right] + \mathrm{MSE}\left[y_i^B\right]\right) - \mathrm{Var}[y_i]. \tag{37}$$

The selection map is then generated based on the estimated MSE maps choosing per pixel the filter with the lowest error. We implement three variants of the one-hot method: the first one (labeled "SBF") pre-filters the estimated MSE maps with the cross bilateral filter from SBF [Li et al. 2012]; the second one (labeled "NFOR") pre-filters the estimated MSE maps with the same non-local means filter as in NFOR; and the last one (labeled "Post") uses the noisy MSE estimates but post-filters the selection maps instead, using the same cross bilateral filter described in Sec. 4.2.

In Table 1 and Table 2, we list the percentages w.r.t. the relMSE metric that our method outperforms the one-hot method across all test scenes with varying sample rates and ensemble sizes. Also, we include visual comparison and error curves for the Bathroom 2 scene in Fig. 10 as an example.

Our method outperforms the one-hot method in most test cases with lower errors. Besides, binary weights make the one-hot method prone to visual discontinuity, while ours always produces clean and

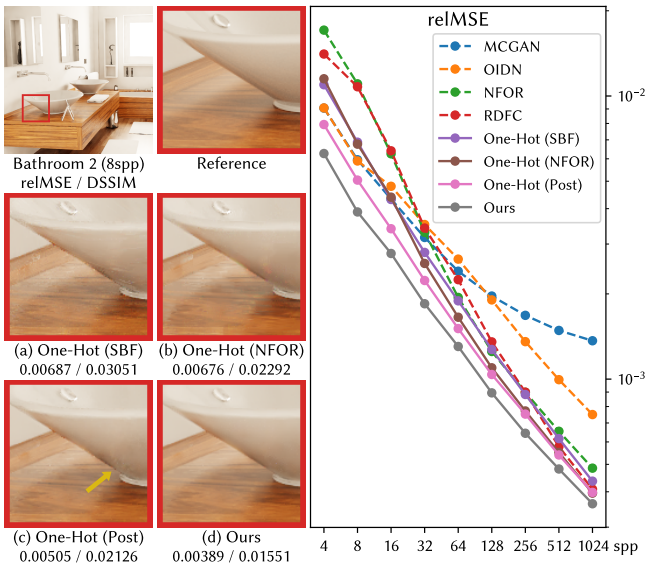

Fig. 10. Comparison with the one-hot selection method, combining the output of MCGAN, OIDN, OptiX and RDFC. Left: visual comparison. The one-hot selection variants that pre-filter the estimated MSE maps (a and b) exhibits visual discontinuities in the inset region. Post-filtering the selection maps (c) alleviates this issue, but artifacts are still noticeable, e.g., along the edge and at the bottom of the washbasin. Our method (d) gives the best visual quality and the lowest quantitative error. Right: error curves. Our method outperforms all the base denoisers and the one-hot selection methods at all sample rates, consistently achieving the lowest relMSE.

	spp	4	8	16	32	64	128	256	Avg.
#Base	2	73.8%	70.4%	64.6%	67.1%	67.5%	66.7%	63.3%	67.6%
	3	79.2%	75.8%	68.3%	63.7%	59.6%	57.1%	52.5%	65.2%
	4	92.9%	91.0%	82.4%	77.6%	76.7%	71.4%	63.8%	79.4%
	5	85.6%	84.4%	76.7%	71.1%	73.3%	77.8%	71.1%	77.1%
	6	89.2%	90.0%	84.2%	77.5%	75.8%	77.5%	67.5%	80.2%
	8	83.3%	80.0%	80.0%	66.7%	66.7%	70.0%	56.7%	71.9%
Avg.		82.9%	80.6%	73.8%	70.3%	69.1%	67.8%	61.7%	72.3%

Table 2. Percentages under the relMSE metric that our ensemble denoising outperforms the post-filtering variant of one-hot selection.

smooth results. Exceptions could happen that the one-hot selection appears more robust than our method, when the noise overwhelms the meaningful information in our ensemble model.

Another interesting observation is that, the pre-filtering implementation variants of one-hot selection usually produce worse results than the post-filtering one, especially at lower sampling rates. One possible explanation is that the MSE of effective base denoisers can be of small magnitudes and thus more sensitive to the bias introduced by the pre-filtering process.

5.2.7 Comparison with weight maps from ground-truth statistics. To validate the effectiveness of weights solved by our algorithm, we compare the weight maps and combined images to those computed

using ground-truth MSE matrices $\boldsymbol{M}$ defined in Eq. (7) for full buffers and $\boldsymbol{M}^{\text{half}}$ defined in Eq. (15) for half buffers.

To compute these statistics, a scene is rendered many times at the same sample rate but with different seeds for random number generation. The respective statistics are repeatedly evaluated in each rendering and accumulated until convergence. For each ground-truth MSE matrix, we compute one-hot selection maps and ensemble weight maps with or without the non-negativity constraints.

Fig. 11 gives an example on the Curly Hair scene rendered with 256spp, combining results from MCGAN, OptiX and RDFC. There are several observations from the figure:

- Weight maps from our algorithm (i) closely match the ground-truth ones solved with the non-negativity constraints (c and f), and the quality of the combined image is on a par as well, which demonstrates the effectiveness of our algorithm;
- Our ensemble model yields lower error than the one-hot selection method with ground-truth matrices (a vs c and d vs f), and with estimated statistics (g vs i), indicating that the expressiveness of our model is superior to the one-hot method, as is discussed in Sec. 3.2.2;
- Ground-truth weight maps w.r.t. half buffers (d, e and f) are close to those of full buffers (a, b and c), and the combined images exhibit similar quantitative error and visual quality, which confirms the rationality of solving weights with the dual-buffer strategy as introduced in Sec. 3.1.3; and
- With ground-truth statistics available, solving weights without non-negativity constraints (b and e) gives more aggressive weights, resulting in lower error than those with constraints (c and f). However, in practice where true MSE matrices are unknown, solving weights with the non-negativity constraints (i) ensures the stability of optimization, otherwise broken results could appear (h).

5.3 Choices of Base Denoisers

With many base denoisers at hand, here come some natural questions. For examples, which are the overall optimal combinations of these base denoisers? Which are the optimal combinations in different cases (e.g. for a specific sample rate)? To find the answers, we enumerate a group of candidate combinations with varying sizes, perform quantitative and perceptual analysis on them, and develop a score-and-sort ranking framework as the selection criterion.

5.3.1 Candidate combinations. Since we have 8 base denoisers, totaling $(2^8 - 8 - 1) = 247$ different combinations, evaluating all those combinations will be a heavy workload and is in fact unnecessary. Instead, we carefully construct a moderate-size pool of candidate combinations. The main considerations and criteria in the pre-selection are twofold: diversity and representativeness. Therefore, we assign more quota to small-sized combinations, since in the primal experiments we found large ensembles are less sensitive to changes of one or two members in them. The final pool consists of 31 combinations. All the 39 candidates (31 ensembles plus 8 base denoisers) are examined and evaluated across all 30 test scenes.

5.3.2 Error metrics. To measure the fidelity of the combined (denoised) images, we adopt relative MSE (relMSE) and structural dissimilarity (DSSIM) as the error metrics. While the former is directly tied to our optimization objective, the latter is more about perceptual quality, providing a supplementary dimension for performance evaluation. For relMSE, we calculate it in the linear high-dynamic-range (HDR) space without any tone mapping or gamma correction. For DSSIM, we beforehand apply ACES tone mapping [Cooper et al. 2017] and sRGB colorspace encoding to the HDR images.

5.3.3 Scores and rankings. In Fig. 13, we plot the error curves w.r.t. sample rates for some candidates in 6 scenes. We do not include all candidates in this plot for a better visualization, since otherwise the lines would overlap a lot, making it hard to distinguish and analyze. While the error metrics are able to measure and reveal the effectiveness of the candidates per test case (i.e., a scene rendered with certain spp), we lack a high-level view for their overall performance. However, simply averaging the errors across all scenes and sample rates for each method is not a valid choice for scoring, since errors in different test cases are usually of distinct scales. Therefore, we propose a ranking framework for a more intuitive and summarized depiction on the overall performance of each candidate.

For each scene rendered at a certain sample rate (i.e. a single test case), we first sort all the 39 candidates together with the noisy MC render according to their errors, and assign linearly spaced scores from 1 to 40 to each of them. For example, the candidate ranked 1^{st} with the lowest error receives a score of 40, while the candidate ranked last receives a score of 1. The ranking scores of all candidate denoisers for two example scenes Bathroom and Dining Room are plotted in Fig. 14 (a, b) and (e, f). The scores are then averaged across all scenes for these candidate denoisers as in Fig. 14 (c) and (g), which are further reduced to the overall scores by averaging across all sample rates, as is shown in Fig. 14 (d) and (h). Then, we compare and analyze the scores and rankings of all candidate denoisers, and some useful and interesting conclusions will be discussed later.

More plots of error curves and ranking curves can be found in the supplemental materials.

5.3.4 Observations from the scores and rankings. First, we find that combining multiple denoisers is effective. A combination of 2 base denoisers almost always outperforms the corresponding individual base denoisers, and combining 3 or 4 base denoisers shows further improvement. Interestingly, however, the marginal benefits begin to diminish when combining more than 5 base denoisers, sometimes even slightly decreased quality compared to smaller ensembles. This phenomenon is not unexpected, given the ensemble weights being solved from imperfect estimated statistics. One such example is Table 3, where the overall ranking scores progressively increase as the ensemble size grows to 5 and start to drop from then on. Nevertheless, this degeneration does not indicate the failure of our algorithm — as is shown in the table, combination #35 (composed of 6 base denoisers) is still very competitive and the overall performance clearly surpasses all the corresponding individual base denoisers.

We also find certain combinations not competitive. For example, combination #28 (KPCN-DC, MCGAN, OIDN and OptiX) performs worse than others of the same ensemble size. We notice that in most

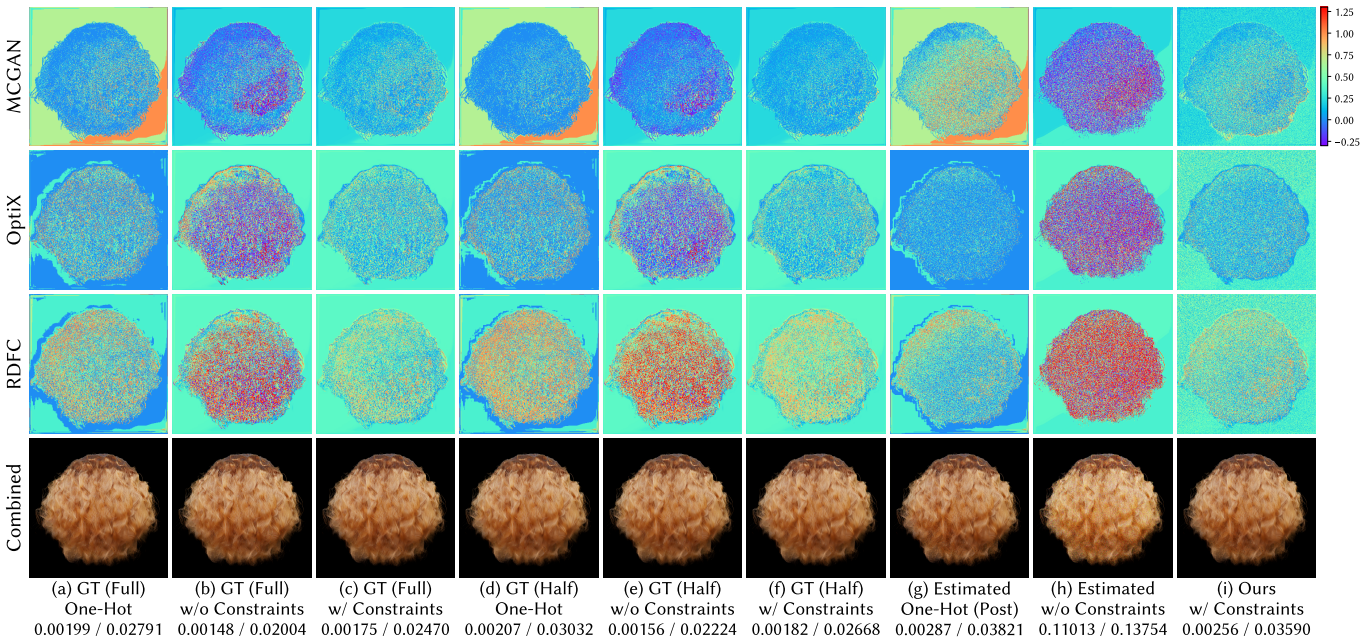

Fig. 11. Comparison with weight maps and combined images from ground-truth statistics evaluated on the Curly Hair scene rendered 1024 times at 256spp combining MCGAN, OptiX and RDFC. Images (a) to (f) are computed with ground-truth statistics, solving for one-hot selection maps, ensemble weight maps without non-negativity constraints and weights with constraints. Images (g) to (i) are computed with estimated statistics using the dual-buffer strategy. In the top three rows, we visualize the weight maps (with channels averaged and colormaps applied) for the base denoisers, and the bottom row contains the combined results with relMSE/DSSIM listed below. Our method gives weight maps close to those from ground-truth MSE matrices with non-negativity constraints (c and f) and outperforms the one-hot selection method (g), demonstrating the effectiveness of our model and solving algorithm. Solving weights using estimated statistics without non-negativity constraints emits broken results (h) while ours with the constraints gives stable weight maps and the high-quality combined image (i).

spp	4	8	16	32	64	128	256	512	1024	Avg.
#1 KPCN	16.5	15.0	15.0	16.0	15.7	16.8	16.7	15.4	12.8	15.5
#3 MCGAN	15.9	14.2	12.7	10.5	7.8	5.9	4.6	3.5	2.6	8.6
#4 NFOR	5.0	5.1	5.8	8.1	9.7	11.4	12.3	12.5	14.0	9.3
#6 OIDN	23.8	17.7	12.0	8.9	7.0	6.6	5.2	4.9	4.2	10.0
#7 OptiX	12.4	9.9	7.5	6.5	5.9	5.0	5.0	4.6	4.5	6.8
#8 RDFC	10.9	10.0	10.7	10.4	11.8	12.9	14.6	16.8	18.1	12.9
#16 (#7+#8)	21.7	18.6	17.3	16.2	17.5	18.5	19.3	21.4	22.0	19.2
#24 (#16+**#6**)	**28.6**	24.8	22.4	19.4	19.5	19.1	19.3	21.0	21.3	21.7
#30 (#24+**#4**)	20.7	22.0	23.2	24.0	25.1	26.7	27.4	27.7	27.9	25.0
#33 (#30+**#1**)	23.5	**25.9**	**28.0**	**29.1**	29.3	**31.0**	**31.1**	**31.5**	**31.8**	**29.0**
#35 (#33+**#3**)	21.1	23.4	27.0	28.5	**29.4**	29.4	28.4	27.6	26.3	26.8

Table 3. Performance of the candidates as the ensemble sizes increase (larger scores are better). The top half of the table lists the individual base denoisers. The bottom half contains the incrementally growing ensembles, composed of 2, 3, 4, 5 and 6 base denoisers respectively. For each candidate individual or ensemble denoiser, we show its relMSE ranking scores w.r.t. sample rates and the overall scores from Fig. 14.

cases, such combinations only involve either learning-based denoisers such as KPCN, or only non-learning-based ones such as NFOR. We also notice that at lower sample rates, learning-based denoisers significantly outperform non-learning-based ones, while at higher sample rates, non-learning-based denoisers perform better. As a result, we do not recommend combining homogeneous denoisers (e.g., #28). Instead, combining heterogeneous denoisers turns out to be more beneficial, as there is greater chance to compensate for one's disadvantages with the distinct strengths of another.

In addition, combinations with DC (Deep Combiner) variants often underperform combinations without DC variants. For instance, #29 (KPCN-DC, NFOR, OIDN and RDFC) ranks lower than #27 (KPCN, NFOR, OIDN and RDFC). Hence, we do not recommend choosing DC variants as base denoisers either.

5.3.5 Best combinations concerning different scenarios. For users' convenience, we recommend combinations concerning different scenarios. For different sizes, we recommend these ensembles:

- Two: #10 (KPCN and NFOR);
- Three: #20 (KPCN, OIDN and RDFC);
- Four: #27 (KPCN, NFOR, OIDN and RDFC).

As for different sample rates, we recommend the following ones:

- Low (4-32 spp): #20 (KPCN, OIDN and RDFC);
- Medium (32-256 spp): #27 (KPCN, NFOR, OIDN and RDFC);
- High (>256 spp): #18 (KPCN, NFOR and RDFC).

5.3.6 Overall best combinations. In general, we found that #27 (KPCN, NFOR, OIDN and RDFC) is the most robust combination

and works reasonably well across all the scenes and sample rates under both the relMSE and DSSIM metrics. We recommend it as the overall best choice. If users would like to use fewer base denoisers, combination #20 (RDFC, KPCN and OIDN) is also a good choice.

5.4 Visual Comparison

Besides the above comprehensive quantitative evaluation, we also provide visual comparison in Fig. 15. A glance of the figure gives the general impression that the majority of the best results fall into our ensemble series across the scenes. Next, we provide detailed analysis on individual scenes to reveal more about the capability and versatility of our method.

For example, in Bathroom 2, NFOR produces lumpy bright noise on the basin. OptiX leaves the wooden table with remaining noise and produces chromatic noise at the edge of the basin, but both defects are wiped out in our cases.

In Dining Room, NFOR, OptiX and RDFC fail to distinguish and remove the outliers in the shadow of the lamp in the noisy input and produce bright spots to different extents. However, our ensemble denoisers significantly alleviate such artifacts by combining results from other base denoisers such as OIDN and KPCN.

In Furball, KPCN and OIDN over-blur the furs, MCGAN suffers from severe color bias, and OptiX leaves remaining noise. Nevertheless, our combined denoisers, #32 as a representative, preserve the fine geometric details of the furs at the reference level, with all patterns of noise and artifacts in base denoisers almost cleaned up.

In Veach Ajar where the white porcelain teapot is severely contaminated by noise, NFOR does well in reconstructing the geometry along the edges of the teapot but the result exhibits obvious low-frequency residual noise at the middle part. OIDN has the noise aggressively reduced at the cost of over-blurring the silhouettes. The result of KPCN is impressive, without obvious defects. Yet our ensemble #27 robustly yields clean results with high fidelity, keeping the excellence of KPCN while circumventing the drawbacks of NFOR and OIDN, achieving a significant decline in both relMSE and DSSIM compared to all the aforementioned base denoisers.

For more results on visual comparison, please refer to the interactive viewer (static web pages) in our supplemental materials.

6 CONCLUSION, LIMITATION AND FUTURE WORK

Conclusion. We have presented ensemble denoising, an optimization-based technique that combines results from multiple base denoisers. With the help of the dual-buffer strategy and our iterative solver, the ensemble weights are efficiently computed in a pixel-wise manner, which is further filtered with a cross bilateral filter to improve the fidelity of the final combined image.

Our ensemble denoiser has nice theoretical properties, e.g., inherited consistency from base denoisers. From a practical perspective, it is demonstrated to be effective and robust, and outperforms any individual denoisers. Furthermore, we have performed a comprehensive analysis on the choices of base denoisers, providing practical guides for users. Besides, our ensemble denoiser does not make any assumption on specific types of base denoisers, but treats them as black boxes. With this feature, other sophisticated individual denoisers can be easily included into our framework.

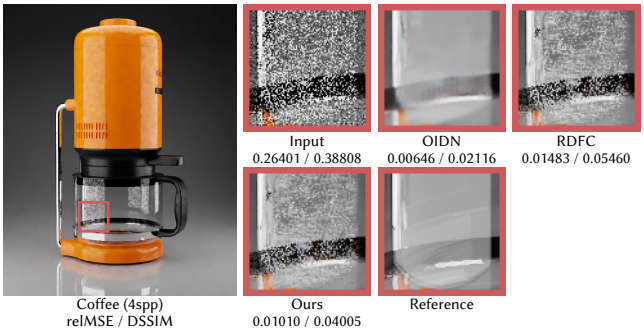

Fig. 12. Less successful case at an extremely low sample rate (4 spp). Two base denoisers (OIDN and RDFC) are used.

Limitation. Our method may perform less successfully when the sample rate is extremely low (e.g. 4 spp). One such case is shown in Fig. 12. This is because the optimization of the ensemble weights relies on the estimation of the MSE matrix. At extremely low sample rates, such estimation has a high variance and is less reliable. Therefore, an immediate future work is investigating ways to further improve the estimation of the MSE matrix.

Future work. Currently, as a general framework for combining multiple denoisers, we do not explicitly consider specialized use cases (e.g., denoising consecutive frames with temporal stability). We look forward to future work optimizing our method with domain-specific knowledge. Besides, our ensemble denoiser combines outputs of all base denoisers in a per-pixel manner. It would be interesting to explore possible formulations considering the local or even non-local neighborhoods, which might benefit from recent advances in general image denoising and super-resolution techniques [Liu et al. 2019; Zin et al. 2021].

ACKNOWLEDGMENT

We would like to thank the reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (Project Numbers: 61822204, 61521002, 61863031) and a research grant from the Beijing Higher Institution Engineering Research Center. Ling-Qi Yan is supported by gift funds from Adobe, Dimension 5 and XVerse.

REFERENCES

Sameer Agarwal and Keir Mierle. 2010. Ceres Solver. http://ceres-solver.org.

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2020. Deep combiner for independent and correlated pixel estimates. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–12.

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 (Proceedings of SIGGRAPH 2017)* 36, 4, Article 97 (2017), 97:1–97:14 pages. https://doi.org/10.1145/3072959.3073708

Pablo Bauszat, Martin Eisemann, Elmar Eisemann, and Marcus Magnor. 2015. General and Robust Error Estimation and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* 34, 2 (2015), 597–608. https://doi.org/10.1111/cgf.12587 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12587

Benedikt Bitterli. 2014. Tungsten Renderer. https://benedikt-bitterli.me/tungsten.html.

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

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively Metropolised Monte Carlo Light Transport Simulation. *ACM Trans. Graph.* 38, 6, Article 153 (Nov. 2019), 10 pages. https://doi.org/10.1145/3355089.3356578

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. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 107–117.

Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, 60–65.

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. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Sean Cooper, Remi Achard, Alex Fry, Robert Molholm, Lars Borg, Lucien Fostier, Anders Langlands, Thomas Mansencal, Steve Agland, Brian Karis, et al. 2017. Retrospective and Enhancements. (2017).

Kevin Egan, Florian Hecht, Frédo Durand, and Ravi Ramamoorthi. 2011. Frequency analysis and sheared filtering for shadow light fields of complex occluders. *ACM Transactions on Graphics (TOG)* 30, 2 (2011), 1–13.

Eduardo S. L. Gastal and Manuel M. Oliveira. 2012. Adaptive Manifolds for Real-Time High-Dimensional Filtering. *ACM TOG* 31, 4, Article 33 (2012), 33:1–33:13 pages. Proceedings of SIGGRAPH 2012.

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

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 (TOG)* 38, 4 (2019), 1–12.

Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-Aware Multiple Importance Sampling. *ACM Trans. Graph.* 38, 6, Article 152 (Nov. 2019), 9 pages. https://doi.org/10.1145/3355089.3356515

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

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 (TOG)* 38, 6 (2019), 1–13.

Yuchi Huo and Sung-eui Yoon. 2021. A survey on deep learning-based Monte Carlo denoising. *Computational Visual Media* 7, 2 (2021), 169–185. https://doi.org/10.1007/s41095-021-0209-9

Intel. 2019. Open Image Denoise. https://www.openimagedenoise.org/.

Nima Khademi Kalantari and Pradeep Sen. 2013. Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms. *Computer Graphics Forum* 32, 2pt1 (2013), 93–102. https://doi.org/10.1111/cgf.12029

Anton S. Kaplanyan and Carsten Dachsbacher. 2013. Adaptive Progressive Photon Mapping. *ACM Trans. Graph.* 32, 2, Article 16 (April 2013), 13 pages. https://doi.org/10.1145/2451236.2451242

Timothy Keller and Ingram Olkin. 2004. Combining Correlated Unbiased Estimators of the Mean of a Normal Distribution. *Lecture Notes-Monograph Series* 45 (2004), 218–227. http://www.jstor.org/stable/4356311

Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Path Tracing. *ACM Trans. Graph.* 34, 4 (2015).

Ivo Kondapaneni, Petr Vevoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. 2019. Optimal Multiple Importance Sampling. *ACM Trans. Graph.* 38, 4, Article 37 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323009

Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint bilateral upsampling. *ACM Transactions on Graphics (ToG)* 26, 3 (2007), 96–es.

Frédéric Lavancier and Paul Rochet. 2016. A general procedure to combine estimators. *Computational Statistics & Data Analysis* 94 (2016), 175–192.

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

Dominic Liao-McPherson and Ilya Kolmanovsky. 2020. FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming. *Automatica* 113 (2020), 108801. https://doi.org/10.1016/j.automatica.2019.108801

Weiheng Lin, Beibei Wang, Lu Wang, and Nicolas Holzschuch. 2020a. A detail preserving neural network model for Monte Carlo denoising. *Computational Visual Media* 6, 2 (2020), 157–168. https://doi.org/10.1007/s41095-020-0167-7

Weiheng Lin, Beibei Wang, Jian Yang, Lu Wang, and Ling-Qi Yan. 2020b. Path-based Monte Carlo Denoising Using a Three-Scale Neural Network. In *Computer Graphics Forum*. Wiley Online Library.

Shuai Liu, Ruipeng Gang, Chenghua Li, and Ruixia Song. 2019. Adaptive deep residual network for single image super-resolution. *Computational Visual Media* 5, 4 (2019), 391–401. https://doi.org/10.1007/s41095-019-0158-8

Michael D. McCool. 1999. Anisotropic Diffusion for Monte Carlo Noise Reduction. *ACM Trans. Graph.* 18, 2 (April 1999), 171–194. https://doi.org/10.1145/318009.318015

Soham Uday Mehta, Brandon Wang, Ravi Ramamoorthi, and Fredo Durand. 2013. Axis-aligned filtering for interactive physically-based diffuse indirect lighting. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Bochang Moon, Jong Yun Jun, JongHyeob Lee, Kunho Kim, Toshiya Hachisuka, and Sung-Eui Yoon. 2013. Robust image denoising using a virtual flash image for Monte Carlo ray tracing. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 139–151.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive polynomial rendering. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–10.

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*, Art B. Owen and Peter W. Glynn (Eds.). Springer International Publishing, Cham, 409–427.

Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine* 6, 3 (2006), 21–45.

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-Local Means Filtering. *ACM Trans. Graph.* 31, 6, Article 195 (Nov. 2012), 11 pages. https://doi.org/10.1145/2366145.2366214

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust denoising using feature and color information. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 121–130.

David Saad. 1998. Online algorithms and stochastic approximations. *Online Learning* 5 (1998), 6–3.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*. 1–12.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. 2020. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation* 12, 4 (2020), 637–672. https://doi.org/10.1007/s12532-020-00179-2

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Vol. 1610. Stanford University PhD thesis.

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*. 419–428.

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 Transactions on Graphics (TOG)* 37, 4 (2018), 1–15.

Kin-Ming Wong and Tien-Tsin Wong. 2019. Deep residual learning for denoising Monte Carlo renderings. *Computational Visual Media* 5, 3 (2019), 239–255. https://doi.org/10.1007/s41095-019-0142-3

Stephen J Wright. 2015. Coordinate descent algorithms. *Mathematical Programming* 151, 1 (2015), 3–34.

Bing Xu, Junfei Zhang, Rui Wang, Kun Xu, Yong-Liang Yang, Chuan Li, and Rui Tang. 2019. Adversarial Monte Carlo Denoising with Conditioned Auxiliary Feature. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019)* 38, 6 (2019), 224:1–224:12.

Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. 2015. Fast 4D sheared filtering for interactive rendering of distribution effects. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 1–13.

Zheng Zeng, Lu Wang, Bei-Bei Wang, Chun-Meng Kang, and Yan-Ning Xu. 2020. Denoising Stochastic Progressive Photon Mapping Renderings Using a Multi-Residual Network. *Journal of Computer Science and Technology* 35, 3 (2020), 506–521. https://doi.org/10.1007/s11390-020-0264-1

Theingi Zin, Yusuke Nakahara, Takuro Yamaguchi, and Masaaki Ikehara. 2021. Improved image denoising via RAISR with fewer filters. *Computational Visual Media* (2021). https://doi.org/10.1007/s41095-021-0213-0

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. Wiley Online Library, 667–681.

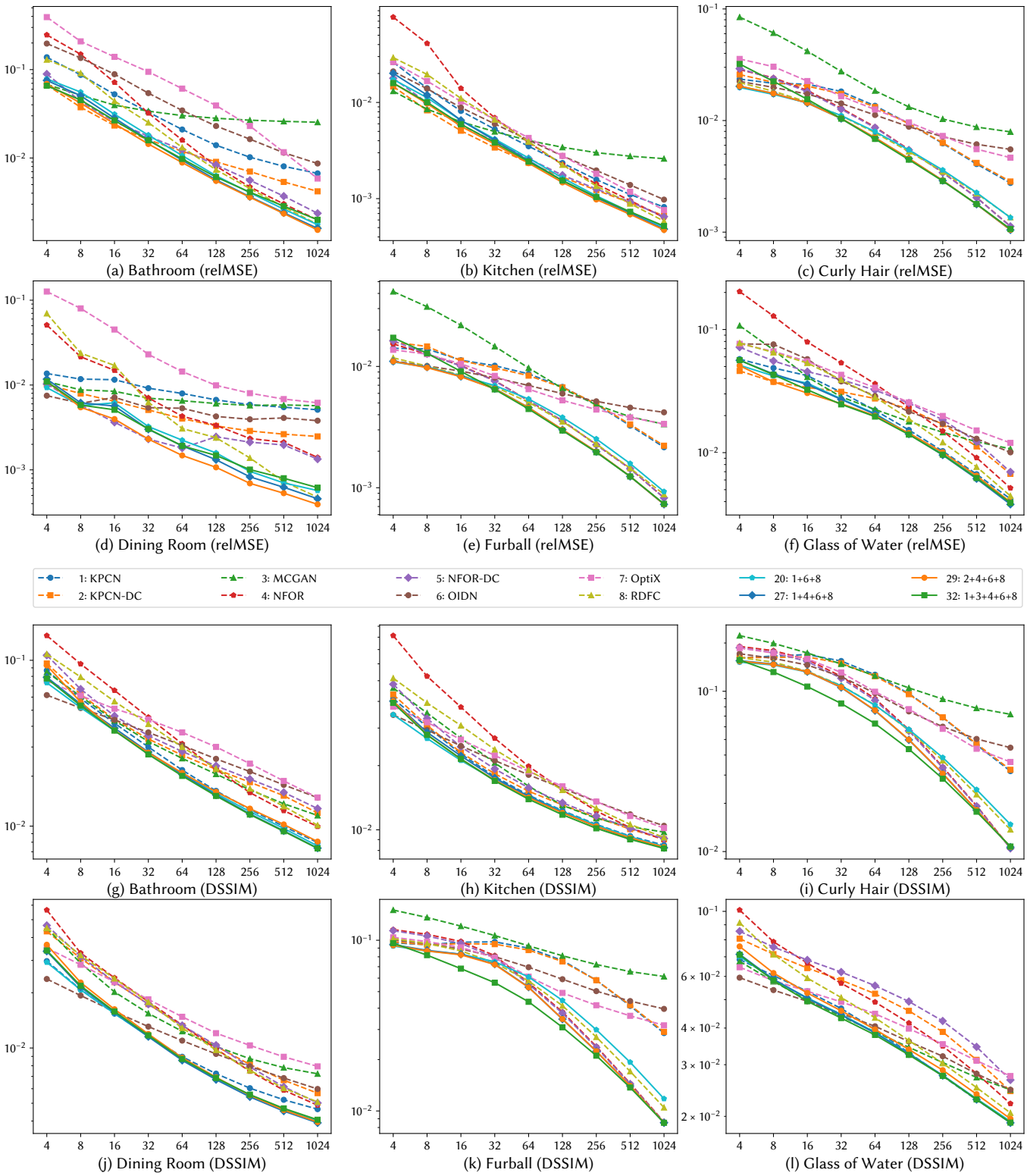

Fig. 13. Example error curves. We compute the relMSE and DSSIM of each method for each scene and sample rate. In most cases, the ensembles outperform the corresponding individual base denoisers.

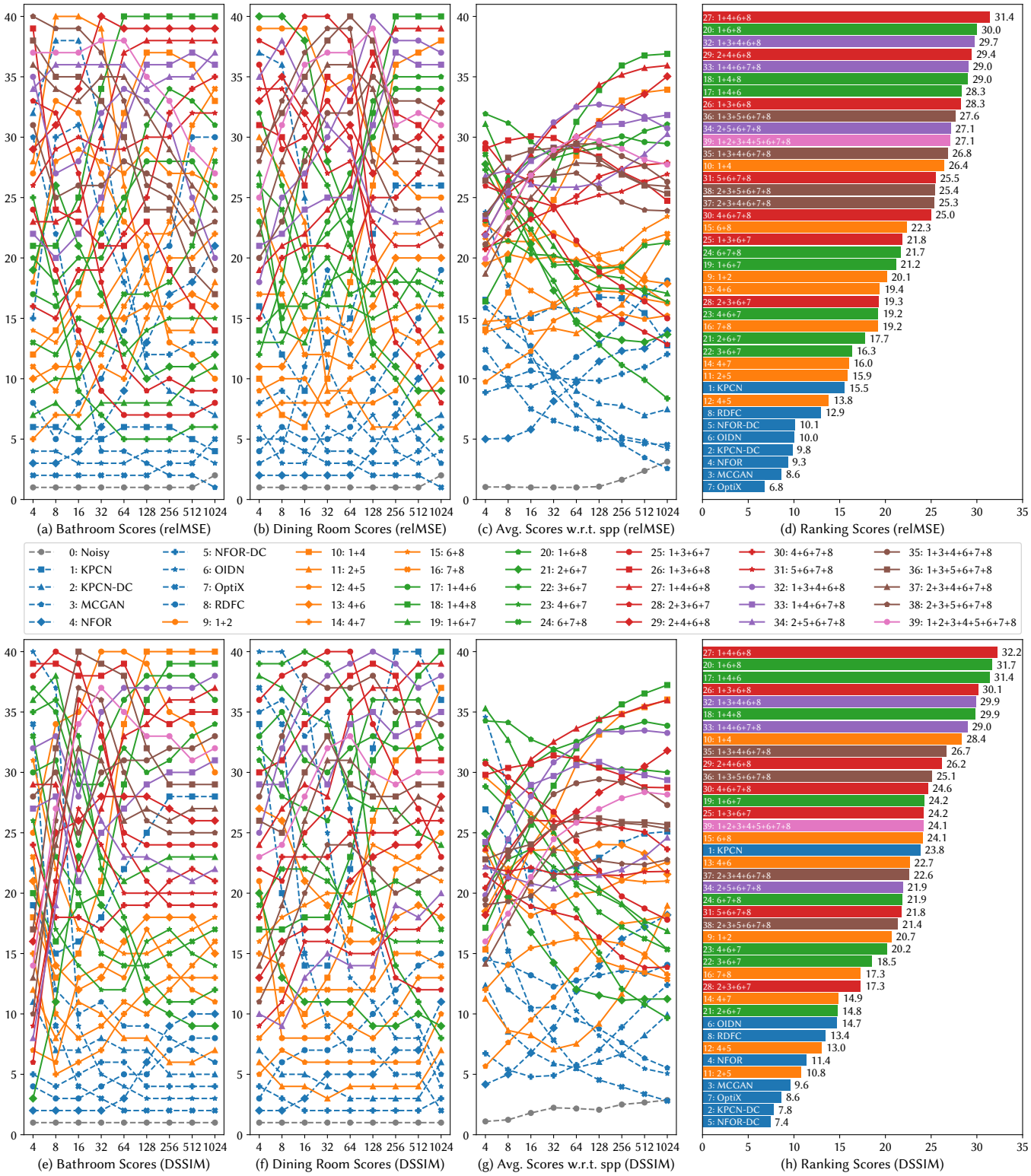

Fig. 14. Scores and rankings. We first compute scores based on the error metrics (relMSE and DSSIM) for each method in individual scenes w.r.t. spp. These scores are then averaged across scenes (c and g), and finally, we further average them across spps to get the overall ranking scores, based upon which we sort the methods and compute the rankings (d and h). The ensembles consistently outperform the individual base denoisers in different scenes and sample rates under both error metrics.

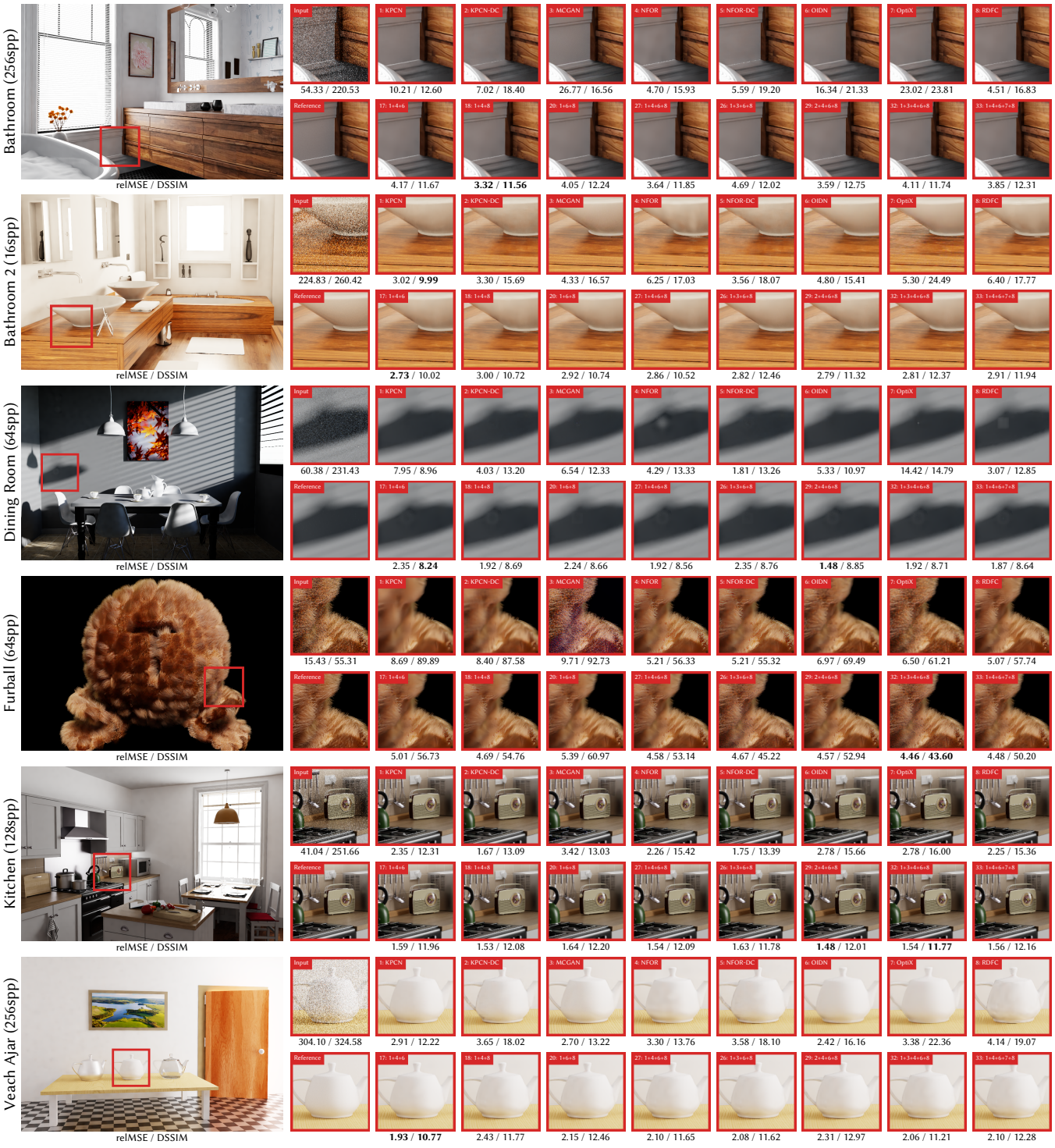

Fig. 15. Visual comparison with base denoisers. RelMSE and DSSIM (both scaled by 1000) for full images are listed below each tile. Our combined images leverage the power of the base denoisers and compensate for most of their artifacts, achieving the lowest error and the best visual quality almost consistently.