Filtering-Based Reconstruction for Gradient-Domain Rendering

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

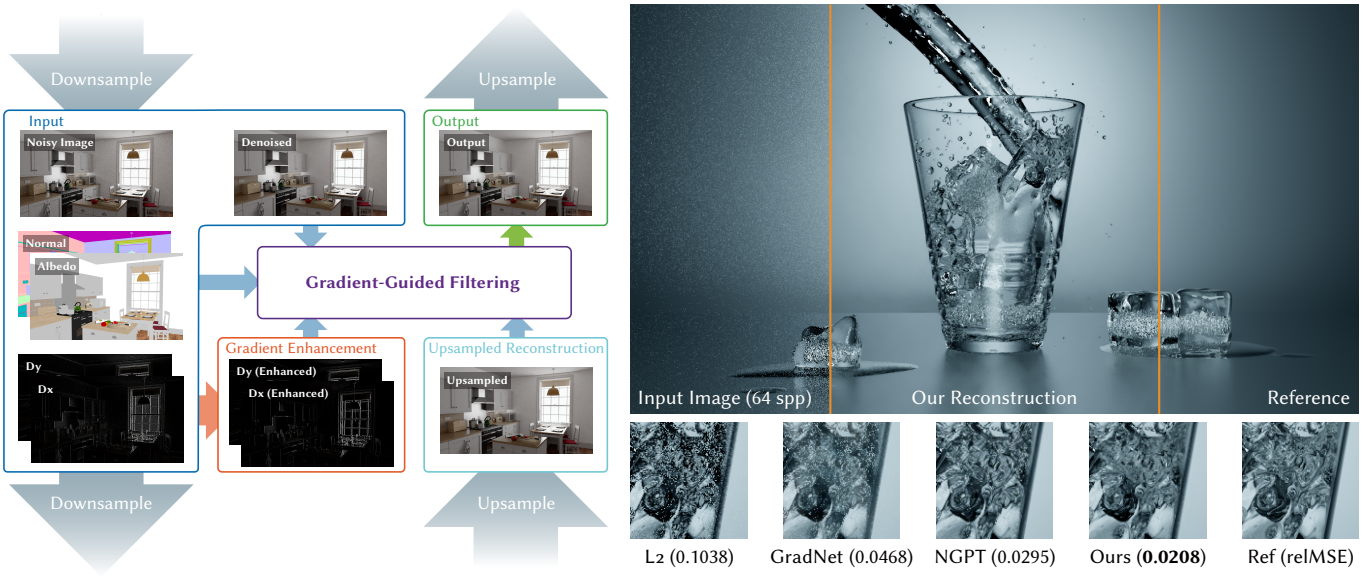

Fig. 1. We propose a filtering-based reconstruction algorithm for gradient-domain rendering. Utilizing gradient-domain information and a coarse-to-fine strategy, we could improve the reconstruction quality across a variety of scenes. Left: the pipeline of our method. Right: our reconstructed results compared to L_2 reconstruction, GradNet [Guo et al. 2019] and NGPT [Kettunen et al. 2019]. Below are close-up views and relMSE evaluations.

Gradient-domain rendering methods reconstruct color images based on the Poisson equation with gradients from correlated sampling. The relatively low variance in the gradient estimation facilitates convergence but the inevitable noises make the solving process prone to unpleasant spiky artifacts.

We propose a gradient-guided filtering approach[1] for reconstruction that avoids instability from directly using noisy gradients. Our method models the output color of each pixel as a weighted combination of its neighboring pixels, utilizing gradients as *guidance* to compute optimized filtering weights. The gradients are enhanced before being applied in the filtering process, and a coarse-to-fine strategy is used to leverage information from a larger scale.

Experiments show that our method achieves superior reconstruction quality for gradient-domain renderings compared to existing techniques. Furthermore, our method has two advantages: (1) it is not learning-based, making it more robust to unseen scenes without requiring extra training or fine-tuning; and (2) it is designed to be asymptotically unbiased.

*Kun Xu is the corresponding author.
[1]Our method is open-sourced at https://github.com/lastmc/FRGR.

SA Conference Papers '24, December 3–6, 2024, Tokyo, Japan

ACM ISBN 979-8-4007-1131-2/24/12.
https://doi.org/10.1145/3680528.3687568

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

Additional Key Words and Phrases: Gradient-Domain Rendering, Reconstruction, Optimization

ACM Reference Format:
Difei Yan, Shaokun Zheng, Ling-Qi Yan, and Kun Xu. 2024. Filtering-Based Reconstruction for Gradient-Domain Rendering. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3680528.3687568

1 INTRODUCTION

Gradient-domain rendering employs correlated sampling to estimate color gradients from differences between neighboring pixels, achieving reduced variance compared to independent sampling. Reconstruction techniques are subsequently applied to combine color and gradient-domain images into the final rendering, typically through solving the Poisson equation.

The underlying rationale is to leverage the relatively low-variance first-order estimates to regularize the estimation of the mean values, i.e., the pixel colors. However, the gradients rendered with Monte Carlo sampling are not perfectly noise-free, which can easily violate the reconstruction process, leading to unpleasant spiky artifacts.

On the other hand, filtering is a widely used noise-reduction approach in conventional Monte Carlo rendering. Sophisticated weighting heuristics are developed to efficiently combine the color estimates from similar pixels. However, typically utilizing only the image-domain information without the true gradients, filtering methods often suffer from blurry artifacts with overly mingled neighboring pixels.

We are thus motivated to find a novel reconstruction technique for gradient-domain rendering that bridges the strengths of both methods: preserving sharp edges with the gradients while suppressing noises and spikes with optimized filters. Instead of solving Poisson equations directly involving noisy gradients, we propose gradient-guided filtering for reconstruction. We model the output color of each pixel as a weighted combination of neighboring pixels, where the gradients are used as guidance in computing optimal filtering weights. The optimization is formulated as a convex quadratic problem and can be efficiently solved.

We enhance the gradients before gradient-guided filtering with an optimization goal involving three aspects: (1) valid gradients, (2) extreme values preserving, and (3) feature-aware. The enhanced gradients are demonstrated to provide better guidance compared to the input noisy gradients. Furthermore, a coarse-to-fine strategy is employed to utilize information from multiple scales. We extend *Guided Linear Upsampling* [Song et al. 2023] with auxiliary features to better maintain details and fine structures.

Experiments demonstrate that our method outperforms the state-of-the-art neural reconstruction approaches, GradNet [Guo et al. 2019] and NGPT [Kettunen et al. 2019], in both quantitative measures and visual quality (Fig. 1 and 11). Compared to traditional Poisson-solving approaches, our method is robust to noisy gradients and avoids spiky artifacts, which are a common issue in previous gradient-domain reconstruction methods. Compared to learning-based approaches which are data-dependent, our method does not require a preprocessing or training step and adapts to diverse scenes. Besides, our method is designed with asymptotic unbiasedness, a desirable property that theoretically ensures convergence.

2 RELATED WORK

2.1 Gradient-Domain Rendering

Gradients provide first-order information about an image. When used meticulously, they help distinguish sharp edges from smooth areas and are thus widespread in image processing [Bhat et al. 2010; Kazhdan and Hoppe 2008; Kou et al. 2015]. In Monte Carlo rendering, Lehtinen et al. [2013] pioneered the use of gradient-domain information for image reconstruction to overcome the uneven convergence in *Metropolis Light Sampling* [Veach and Guibas 1997]. They compute gradients by sampling pairs of paths and reconstruct the rendering by solving the Poisson equation. Soon their approach was improved [Manzi et al. 2014] and extended to other Monte Carlo rendering techniques, including unidirectional path tracing [Bauszat et al. 2017; Kettunen et al. 2015], bidirectional path tracing [Manzi et al. 2015], photon mapping [Hua et al. 2017], vertex connection and merging [Sun et al. 2017], etc. All follow the framework of correlated path/photon sampling and Poisson equation solving. The rendered gradients can also serve as guidance for adaptive rendering. Back et al. [2018] introduced an adaptive guiding technique utilizing features derived from gradients.

While correlated sampling reduces gradient estimation variance and the first-order information benefits edge preservation, equation-solving methods still suffer from the inevitable gradient-domain noise that leads to spiky artifacts in the color space. Reconstruction methods, such as Ha et al. [2019]; Manzi et al. [2016], have been developed to better combine color-space and gradient-space. However, their direct use of noisy gradients still results in spiky artifacts. For better visual quality, learning-based methods are introduced [Guo et al. 2019; Kettunen et al. 2019]. However, they typically require large datasets for training and can hardly deal with noise patterns not captured by the training distribution. Our method opts to apply reconstruction with enhanced gradients to achieve a more robust solution while remaining independent of large datasets.

2.2 Image-Space Reconstruction

Filtering is a common facility in image reconstruction. In Monte Carlo rendering, auxiliary features, such as albedo and normal, are available as a byproduct and extensively exploited in today's Monte Carlo denoisers. Representative filter-based techniques include *Joint Bilateral Filtering* (JBF) [Kopf et al. 2007], *Robust Denoising Using Feature and Color Information* (RDFC) [Rousselle et al. 2013], *Adaptive rendering based on weighted local regression* [Moon et al. 2014], and *Nonlinearly Weighted First-Order Regression for Denoising Monte Carlo Renderings* (NFOR) [Bitterli et al. 2016]. These methods utilize auxiliary buffers to aid in discerning the similarity of neighboring pixels and constructing robust weights for hand-designed filtering/regression models. With the prevalence of deep learning, *Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings* (KPCN) [Bako et al. 2017] predicts the filter weights with learned priors. *Monte Carlo Denoising via Auxiliary Feature Guided Self-Attention* (AFGSA) [Yu et al. 2021] introduced the attention mechanism to enlarge the receptive field. These data-driven techniques allow neural networks to learn prior knowledge and discover effective denoising strategies. Post-denoising is also considered important for further correction. Utilizing statistical techniques, the denoised images are processed to gain more advantages such as unbiasedness. Zheng et al. [2021] combined the outputs of multiple denoisers to obtain a better result, and Gu et al. [2022] combined biased denoised outputs with unbiased ones. Back et al. [2022] designed a self-supervised framework to boost supervised denoising. However, handcrafted or data-driven, the averaging nature of filters results in unavoidable over-blurry.

The distinct characteristics of gradient-domain and color-space methods indicate a chance of mutual complement when combined. We are thus inspired to propose the filtering-based reconstruction method for gradient-domain rendering.

2.3 Image Upsampling

In image processing, manipulating images at a low resolution and then upsampling them back to the original size is an effective strategy for reducing computation. However, the information loss in this procedure could not be ignored sometimes. Kopf et al. [2007]

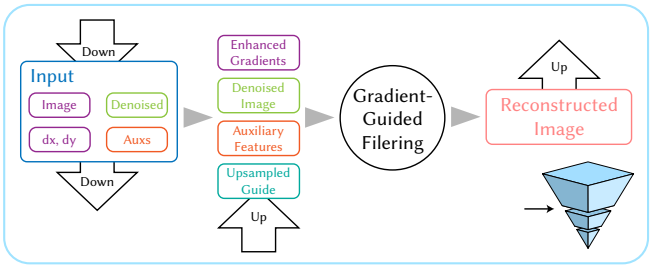

Fig. 2. The overall pipeline of our method leverages a coarse-to-fine strategy. At each level, input images are downsampled, filtered, and then upsampled to guide the finer level. The downsampling and upsampling processes use an algorithm similar to GLU [Song et al. 2023].

first introduced the concept of guided upsampling, where the pixels in the upsampled image are represented with a weighted sum of the low-resolution image. In their work, the weights are computed from bilateral weighting techniques [Tomasi and Manduchi 1998]. As a follow-up work, Song et al. [2023] proposed a pixel representation where two low-resolution pixels interpolate the output one and an optimization algorithm minimizes the reconstruction loss. We employ their technique to mitigate the loss of details during downsampling and upsampling.

3 OUR METHOD

Given a scene, we first use a gradient-domain path tracer to generate a noisy color image $\mathbf{I}_\mathrm{b}$ and two noisy gradient images $\mathbf{I}_\mathrm{dx}$ and $\mathbf{I}_\mathrm{dy}$ at a relatively low sample rate. Our method can be formulated as a function G that takes as input the rendered noisy color and gradient images ($\mathbf{I}_\mathrm{b}$, $\mathbf{I}_\mathrm{dx}$, and $\mathbf{I}_\mathrm{dy}$), together with some auxiliary feature images $\mathbf{F}$ (normal, albedo, etc.), and outputs a reconstructed image $\mathbf{I}$:

$$\mathbf{I} = G(\mathbf{I}_\mathrm{b}, \mathbf{I}_\mathrm{dx}, \mathbf{I}_\mathrm{dy}, \mathbf{F}). \tag{1}$$

To leverage information from multiple scales, we employ a *coarse-to-fine* strategy, which proceeds from the lowest (coarsest) level to the highest (finest) level. At each level, we first downsample the inputs from the higher level; then, we perform reconstruction; finally, we upsample the reconstructed image to guide the higher level. We adopt *Guided Linear Upsampling* (GLU) [Song et al. 2023] for both downsampling and upsampling to ensure quality (Sec. 3.1).

Specifically, two steps are performed at each level k ($1 \le k \le K$):

(1) *Gradient enhancement.* To preserve sharp edges while suppressing noise and spikes in the gradient images, we apply an enhancement step to the gradient images:

$$(\hat{\mathbf{I}}^k_\mathrm{dx}, \hat{\mathbf{I}}^k_\mathrm{dy}) = G_s(\mathbf{I}^k_\mathrm{dx}, \mathbf{I}^k_\mathrm{dy}, \mathbf{F}^k), \tag{2}$$

where the superscript $(\cdot)^k$ denotes the images at resolution level k. Details are provided in Sec. 3.2.

(2) *Gradient guided filtering.* The reconstructed image is computed through a filtering pass on the input noisy image. The filtering weights are obtained through an optimization process considering the smoothed gradients:

$$\mathbf{I}^k = G_f\left(\mathbf{I}^k_\mathrm{b}, \hat{\mathbf{I}}^k_\mathrm{dx}, \hat{\mathbf{I}}^k_\mathrm{dy}, \mathbf{F}^k, \mathrm{Up}(\mathbf{I}^{k-1})\right), \tag{3}$$

where $\mathrm{Up}(\cdot)$ denotes the upsampling operator. Details are provided in Sec. 3.3.

The final output reconstructed image is simply the output at the finest level, i.e., $\mathbf{I} = \mathbf{I}^K$. Fig. 2 illustrates the pipeline of our method.

To summarize the general flow of our method: We begin by generating multiple scales using the GLU method, along with noisy renderings. Starting from the lowest level (i.e., the smallest scale), we reconstruct the image using our filter-based algorithm. The reconstructed output is then upsampled using the appropriate strategy, serving as one of the auxiliary buffers for reconstructing the next level. By repeating this process, we ultimately reconstruct the image at its original resolution, which is the final output of our method.

3.1 Coarse-to-Fine Strategy

In our coarse-to-fine strategy, the input at each resolution level includes the noisy rendering, noisy gradients, and auxiliary features, all downsampled from the higher level. After the reconstruction process, the output will be upsampled and provided back to the previous level as a guidance image.

The output reconstructed image utilizes the low-variance guidance from the lower level and benefits from the wide perception field. However, details such as fine structures are often lost during downsampling. To mitigate the loss of spatial information, we adopt the *Guided Linear Upsampling* (GLU) method, which regresses the downsampling and upsampling weights iteratively to achieve minimum reconstruction loss. Also, we follow RDFC to pre-denoise the noisy images for more feasible inputs for GLU.

Below, we briefly review GLU in Sec. 3.1.1 and explain how it is used in our coarse-to-fine strategy in Sec. 3.1.2. The RDFC-based pre-denoising algorithm is described in Sec. 3.1.3.

3.1.1 Review of Guided Linear Upsampling (GLU). GLU does not use a simple predefined interpolation weighting scheme (e.g., bilinear interpolation) for downsampling and upsampling. Instead, given an input guidance image $\mathbf{J}$, the downsampling and upsampling weights are obtained by optimizing the reconstruction loss:

$$\min_{W_d, W_u} \|\mathbf{J} - \mathrm{Up}(\mathrm{Down}(\mathbf{J}, W_d), W_u)\|, \tag{4}$$

where W_d, W_u denote the weights (and parameters) used for downsampling and upsampling, respectively.

In the case of 2× downsampling, the value of each pixel in the downsampled image is interpolated from four neighboring pixels in the input image. The downsampling weights can be assigned in two ways: (1) all set to 1/4 for basic averaging; or (2) one set to 1 and others set to 0, to preserve detailed structures.

During 2× upsampling, each pixel in the upsampled image is reconstructed via linear interpolation of just two pixels within its small neighborhood (a 3×3 window) of the downsampled image.

All weights and parameters used in downsampling and upsampling are jointly optimized according to Eq. 4. Given the combinatorial nature of the optimization problem, which is challenging to solve directly, an iterative and efficient scheme is employed. For more details, please refer to the GLU paper [Song et al. 2023].

3.1.2 Downsampling and upsampling. Our coarse-to-fine strategy involves downsampling images from resolution level k to $k-1$

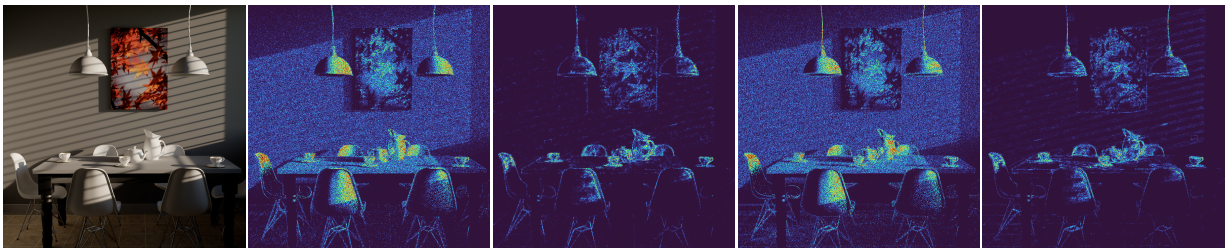

Fig. 3. A visualization of enhanced gradients and the input noisy gradients compared to the ground truth gradients. Above, we show their relMSE (relative mean square error) error maps. The scene in this figure is *Dining Room* sampled at 32 samples per pixel.

and then upsampling them back to level k after processing. For guidance, we use a multi-channel image $\mathbf{J}^k$, formed by stacking the pre-denoised image $\mathbf{I}_{\mathrm{d}}^k$ (as detailed in Sec. 3.1.3) and auxiliary features $\mathbf{F}^k$, so that $\mathbf{J}^k = [\mathbf{I}_{\mathrm{d}}^k, \mathbf{F}^k]$. In our implementation, $\mathbf{J}^k$ consists of 9 channels: 3 each for color, albedo, and normal. These channels are scaled by different coefficients to ensure matching magnitudes.

We then use the guidance image to compute the downsampling and upsampling weights using Eq. 4. We then apply these weights to downsample all inputs from level k to $k-1$, including the noisy rendering $\mathbf{I}_{\mathrm{b}}^k$, the noisy gradients $\mathbf{I}_{\mathrm{dx}}^k, \mathbf{I}_{\mathrm{dy}}^k$, the pre-denoised image $\mathbf{I}_{\mathrm{d}}^k$, and the auxiliary feature images $\mathbf{F}^k$. After completing the reconstruction process at level $k-1$, we upsample the reconstructed image $\mathbf{I}^{k-1}$ from level $k-1$ back to level k.

3.1.3 Pre-denoising algorithm. The noisy guidance image is unsuitable for calculating GLU weights because noise and outliers could be mistaken for fine structures and preserved at lower levels. Therefore, we pre-denoise it using RDFC [Rousselle et al. 2013], which jointly combines the *non-local means* (NLM) weights from the color image with the bilateral weights from the auxiliary feature buffers.

The per-pixel, per-channel color distance between two pixels, p and q, is defined as

$$\Delta^2(p,q) = \frac{(c(p)-c(q))^2 - (\mathrm{Var}[p] + \min(\mathrm{Var}[p], \mathrm{Var}[q]))}{10^{-10} + k_c^2(\mathrm{Var}[p] + \mathrm{Var}[q])}. \quad (5)$$

Based on this, we compute the NLM distance by averaging the pixel distances within the neighborhoods of p and q:

$$d_c^2(p,q) = \frac{1}{(2f+1)^2} \sum_{\|n\|_1 \le f} \Delta^2(p+n, q+n). \quad (6)$$

The bilateral feature distance is defined as

$$d_f^2(p,q) = \frac{(f(p)-f(q))^2 - (\mathrm{Var}_f[p] + \min(\mathrm{Var}[p], \mathrm{Var}[q]))}{k_f \max(10^{-10}, \max(\mathrm{Var}[p], \|\mathrm{Grad}[p]\|^2))}. \quad (7)$$

Here, $\mathrm{Var}[p]$ refers to the sample variance of the corresponding buffer, and $\mathrm{Grad}[p]$ refers to its gradient. The final filter weight is

$$\exp(-d_c^2 \prod_f d_f^2) \quad (8)$$

We first filter the image using three different sets of parameters for (f, k_c, k_f): $(1, 0.45, 0.6)$, $(3, 0.45, 0.6)$, and $(3, 10^{10}, 0.6)$. Then we estimate the SURE (Stein's Unbiased Risk Estimate) error of each filtered part and select the pixels with the lowest error from the three candidates to produce the final pre-denoised image.

3.2 Gradient Enhancement

The input gradients suffer from sampling noise in two major ways:

(1) Invalid gradients. Even for the same two pixels, inconsistent pixel differences are accumulated along different paths. Although the gradients rendered from Monte Carlo sampling are unbiased, their noise may disrupt the divergence-free property of image gradients.
(2) Spikes from outliers. Outliers in the gradient domain are reconstructed into spiky artifacts by the Poisson equation.

Therefore, it is necessary to enhance the gradients before image reconstruction. To better preserve the first-order information, three requirements are to be met:

(1) *Valid.* The processed gradient images should be divergence-free vector fields, ensuring that any path between the same two pixels sums to the same pixel difference.
(2) *Extreme value preserving.* Gradients of small and great magnitudes should be less changed because they typically correspond to the smooth regions and sharp edges, which are the most perceptually noticeable.
(3) *Feature-aware.* Auxiliary buffers (e.g., albedo and normal) should be utilized, as they provide additional information about geometry and shading and contain significantly less noise than the full rendering.

Considering these requirements, we formulate an optimization target for gradient enhancement with three components: the validity constraint, the color-guided regularization, and the feature-guided smoothing. Since gradient enhancement is applied at every resolution level k, we will omit the superscripts $(\cdot)^k$ (e.g., using $\mathbf{I}_{\mathrm{b}}$ instead of $\mathbf{I}_{\mathrm{b}}^k$) for simplicity in notation from now on.

3.2.1 Validity constraint. This constraint ensures that the sum of differences along any path between the same two pixels remains constant, effectively making the gradient vector field divergence-free. Let $\hat{\mathbf{I}}$ represent any color image, with $\mathbf{I}_{\mathrm{dx}}$ and $\mathbf{I}_{\mathrm{dy}}$ denoting its gradients along the x and y axes, respectively. For every pixel (i, j), the following equation holds:

$$\begin{aligned} \hat{\mathbf{I}}(i+1, j+1) - \hat{\mathbf{I}}(i, j) &= \hat{\mathbf{I}}_{\mathrm{dx}}(i, j) + \hat{\mathbf{I}}_{\mathrm{dy}}(i, j+1) \\ &= \hat{\mathbf{I}}_{\mathrm{dy}}(i, j) + \hat{\mathbf{I}}_{\mathrm{dx}}(i+1, j) \end{aligned} \quad (9)$$

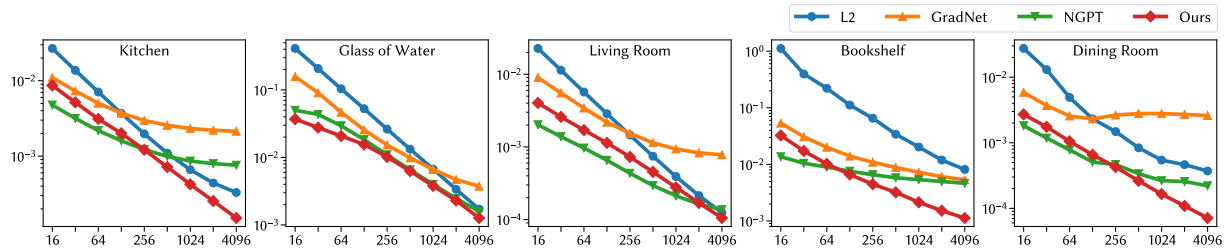

Fig. 4. Error curves of relMSE with respect to the number of spp (samples per pixel).

This equation represents the validity constraint. Gradients adhering to this constraint are considered valid, and from them, a color image can be uniquely reconstructed.

3.2.2 Color-guided regularization. To meet the second requirement, pixel-wise weights are applied to the differences between the input gradients ($\mathbf{I}_{\mathrm{dx}}$ and $\mathbf{I}_{\mathrm{dy}}$) and the enhancement targets ($\hat{\mathbf{I}}_{\mathrm{dx}}$ and $\hat{\mathbf{I}}_{\mathrm{dy}}$), controlling how much the original values are preserved. Specifically, the regularization loss is derived as follows:

$$\mathcal{L}_g = \sum_p \Big(m_x(p)(\hat{\mathbf{I}}_{\mathrm{dx}}(p) - \mathbf{I}_{\mathrm{dx}}(p))^2 + m_y(p)(\hat{\mathbf{I}}_{\mathrm{dy}}(p) - \mathbf{I}_{\mathrm{dy}}(p))^2 \Big). \tag{10}$$

This equation iterates over all pixels p. Per-pixel weights m_x and m_y are defined as

$$m_x(p) = \frac{|\mathbf{I}_{\mathrm{dx}}(p)|}{t} + \frac{1}{|\mathbf{I}_{\mathrm{dx}}(p)| + \epsilon}, \quad m_y(p) = \frac{|\mathbf{I}_{\mathrm{dy}}(p)|}{t} + \frac{1}{|\mathbf{I}_{\mathrm{dy}}(p)| + \epsilon}, \tag{11}$$

where t is set to a constant value of 0.09, controlling the minimum value of the weights and matching the magnitudes of other coefficients. ϵ, which is 0.01, is added to the denominator to avoid division by 0. The weights are large when the absolute values of the corresponding gradient are either great or small. This design is based on our observation that large and small gradient values typically correspond to sharp edges and smooth regions, which are the most visually noticeable structures and should be preserved. Any noise or outliers that might produce large gradient values will be suppressed by the feature-guided loss described in the next paragraph.

3.2.3 Feature-guided smoothing. Feature buffers, such as albedo and normal maps, often exhibit less noise and clearer edges, making them valuable for distinguishing between smooth regions and sharp boundaries. To mitigate jittering and enhance smoothness within flat areas, we propose a feature-guided loss that involves a pixel-wise weighting of the feature gradients. This loss is mathematically expressed as follows:

$$\mathcal{L}_f = \sum_p \Big(a_x(p)|\hat{\mathbf{I}}_{\mathrm{dx}}(p)|^2 + a_y(p)|\hat{\mathbf{I}}_{\mathrm{dy}}(p)|^2 \Big), \tag{12}$$

The per-pixel weights a_x and a_y are designed to be large when the corresponding gradients of feature buffers are small:

$$a_x(p) = \frac{1}{\sum_j |\mathbf{F}^{(j)}_{\mathrm{dx}}(p)| + \epsilon}, \quad a_y(p) = \frac{1}{\sum_j |\mathbf{F}^{(j)}_{\mathrm{dy}}(p)| + \epsilon}, \tag{13}$$

where $\mathbf{F}_{\mathrm{dx}}$ and $\mathbf{F}_{\mathrm{dy}}$ denote the screen-space gradients of the feature maps $\mathbf{F}$, and j traverses the feature map channels. ϵ is set to 0.01.

3.2.4 Final enhancement objective. In general, the color-guided term preserves both large and small gradient values, whereas the feature-guided term smooths gradients within regions indicated by feature buffers. By incorporating these losses alongside the hard validity constraint, we formulate the final objective $\mathcal{L}_{grad}$ to solve for the enhanced gradients $\hat{\mathbf{I}}_{\mathrm{dx}}$ and $\hat{\mathbf{I}}_{\mathrm{dy}}$:

$$\min \mathcal{L}_{grad} = \mathcal{L}_g + \lambda_f \mathcal{L}_f, \quad \text{s.t. Eq.(9) holds for all pixels.} \tag{14}$$

Here, $\lambda_f = 8/\sqrt{N}$ (where N represents the number of samples per pixel) is employed to balance the color- and feature-guided losses. Note that $\mathcal{L}_{grad}$ is a convex function. Thus, a gradient descent method is leveraged to solve the minimization problem.

Meanwhile, to enforce the validity constraint during optimization, we introduce a change of variable into the optimization objective:

$$\hat{\mathbf{I}}_{\mathrm{dx}}(i,j) = \hat{\mathbf{I}}(i+1,j) - \hat{\mathbf{I}}(i,j), \quad \hat{\mathbf{I}}_{\mathrm{dy}}(i,j) = \hat{\mathbf{I}}(i,j+1) - \hat{\mathbf{I}}(i,j). \tag{15}$$

By implementing this substitution, we shift the optimization focus to the reconstructed image. Consequently, the validity of the image gradients is inherently satisfied during the optimization process. We perform the gradient descent method on these convex losses to assure fast and global convergence. Specifically, we perform multiple one-dimension searches with random directions to decrease the losses. For more details, please refer to our source code.

Fig. 3 compares gradients before and after enhancement, showing that the enhanced gradients are less noisy and closer to the ground truth than the original ones.

3.2.5 Discussion of asymptotic convergence. While the term $\mathcal{L}_f$ introduces bias, its weight λ_f will decrease to zero as the sample rate N goes to infinity. Therefore, the enhanced gradients will asymptotically converge to the correct gradients.

3.3 Gradient-Guided Filtering

The enhanced gradients from the previous step exhibit low variance. We introduce a filtering-based method that utilizes these low-variance gradients for improved image reconstruction. Compared to traditional Poisson-based approaches, this method is more robust and effectively avoids the typical spiky artifacts. For simplicity in notation, the superscript $(\cdot)^k$ is also omitted in this subsection.

3.3.1 Formulation. The reconstructed color at pixel p is computed as the linear combination among $\mathcal{N}(p)$, the neighborhood of p (note

that $p \in \mathcal{N}(p)$). We have

$$\mathbf{I}(p) = \sum_{q \in \mathcal{N}(p)} w_q \mathbf{I}_\mathrm{b}(q), \quad \text{s.t. } w_q \geq 0, \sum_{q \in \mathcal{N}(p)} w_q = 1. \tag{16}$$

Next, we will detail how the filter weights w_q are computed by optimizing a combination of several losses.

3.3.2 Gradient-guided loss. This loss is defined by the expectation of the squared error between the filtered color $\mathbf{I}(p)$ and the noise-free ground-truth color μ_p at pixel p. We also denote the variance of the input noisy color $\mathbf{I}_\mathrm{b}(q)$ as v_p. The loss is formulated as

$$\begin{aligned}\mathcal{L}_\mathrm{opt} &= \mathbb{E}\left[(\mathbf{I}(p) - \mu_p)^2\right] = \left(\mathbb{E}\left[\mathbf{I}(p) - \mu_p\right]\right)^2 + \mathrm{Var}\left[\mathbf{I}(p) - \mu_p\right] \\ &= \left(\sum_{q \in \mathcal{N}(p)} w_q(\mu_q - \mu_p)\right)^2 + \sum_{q \in \mathcal{N}(p)} w_q^2 v_q\end{aligned} \tag{17}$$

Minimizing $\mathcal{L}_\mathrm{opt}$ leads to the optimal weights for filtering. The color variance v_p can be unbiasedly estimated by the sample variance of the input noisy colors $\mathbf{I}_\mathrm{b}(q)$. However, the true colors μ_p and μ_q are unknown in the equation, and their estimation via vanilla path tracing is typically very noisy. Fortunately, we can now derive a low-variance estimate of their differences $(\mu_q - \mu_p)$ using the enhanced gradients obtained in the previous step (Sec. 3.2).

Noticeably, with the validity constraints outlined in Section 3.2.1, this estimation becomes straightforward, using the pixel differences $\hat{\mathbf{I}}(p) - \hat{\mathbf{I}}(q)$ in the corresponding image of the enhanced gradients.

3.3.3 Auxiliary self-regression loss. To reduce noise and prevent blurring at object boundaries, we introduce a self-regression loss using auxiliary buffers, including albedo and normal. The loss for each auxiliary map, $\mathcal{L}_\mathrm{albedo}$ and $\mathcal{L}_\mathrm{normal}$, is defined as

$$\mathcal{L}_x = \left(\mathbf{F}_x(p) - \sum_{q \in \mathcal{N}(p)} w_q \mathbf{F}_x(q)\right)^2, \quad x \in \{\text{albedo}, \text{normal}\}. \tag{18}$$

3.3.4 Additional losses. Recall that we have two additional images at our disposal. One is the upsampled reconstructed image from the lower level, which we denote as $\mathbf{I}_\mathrm{up}$. Similar to the approach with auxiliary buffers, we introduce a self-regression loss for this upsampled reconstructed image:

$$\mathcal{L}_\mathrm{up} = \left(\mathbf{I}_\mathrm{up}(p) - \sum_{q \in \mathcal{N}(p)} w_q \mathbf{I}_\mathrm{up}(q)\right)^2. \tag{19}$$

The other is the pre-denoised color image $\mathbf{I}_\mathrm{d}$ (see Sec. 3.1.2), which is already reasonably clean. We define a loss to constrain the reconstructed color to be close to the pre-denoised color:

$$\mathcal{L}_\mathrm{d} = (\mathbf{I}_\mathrm{d}(p) - \mathbf{I}(p))^2 = \left(\mathbf{I}_\mathrm{d}(p) - \sum_{q \in \mathcal{N}(p)} w_q \mathbf{I}_\mathrm{b}(q)\right)^2. \tag{20}$$

3.3.5 The final combined loss. The final combined loss is a weighted sum of all previously defined losses, including the gradient-guided loss, the auxiliary self-regression loss, and additional losses from the lower-level reconstructed image and the pre-denoised image:

$$\mathcal{L} = \mathcal{L}_\mathrm{opt} + \lambda_\mathrm{a}\mathcal{L}_\mathrm{albedo} + \lambda_\mathrm{n}\mathcal{L}_\mathrm{normal} + \lambda_\mathrm{u}\mathcal{L}_\mathrm{up} + \lambda_\mathrm{d}\mathcal{L}_\mathrm{d}. \tag{21}$$

The filter weights w_q are computed by minimizing the final loss:

$$\underset{w_q}{\mathrm{argmin}}\, \mathcal{L}, \quad \text{s.t. } w_q \geq 0, \sum_{q \in \mathcal{N}(p)} w_q = 1. \tag{22}$$

Since the loss function is a convex quadratic form, the minimization can be treated as a quadratic programming problem and solved using the gradient descent method as mentioned in Sec. 3.2.4.

In our implementation, we use a 9×9 neighborhood size. Although minimization is required for every pixel, each pixel's computation is independent and involves only 81 unknown values. This allows for efficient parallel processing on a GPU.

The combination factors in the final loss are empirically set as $\lambda_\mathrm{a} = 16/\sqrt{N}$, $\lambda_\mathrm{n} = 0.16/\sqrt{N}$, $\lambda_\mathrm{u} = \lambda_\mathrm{d} = 8/\sqrt{N}$, where N denotes the number of samples per pixel. These factors are designed to ensure the asymptotic unbiasedness of our method: as the sample rate N goes to infinity, all loss terms except for $\mathcal{L}_\mathrm{opt}$ will diminish to zero.

4 EXPERIMENTS

Our method is implemented on a PC with an AMD Ryzen 9 7950X CPU and an NVIDIA 3080Ti GPU. We use the gradient-domain path tracing algorithm [Kettunen et al. 2015] implemented on LuisaRender [Zheng et al. 2022] to render the inputs, including the noisy color images, noisy gradient images, and auxiliary feature images. We use a fixed rendering resolution of 1280×720. Our method is efficient, taking about 5 seconds for the whole process of the reconstruction of one image (excluding rendering time).

4.1 Evaluation

In this section, we evaluate the effectiveness of the different components in our method.

4.1.1 The coarse-to-fine strategy. In Fig. 8, we evaluate our method using different numbers of resolution levels, from only 1 level to 4 levels. From the results, we find that using multiple levels improves the reconstruction quality compared to a single level. The optimal number of levels might vary with the scenes. For scenes with simple lighting and shading, such as the *Living Room*, using 2 levels is preferable. However, for more complex scenes like the *Bookshelf*, which have intricate lighting conditions, employing 4 levels produces better results. Overall, in our experiments, we fix the number of levels to 2 for its good general quality.

In Fig. 5, we evaluate the effectiveness of the *Guided Linear Upsampling* (GLU) method used in our coarse-to-fine strategy. In contrast, we use a simple bilinear downsampling/upsampling approach as a comparing baseline. From the results, we could find that GLU helps in preserving details and fine structures.

4.1.2 The overall loss $\mathcal{L}$ in gradient-guided filtering. Recall that the overall loss (Eq. 21) is defined as a weighted of the gradient-guided loss and several regression losses using the upsampled reconstructed image ($\mathcal{L}_\mathrm{up}$), the auxiliary maps ($\mathcal{L}_\mathrm{albedo}$ and $\mathcal{L}_\mathrm{normal}$), and the denoised image ($\mathcal{L}_\mathrm{d}$), respectively. We would like to evaluate the importance of those regression losses.

Removing the regression loss using the upsampled reconstructed image ($\mathcal{L}_\mathrm{up}$) is equivalent to setting the number of resolution levels to 1, i.e., not using the coarse-to-fine strategy at all. We have already

tested such settings in Fig. 8 and the results suggest the coarse-to-fine strategy helps in improving reconstruction qualities.

In Fig. 6, we show that removing the feature-guided losses ($\mathcal{L}_{\text{albedo}}$ and $\mathcal{L}_{\text{normal}}$) causes artifacts around textures and edges, highlighting their importance in preserving texture and geometry details.

In Fig. 7, we show that using a pre-denoised image as a regression loss ($\mathcal{L}_{\text{d}}$) significantly improves reconstruction quality.

4.1.3 Convergence. As explained in Sec. 3.3, our method produces asymptotically converged results. To validate this property, we measure how the relative MSE (relMSE) of our reconstructed images change with the sample rates on different scenes, as shown in Fig. 4. We also compare the relMSE of several other methods, including L_2 reconstruction, GradNet [Guo et al. 2019] and NGPT [Kettunen et al. 2019]. The results show that our method has a better convergence property compared to all those competitive methods.

4.2 Comparison

We have compared our method with several reconstruction methods for gradient-domain rendering, including the basic L_2 reconstruction methods by solving Poisson equations, and two state-of-art learning-based reconstruction methods, GradNet [Guo et al. 2019] and NGPT [Kettunen et al. 2019]. We also compared our method with non-gradient techniques including NFOR [Bitterli et al. 2016], AFGSA [Yu et al. 2021], and NPPD [Balint et al. 2023].

In our experiments, we compare our method with the pre-trained models provided by the authors of GradNet, NGPT, AFGSA, and NPPD. Since our method is not dependent on any specific dataset, we opted not to retrain or fine-tune the compared methods.

Six scenes from the Rendering Resources website [Bitterli 2016] and GradNet, including *Glass of Water*, *Kitchen*, *Living Room*, *Bathroom*, *Bookshelf*, and *Dining Room*, are tested. These scenes cover a variety of materials and lighting conditions. We report relative MSE (relMSE) for all reconstructed images. Different numbers of spps (i.e., from 32 spps to 128 spps) are tested. More results of comparisons can be found in the supplemental material.

4.2.1 Comparisons to gradient-domain approaches. As depicted in Fig. 11, the L_2 reconstructions show noticeable noise and spikes, while GradNet and NGPT effectively suppress most of these artifacts. In terms of reconstruction time, GradNet takes approximately 800ms, whereas NGPT takes around 600ms. However, GradNet struggles to eliminate outliers present in both color and gradient images, whereas NGPT introduces wave-like artifacts. In contrast, our method markedly outperforms GradNet and yields cleaner and smoother results than NGPT. Also, it is noticeable that on scenes out of NGPT's training set (e.g., *Bookshelf* and *Glass of Water*), our method achieves better results both numerically and visually, showing more robustness than learning-based methods.

We provide an indirect comparison with Manzi et al. [2016] on the reduction in RMSE over the L1 baseline due to the unavailability of the original implementation. In the Bookshelf scene, Manzi et al. [2016] reported an RMSE that is 21.3% of the L1 reconstruction's RMSE, while our method achieves an RMSE that is 11.7% of the L1 reconstruction's RMSE. This result suggests that our method offers a more substantial improvement in denoising performance.

4.2.2 Comparisons to non-gradient approaches. We also compare our method to image-space Monte Carlo denoising approaches, including NFOR, a filtered-based method, and AFGSA, a recent learning-based method. Since generating gradients requires additional time, our comparisons were conducted within the same time budget. To elaborate, our implementation of gradient path tracing takes slightly less than twice the time of path tracing with equivalent sample rates. Thus we provide NFOR and AFGSA with twice the samples of our inputs. Furthermore, the gradient path tracing algorithm yields pixel-correlated results that deviate from the assumptions of NFOR and lie outside the distribution of AFGSA's training set. Therefore, we provided noisy renderings generated by path tracing to these methods. NFOR takes around 5s for denoising, while AFGSA takes about 400ms.

As illustrated in Fig. 9, our method generally produces better results than NFOR and AFGSA. NFOR tends to blur the output at a low sample rate and occasionally fails due to the instability of first-order regression. On the other hand, AFGSA tends to blur the scene even at high sample rates. In contrast, our method surpasses them within the same rendering time budget.

Additionally, we compared our method to a state-of-the-art non-gradient neural denoising technique, NPPD [Balint et al. 2023]. NPPD requires every sample of a noisy input (e.g., 64 images for a 64spp input). Thus we reduced the sample rates to half of the previous comparisons in Fig. 9, as too many samples would cause NPPD to consume excessive GPU memory. In current settings, NPPD needs 1s to handle 32spp, but over 10s for 128spp due to memory exceeding. Since our experimental setup does not include temporal sequences, we used the first frame output of NPPD for comparison. As shown in Fig. 10, we conducted equal-time experiments on the same scenes as AFGSA and NFOR but with half the sample rate. Our method demonstrated better results compared to NPPD.

5 CONCLUSION

In this paper, we have proposed a filtering-based reconstruction method for gradient-domain rendering. The basic idea is to model each output pixel color as a weighted combination of noisy colors from neighboring pixels. The gradients are used to optimize the filtering weights. The optimization is formulated as a convex quadratic problem and could be efficiently solved. We also introduce a gradient enhancement step before gradient-guided filtering to obtain higher-quality gradients. Furthermore, a coarse-to-fine strategy is employed to utilize information from multiple scales. Experiments show that our method achieves the best reconstruction results for gradient-domain rendering compared to other methods.

Our method would perform less successfully at extremely low sample rates where the high variance might lead to less accurate estimates. Possible ways to alleviate this problem might be using a larger filter radius for filtering or using ensemble denoising [Zheng et al. 2021] to combine with other learning-based methods.

In recent years, we have seen a renaissance of path reuse techniques, such as ReSTIR [Bitterli et al. 2020], in which correlated sampling and path shifting techniques are also widely leveraged. It is worthwhile to investigate how to utilize our gradient-guided filtering idea in such techniques to further reduce noise.

ACKNOWLEDGMENTS

We would like to thank the reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (Project No. 62372257).

REFERENCES

Jonghee Back, Binh-Son Hua, Toshiya Hachisuka, and Bochang Moon. 2022. Self-supervised post-correction for Monte Carlo denoising. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–8.

Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2018. Feature Generation for Adaptive Gradient-Domain Path Tracing. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 65–74.

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 (2017), 97–1.

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*. 1–11.

Pablo Bauszat, Victor Petitjean, and Elmar Eisemann. 2017. Gradient-domain path reusing. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–9.

Pravin Bhat, C Lawrence Zitnick, Michael Cohen, and Brian Curless. 2010. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Transactions on Graphics (TOG)* 29, 2 (2010), 1–14.

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

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.

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 Transactions on Graphics (TOG)* 39, 4 (2020), 148–1.

Jeongmin Gu, Jose A Iglesias-Guitian, and Bochang Moon. 2022. Neural James-Stein combiner for unbiased and biased renderings. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–14.

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.

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

Binh-Son Hua, Adrien Gruson, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2017. Gradient-domain photon density estimation. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 31–38.

Michael Kazhdan and Hugues Hoppe. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Transactions on graphics (TOG)* 27, 3 (2008), 1–10.

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. Deep convolutional reconstruction for gradient-domain rendering. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

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

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

Fei Kou, Weihai Chen, Changyun Wen, and Zhengguo Li. 2015. Gradient domain guided image filtering. *IEEE Transactions on Image Processing* 24, 11 (2015), 4528–4539.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

Marco Manzi, Markus Kettunen, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. 2015. Gradient-Domain Bidirectional Path Tracing.. In *EGSR (EI&I)*. 65–74.

Marco Manzi, Fabrice Rousselle, Markus Kettunen, Jaakko Lehtinen, and Matthias Zwicker. 2014. Improved sampling for gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 1–12.

Marco Manzi, Delio Vicini, and Matthias Zwicker. 2016. Regularizing image reconstruction for gradient-domain rendering with feature patches. In *Computer graphics forum*, Vol. 35. Wiley Online Library, 263–273.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive rendering based on weighted local regression. *ACM Transactions on Graphics (TOG)* 33, 5 (2014), 1–14.

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.

Shuangbing Song, Fan Zhong, Tianju Wang, Xueying Qin, and Changhe Tu. 2023. Guided Linear Upsampling. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–12.

Weilun Sun, Xin Sun, Nathan A Carr, Derek Nowrouzezahrai, and Ravi Ramamoorthi. 2017. Gradient-Domain Vertex Connection and Merging.. In *EGSR (EI&I)*. 83–92.

Carlo Tomasi and Roberto Manduchi. 1998. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 839–846.

Eric Veach and Leonidas J Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 65–76.

Jiaqi Yu, Yongwei Nie, Chengjiang Long, Wenjun Xu, Qing Zhang, and Guiqing Li. 2021. Monte Carlo denoising via auxiliary feature guided self-attention. *ACM Trans. Graph.* 40, 6 (2021), 273–1.

Shaokun Zheng, Fengshi Zheng, Kun Xu, and Ling-Qi Yan. 2021. Ensemble Denoising for Monte Carlo Renderings. *ACM Transactions on Graphics* 40, 6, Article 274 (2021), 17 pages.

Shaokun Zheng, Zhiqian Zhou, Xin Chen, Difei Yan, Chuyan Zhang, Yuefeng Geng, Yan Gu, and Kun Xu. 2022. LuisaRender: A High-Performance Rendering Framework with Layered and Unified Interfaces on Stream Architectures. *ACM Trans. Graph.* 41, 6, Article 232 (nov 2022), 19 pages. https://doi.org/10.1145/3550454.3555463

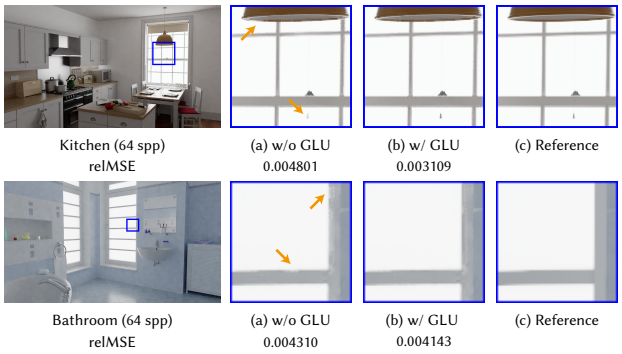

Fig. 5. Comparisons between using GLU and using simple bilinear downsampling/upsampling.

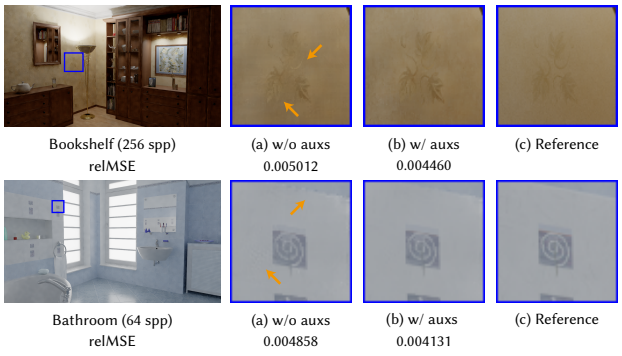

Fig. 6. Evaluation on the importance of the regression losses using the auxiliary features ($\mathcal{L}_{\text{albedo}}$, $\mathcal{L}_{\text{normal}}$).

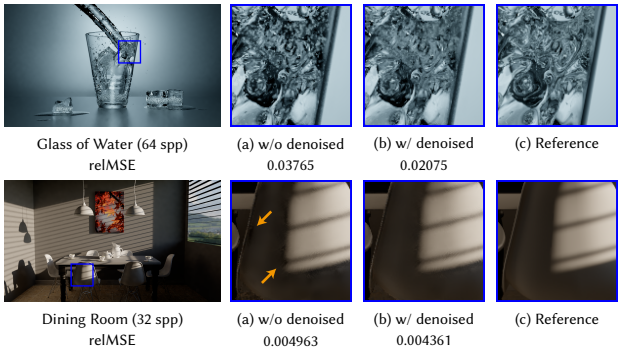

Fig. 7. Evaluation on the importance of the regression losses using the denoised image ($\mathcal{L}_{\text{d}}$).

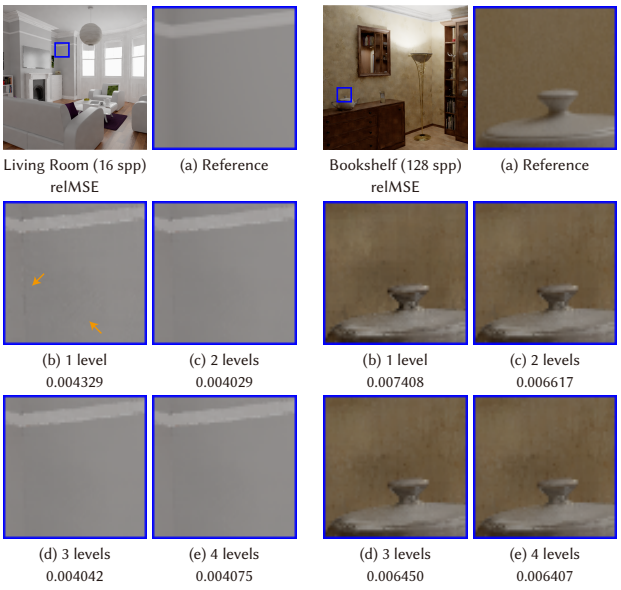

Fig. 8. Evaluation on the number of levels in the coarse-to-fine strategy.

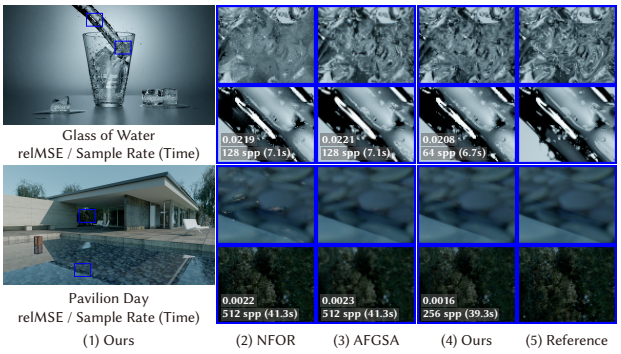

Fig. 9. Comparisons of our method with NFOR [Bitterli et al. 2016] and AFGSA [Yu et al. 2021]. We doubled the sample rates for NFOR and AFGSA to achieve roughly equal rendering times as ours. The rendering times are indicated in the figures. Remarkably, our method achieves better results within a shorter rendering time.

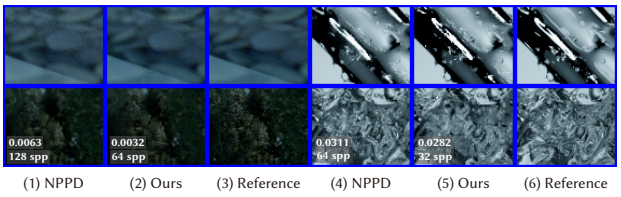

Fig. 10. Comparisons of our method with NPPD [Balint et al. 2023]. The same scenes as the previous comparisons are used and the rendering times are roughly equal. All sample rates are halved to reduce the GPU memory occupancy of NPPD. Our method achieves better results.

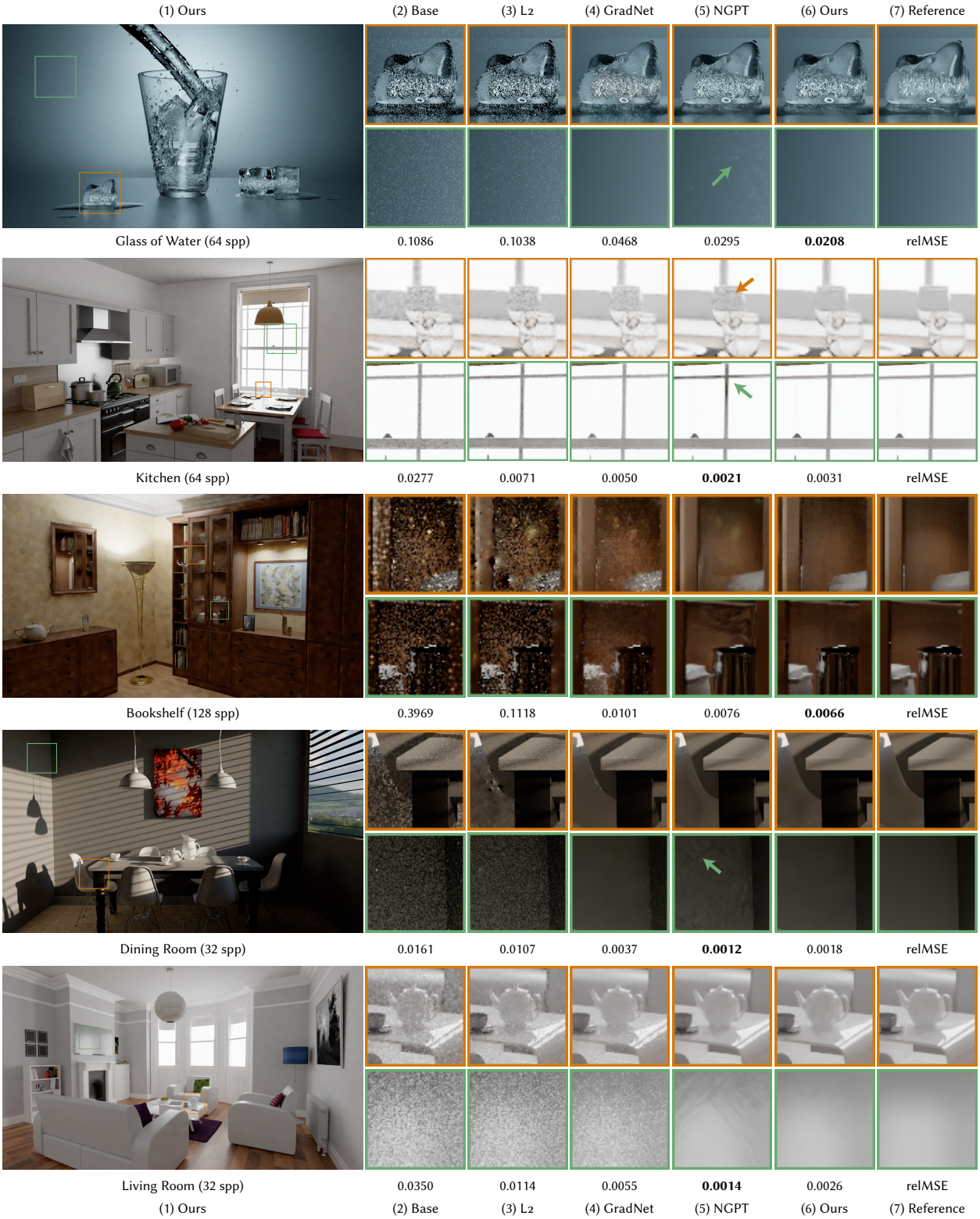

Fig. 11. Image reconstruction results comparing to several existing reconstruct methods: L_2 reconstruction, GradNet [Guo et al. 2019] and NGPT [Kettunen et al. 2019]. Note that the *Dining Room* and *Living Room* scenes are in the training set of NGPT. We apply relative MSE (relMSE) as the metric in comparisons. Obvious artifacts could be observed in the outputs of NGPT though their numerical metrics look better in some cases.