RESEARCH-ARTICLE

Target-Aware Image Denoising for Inverse Monte Carlo Rendering

JEONGMIN GU, Gwangju Institute of Science and Technology, Gwangju, South Korea

JONGHEE BACK, Gwangju Institute of Science and Technology, Gwangju, South Korea

SUNG-EUI YOON, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

BOCHANG MOON, Gwangju Institute of Science and Technology, Gwangju, South Korea

Target-Aware Image Denoising for Inverse Monte Carlo Rendering

JEONGMIN GU, Gwangju Institute of Science and Technology, South Korea
JONGHEE BACK, Gwangju Institute of Science and Technology, South Korea
SUNG-EUI YOON, KAIST, South Korea
BOCHANG MOON, Gwangju Institute of Science and Technology, South Korea

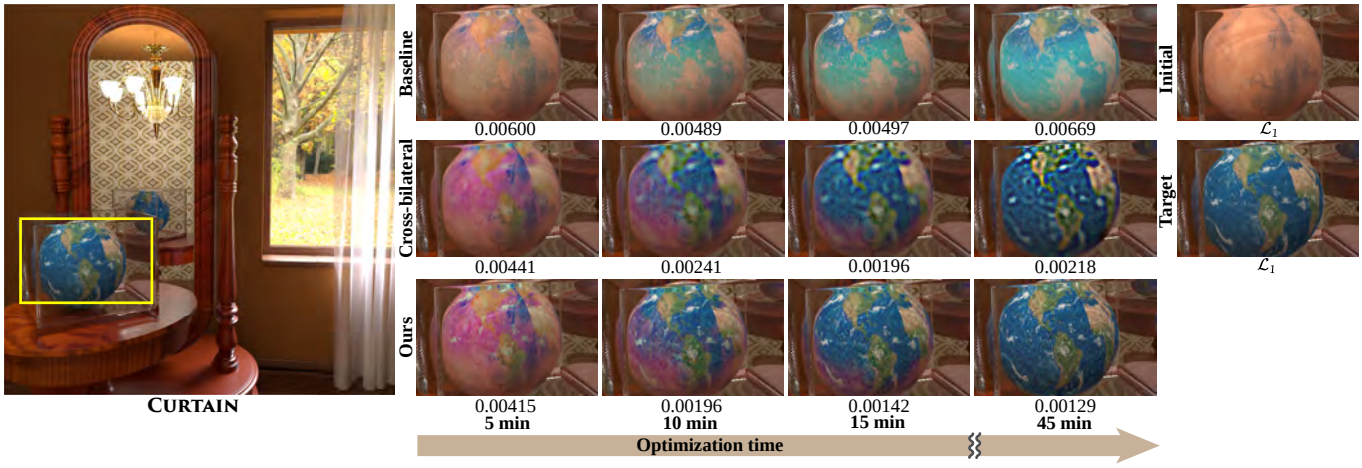

Fig. 1. Optimization results where we use a gradient-based optimizer that infers the scene parameters (i.e., textures within the yellow box) from its initial (a reddish one) so that the rendered image with the inferred textures is close to the target image. We compare the images rendered using the parameters inferred by the inverse rendering optimization without and with image denoising, i.e., the baseline and two denoisers (a cross-bilateral filter and our denoiser). Adopting an existing denoiser (i.e., the cross-bilateral filter) allows faster convergence than the baseline without image denoising, but the optimization goes into an undesirable local minimum, i.e., over-blurred textures. On the other hand, our image denoiser makes the scene inference robust by guiding the optimizer to preserve the texture details.

Physically based differentiable rendering allows an accurate light transport simulation to be differentiated with respect to the rendering input, i.e., scene parameters, and it enables inferring scene parameters from target images, e.g., photos or synthetic images, via an iterative optimization. However, this inverse Monte Carlo rendering inherits the fundamental problem of the Monte Carlo integration, i.e., noise, resulting in a slow optimization convergence. An appealing approach to addressing such noise is exploiting an image denoiser to improve optimization convergence. Unfortunately, the direct adoption of existing image denoisers designed for ordinary rendering scenarios can drive the optimization into undesirable local minima due to denoising bias. It motivates us to reformulate a new image denoiser specialized for inverse rendering. Unlike existing image denoisers, we conduct our denoising by considering the target images, i.e., specific information in inverse rendering. For our target-aware denoising, we determine our denoising weights via a linear regression technique using the target. We demonstrate that our denoiser enables inverse rendering optimization to infer scene parameters robustly through a diverse set of tests.

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

Additional Key Words and Phrases: linear regression, image denoising, differentiable rendering, inverse rendering

ACM Reference Format:
Jeongmin Gu, Jonghee Back, Sung-Eui Yoon, and Bochang Moon. 2024. Target-Aware Image Denoising for Inverse Monte Carlo Rendering. *ACM Trans. Graph.* 43, 4, Article 124 (July 2024), 11 pages. https://doi.org/10.1145/3658182

Authors' Contact Information: Jeongmin Gu, Gwangju Institute of Science and Technology, Gwangju, South Korea, jeong755@gm.gist.ac.kr; Jonghee Back, Gwangju Institute of Science and Technology, Gwangju, South Korea, jongheeback@gm.gist.ac.kr; Sung-Eui Yoon, KAIST, Deajeon, South Korea, sungeui@kaist.edu; Bochang Moon, Gwangju Institute of Science and Technology, Gwangju, South Korea, bmoon@gist.ac.kr.

ACM 1557-7368/2024/7-ART124
https://doi.org/10.1145/3658182

1 INTRODUCTION

Monte Carlo rendering, such as path tracing [Kajiya 1986], has become a standard solution for scenarios where photorealistic images are necessary since it enables an accurate simulation of various rendering effects from 3D virtual scenes. Recent breakthroughs [Li et al. 2018; Zhang et al. 2020, 2019] have shown that this physically based rendering can be differentiable with respect to its input, allowing for obtaining the derivatives of a rendered image with respect to scene parameters. This makes it possible to infer the parameters through a gradient-based optimization with a loss function, which measures a discrepancy between a rendered image and a target image, since the gradient of the loss with respect to scene parameters can be computed.

Such differentiable rendering provides a systematic and general way to create 3D content [Kato et al. 2020], but it shares the problem of Monte Carlo rendering, i.e., noise from random sampling, leading to noise in the gradients generated by differentiable rendering. We can reduce the noise by increasing the sample size for rendering, but unfortunately, taking a large sample size per optimization step is not practical as inverse rendering optimization usually takes many iteration steps [Balint et al. 2023; Chang et al. 2023; Hasselgren et al. 2022; Nicolet et al. 2023; Wang et al. 2023].

An attractive approach for reducing noise in the gradients is to adopt image denoising as a post-sampling process so that the gradients can be computed from denoised images instead of noisy images. For example, one can exploit an existing denoiser (e.g., a cross-bilateral filter [Li et al. 2012; Sen and Darabi 2012]) designed for ordinary forward rendering so that inverse rendering optimization exploits the gradients with reduced noise. Nonetheless, this direct use of an existing denoiser can let optimization reach an undesirable local minimum due to its inevitable side effect (i.e., denoising bias), as shown in Fig. 1.

In this paper, we present a reformulated image denoiser that aims to enhance the convergence of inverse rendering optimization while ameliorating the side effect of image denoising. We devise our denoiser using inverse rendering-specific information, i.e., a target image, instead of exploiting conventional information (G-buffers), i.e., a popular choice of existing denoisers. Our main contributions are as follows.

- We incorporate the inverse rendering-specific information, the target image, into image denoising for the first time to make inverse rendering optimization with denoising robust.
- We present a new linear regression using the target image to fulfill our high-level idea of adjusting denoising weights according to the pixel colors in the target image.

We demonstrate that our target-aware denoising can improve the convergence of inverse rendering optimization while preventing the optimization from converging to an undesirable local minimum through various tests.

2 RELATED WORK

Differentiable Monte Carlo rendering has received substantial attention recently as it allows for inferring various scene parameters through simulating light transport equations [Kato et al. 2020]. This section discusses previous studies related to our denoising.

Physically based differentiable rendering. Iterative optimization (e.g., stochastic gradient descent) for inverse rendering requires the differentiation of the light transport integrals with respect to scene parameters to be optimized. A major technical challenge is to differentiate light transport equations, including discontinuous terms (e.g., the visibility term), efficiently and accurately. Li et al. [2018] proposed a boundary sampling on silhouette edges and computed unbiasedly estimated derivatives for path tracing. Later, the reparameterizations of the light transport equation [Bangaru et al. 2020; Loubet et al. 2019] were proposed to estimate the derivatives more efficiently. In addition, differentiation techniques for more advanced light transport forms have been proposed, e.g., the differentiable radiative transfer equations for volumes [Nimier-David et al. 2022; Zhang et al. 2019, 2021] and the derivative estimation for the path integral form [Xu et al. 2023; Yu et al. 2022; Zhang et al. 2020].

Computing the gradient of a loss function with respect to scene parameters via backpropagation is often much more efficient than its counterpart (i.e., forward differentiation) for inverse rendering with a scalar loss function, and this backward differentiation was performed efficiently via splitting the differentiable rendering into two disjoint parts (i.e., the primal and adjoint rendering phases) [Nimier-David et al. 2020; Vicini et al. 2021]. We refer to a recent survey [Zeltner et al. 2021] that provides an in-depth discussion of various sampling and optimization techniques.

The differentiable rendering frameworks mentioned above rely on Monte Carlo sampling. Thus, those deliver noisy gradient estimates to an iterative optimizer unless we use a large sample size (and thus significant optimization times). Our image denoiser is a post-sampling process to reduce noise in the gradient estimates, which does not alter their sampling. In this paper, we show that our denoising can improve the convergence speed of inverse rendering optimization by plugging our method into a well-known differentiable rendering framework [Jakob et al. 2022; Vicini et al. 2021].

Variance reduction using spatio-temporal coherence. Variance reduction techniques allow noise reduction in the gradient of a loss function with respect to scene parameters without simply taking more samples. A recent approach was to use correlated sampling to reduce the gradient noise using temporal coherence across iterations in gradient descents. For example, Wang et al. [2023] and Chang et al. [2023] adapted path-reusing techniques (e.g., ReSTIR [Bitterli et al. 2020; Lin et al. 2022]) for inverse rendering optimization, and Nicolet et al. [2023] devised a recursive version of the image-space control variate [Rousselle et al. 2016] to reduce noise in the gradients using temporal coherence. Also, Balint et al. [2023] reused the gradients temporally using the finite differences in adjacent gradient steps.

An alternative approach for variance reduction is to adopt an image denoiser so that the gradient estimates can be computed from denoised images (not from noisy rendered images). For example, Hasselgren et al. [2022] showed that adopting an image denoiser for inverse rendering can be beneficial, especially with the small number of samples per gradient iteration.

Inspired by this observation, we investigate a robust way of exploiting image denoising for inverse rendering. However, unlike adopting existing denoisers, we reformulate image denoising using problem-specific information, i.e., a target image. We also show that our denoising is compatible with the existing variance reduction techniques using temporal coherence (e.g., [Chang et al. 2023; Nicolet et al. 2023]).

3 PROBLEM SPECIFICATION

Let us consider a loss function $\mathcal{L}$ that inverse rendering optimization aims to minimize:

$$\mathcal{L} = \frac{1}{m}\|f(\pi) - I\|_p^p, \tag{1}$$

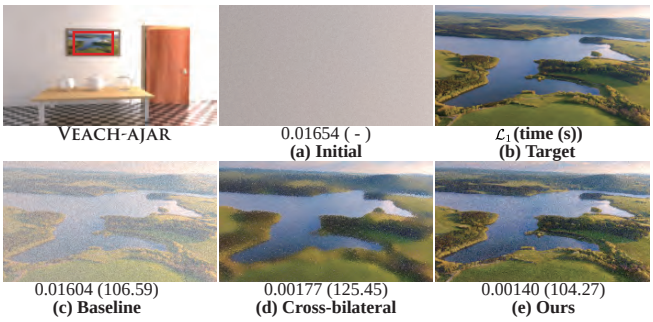

Fig. 2. Optimization results where we infer the textures (within the red box) from the initial guess (a constant texture (a)) so that rendering results using the inferred textures match the user-provided target image (b). We test a gradient-based optimizer that uses noisy gradients without denoising (the baseline (c)) and less noisy gradients with image denoising (a cross-bilateral filter (d) and ours (e)). Once their texture inferences are complete, we render the scene with their inferred parameters using a large sample count for the visualization. The baseline (c) does not effectively reduce the $\mathcal{L}_1$ errors (i.e., a similar error with the starting point (a)). The cross-bilateral filter (d) reduces the error much more than the baseline, but its output is severely blurred. On the other hand, ours (e) enables the optimizer to infer more accurate textures while keeping the details.

where $f(\pi)$ is the rendered image with scene parameters π and I is a user-provided target image. m is the number of elements in the color images $f(\pi)$ and I, i.e., the pixel count $\times$ 3. We refer to the $f(\pi)$ as an accurate image, e.g., a rendered image using unbiased rendering with an infinite sample size, which is unknown in practice. We set the order p to one (and thus $\mathcal{L}_1$ loss) unless otherwise mentioned. When a rendering function f becomes differentiable with respect to the parameters π, one can seek the optimal parameters π_{opt} that minimize the loss $\mathcal{L}$ via a gradient-based optimization since the gradient of the loss $\mathcal{L}$ with respect to the parameters can be obtained via a chain rule:

$$\frac{\partial \mathcal{L}}{\partial \pi} = \frac{\partial \mathcal{L}}{\partial f(\pi)} \frac{\partial f(\pi)}{\partial \pi}. \tag{2}$$

Such gradient-based optimization allows us to take any rendering function that is differentiable concerning the parameters. Still, we limit the choice of the rendering function to a physically based differentiable rendering. Specifically, one can replace the unknown image $f(\pi)$ (in Eq. 1) with a noisy estimate $\tilde{f}(\pi)$ rendered with a small number of samples, i.e., $\tilde{\mathcal{L}} = m^{-1}\|\tilde{f}(\pi) - I\|_p^p$, and perform the gradient-based optimization with the noisy image $\tilde{f}(\pi)$:

$$\frac{\partial \tilde{\mathcal{L}}}{\partial \pi} = \frac{\partial \tilde{\mathcal{L}}}{\partial \tilde{f}(\pi)} \frac{\partial \tilde{f}(\pi)}{\partial \pi}. \tag{3}$$

The two derivatives, $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ and $\partial\tilde{f}(\pi)/\partial\pi$, can be computed independently through two separate sampling processes [Nimier-David et al. 2020; Vicini et al. 2021].

While this general approach with differentiable Monte Carlo rendering provides a systematic means to optimize scene parameters, noise in the gradient $\partial\tilde{\mathcal{L}}/\partial\pi$ can slow the convergence toward the optimal parameters. A tempting approach for mitigating this problem is to reduce the noise in the image-space gradient $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ via image denoising so that the parameter-space gradient $\partial\tilde{\mathcal{L}}/\partial\pi$ can have reduced noise.

Specifically, let us consider an image denoiser that produces a denoised image $\hat{f}(\pi)$ from a noisy image $\tilde{f}(\pi)$ and set an alternative loss $\hat{\mathcal{L}} = m^{-1}\|\hat{f}(\pi) - I\|_p^p$ with the denoised image $\hat{f}(\pi)$, which is a less noisy estimate of the unknown $\mathcal{L}$ (Eq. 1). It results in a different chain rule:

$$\frac{\partial \hat{\mathcal{L}}}{\partial \pi} = \frac{\partial \hat{\mathcal{L}}}{\partial \hat{f}(\pi)} \frac{\partial \hat{f}(\pi)}{\partial \tilde{f}(\pi)} \frac{\partial \tilde{f}(\pi)}{\partial \pi} + \frac{\partial \hat{\mathcal{L}}}{\partial \hat{f}(\pi)} \frac{\partial \hat{f}(\pi)}{\partial g(\pi)} \frac{\partial g(\pi)}{\partial \pi}, \tag{4}$$

where $g(\pi)$ is an auxiliary input that a denoiser can take, e.g., G-buffers. Note that the second term on the right side of Eq. 4 becomes nonzero when the auxiliary input $g(\pi)$ depends on the learnable parameters π. An example is when we infer G-buffers (e.g., albedos) while employing an existing denoiser using such buffers simultaneously.

Nonetheless, we observed that including the second term can result in more noisy estimation results than the ones without the term when adopting existing G-buffer-based denoisers. Please see the supplemental report for this analysis. Therefore, we shall use only the first term for G-buffer-based denoisers unless otherwise mentioned. On the other hand, in Sec. 4, we will design our denoising without taking the learnable parameters as input to avoid this approximation of the chain rule.

One may consider any image denoiser that is differentiable with respect to its noisy input, but it can be preferable to use a simple denoiser for efficient computation of its denoising output $\hat{f}(\pi)$ and derivatives $\partial\hat{f}(\pi)/\partial\tilde{f}(\pi)$ since such computation should be conducted per optimization step.

As an example of such simple denoisers, we can think of using a cross-bilateral filter that produces a denoised color $\hat{f}_c(\pi)$ at pixel c by weight-averaging noisy colors $\tilde{f}_i(\pi)$ of pixel i within a denoising window Ω_c (e.g., 31 $\times$ 31 window) centered at pixel c:

$$\hat{f}_c(\pi) = \sum_{i \in \Omega_c} \left(w_{i|c}^{cb} \tilde{f}_i(\pi) \right) / \sum_{i \in \Omega_c} w_{i|c}^{cb}. \tag{5}$$

The denoising weight $w_{i|c}^{cb}$, assigned to a neighboring pixel i from the center pixel c, is computed as

$$w_{i|c}^{cb} = \exp\left(-\frac{\|q_i - q_c\|^2}{2(b^q)^2} - \frac{\|\rho_i - \rho_c\|^2}{2(b^\rho)^2} - \frac{\|n_i - n_c\|^2}{2(b^n)^2} - \frac{(d_i - d_c)^2}{2(b^d)^2} \right), \tag{6}$$

where b^q, b^ρ, b^n, and b^d are the bandwidth terms for the pixel positions (q_i and q_c), albedos (ρ_i and ρ_c), normals (n_i and n_c), and normalized depths (d_i and d_c) of pixel i and pixel c.

Adopting this cross-bilateral filter for inverse rendering is straightforward as differentiating this simple filter (Eqs. 5 and 6) with respect to its input $\tilde{f}(\pi)$ is trivial and it does not modify the underlying sampling processes, i.e., computing $\tilde{f}(\pi)$ and $\partial\tilde{f}(\pi)/\partial\pi$.

Challenges of image denoising and our motivation. Fig. 2 shows the results of a gradient-based optimization without and with image denoising, i.e., using the noisy gradient $\partial\tilde{\mathcal{L}}/\partial\pi$ (Eq. 3) or the less noisy gradient $\partial\hat{\mathcal{L}}/\partial\pi$ (Eq. 4). For the test of the cross-bilateral filter (Eqs. 5 and 6), we set the b^q to one-third of the half-width of the Ω_c, and the other terms for G-buffers are $b^\rho = 0.05$, $b^n = b^d = 0.1$.

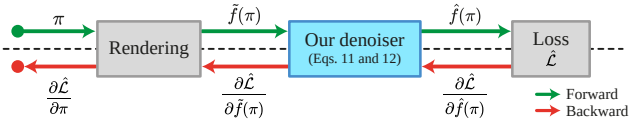

Fig. 3. The overview of an inverse rendering framework with our denoising. In the forward phase, denoted by green arrows, our denoiser takes a noisy image $\tilde{f}(\pi)$ rendered using scene parameters π and generates a denoised image $\hat{f}(\pi)$ via our target-aware weighting (Eq. 11) to compute the loss $\hat{\mathcal{L}} = m^{-1}\|\hat{f}(\pi) - I\|_p^p$. During the backward phase, marked by red arrows, the gradient of the loss with respect to the noisy image, $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$, is computed by locally averaging the $\partial\hat{\mathcal{L}}/\partial\hat{f}(\pi)$ with our weighting (Eq. 12). Finally, we multiply the $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ by the $\partial\tilde{f}(\pi)/\partial\pi$ (in Eq. 4), and then pass the resulting gradient $\partial\hat{\mathcal{L}}/\partial\pi$ to a gradient-based optimizer for updating the scene parameters π.

While the baseline without denoising computes its gradient $\partial\tilde{\mathcal{L}}/\partial\pi$ using an unbiased image $\tilde{f}(\pi)$, it shows a slow convergence (e.g., see the $\mathcal{L}_1$ errors of the baseline and the initial in Fig. 2). Using the cross-bilateral filter improves the convergence speed of the baseline, but it infers undesirable parameters, i.e., blurred textures. It motivates us to design a new image denoiser specialized for inverse rendering instead of the straightforward option of taking an existing denoiser, which will be presented in the subsequent section.

4 TARGET-AWARE IMAGE DENOISING

This section proposes how to compute the less noisy gradient $\partial\hat{\mathcal{L}}/\partial\pi$ (Eq. 4) with our image denoising. Fig. 3 shows the overview of an inverse rendering framework with a plug-in module, our denoising.

Our key idea is to replace conventional weighting (e.g., G-buffer-based weighting in Eq. 6) with a new weighting formed by considering the target image I. To fullfil this idea, we incorporate the inverse rendering-specific information I into our weighting via local regression [Loader 2006; Moon et al. 2014]. Specifically, we locally approximate the unknown image $f(\pi)$ with a linear function of the pixel colors in the target image I (i.e., a Taylor polynomial of degree one):

$$f_i(\pi) \approx f_c(\pi) + f_c'(\pi)(I_i - I_c), \tag{7}$$

where $f_c'(\pi)$ is the first derivative of the unknown image $f(\pi)$ with respect to the target I at a center pixel c. For brevity, we shall treat the colors, e.g., $f_i(\pi)$ and $f_c(\pi)$, as 1D values since we can apply our denoising to each color channel independently.

The unknowns $f_c(\pi)$ and $f_c'(\pi)$ in Eq. 7 can be estimated using a weighted least-squares objective function:

$$\begin{bmatrix}\hat{\alpha}_c\\ \hat{\beta}_c\end{bmatrix} = \underset{\alpha_c,\beta_c}{\operatorname{argmin}} \sum_{i\in\Omega_c} w_{i|c}\left\|\tilde{f}_i(\pi) - \alpha_c - \beta_c(I_i - I_c)\right\|^2, \tag{8}$$

where the outputs $\hat{\alpha}_c$ and $\hat{\beta}_c$ are the estimates of the unknowns $f_c(\pi)$ and $f_c'(\pi)$ in Eq. 7, respectively. The weight $w_{i|c}$ controls the relative importance of the squared error $\|\tilde{f}_i(\pi) - \alpha_c - \beta_c(I_i - I_c)\|^2$ at pixel i, and we define the weight using the target I:

$$w_{i|c} = \exp\left(-\frac{\|\log_e(I_i+1) - \log_e(I_c+1)\|^2}{2(b^I)^2}\right), \tag{9}$$

where the bandwidth b^I is set to 0.1. We exploit the log transformation for robust weighting since the target colors, I_i and I_c, are in HDR space. We then compute the optimal coefficients $\hat{\alpha}_c$ and $\hat{\beta}_c$ via the closed-form solution (i.e., normal equation) [Loader 2006]:

$$\begin{bmatrix}\hat{\alpha}_c\\ \hat{\beta}_c\end{bmatrix} = (X^TWX)^{-1}X^TWY, \tag{10}$$

where the i-th row in the design matrix X of size $|\Omega_c| \times 2$ is set to $[1, I_i - I_c]$ and the i-th element in the diagonal matrix W of size $|\Omega_c| \times |\Omega_c|$ is set to $w_{i|c}$ in Eq. 9. Also, the column vector Y of size $|\Omega_c|$ is set by the noisy input $\tilde{f}_i(\pi)$. Once we solve the normal equation (Eq. 10) per pixel c, we set our denoising output to $\hat{\alpha}_c$, i.e., $\hat{f}_c(\pi) = \hat{\alpha}_c$. We then compute the loss $\hat{\mathcal{L}} = m^{-1}\|\hat{f}(\pi) - I\|_p^p$ and its gradient with respect to the denoising output, $\partial\hat{\mathcal{L}}/\partial\hat{f}(\pi)$ (see Fig. 3).

Our next task is to compute the $\frac{\partial\hat{\mathcal{L}}}{\partial\tilde{f}(\pi)}$, i.e., $\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}(\pi)}\frac{\partial\hat{f}(\pi)}{\partial\tilde{f}(\pi)}$, and it requires to calculate the derivatives of the $\hat{f}(\pi)$ with respect to its noisy input $\tilde{f}(\pi)$, $\partial\hat{f}(\pi)/\partial\tilde{f}(\pi)$ (see Eq. 4). To make this differentiation trivial, let us represent our denoising as a linear smoother:

$$\hat{f}_c(\pi) = \hat{\alpha}_c = e_1^T(X^TWX)^{-1}X^TWY = \sum_{i\in\Omega_c} l_{i|c}\tilde{f}_i(\pi), \tag{11}$$

where $e_1^T = [1, 0]$. Note that our denoising is a linear combination of the noisy input values $\tilde{f}_i(\pi)$, like the cross-bilateral filter (Eq. 5). However, the difference is that the problem-specific information, the target I, controls our weighting $l_{i|c}$, unlike the existing weighting (Eq. 6). Also, our weighting is independent of scene parameters π; thus, we can safely ignore the second term $\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}(\pi)}\frac{\partial\hat{f}(\pi)}{\partial g(\pi)}\frac{\partial g(\pi)}{\partial\pi}$ on the right side in Eq. 4 without approximation.

Let us represent the $\partial\hat{\mathcal{L}}/\partial\tilde{f}_i(\pi)$ at pixel i using the linear smoother (Eq. 11) into

$$\frac{\partial\hat{\mathcal{L}}}{\partial\tilde{f}_i(\pi)} = \frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}(\pi)}\frac{\partial\hat{f}(\pi)}{\partial\tilde{f}_i(\pi)} = \sum_{c\in\Omega_i}\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}_c(\pi)}\frac{\partial\hat{f}_c(\pi)}{\partial\tilde{f}_i(\pi)} = \sum_{c\in\Omega_i}\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}_c(\pi)}l_{i|c}, \tag{12}$$

where the weight $l_{i|c}$ assigned to pixel i from pixel c locally averages the $\partial\hat{\mathcal{L}}/\partial\hat{f}_c(\pi)$. Note that the $l_{i|c}$ is used twice when applying a denoiser into inverse rendering, one for the denoised image $\hat{f}(\pi)$ (Eq. 11) and another for the gradient $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ (Eq. 12). Therefore, the weighting should mitigate blurring details in the two inputs, the noisy input ($\tilde{f}_i(\pi)$ in Eq. 11) and the derivative input ($\partial\hat{\mathcal{L}}/\partial\hat{f}_c(\pi)$ in Eq. 12), to alleviate the bias of the $\partial\hat{\mathcal{L}}/\partial\tilde{f}_i(\pi)$ (Eq. 12). We will analyze our denoising bias in the subsequent sections.

Our final step is to compute the $\partial\hat{\mathcal{L}}/\partial\pi$ (in Fig. 3) and pass it to a gradient-based optimizer for updating scene parameters π. To this end, we multiply the computed $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ (Eq. 12) by the $\partial\tilde{f}(\pi)/\partial\pi$ (in Eq. 4), which is independent of image denoising. This process is conducted per gradient step.

4.1 Bias analysis of our denoised image

Let us suppose the unknown function $f(\pi)$ is twice differentiable, allowing for expressing the Taylor polynomial of degree two:

$$f_i(\pi) \approx f_c(\pi) + f_c'(\pi)(I_i - I_c) + \frac{f_c''(\pi)}{2}(I_i - I_c)^2. \tag{13}$$

Then, our denoising bias $E[\hat{f}_c(\pi)] - f_c(\pi)$ can be derived as

$$\begin{aligned} E[\hat{f}_c(\pi)] - f_c(\pi) &= \sum_{i \in \Omega_c} l_{i|c} E[\tilde{f}_i(\pi)] - f_c(\pi) \\ &= \sum_{i \in \Omega_c} l_{i|c} f_i(\pi) - f_c(\pi). \end{aligned} \tag{14}$$

Note that the $l_{i|c}$ is independent of the noisy input $\tilde{f}(\pi)$ and is fixed unless we change the target image I during optimization, i.e., $E[l_{i|c}] = l_{i|c}$. Plugging the Taylor expansion (Eq. 13) into this exact bias equation (Eq. 14) leads to an approximate bias:

$$\begin{aligned} E[\hat{f}_c(\pi)] - f_c(\pi) \approx f_c(\pi) \sum_{i \in \Omega_c} l_{i|c} + f_c'(\pi) \sum_{i \in \Omega_c} l_{i|c}(I_i - I_c) \\ + \frac{f_c''(\pi)}{2} \sum_{i \in \Omega_c} l_{i|c}(I_i - I_c)^2 - f_c(\pi). \end{aligned} \tag{15}$$

As we choose a linear regression for our denoising, $\sum_{i \in \Omega_c} l_{i|c} = 1$ and $\sum_{i \in \Omega_c} l_{i|c}(I_i - I_c) = 0$ [Loader 2006]. Hence, our denoising bias can be compactly represented into

$$E[\hat{f}_c(\pi)] - f_c(\pi) \approx \frac{f_c''(\pi)}{2} \sum_{i \in \Omega_c} l_{i|c}(I_i - I_c)^2, \tag{16}$$

which provides an intuition on our denoising bias that is not related to the first derivative $f_c'(\pi)$ as we produce a denoising output that linearly correlates with the target, i.e., a linear function of the target. Also, the temporal change of our denoising bias during optimization depends only on the second derivative $f_c''(\pi)$ as our denoising weight $l_{i|c}$ is fixed over time. Additionally, our denoising bias decreases when the optimized scene parameters π become close to the ideal ones π_{opt} that match the unknown image $f(\pi)$ to the target I, i.e., $f(\pi_{opt}) = I$.

Importance of the linear regression. A different way to fulfill our high-level idea, i.e., exploiting the target image, is to use the target-based weighting $w_{i|c}$ (Eq. 9) directly without the linear regression, i.e., $\hat{f}_c(\pi) = \sum_{i \in \Omega_c}(w_{i|c}\tilde{f}_i(\pi))/\sum_{i \in \Omega_c} w_{i|c}$. It corresponds to a zeroth-order regression that approximates the unknown $f(\pi)$ with constant models, as discussed in [Bitterli et al. 2016]. Unlike the linear regression, its bias depends on the first derivatives (and also the second derivatives) since constant functions cannot approximate linear functions without errors. Fig. 4 shows the comparisons between the constant and linear models, which take advantage of the target image I differently. As shown in the figure, our chosen design, the linear regression, makes the optimization more accurate thanks to its smaller bias than the alternative.

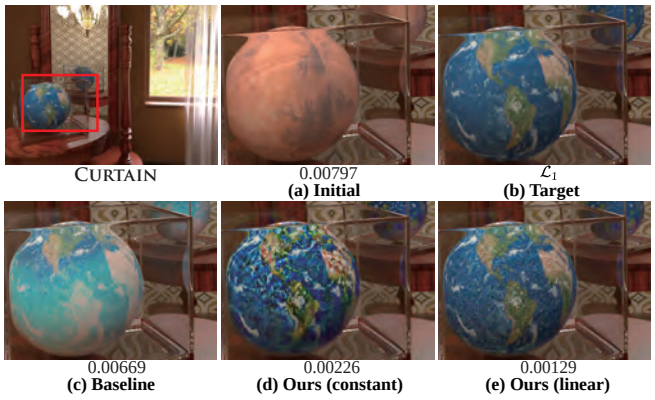

Fig. 4. Texture optimization results of our denoiser that approximates the unknown image as either constant (d) or linear functions (e) of the target image (b). Both options, which fulfill our high-level idea of adjusting the denoising weights by taking account of the target, produce better results than the baseline (c) without denoising. Nevertheless, it is noticeable that our chosen option (e), the linear approximation, provides more accurate inferences than the constant approximation (d), thanks to its smaller denoising bias.

4.2 Bias analysis of our image-space gradient

The bias of our image-space gradient $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ at pixel i, i.e., $\partial\hat{\mathcal{L}}/\partial\tilde{f}_i(\pi)$ in Eq. 12, can be represented into

$$\begin{aligned} E\left[\frac{\partial\hat{\mathcal{L}}}{\partial\tilde{f}_i(\pi)}\right] - \frac{\partial\mathcal{L}}{\partial f_i(\pi)} &= \sum_{c \in \Omega_i} E\left[\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}_c(\pi)}\frac{\partial\hat{f}_c(\pi)}{\partial\tilde{f}_i(\pi)}\right] - \frac{\partial\mathcal{L}}{\partial f_i(\pi)} \\ &= \sum_{c \in \Omega_i} E\left[\frac{\partial\hat{\mathcal{L}}}{\partial\hat{f}_c(\pi)}\right] l_{i|c} - \frac{\partial\mathcal{L}}{\partial f_i(\pi)}. \end{aligned} \tag{17}$$

Note that one can derive the bias of other denoisers with a linear smoother form (e.g., the cross-bilateral filter) analogously.

The equation above indicates that the denoising weight $l_{i|c}$, constructed for a denoising output $\hat{f}(\pi)$, also affects the bias of the derivative $\partial\hat{\mathcal{L}}/\partial\tilde{f}_i(\pi)$ as it is the weighted average of the $\partial\hat{\mathcal{L}}/\partial\hat{f}_c(\pi)$ of pixel c ($c \in \Omega_i$). Hence, it is desirable to keep the details in the $\partial\hat{\mathcal{L}}/\partial\hat{f}_c(\pi)$ so that the bias (Eq. 17) becomes small.

An existing denoiser can effectively estimate the unknown image $f(\pi)$ when considering rendering-specific features (i.e., G-buffers). However, its weighting can become less effective (i.e., a large denoising bias) for the gradient $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ when the G-buffers do not capture the details in the $\partial\hat{\mathcal{L}}/\partial\hat{f}(\pi)$. Note that the $\partial\hat{\mathcal{L}}/\partial\hat{f}(\pi)$ can inherit high-frequency information from the target I due to $\hat{\mathcal{L}} = m^{-1}\|\hat{f}(\pi) - I\|_p^p$. On the other hand, our weighting, which takes advantage of the target image, enables us to conduct an edge-avoiding smoothing for the image-space gradient $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$, as shown in Fig. 5.

5 RESULTS AND DISCUSSION

In this section, we verify our denoising by applying our denoising to the path replay backpropagation framework [Vicini et al. 2021]

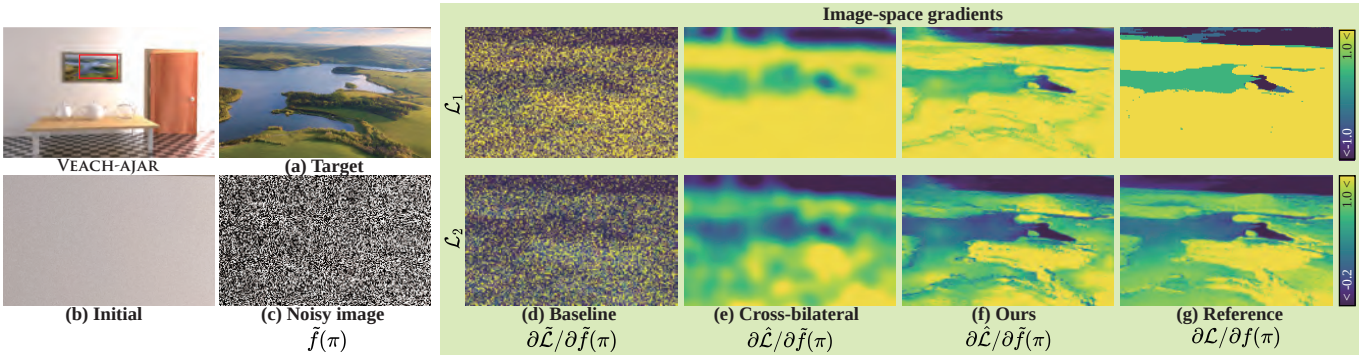

Fig. 5. Comparisons of the image-space gradients, i.e., $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ for the baseline (d) and $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ for the denoisers ((e) and (f)), for the first optimization iteration with the initial scene parameters π (a constant texture (b)) and the target I (the landscape texture (a)). We visualize their gradients when we use the $\mathcal{L}_1$ (in the top row) or $\mathcal{L}_2$ loss (in the bottom). To visualize the gradients clearly, we multiply those by m (in Eq. 1). We use 16 samples to generate the noisy image $\tilde{f}(\pi)$ (c), and the baseline and two denoisers produce their gradients using the image $\tilde{f}(\pi)$ without denoising and with denoising (the cross-bilateral filter and ours), respectively. We use 131K samples for the reference gradient $\partial\mathcal{L}/\partial f(\pi)$ (g). While the cross-bilateral (e) reduces noise in its gradient compared to the baseline (d), it blurs the details severely. Note that G-buffers used for its weighting cannot capture the high-frequency information in the reference (g) since the details come from the target (a) (not from the current scene parameters (b)). This excessive blur can drive the optimization incorrectly (Fig. 2 (d)). On the other hand, our method (f) keeps the details better than the cross-bilateral filter while producing much less noise than the baseline. It results in a more accurate scene inference (Fig. 2 (e)).

in Mitsuba 3 [Jakob et al. 2022]. This general differentiable framework enables various scene optimizations, including volumes via the differentiable ratio tracking [Nimier-David et al. 2022] and geometries via the reparameterization [Loubet et al. 2019]. We have used Adam [Kingma and Ba 2014] as the optimizer when inferring materials and volumes, and used its variant [Nicolet et al. 2021] for optimizing geometries. We have conducted all tests using an Nvidia RTX 3090 GPU.

5.1 Applications

In Fig. 7, we apply our denoising for various inverse rendering tasks and compare optimization results without and with our denoising. The baseline without denoising uses the noisy gradient $\partial\tilde{\mathcal{L}}/\partial\pi$ (Eq. 3), and ours employs the less noisy gradient $\partial\hat{\mathcal{L}}/\partial\pi$ (Eq. 4). We conduct same-time comparisons with the baseline. Specifically, the standalone time of our method (Eqs. 11 and 12) is less than 0.01 seconds per iteration, given a 1K × 1K image. This overhead can be considered small but nonzero; thus, we allocate slightly more iterations to the baseline.

Materials. For the CURTAIN and VEACH-AJAR scenes, we optimize the albedo textures from the initial parameters (a reddish one for the CURTAIN and a constant (set by 0.5) for the VEACH-AJAR). We set the sample count for the rendering to 128 for the CURTAIN and 16 for the VEACH-AJAR, respectively. As shown in Fig. 7, the training convergence speed of the baseline is very slow (see Fig. 7 (e)), and it makes their output (Fig. 7 (b)) far from the target (Fig. 7 (d)). Our method helps the optimization produce a more accurate inference of the material parameters than the baseline, e.g., 11.46× lower $\mathcal{L}_1$ error for the VEACH-AJAR scene.

Volumes. For the JANGA and DUST DEVIL scenes in the third and fourth rows of Fig. 7, we infer heterogeneous volumes with spatially varying coefficients (albedos and densities). We use 4 samples for the JANGA scene and 16 samples for the DUST DEVIL, respectively. The optimization without denoising (i.e., the baseline in Fig. 7 (b)) shows an effective error reduction compared to the starting point (i.e., the initial in Fig. 7 (a)) but still misses some fine details in the target. Our denoising makes this optimization more effective while restoring the details, resulting in much lower $\mathcal{L}_1$ errors, e.g., the 3.96× smaller error for the JANGA scene.

Geometries. In the fifth and sixth rows of Fig. 7 (the BUNNY and NEFERTITI scenes), we iteratively optimize the geometries from an initial guess (a sphere in Fig. 7 (a)). We measure the Hausdorff distances between inferred geometries and the target geometries to compare the accuracy of the geometries inferred by the baseline and ours. We set the sample size to 4. The baseline misses the high-frequency details in the targets, but our denoising enables us to recover such details more accurately than the baseline.

5.2 Comparisons and analysis

Comparisons with other denoisers. The alternative to our approach is adopting an existing denoiser without any problem-specific adaptation. The two classical denoisers related to our method are the cross-bilateral filter (Eqs. 5 and 6) and a linear regression with G-buffers [Moon et al. 2014]. The G-buffer-based linear regression approximates the unknown with a linear function of their features (G-buffers). It relies on the local regression theory like ours. Thus, we can estimate its linear function (and its denoised image) using the normal equation (Eq. 10), i.e., the same process as ours but with different features (G-buffers for the existing approach and the target for ours). Also, we assign the weight in its least-squares objective function (e.g., $w_{i|c}$ in Eq. 8) by the cross-bilateral weight $w_{i|c}^{cb}$ (Eq. 6).

We also test a neural denoiser, the Open Image Denoise (OIDN) [Áfra 2023]. A subtlety for using such a neural denoiser in inverse rendering is that its full pipeline should be differentiable, i.e., from a

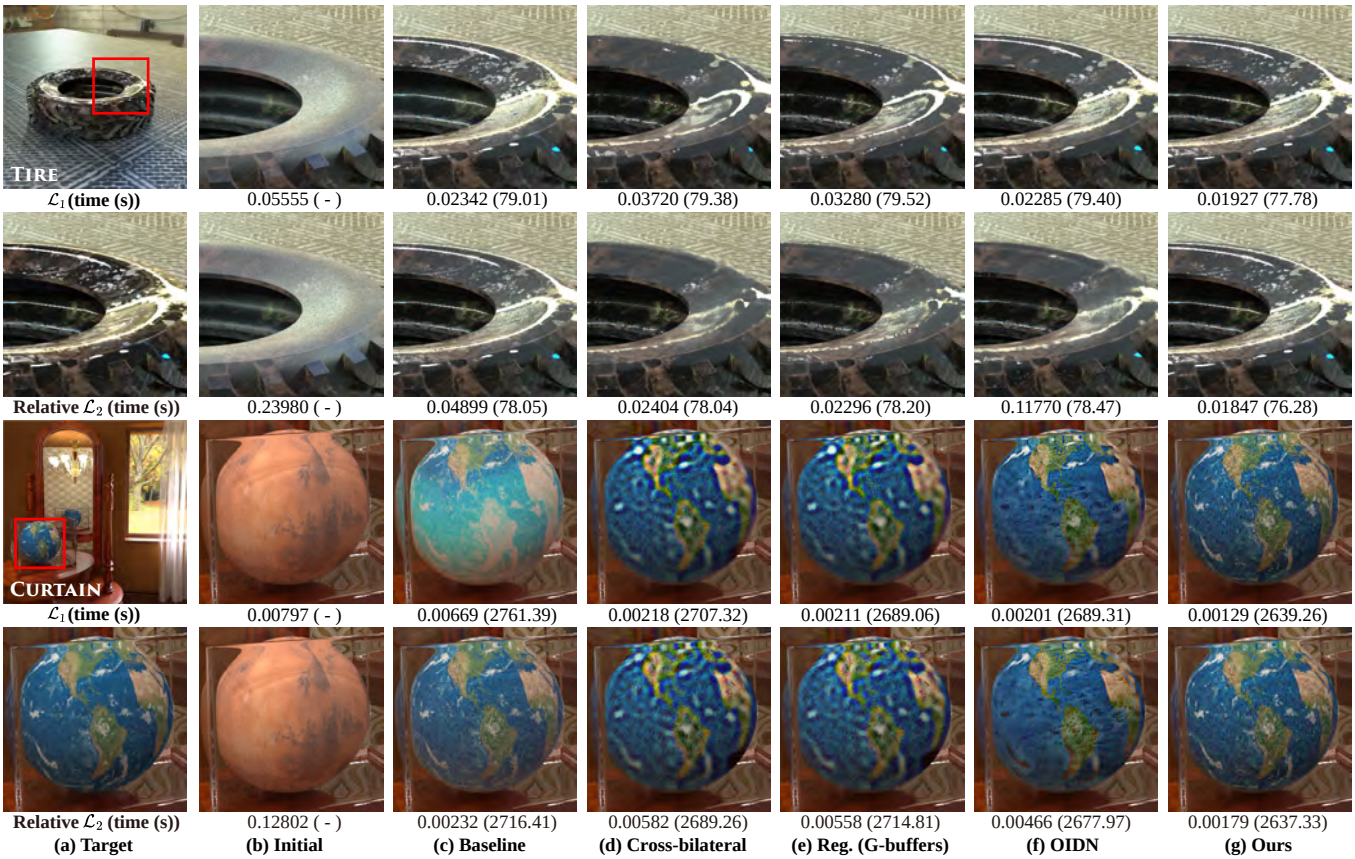

Fig. 6. Equal-time comparisons of optimization results with different denoising methods. Directly adopting the existing denoisers in inverse rendering leads to undesirable local minima, e.g., over-blurred artifacts by the cross-bilateral (d), regression with G-buffers (e), and OIDN (f). On the other hand, our method (g) helps the inverse rendering optimization infer more accurate results while preserving the details.

noisy input $\tilde{f}(\pi)$ to a denoised output $\hat{f}(\pi)$. Particularly, a neural denoiser in rendering usually takes a tone-mapped image for robust training, and this tone mapping also needs to be differentiable. To this end, we chose the log transformation [Bako et al. 2017] and trained the OIDN network using the publicly released code. For the training, we generated 1163 noisy/reference image pairs (also G-buffers) using public scenes [Bitterli 2016]. Then, we froze the trained network while optimizing scene parameters rather than fine-tuning the network, as suggested in [Hasselgren et al. 2022].

In Fig. 6, we infer the roughness for the TIRE scene and albedo textures for the CURTAIN scene, using either the $\mathcal{L}_1$ or relative $\mathcal{L}_2$ loss. The direct adoption of the existing denoisers drives inverse rendering optimization into undesirable local minima, e.g., blurred details. In particular, the cross-bilateral and G-buffer-based regression produce higher errors than the baseline for the TIRE scene when using the $\mathcal{L}_1$ loss. Also, the previous denoisers do not improve the scene optimization for the CURTAIN scene when using the relative $\mathcal{L}_2$ loss, i.e., higher errors than the baseline. On the other hand, adjusting denoising weights by considering the target image allows a robust use of image denoising for the inverse rendering tasks, as shown in Fig. 6 (g). Our supplemental report includes additional results, e.g., comparisons with the cross-bilateral filter for the BUNNY and NEFERTITI scenes and analysis of our method with and without G-buffers.

Our denoising with path-reusing techniques. In Fig. 8, we test a combination of our denoising and path-reusing techniques to see if such a combination can further enhance inverse rendering optimization. For the JANGA scene, we test the recursive control variates (R-CV) [Nicolet et al. 2023] that produces an improved image estimate $\tilde{f}(\pi)$ with reduced noise due to their temporal reusing. Note that this combination does not require a change in our denoising since it simply provides a less noisy input (i.e., their output) to our denoising. The combined method, i.e., ours + R-CV in Fig. 8, outperforms either R-CV or ours since this combination allows the use of both temporal coherence by R-CV and spatial coherence by ours for a more effective variance reduction in the gradient $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$.

Another interesting combination is to exploit our method within the parameter-space ReSTIR (P-ReSTIR) framework [Chang et al. 2023]. Specifically, we pass our image-space gradient $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ to P-ReSTIR instead of its original input, the noisy gradient $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$. As P-ReSTIR uses the relative $\mathcal{L}_2$ error [Chang et al. 2023], we exploit

their loss to conduct a fair comparison. Note that our denoising process is identical unless an inverse rendering optimization does not include a target image. As can be seen in Fig. 8, this combination, ours + P-ReSTIR, lets the optimization produce a more accurate inference than the optimization with either P-ReSTIR or ours.

Analysis of our denoising with different sample counts and bandwidths. Table 1 shows our inference errors when varying the bandwidth b^I (in Eq. 9) from 0.05 to 0.4 for the Curtain scene. The differences in the errors of the baseline and ours decrease as the samples per pixel (spp) increase since noise in the rendered image $\tilde{f}(\pi)$ (i.e., our input) becomes smaller with larger sample sizes. Nonetheless, utilizing our denoising in an inverse rendering framework can be a practical option for reducing the required number of samples (and thus optimization times). For example, our errors with 4 spp are smaller than the baseline error with 256 spp for the scene.

Also, as shown in Table 1, the accuracy of our inference remains mostly the same across the bandwidth, and it indicates that our method is robust against the selection of the global bandwidth (i.e., the same value for all pixels). Nonetheless, it would be interesting to adjust the bandwidth per pixel, i.e., a widely adopted optimization for image denoising [Zwicker et al. 2015]. We leave this per-pixel bandwidth selection as future work.

Analysis of different regression orders. We approximate the unknown image $f(\pi)$ locally with the Taylor polynomial of degree one (i.e., the linear function in Eq. 7). However, setting the order to a higher one may be tempting. To analyze our method with different regression orders, we have conducted the same optimization for the Tire and Curtain scenes in Fig. 6 but with different orders (cubic and quintic) and observed that the errors of these higher-order regressions are similar to the ones with the chosen linear order, as shown in Table 2. Nonetheless, it does not indicate that the order selection is unnecessary. Adjusting the regression order per pixel to minimize denoising errors can be technically desirable [Moon et al. 2016], and we leave this per-pixel order selection for future work.

Bias analysis of the $\mathcal{L}_1$ loss. In Fig. 9, we compare the biases of the image-space gradients of the baseline and our method when using the $\mathcal{L}_1$ loss. Note that the baseline produces unbiased gradient estimates for an $\mathcal{L}_2$ loss (e.g., the relative $\mathcal{L}_2$ loss), but their noise results in biased estimates for a non-$\mathcal{L}_2$ loss (e.g., the $\mathcal{L}_1$ loss) [Nicolet et al. 2023]. As shown in Fig. 9, our denoiser produces gradient estimates with a much smaller bias than the baseline, thanks to our noise reduction in the gradient estimates.

Our limitation. A limitation of our method is that we can only reduce noise in the image-space gradient $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ (in Eq. 3) since our technique is an image-space denoiser. As a result, noise in the $\partial\tilde{f}(\pi)/\partial\pi$ (in Eq. 4), which is not affected by image denoising, can hamper the convergence of inverse rendering optimization with our denoising. Fig. 10 shows the two references without denoising, which use a large number of samples (e.g., 1K) only for the $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ and for both the $\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ and $\partial\tilde{f}(\pi)/\partial\pi$. Our method enables reducing the difference in the numerical accuracy between the baseline with small samples and the first reference (Fig. 10 (d)) without allocating more samples. Nonetheless, there is a noticeable gap between ours and the second reference (Fig. 10 (f)). As an option for reducing the gap, we would like to investigate target-aware denoising in parameter space. While we show that a regression-based denoiser using the target image can become a practical option for improving inverse rendering tasks, it would be interesting to devise a neural network that fulfills our high-level idea, i.e., target-aware denoising. We leave these investigations to future work.

Table 1. $\mathcal{L}_1$ errors in our inference with various bandwidths for the Curtain.

	Bandwidth				
spp	0.05	0.1	0.2	0.4	Baseline
4	0.00232	0.00224	0.00223	0.00233	0.01639
8	0.00195	0.00192	0.00193	0.00193	0.01641
16	0.00171	0.00170	0.00172	0.00173	0.01537
32	0.00154	0.00155	0.00156	0.00157	0.01389
64	0.00140	0.00142	0.00144	0.00146	0.01149
128	0.00127	0.00129	0.00132	0.00133	0.00669
256	0.00116	0.00118	0.00120	0.00121	0.00359

Table 2. $\mathcal{L}_1$ errors in our inference with different Taylor polynomial orders.

	Order		
Scene	1	3	5
Tire	0.01927	0.01895	0.01913
Curtain	0.00129	0.00129	0.00130

6 CONCLUSION

This paper proposes an image denoiser that alleviates the gradient noise while preventing our denoising from driving inverse rendering optimization to undesirable local minima. Reducing noise in rendered images via image denoising has been extensively explored in ordinary rendering scenarios. Hence, exploiting an existing denoiser for inverse rendering is a natural and appealing option. Nonetheless, such a direct adoption of existing denoisers for inverse rendering can have a side effect, e.g., over-blurred scene parameters. We have explored a robust use of image denoising and devised a denoiser whose weights are adjusted using a target image. While we have designed a simple but effective way for considering the target image, i.e., a linear regression using the image, we believe that our high-level idea of incorporating the problem-specific information into image denoising would inspire various ways of taking advantage of the target image to improve inverse rendering optimization further.

ACKNOWLEDGMENTS

We appreciate the reviewers for their comments and thank the authors of the 3D models: Benedikt Bitterli (Veach-ajar), HorusZ (Tire), andrejammes (Nefertiti), Stanford 3D repository (Bunny) and JangaFX (Janga and Dust-devil). Bochang Moon is the corresponding author of the paper. This work was partly supported by the National Research Foundation of Korea (NRF) and Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korean government (MSIT) (Nos. RS-2023-00207939, 2022-0-00566, and RS-2023-00237965).

REFERENCES

Attila T. Áfra. 2023. Intel® Open Image Denoise. https://www.openimagedenoise.org.

Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages.

Martin Balint, Karol Myszkowski, Hans-Peter Seidel, and Gurprit Singh. 2023. Joint Sampling and Optimisation for Inverse Rendering. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA '23)*. Article 29, 10 pages.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 245 (Nov. 2020), 18 pages.

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. *Computer Graphics Forum* 35, 4 (2016), 107–117.

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

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-Space ReSTIR for Differentiable and Inverse Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Article 18, 10 pages.

Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, light, and material decomposition from images using Monte Carlo rendering and denoising. *Advances in Neural Information Processing Systems* 35 (2022), 22856–22869.

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

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable Rendering: A Survey. *CoRR* abs/2006.12057 (2020). arXiv:2006.12057 https://arxiv.org/abs/2006.12057

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph.* 37, 6, Article 222 (Dec. 2018), 11 pages.

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.

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Trans. Graph.* 41, 4, Article 75 (July 2022), 23 pages.

Clive Loader. 2006. *Local regression and likelihood.* Springer Science & Business Media.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.* 38, 6, Article 228 (Nov. 2019), 14 pages.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Trans. Graph.* 33, 5, Article 170 (Sept. 2014), 14 pages.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive Polynomial Rendering. *ACM Trans. Graph.* 35, 4, Article 40 (jul 2016), 10 pages.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Trans. Graph.* 40, 6, Article 248 (Dec. 2021), 13 pages.

Baptiste Nicolet, Fabrice Rousselle, Jan Novak, Alexander Keller, Wenzel Jakob, and Thomas Müller. 2023. Recursive Control Variates for Inverse Rendering. *ACM Trans. Graph.* 42, 4, Article 62 (July 2023), 13 pages.

Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Trans. Graph.* 41, 4, Article 44 (July 2022), 20 pages.

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph.* 39, 4, Article 146 (Aug. 2020), 15 pages.

Fabrice Rousselle, Wojciech Jarosz, and Jan Novák. 2016. Image-space control variates for rendering. *ACM Trans. Graph.* 35, 6, Article 169 (Dec. 2016), 12 pages.

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

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths Using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4, Article 108 (July 2021), 14 pages.

Yu-Chen Wang, Chris Wyman, Lifan Wu, and Shuang Zhao. 2023. Amortizing Samples in Physics-Based Inverse Rendering Using ReSTIR. *ACM Trans. Graph.* 42, 6, Article 214 (Dec. 2023), 17 pages.

Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* 42, 6, Article 213 (Dec. 2023), 18 pages.

Zihan Yu, Cheng Zhang, Derek Nowrouzezahrai, Zhao Dong, and Shuang Zhao. 2022. Efficient Differentiation of Pixel Reconstruction Filters for Path-Space Differentiable Rendering. *ACM Trans. Graph.* 41, 6, Article 191 (Nov. 2022), 16 pages.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo Estimators for Differential Light Transport. *ACM Trans. Graph.* 40, 4, Article 78 (July 2021), 16 pages.

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39, 4, Article 143 (Aug. 2020), 19 pages.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph.* 38, 6, Article 227 (Nov. 2019), 16 pages.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021. Path-Space Differentiable Rendering of Participating Media. *ACM Trans. Graph.* 40, 4, Article 76 (July 2021), 15 pages.

M. Zwicker, W. Jarosz, J. Lehtinen, B. Moon, R. Ramamoorthi, F. Rousselle, P. Sen, C. Soler, and S.-E. Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* 34, 2 (2015), 667–681.

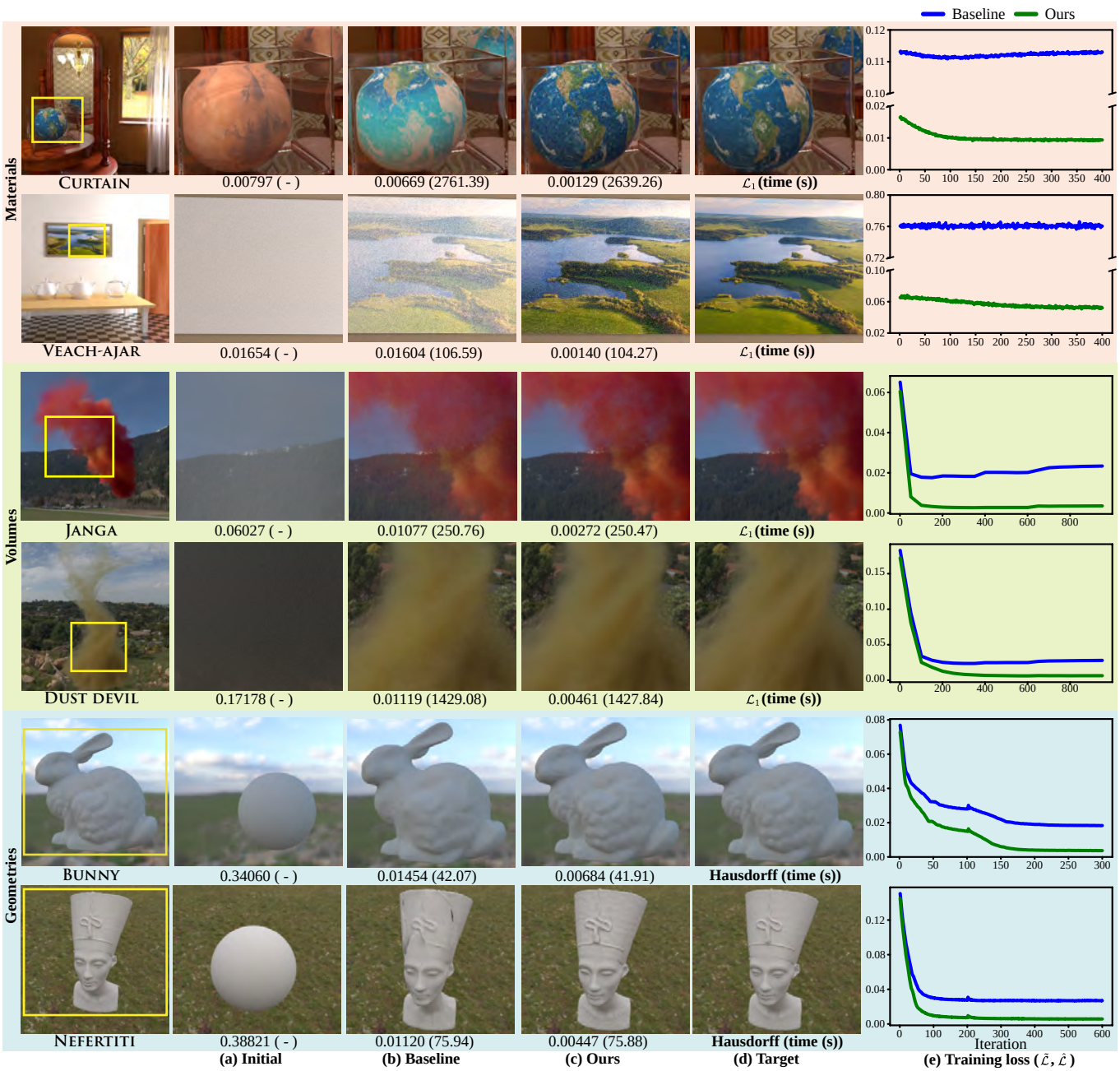

Fig. 7. Same-time comparisons with the baseline that does not use image denoising. We visualize the optimized scene parameters of the baseline (b) and ours (c) by rendering the inferred scenes with a large sample count, i.e., $f(\pi)$ with the two resulting parameters π, one from the baseline and another from ours. Also, we plot the training errors of the methods in (e), $\tilde{\mathcal{L}}$ for the baseline and $\hat{\mathcal{L}}$ for ours. Our image denoising makes the scene inferences more accurate than the baseline for various inverse rendering tasks, e.g., material estimation in the first two rows, inference of volume parameters in the third and fourth rows, and geometry optimization in the last two rows. We use the $\mathcal{L}_1$ errors to compute the numerical accuracy in inferred materials and volumes (in the first four rows) and measure the Hausdorff distance between the inferred and target geometries (in the last two rows).

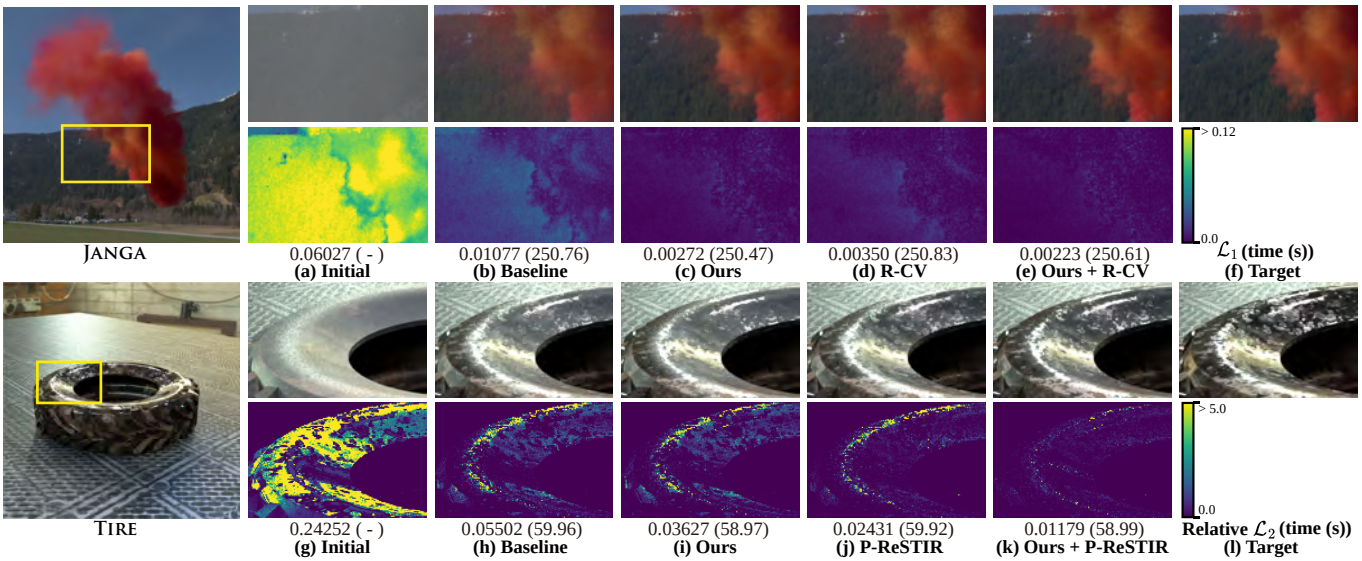

Fig. 8. Combination results with path-reusing techniques, the recursive control variates (R-CV) for the JANGA scene and the parameter-space ReSTIR (P-ReSTIR) for the TIRE scene. We visualize the inference results and error maps ($\mathcal{L}_1$ errors for the JANGA and relative $\mathcal{L}_2$ errors for the TIRE). Exploiting our denoising or a path-reusing technique lowers the inference errors compared to the baseline. Combining the two kinds of variance reduction techniques, i.e., a path-reusing technique together with our denoising, enables us to infer the scene parameters more accurately than the optimizations with ours or path-reusing alone.

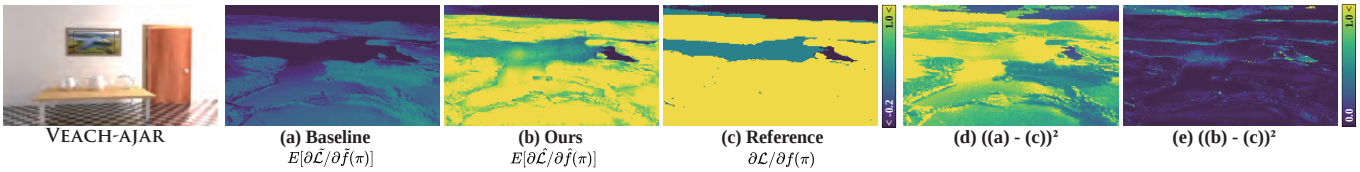

Fig. 9. Comparisons of the biases in the estimated image-space gradients with and without our method when using the $\mathcal{L}_1$ loss. We generate 10K image-space gradients of the baseline and our method while varying the random seed, using the scene setting for the VEACH-AJAR scene in Fig. 5. We then average the gradients from each method ((a) for the baseline and (b) for our technique). We also generate the reference gradients (c) using a large sample count. Noticeably, the squared bias of our gradient (e) is smaller than the baseline (d), thanks to our noise reduction. For clear visualization, we multiply the gradients by m (in Eq. 1).

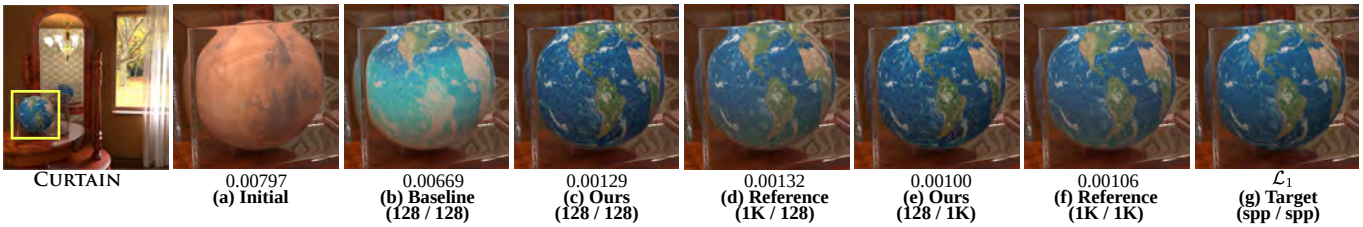

Fig. 10. A limitation of our denoising. We vary the number of samples per pixel (spp) for computing the image-space gradient ($\partial\tilde{\mathcal{L}}/\partial\tilde{f}(\pi)$ for the baseline and $\partial\hat{\mathcal{L}}/\partial\tilde{f}(\pi)$ for our denoiser) and the $\partial\tilde{f}(\pi)/\partial\pi$. Our method with 128 / 128 samples (c) shows a much smaller numerical error than the baseline with the same samples (b) and a similar error to the baseline with 1K / 128 samples (i.e., reference (d)). Nonetheless, our error (0.00129) is still noticeably higher than the error (0.00106) of another reference with 1K / 1K samples (f).