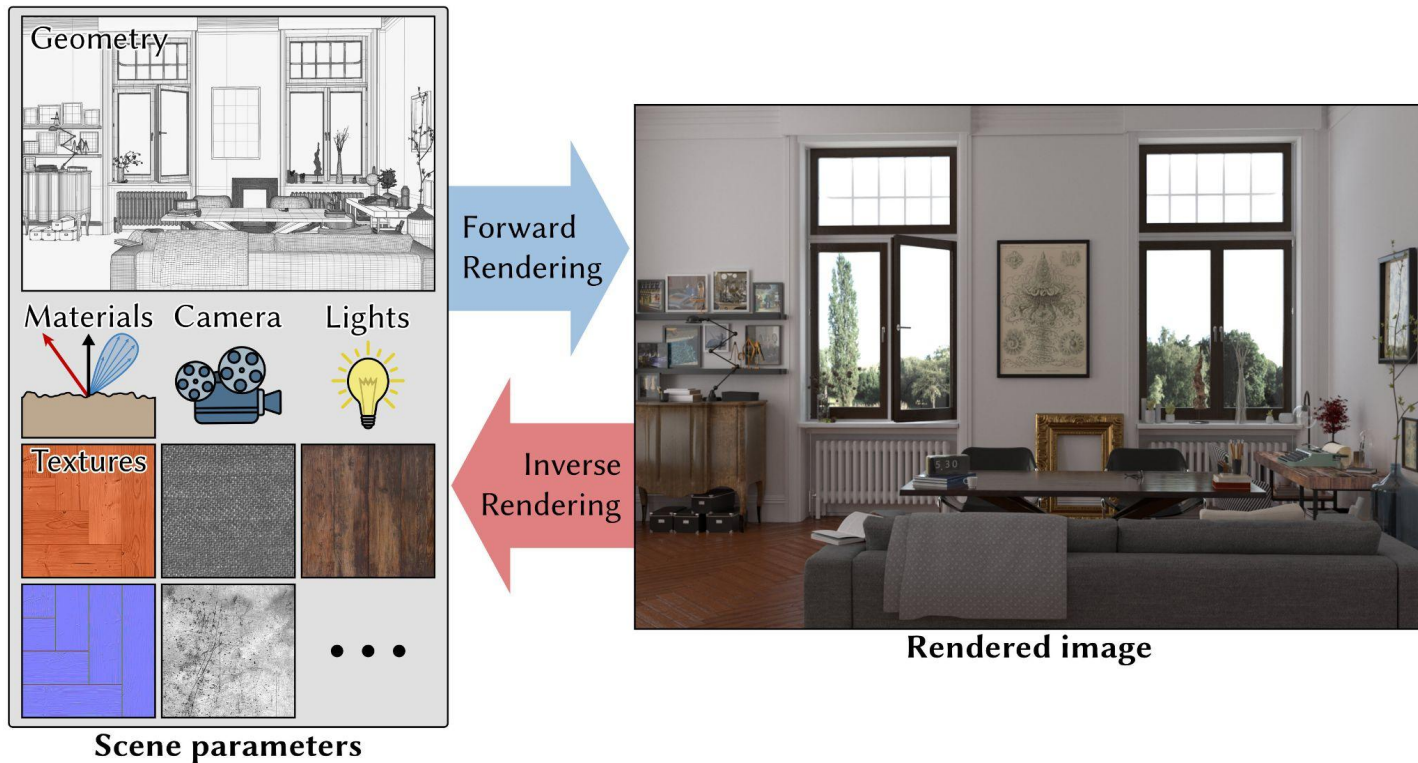


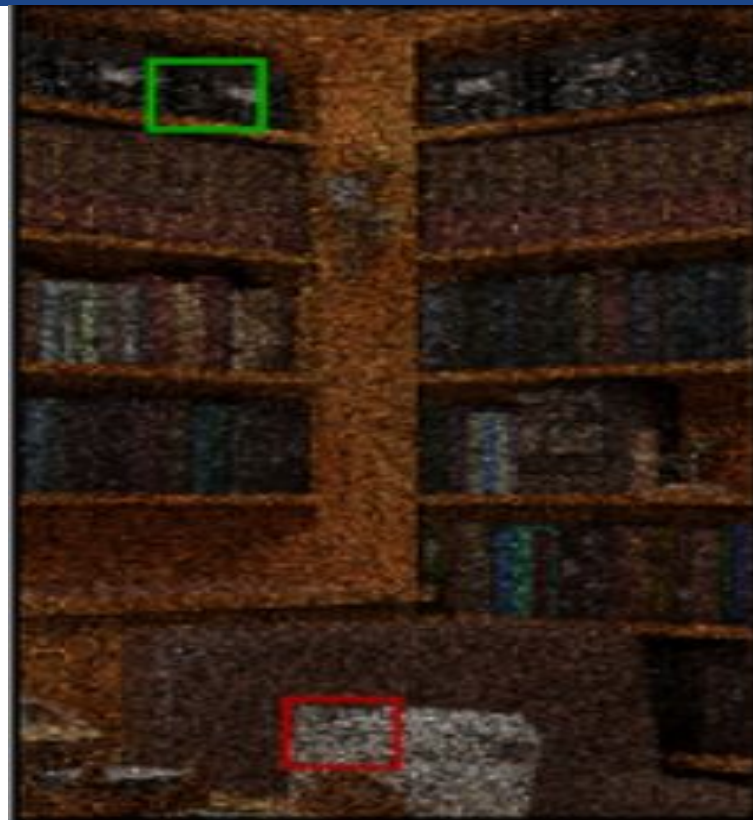
Target-Aware Image Denoising for Inverse Monte Carlo Rendering

Jeongmin Gu, Jonghee Back , Sung-Eui Yoon, Bochang Moon
ToG 2024

Inverse rendering



Inverse rendering uses low spp rendering



Denoiser :

En rendu différentiable :

- Beaucoup de images low spp
- Très bruitées

Que veut dire une loss entre avec une image 1spp ?



(1 spp)

[Chaitanya 2017]

Ground truth

The idea :

Venir denoiser les images pour chaque pas d'optimisation afin d'obtenir une loss plus sensée.

The idea :

Venir denoiser les images pour chaque pas d'optimisation afin d'obtenir une loss plus sensée.

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:

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

where $g(\pi)$ is an auxiliary input that a denoiser can take

Cas 1: Cross-bilateral filter

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., 31×31 window) centered at pixel c :

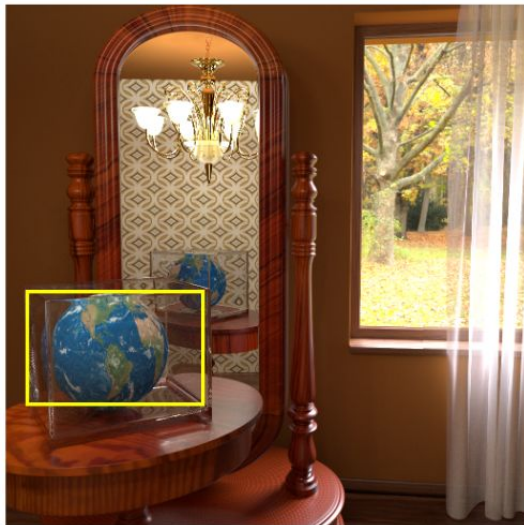
$$\hat{f}_c(\pi) = \sum_{i \in \Omega_c} \left(w_{i|c}^{cb} \tilde{f}_i(\pi) \right) / \sum_{i \in \Omega_c} w_{i|c}^{cb}. \quad (5)$$

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

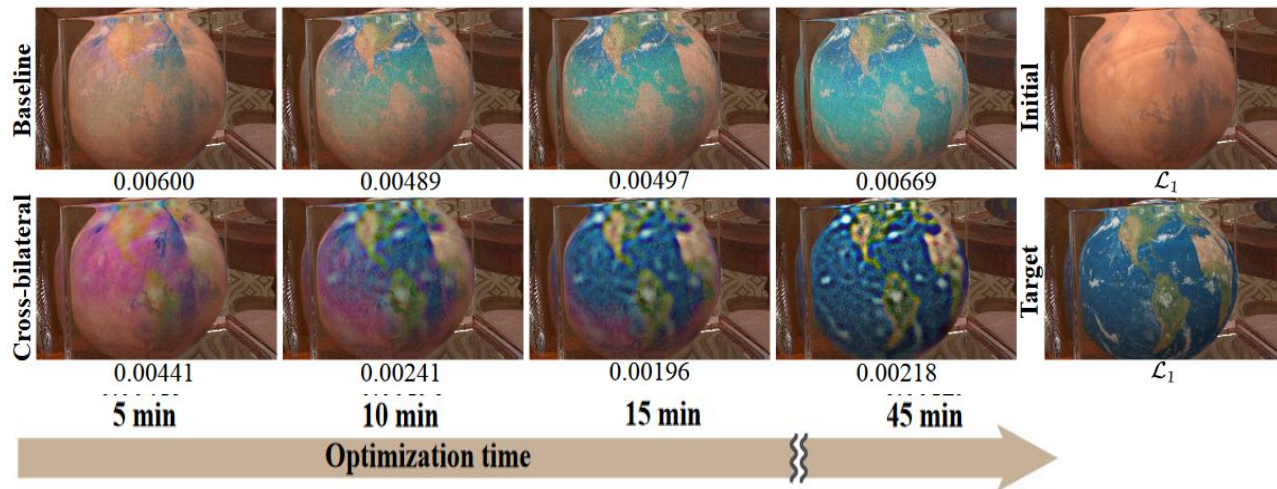
$$w_{i|c}^{cb} = \exp \left(-\frac{\|q_i - q_c\|^2}{2(b^q)^2} - \frac{\|\rho_i - \rho_c\|^2}{2(b^\rho)^2} - \frac{\|n_i - n_c\|^2}{2(b^n)^2} - \frac{(d_i - d_c)^2}{2(b^d)^2} \right), \quad (6)$$

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 .

Résultats:



CURTAIN



Cas 2: Their contribution

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):

$$f_i(\pi) \approx f_c(\pi) + f'_c(\pi)(I_i - I_c), \quad (7)$$

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 .

Cas 2: Their contribution

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

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

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 $\|\tilde{f}_i(\pi) - \alpha_c - \beta_c(I_i - I_c)\|^2$

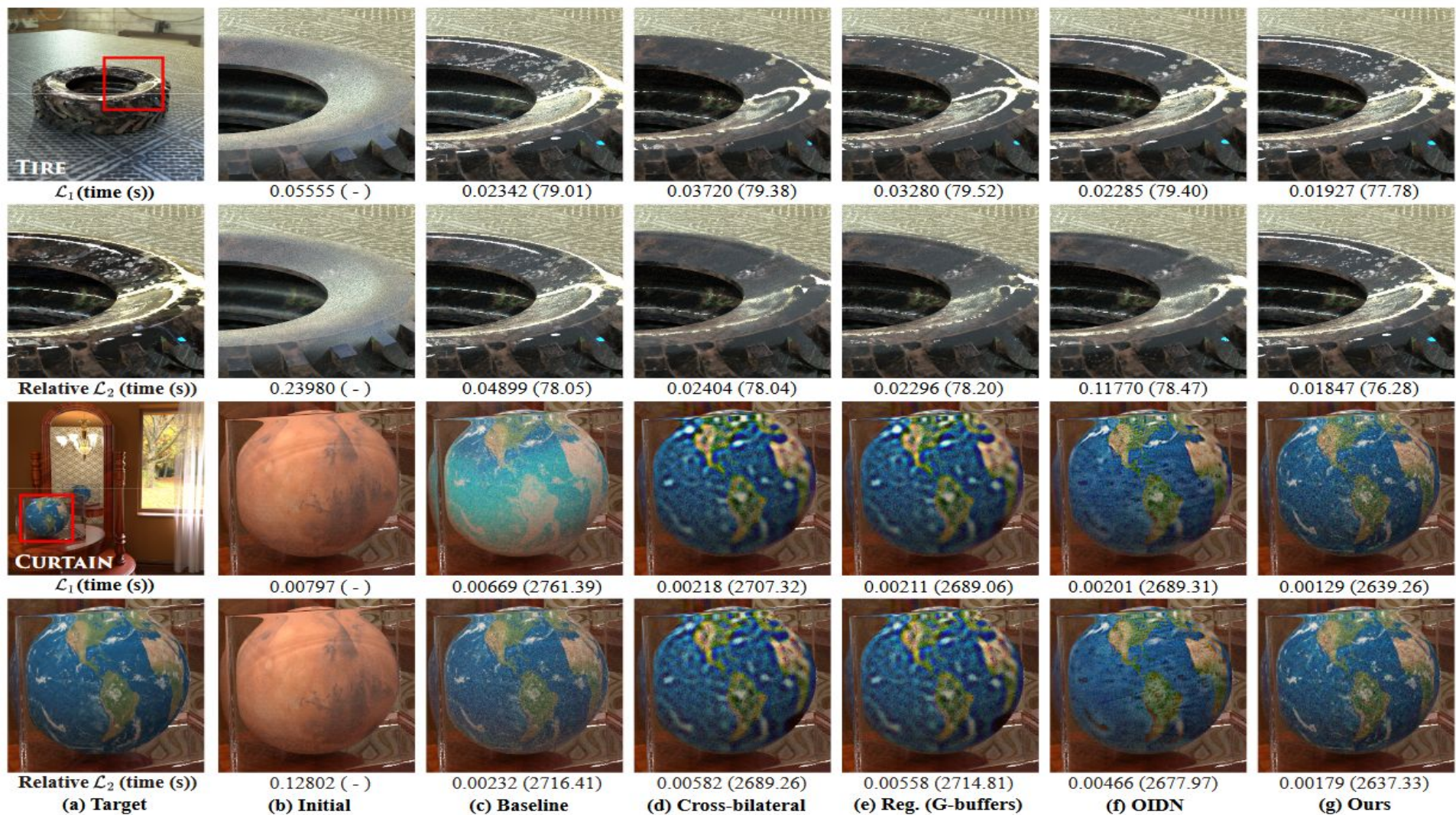
Cas 2: Their contribution (Analysis)

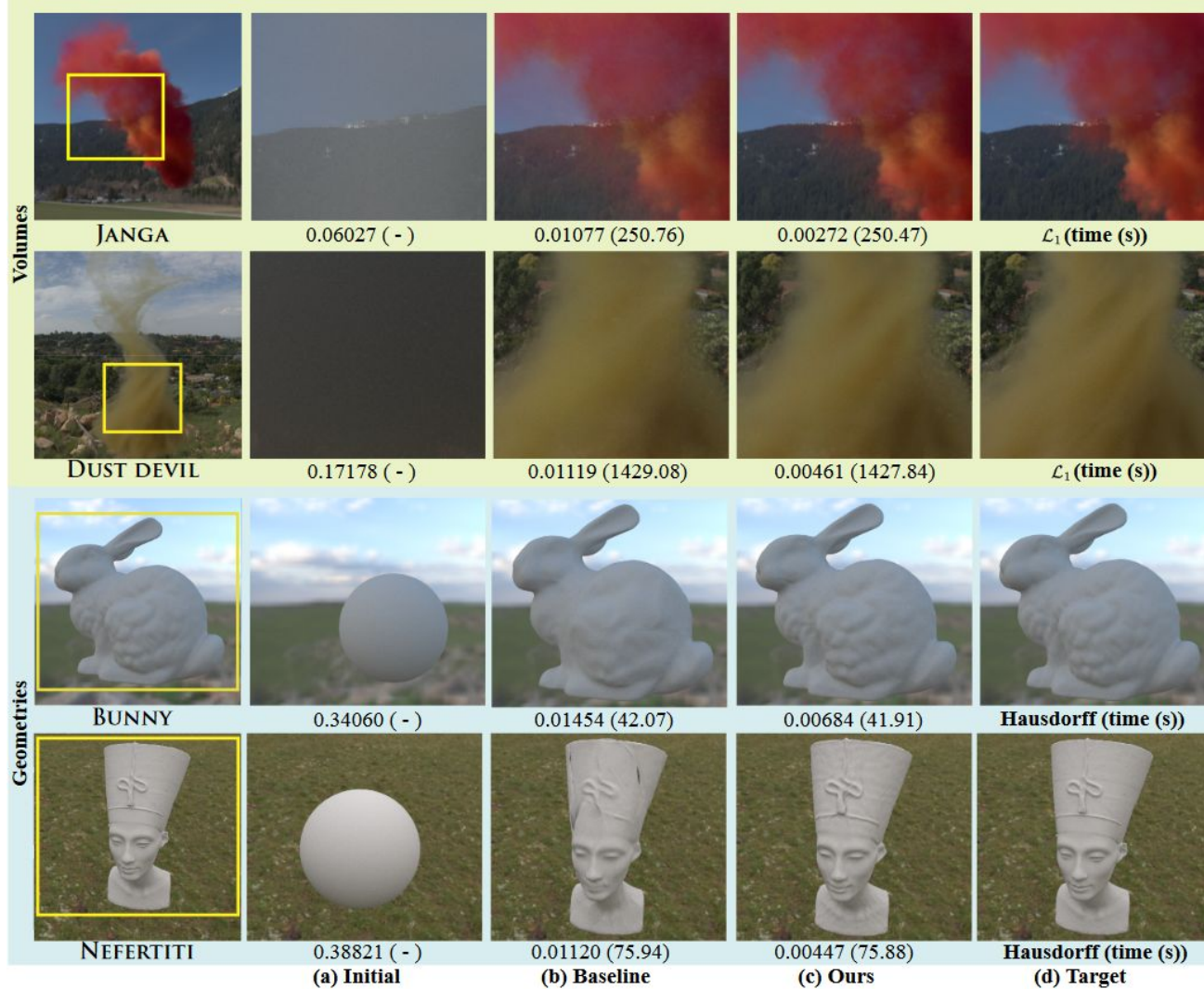
Bias:

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

Gradient Bias:

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





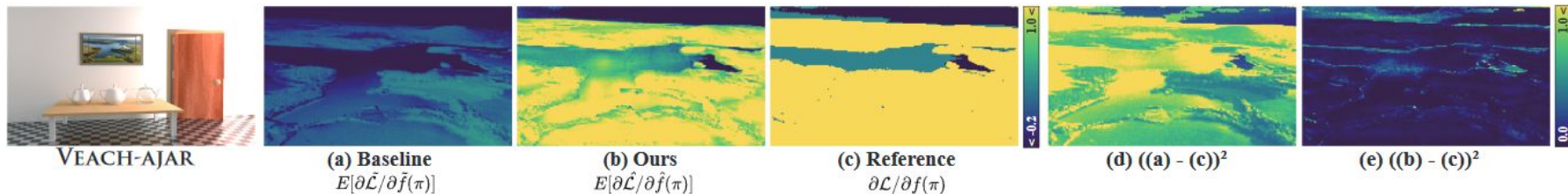


Fig. 9. Comparisons of the biases in the estimated image-space gradients with and without our method when using the $\mathcal{L}_1$ loss. We generate 10K image-space gradients of the baseline and our method while varying the random seed, using the scene setting for the VEACH-AJAR scene in Fig. 5. We 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).

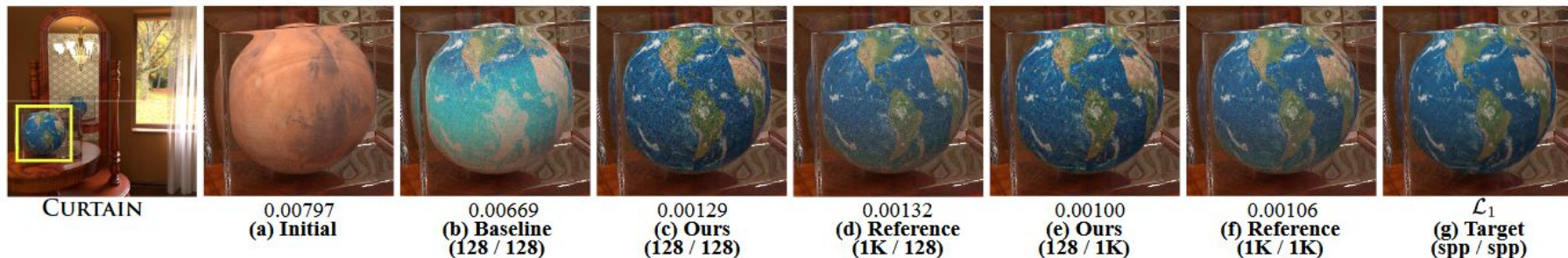


Fig. 10. A limitation of our denoising. We vary the number of samples per pixel (spp) for computing the image-space gradient ($\partial \tilde{\mathcal{L}} / \partial \tilde{f}(\pi)$ for the baseline and $\partial \tilde{\mathcal{L}} / \partial \tilde{f}(\pi)$ for our denoiser) and the $\partial \tilde{f}(\pi) / \partial \pi$. 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).

FIN