Supplementary Document: Spatiotemporal Bilateral Gradient Filtering for Inverse Rendering

WESLEY CHANG*, University of California San Diego, USA
XUANDA YANG*, University of California San Diego, USA
YASH BELHE*, University of California San Diego, USA
RAVI RAMAMOORTHI, University of California San Diego, USA
TZU-MAO LI, University of California San Diego, USA

ACM Reference Format:
Wesley Chang, Xuanda Yang, Yash Belhe, Ravi Ramamoorthi, and Tzu-Mao Li. 2024. Supplementary Document: Spatiotemporal Bilateral Gradient Filtering for Inverse Rendering. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 3–6, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3680528.3687606

1 PREFILTERING AND POSTFILTERING

In Algorithm 2 of the main text, we propose to extend Adam's temporal filtering to a spatiotemporal filter. We can apply the spatial filter before the exponential averages (prefiltering) or after the exponential averages (postfiltering). It is also possible to apply both a prefilter and a postfilter, whereas, applying neither reduces to the standard Adam algorithm.

In this section, we study the differences between the different configurations theoretically, followed by an experimental analysis. To recapitulate, we are interested in minimizing an objective function $f(\theta) : \mathbb{R}^n \to \mathbb{R}$, with gradient $\nabla_{\theta_t} f$ at time t, which we denote as g_t going forward.

Both prefiltering and postfiltering apply cross-bilateral filters. These are linear operators [Milanfar 2013] which we denote as $A, B \in \mathbb{R}^{n\times n}$ respectively. The prefiltered gradient is

$$\tilde{g}_t = A_t g_t, \tag{1}$$

and its (prefiltered) first and second moments are

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\tilde{g}_t && (2)\\ &= \sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}A_i g_i, && (3)\\ v_t &= \beta_2 v_{t-1} + (1-\beta_2)\tilde{g}_t^2 && (4)\\ &= \sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}(A_i g_i)^2, && (5)\end{aligned}$$

where the square is component-wise. Postfiltering gives us

$$\begin{aligned} \tilde{m}_t &= B_t m_t && (6)\\ &= \sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}B_t A_i g_i, && (7)\\ \tilde{v}_t &= B_t v_t && (8)\\ &= \sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}B_t(A_i g_i)^2. && (9)\end{aligned}$$

Notice that since prefiltering is applied to the gradient g_t *before* exponential moving averages, m_t and v_t depend on *all* past prefilters A_i. On the other hand, postfiltering is applied *after* exponential moving averages, so m_t and v_t do not accumulate any past postfilters B_i; they only depend on the current postfilter B_t.

The resulting update step[1] which includes both prefiltering and postfiltering is

$$\begin{aligned} \Delta_t &= \frac{\tilde{m}_t}{\sqrt{\tilde{v}_t}} && (10)\\ &= \frac{\sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}B_t A_i g_i}{\sqrt{\sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}B_t(A_i g_i)^2}}. && (11)\end{aligned}$$

From this general update step, we can recover the update step for Adam, prefiltering only and postfiltering only by appropriately setting A_i and/or B_i to identity.

$$\begin{aligned} \Delta_t^{\text{adam}} &= \frac{\sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}g_i}{\sqrt{\sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}g_i^2}}, && (12)\\ \Delta_t^{\text{pre}} &= \frac{\sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}A_i g_i}{\sqrt{\sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}(A_i g_i)^2}}, && (13)\\ \Delta_t^{\text{post}} &= \frac{\sum_{i=1}^{t}(1-\beta_1)\beta_1^{t-i}B_t g_i}{\sqrt{\sum_{i=1}^{t}(1-\beta_2)\beta_2^{t-i}B_t(g_i)^2}}. && (14)\end{aligned}$$

The above update steps are hard to theoretically analyze directly. Instead, we consider a simplified setting with $\beta_1 = 0$, i.e. with no temporal filtering in Section 1.1. This setting is indeed close to the optimal value we practically use ($\beta_1 = 0.2$) under a restricted iteration budget (see Section 5.1 in main text).

*The three authors contributed equally to this research.

[1]We omit bias correction and multiplication with learning rate for ease of exposition. The bias corrected versions of $\tilde{m}_t$ and $\tilde{v}_t$ are given by $\tilde{m}_t/(1-\beta_1^t)$ and $\tilde{v}_t/(1-\beta_2^t)$ respectively.

Authors' Contact Information: Wesley Chang, wec022@ucsd.edu, University of California San Diego, USA; Xuanda Yang, xuy008@ucsd.edu, University of California San Diego, USA; Yash Belhe, ybelhe@ucsd.edu, University of California San Diego, USA; Ravi Ramamoorthi, ravir@ucsd.edu, University of California San Diego, USA; Tzu-Mao Li, tzli@ucsd.edu, University of California San Diego, USA.

1.1 Analysis with no temporal filtering

Setting $\beta_1 = \beta_2 = 0$, the update steps simplify to

$$\Delta_t^{\text{adam}} = \frac{g_t}{\sqrt{g_t^2}} = \operatorname{sign}(g_t), \tag{15}$$

$$\Delta_t^{\text{pre}} = \frac{A_t g_t}{\sqrt{(A_t g_t)^2}} = \operatorname{sign}(A_t g_t), \tag{16}$$

$$\Delta_t^{\text{post}} = \frac{B_t g_t}{\sqrt{B_t (g_t)^2}}. \tag{17}$$

Both Adam and prefiltering make a step solely based on the *sign* of the gradient g_t and prefiltered gradient $A_t g_t$ respectively. Since prefiltering reduces noise in the gradient, it also reduces noise in the sign, which improves convergence. For an in-depth discussion on the benefits of decreasing sign variance in inverse rendering, please see Belhe et al. [2024] and Chang et al. [2023].

Postfiltering's step is slightly more nuanced. To analyze it, we inspect its j^{th} component

$$\Delta_{t,j}^{\text{post}} = \frac{\sum_{k=1}^{n} B_{t,jk} g_{t,k}}{\sqrt{\sum_{k=1}^{n} B_{t,jk} g_{t,k}^2}}, \tag{18}$$

where $B_{t,jk}$ is the (j,k)th element in B_t and $g_{t,k}$ is the kth component of g.

Comparing the equation above with Equation (12) shows an interesting similarly between postfiltering and the standard Adam step. Both methods make a step based on ratio of the first moment to the square root of the second moment. The key difference is that **Adam computes moments over time, while postfiltering computes them over space.**

This perspective also explains several of the practical advantages we observe within the low β_1 regime for the postfilter. Variance adaptive optimizers, like our postfilter Equation (17), which adjust step sizes based on the estimated variance (the denominator in Equation (18)) consistently outperform their sign-only counterparts [Balles and Hennig 2018], i.e Adam and prefiltering which make constant sized steps Equations (15) and (16). Practically, we find that with no temporal filtering or low values of β_1, postfiltering converges much faster than Adam, see Figure 1 in this text and Figures 5, 8 in the main text.

1.2 Postfiltering is adaptive to noise.

We decompose the gradient $g_t = \hat{g}_t + n_t$, as a sum of its true value $\hat{g}_t$ and zero-mean noise n_t. An ideal postfilter B_t should preserve the moments of the true gradient $B_t g_t = \hat{g}_t$ and $B_t g_t^2 = \hat{g}_t^2$. It should also filter out noise $B_t n_t = 0$ as well as its product with the true gradient $B_t(n_t g_t) = 0$, where the product $n_t g_t$ is component-wise. Thus, filtering the noisy gradient gives us the true gradient

$$B_t g_t = B_t(\hat{g}_t + n_t) = \hat{g}_t. \tag{19}$$

It can also be written component-wise as

$$\sum_{k=1}^{n} B_{t,jk} g_{t,k} = \hat{g}_{t,j}. \tag{20}$$

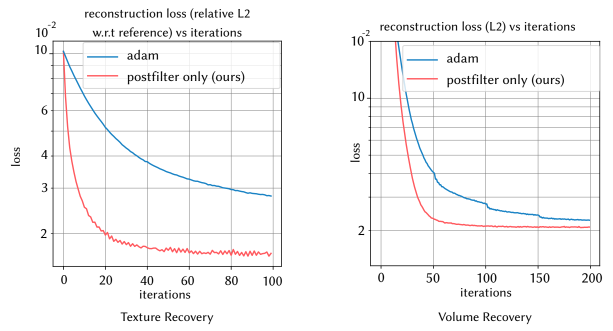

Fig. 1. Recovery with no temporal filtering. We compare Adam and postfiltering with no temporal filtering, that is, with $\beta_1 = \beta 2 = 0$ for texture and volume recovery (Figures 5 and 8 in main text). Without temporal filtering, both our method and Adam are *memoryless*, this reduces each of their memory overhead by 2/3rd since they no longer need to maintain buffers for m_t and v_t. For Adam, however, this comes at the cost of making fixed sized steps, slowing down convergence in the presence of noise and anisotropy, see Equation (15). On the other hand, our method can still estimate the signal-to-noise ratio via spatial filtering, see Equation (17). This enables it to make dynamic step sizes, which converges faster in the presence of noise and anisotropy, see loss curves above.

Now, substituting this into the postfilter step Equation (18), we get

$$\Delta_{t,j}^{\text{post}} = \frac{\sum_{k=1}^{n} B_{t,jk} g_{t,k}}{\sqrt{\sum_{k=1}^{n} B_{t,jk} g_{t,k}^2}}, \tag{21}$$

$$= \frac{\sum_{k=1}^{n} B_{t,jk} (\hat{g}_t + n_t)}{\sqrt{\sum_{k=1}^{n} B_{t,jk} (\hat{g}_t + n_t)^2}}, \tag{22}$$

$$= \frac{\hat{g}_{t,j}}{\sqrt{\sum_{k=1}^{n} B_{t,jk} (\hat{g}_{t,k}^2 + 2\hat{g}_{t,k} n_{t,k} + n_{t,k}^2)}}, \tag{23}$$

$$= \frac{\hat{g}_{t,j}}{\sqrt{\hat{g}_{t,j}^2 + \sum_{k=1}^{n} B_{t,jk} n_{t,k}^2}}, \tag{24}$$

where the third step comes from Equation (20) and the fourth step comes from the properties of an ideal postfilter listed in the beginning of this section. Now, examining Equation (24), we see that since $B_{t,jk} n_{t,k}^2 \geq 0$ (both terms are non-negative), the denominator grows as the noise increases, reducing the learning rate. Therefore, even with $\beta_1 = \beta_2 = 0$ postfiltering adapts its learning rate according to the signal-to-noise ratio, unlike prefiltering and Adam in this regime, which make constant sized steps Equations (15) and (16).

1.3 Postfiltering m_t and v_t keeps the update step bounded

In the main text, we advocate to postfilter both m_t and v_t jointly, since this maintains Adam's bounded step size. Formally, given that

$$\left| \frac{m_t}{\sqrt{v_t}} \right| \leq 1, \tag{25}$$

we want to prove that

$$\left| \frac{\tilde{m}_t}{\sqrt{\tilde{v}_t}} \right| \leq 1. \tag{26}$$

Note that in the identities above, m_t and v_t are vectors so the inequalities hold for each of their components.

We now prove the inequality above for the j^{th} component. The filtered moments are given by

$$\tilde{m}_{t,j} = \sum_{k=1}^{n} B_{t,jk} m_{t,k}, \tag{27}$$

$$\tilde{v}_{t,j} = \sum_{k=1}^{n} B_{t,jk} v_{t,k}, \tag{28}$$

where $\sum_{k=1}^{n} B_{t,jk} = 1$ since the cross-bilateral filter is normalized and its weights sum up to 1 and $0 \leq B_{t,jk} \leq 1 \; \forall k \in \{1, ..., n\}$ since the weights are always non-negative. Then we have

$$\sum_{k=1}^{n} B_{t,jk} m_{t,k} = \sum_{k=1}^{n} \left(\sqrt{B_{t,jk}}\right)\left(\sqrt{B_{t,jk}} m_{t,k}\right) \tag{29}$$

$$\left|\sum_{k=1}^{n} B_{t,jk} m_{t,k}\right|^2 \leq \sum_{k=1}^{n} \left(\sqrt{B_{t,jk}}\right)^2 \sum_{k=1}^{n} \left(\sqrt{B_{t,jk}} m_{t,k}\right)^2 \tag{30}$$

$$= \sum_{k=1}^{n} B_{t,jk} \sum_{k=1}^{n} B_{t,jk} m_{t,k}^2 \tag{31}$$

$$= \sum_{k=1}^{n} B_{t,jk} m_{t,k}^2 \tag{32}$$

$$\leq \sum_{k=1}^{n} B_{t,jk} v_{t,k}, \tag{33}$$

where the second step is the Cauchy-Schwarz inequality and the fourth step holds because $\sum_{k=1}^{n} B_{t,jk} = 1$. The final step holds because $m_{t,k}^2 \leq v_{t,k}$. Therefore, we have

$$\left|\sum_{k=1}^{n} B_{t,jk} m_{t,k}\right| \leq \sqrt{\sum_{k=1}^{n} B_{t,jk} v_{t,k}}. \tag{34}$$

Now, substituting Equations (27) and (28) into the equation above, we get

$$\left|\tilde{m}_{t,j}\right| \leq \sqrt{\tilde{v}_{t,j}}, \tag{35}$$

$$\left|\frac{\tilde{m}_{t,j}}{\sqrt{\tilde{v}_{t,j}}}\right| \leq 1, \tag{36}$$

which completes the proof.

2 OVERHEAD FOR OUR METHOD

Our method has little overhead over Adam (< 4%) for grid-based applications Table 1 and inverse mesh recovery (< 10%) Section 2.

3 HYPERPARAMETER ABLATIONS

For grid-based applications, our method introduces two hyperparameters: the filter size F and the edge-preserving data term weight σ_d. These hyperparameters effectively trade off Monte Carlo noise in the gradient with spatial smoothing. When F is small, our method approaches Adam, and the noise in the gradient results in noise in the recovered texture. When F is large, our method approaches low-pass filtering, and exhibits little noise, but results in overblurred reconstructions. Similarly, as σ_d decreases, edges are more well-preserved at the cost of more noise and vice versa.

For mesh recovery, our method inherits the Laplacian smoothing weight λ from Large Steps and introduces an additional σ_d to prevent smoothing over edges. We show inverse mesh recovery of the cube scene over different values of λ and σ_d in Fig. 2. Like Large Steps, low values of λ can introduce non-uniformities in the recovery, while larger values can smooth over edges. When σ_d is small, the smoothing is reduced at the edges, which can cause cracks due to aggressive silhouette gradient updates, as discussed by Nicolet et al. [2021]. As σ_d increases, our method approaches Large Steps. We find σ_d around 0.5 to be a reasonable default.

Table 1. TEXTURE AND VOLUME FILTERING OVERHEAD. We report the average per-iteration timings for our method and Adam, for texture and volume optimization. For both tasks, this includes time for both (differentiable) rendering and updating parameters. For texture optimization (Figure 5 in main text), our method incurs up to 3.4% overhead. For volume optimization (Figure 9 in main text), the runtime is dominated by forward rendering and derivative estimation (> 99%). Interestingly, we find that our method has a lower wall clock time than Adam for (differentiable) volume rendering. This is because volume rendering time is highly dependent on parameter values [Nimier-David et al. 2022].

Example	Method	Time	% Change	Grid
Texture	Adam	247ms	-	512^2
Texture	Ours ($F = 3$)	251ms	1.78%	512^2
Texture	Ours ($F = 5$)	255ms	3.40%	512^2
Volume	Adam	1380ms	-	64^3
Volume	Ours ($F = 3$)	1046ms	-24.2%	64^3
Volume	Ours ($F = 5$)	1058ms	-23.3%	64^3

Table 2. MESH FILTERING OVERHEAD We measure the average time per optimization iteration for the dragon scene in the main text at different vertex counts.

Large Steps	Ours	Overhead	Vertex Count
148.52ms	152.52ms	2.69%	10242
163.52ms	168.87ms	3.27%	34789
189.41ms	207.11ms	9.34%	166103

4 SPATIAL BIAS

For grid-based applications, our method trades off Monte Carlo noise in the gradient with spatial smoothing bias. As discussed in the main text, even though this biases the gradients in the sense that each filtered partial derivative no longer matches the true derivative in expectation, our filter is a positive semi-definite preconditioner, and will thus still decrease or maintain the loss on average. In extreme cases like when recovering high frequency textures (Figure 4) or high frequency volumes (Figure 5), spatial filtering can cause over-smoothed results. In these cases, it may help to start optimization with our method and progressively update the filtering parameters so that our method approaches Adam near the end to reconstruct the high frequency details.

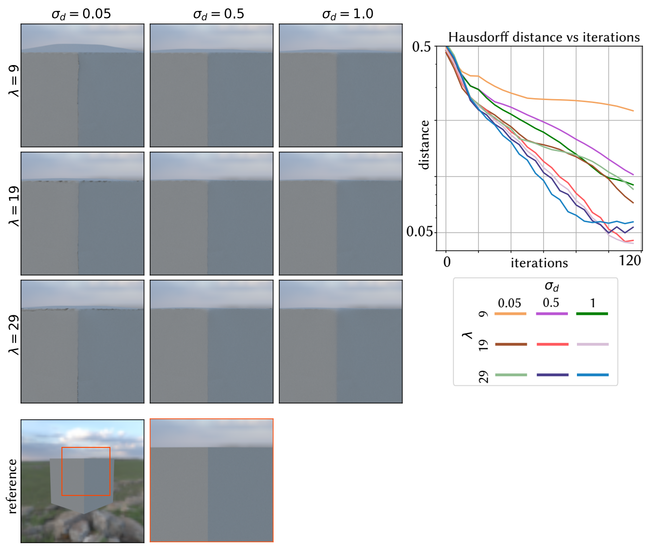

Fig. 2. Mesh hyperparameter ablation We compare mesh recovery at different values of λ (smoothing weight) and σ_d (edge-preservation data weight) and show both visual results and Hausdorff (geometric) distance. Scene adapted from Kloppenheim 06 ©Greg Zaal.

For mesh recovery, our method inherits the spatial bias from smoothing the gradients using Large Steps, which is shown to be crucial to recover geometry [Nicolet et al. 2021]. Due to the smoothing, it can be challenging for both our method and Large Steps to converge to high frequency geometry, especially when the smoothing weight is high. Figure 6 shows an extreme case of recovering the cube scene with $\lambda = 999$. While our bilateral filter can form edges faster than Large Steps, both do not converge within a reasonable number of iterations. Similar to the grid-based applications, it may help to decay the smoothing weight over optimization to recover fine details.

REFERENCES

Lukas Balles and Philipp Hennig. 2018. Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients. https://openreview.net/forum?id=S1EwLkW0W

Yash Belhe, Bing Xu, Sai Praveen Bangaru, Ravi Ramamoorthi, and Tzu-Mao Li. 2024. Importance Sampling BRDF Derivatives. *ACM Trans. Graph.* 43, 3 (2024).

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for Differentiable and Inverse Rendering. In *SIGGRAPH Conference Proceedings*. 10 pages.

Peyman Milanfar. 2013. Symmetrizing Smoothing Filters. *SIAM Journal on Imaging Sciences* 6, 1 (2013), 263–284. https://doi.org/10.1137/120875843

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 40, 6 (2021).

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

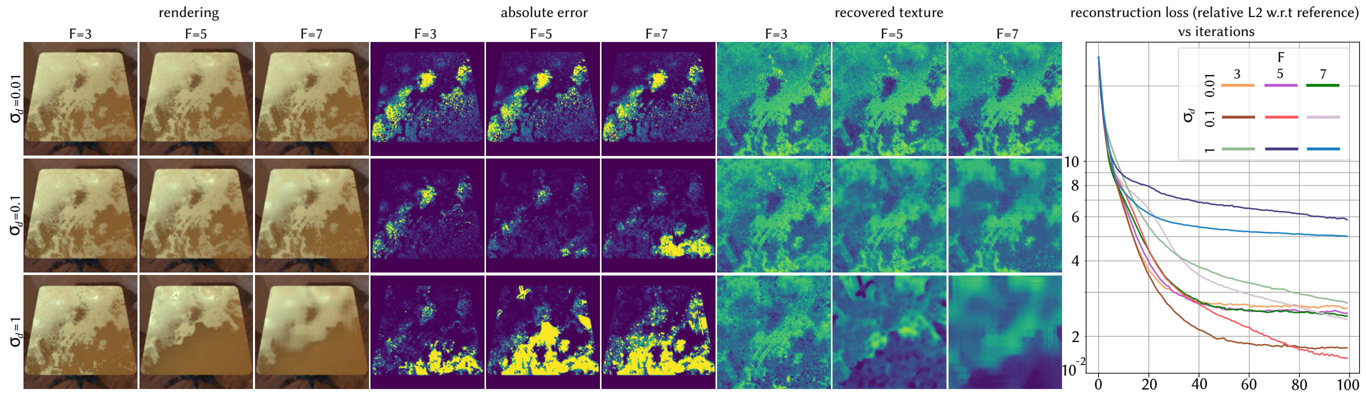

Fig. 3. Texture hyperparameter ablation. We compare the inverse texture recovery from Figure 5 of the main text at different values of F (filter size) and σ_d (edge-preservation weight). We show the rendered image, image error, recovered roughness texture, and loss plots. These parameters trade off spatial smoothing bias with noise in the recovery. Scene adapted from At the Window ©Bernhard Vogl.

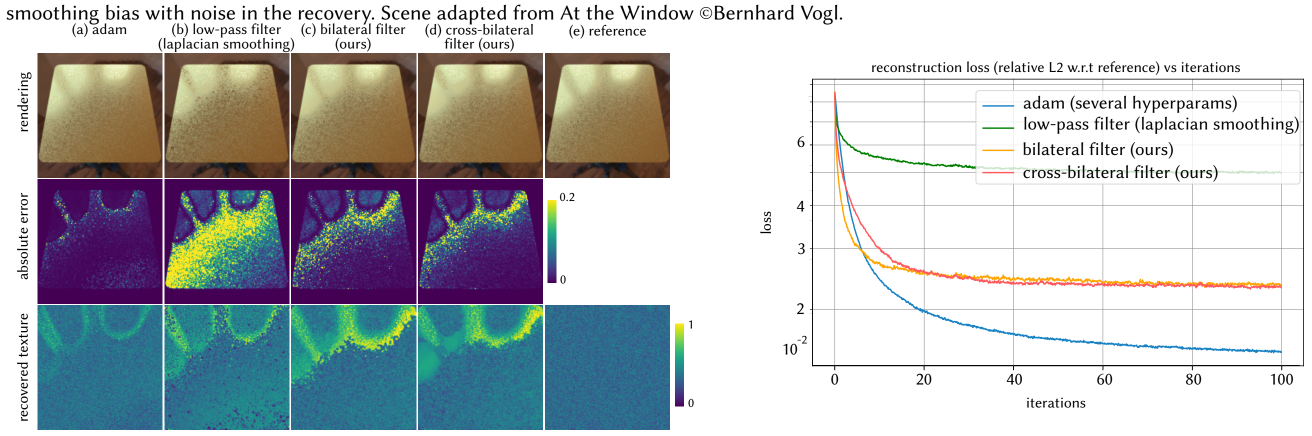

Fig. 4. Bias in high-frequency texture recovery. We keep the same setup at Figure 5 from the main text, only replacing the roughness texture with high frequency noise (importantly, we use the same hyperparameters). Since high frequency noise is not piecewise-smooth, both our method (d) and bilateral filtering (c) have larger error than Adam (a), which optimizes each parameter independently; low-pass filtering (b) oversmooth s the result the most and has highest error. Scene adapted from At the Window ©Bernhard Vogl.

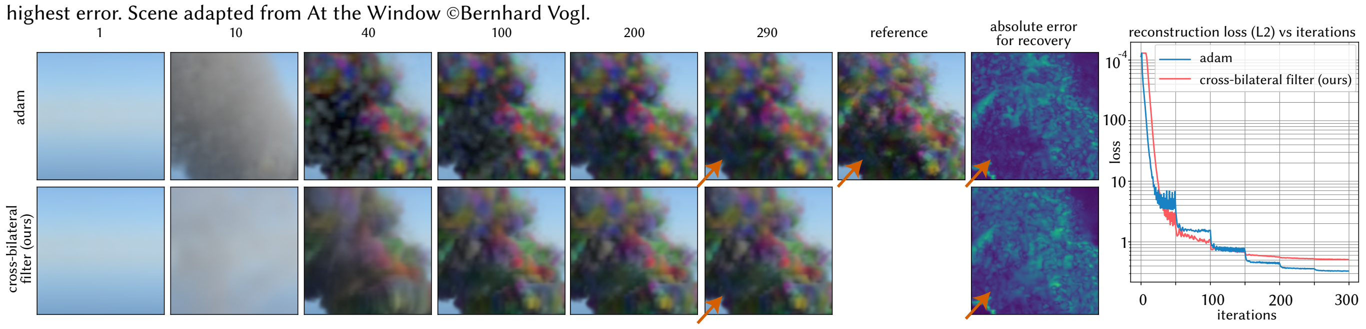

Fig. 5. Bias in low-noise volume recovery. We show an extreme case of volume recovery of density σ_t and high frequency albedo ρ at high sample counts (512 spp primal, 128 spp grad). At such high sample counts, the gradients have little noise, and so Adam can recover the details well, while our spatial filtering causes some oversmoothing, especially at the earlier stages of the optimization. Even in the final result, Adam is able to better resolve the dark regions (red arrow) than our method which oversmooths. Scene adapted from Autumn Field ©Jarod Guest and Sergej Majboroda, and High-Res Smoke Plume ©JangaFX.

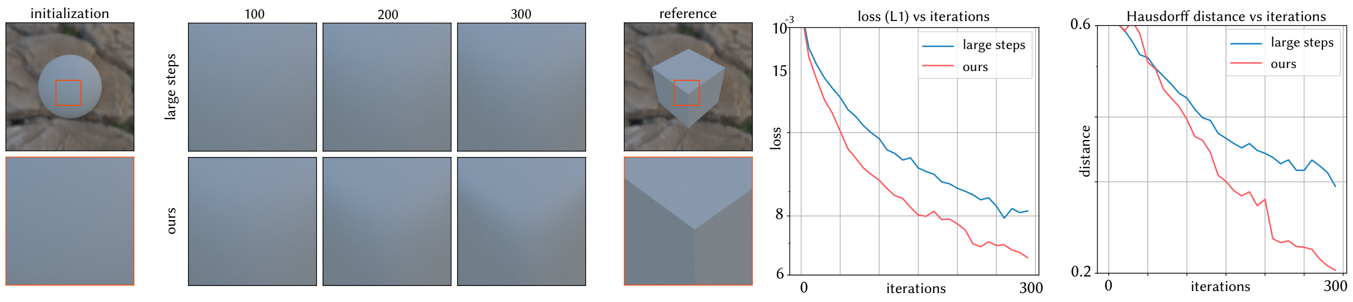

Fig. 6. Bias in mesh recovery. We keep the same setup at Figure 10 from the main text, but set the smoothing weight λ to be extremely large (= 999). As a result, both our method and Large Steps greatly oversmooth the gradient, resulting in a very uniform step across all vertices. Both therefore struggle to form the edges required in the cube in a reasonable number of iterations. Scene adapted from Kloppenheim 06 ©Greg Zaal.