Adaptive Manifolds for Real-Time High-Dimensional Filtering

(Supplementary Materials)

Eduardo S. L. Gastal* Manuel M. Oliveira†
Instituto de Informática – UFRGS

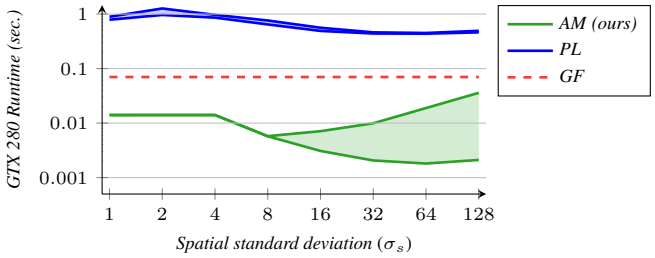


Figure 1: Performance on a GTX 280 GPU *of our adaptive-manifold filter (AM) versus the permutohedral lattice (PL), and the guided filter (GF). The vertical axis shows time, in seconds, to filter a 1 Megapixel RGB color image. The shaded areas represent performance changes when σ_r varies from 1 (bottom curve) to 0.05 (top curve). The guided filter performance curve, in dashed red, is based on performance numbers reported by Bauszat et al. [2011] on a GTX 285 GPU.*

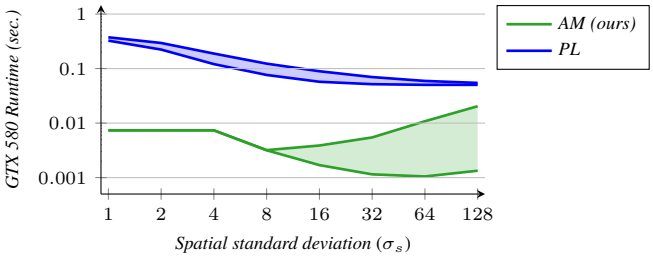


Figure 2: Performance on a GTX 580 GPU *of our adaptive-manifold filter (AM) versus the permutohedral-lattice filter (PL). The guided filter (GF) has no performance data available for such a GPU. The vertical axis shows time, in seconds, to filter a 1 Megapixel RGB color image. The shaded areas represent performance changes when σ_r varies from 1 (bottom curve) to 0.05 (top curve).*

See our project website for up-to-date information:

http://inf.ufrgs.br/~eslgastal/AdaptiveManifolds/

1 GPU Performance and Filtering Quality

Figures 1 and 2 show GPU performance graphs for 5-D color image filtering, comparing our *adaptive-manifold* filter (AM) against the *permutohedral lattice* (PL) [Adams et al. 2010], which is fastest color bilateral filter currently available, and also against the *guided filter* (GF) [He et al. 2010]. Due to the simple and parallel operations used by our approach, our filter is 3 to 200× faster than PL, depending on filtering parameters and GPU used. Furthermore, our filter is also 2 to 40× faster than the guided filter, while generating results indistinguishable from brute-force bilateral filtering (Figure 3).

The guided filter does not compute true Euclidean distances between pixels. As a result, it may introduce artifacts in the filtered images. Figure 4 shows one example where a noisy path-traced global illumination image, shown in (a), was filtered using our adaptive manifolds filter, shown in (b), and using the guided filter, shown in (c). Note how the guided filter introduces haloing artifacts in certain regions of the image. These artifacts are not present in the result produced by our filter. Further examples are presented and discussed in our paper.

2 Values of K for RGB Color Image Filtering

Table 1 shows several values for the number of manifolds (K), computed using Eq. (10) (from the paper) for performing RGB color image filtering. Note how these values follow the guidelines outlined in the beginning of Section 5.1: the number of manifolds should *increase* with increasing spatial standard deviation, and should *decrease* with increasing range standard deviation. Please refer to Section 5.1 of our paper for a more in-depth discussion.

σ_r \ σ_s	1	4	8	16	32	64	128
0.01	3	3	3	7	15	31	63
0.10	3	3	3	7	15	31	63
0.20	3	3	3	7	15	15	31
0.40	3	3	3	3	7	7	15
1.00	3	3	3	3	3	3	3

Table 1: *Several values of K computed for RGB color image filtering.*

3 Approximate Linearity

Appendix A in our paper proves that a good approximation for Eq. (1) can be obtained using nonlinear manifolds if they are approximately linear in all local neighborhoods. Steps 1 and 4 of our algorithm generate these approximately-linear manifolds by applying a low-pass filter h to pixels from the original signal. This section shows why this procedure indeed generates the desired results. Although this analysis is performed for 1-D signals, the same arguments generalizes to arbitrary dimensions due to the separability of the filtering operations.

Definition S.1: A 1-D function (*e.g.*, manifold) $\eta(x)$ is linear if its second derivative (or *curvature*) is zero for all x: $\partial_{xx}\eta = 0$. It is said to be approximately-linear in an interval $[a, b]$ if the total curvature over this interval is less than some small value ϵ:

$$\int_a^b \left|\partial_{xx}\eta\right| dx < \epsilon. \qquad \text{(i)}$$

Proposition S.1: For any $\epsilon > 0$, any interval $[a, b]$, and any signal f whose derivatives follow natural-image distributions [Weiss and Freeman 2007], there exists a low-pass filter h_σ with standard deviation σ for which $\eta = f * h_\sigma$ is approximately linear according

*eslgastal@inf.ufrgs.br
†oliveira@inf.ufrgs.br

to Eq. (i).

Proof. The derivative of a convolution can be decomposed as:

$$\partial_{xx}\eta = \partial_{xx}(f * h_\sigma) = (\partial_{xx} f) * h_\sigma. \tag{ii}$$

This convolution performs local averaging of the values $\partial_{xx} f$, which are realizations of a random variable. Eq. (ii) converges to the population's expected value $E[\partial_{xx} f]$ as more samples are averaged — *i.e.*, as the standard deviation σ of the filter increases. For a particular σ value, the variance of the estimator given by Eq. (ii) is proportional to $Var[\partial_{xx} f]/\sigma$. Thus, for some constant γ:

$$\left|(\partial_{xx} f) * h_\sigma - E[\partial_{xx} f]\right| < \frac{\gamma}{\sigma} Var[\partial_{xx} f]. \tag{iii}$$

The output of derivative filters applied to natural images have (non-Gaussian) distributions which peak at, and are symmetric about, zero [Weiss and Freeman 2007], and hence $E[\partial_{xx} f] \approx 0$. Plugging this into Eq. (iii) and integrating both sides in the interval $[a, b]$:

$$\int_a^b \left|(\partial_{xx} f) * h_\sigma\right| dx < \int_a^b \frac{\gamma}{\sigma} Var[\partial_{xx} f]\, dx. \tag{iv}$$

Evaluating the integral on the right-hand-side yields:

$$\int_a^b \left|(\partial_{xx} f) * h_\sigma\right| dx < \frac{\gamma}{\sigma}(b-a)\, Var[\partial_{xx} f]. \tag{v}$$

Since the standard deviation σ is a free parameter of the low-pass filter, for any $\epsilon > 0$ and any interval $[a, b]$ one can always find a filter h_σ with standard deviation σ sufficiently large such that the right-hand-side of Eq. (v) evaluates to less than ϵ, and thus $\eta = f * h_\sigma$ is approximately linear according to *Definition* S.1. □

Note that *Proposition* S.1 is not valid for a general signal f. One particular example would be the signal defined by $f(x) = x^2$, for which a convolution with any symmetric low-pass filter (disregarding boundary conditions) does not alter its curvature $\partial_{xx} x^2 = 2$.

4 Splatting onto the Adaptive Manifolds

The term $\Psi^{P_k}_{splat}(\hat{\eta}_{kj})$ in Eq. (20) defines a Gaussian distance-weighted splatting of a pixel's color f_j onto a flat P_k. Note that the parameter of the Gaussian is the distance of $\hat{p}_j$ to the flat P_k (in a space scaled by $\Sigma_\eta^{-1/2}$).

Our approach uses nonlinear manifolds instead of flats, thus, when splatting a pixel's color f_j onto an adaptive manifold M_k, one must compute the distance from $\hat{p}_j$ to M_k. Since this manifold is only approximately linear, this can be done in a few ways.

Following the term $\Psi^{P_k}_{splat}(\hat{\eta}_{kj})$ in Eq. (20), one way to compute the distance to a manifold would be to find the flat P_{kj} which approximates this manifold in the neighborhood around p_j. The Gaussian distance-weighted projection should then be performed using the closest distance between $\hat{p}_j$ and P_{kj}. However, finding this flat would require some sort of linear regression on points sampled from η_k; or the computation of tangent flats.

Another way of computing this distance is by projecting the point $\hat{p}_j$ onto the manifold M_k along the dimensions of the range $\mathcal{R}$. With this approach, the projection point on the manifold would be, by definition, $\hat{\eta}_{kj}$. Furthermore, since this projection is along $\mathcal{R}$, the complexity of computing the distance from $\hat{p}_j$ to $\hat{\eta}_{kj}$, used for splatting, is $O(d_\mathcal{R})$ (remember that $\hat{p}_j$ and $\hat{\eta}_{kj}$ have exactly the same coordinates in $\mathcal{S}$). This is the approach we use to splat onto the manifolds (Eq. (3)). However, notice that at the projection point $\hat{\eta}_{kj}$ on the manifold, the basis which locally spans the manifold (and which defines, in blurring stage, the direction of blurring) is not orthogonal to the direction of this projection (*i.e.*, not orthogonal to $\hat{\eta}_{kj} - \hat{p}_j$). Thus, this introduces an accuracy penalty, since the Gaussian is only truly separable for orthogonal directions. A similar error appears in other approaches which discretize the high-dimensional space (*i.e.*, it is a sampling error).

5 Sampling Rate for Recursive Filter

The recursive filter h described by Eq. (13), when applied twice (once in each direction) has discrete impulse response (for an impulse at $n = 0$) given by

$$h[n] = -\frac{1}{\sqrt{2}\,\sigma_l} \exp\left(-\frac{\sqrt{2}\,|n|}{\sigma_l}\right), \tag{vi}$$

where $|n|$ is the absolute value of n. The discrete-time Fourier transform of this filter is given by

$$H[\omega] = \frac{\sinh\left(\frac{\sqrt{2}}{\sigma_l}\right)}{\sqrt{2}\,\sigma_l\left(\cosh\left(\frac{\sqrt{2}}{\sigma_l}\right) - \cos(\omega)\right)}, \tag{vii}$$

where $\omega \in [-\pi, \pi]$ is the frequency parameter. Note that H is periodic outside the interval $[-\pi, \pi]$, since the filter is discrete-time. The cut-off frequency ω_c of h (*i.e.*, the frequency above which all frequencies are attenuated by h below a small threshold τ), is obtained solving $H[\omega_c] = \tau$ for ω_c. This yields

$$\omega_c = \operatorname{acos}\left(\cosh\left(\frac{\sqrt{2}}{\sigma_l}\right) - \frac{\sinh\left(\frac{\sqrt{2}}{\sigma_l}\right)}{\sqrt{2}\,\tau\,\sigma_l}\right). \tag{viii}$$

When this equation is solvable for $\omega_c \in \mathbb{R}$, any signal f filtered with h can be considered bandlimited to the interval $[-\omega_c, \omega_c]$. Thus, in practice, if the original signal f is represented by N samples, the filtered version of f, obtained with the filter h, should be represented by at least $N\,\omega_c/\pi$ samples. In our implementation, we represent this filtered signal with $min(N,\, 4\,N/\sigma_l)$ samples, which is equivalent to a cut-off frequency ω_c obtained with threshold $\tau = 0.0125$.

6 Eigenvector Computation

Proposition S.2: For a positive-definite matrix XX^T of size $d_\mathcal{R} \times d_\mathcal{R}$, with eigenvectors $v_1, \ldots, v_{d_\mathcal{R}}$ and eigenvalues $\lambda_1 > \lambda_2 \geq \ldots \geq \lambda_{d_\mathcal{R}} > 0$,

$$\text{if} \quad (\forall i \neq 1,\, \lambda_1 > \lambda_i) \quad \text{then} \quad \left(\lim_{m\to\infty} (XX^T)^m w \propto v_1\right);$$

where w is a random vector not orthogonal to v_1.

Proof. The eigenvectors $v_1, \ldots, v_{d_\mathcal{R}}$ form an orthonormal basis for $\mathbb{R}^{d_\mathcal{R}}$. The vector w can be expressed on this basis as

$$w = a_1 v_1 \;+\; a_2 v_2 \;+\; \cdots \;+\; a_{d_\mathcal{R}} v_{d_\mathcal{R}}, \tag{ix}$$

for scalars $a_1, \ldots, a_{d_\mathcal{R}}$. From this it follows that

$$\begin{aligned}
&(XX^T)^m w \\
&= (XX^T)^m (a_1 v_1 \;+\; a_2 v_2 \;+\; \cdots \;+\; a_{d_\mathcal{R}} v_{d_\mathcal{R}}) \\
&= a_1 (XX^T)^m v_1 \;+\; a_2 (XX^T)^m v_2 \;+\; \cdots \;+\; a_{d_\mathcal{R}} (XX^T)^m v_{d_\mathcal{R}} \\
&= a_1 \lambda_1^m v_1 \;+\; a_2 \lambda_2^m v_2 \;+\; \cdots \;+\; a_{d_\mathcal{R}} \lambda_{d_\mathcal{R}}^m v_{d_\mathcal{R}} \\
&= a_1 \lambda_1^m \left(v_1 + \frac{a_2}{a_1}\left(\frac{\lambda_2}{\lambda_1}\right)^m v_2 + \cdots + \frac{a_{d_\mathcal{R}}}{a_1}\left(\frac{\lambda_{d_\mathcal{R}}}{\lambda_1}\right)^m v_{d_\mathcal{R}}\right).
\end{aligned}$$

The last line is well defined since w is not orthogonal to v_1 (*i.e.*, $a_1 \neq 0$). Noting that

$$\text{if} \quad (\forall i \neq 1, \lambda_1 > \lambda_i) \quad \text{then} \quad \left(\lim_{m \to \infty} \left(\frac{\lambda_i}{\lambda_1} \right)^m = 0 \right),$$

we have

$$\lim_{m \to \infty} (XX^T)^m \, w \;=\; a_1 \lambda_1^m v_1 \;\propto\; v_1. \;\square$$

Observation: if the largest eigenvalue λ_1 is not unique (*i.e.*, the characteristic polynomial has a multiple root at λ_1, with multiplicity r), $(XX^T)^m w$ will converge to a linear combination of all r eigenvectors $v_1, \ldots, v_r$ associated with λ_1. This combination will be defined by the scalars $a_1, \ldots, a_r$ — *i.e.*, will be defined by the choice of random vector w. For our filter, this means that the pixels are isotropically distributed around the manifolds for the r directions of maximum variation. Thus this combination of eigenvectors will provide a good segmentation for Step 3 of our manifold creation process (Section 4).

Note also that using a value of $m + 1$ for the exponent always produces a better approximation for v_1 than using a value of m.

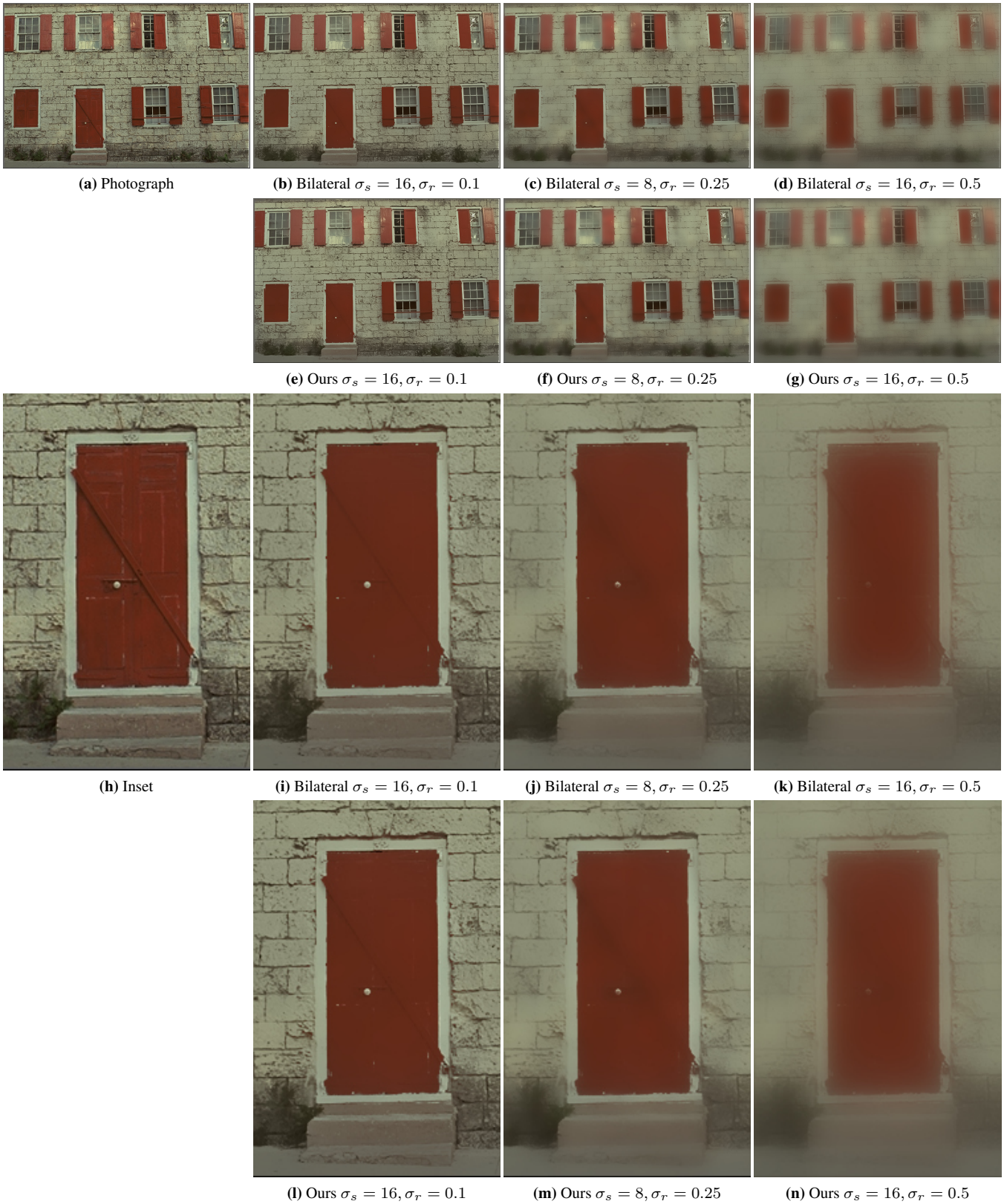
References

ADAMS, A., BAEK, J., AND DAVIS, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. *CGF 29*, 2, 753–762.

BAUSZAT, P., EISEMANN, M., AND MAGNOR, M. 2011. Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum 30*, 4, 1361–1368.

HE, K., SUN, J., AND TANG, X. 2010. Guided image filtering. In *ECCV*. Springer Berlin / Heidelberg, 1–14.

WEISS, Y., AND FREEMAN, W. T. 2007. What makes a good model of natural images? *CVPR 3*, 8, 1–8.



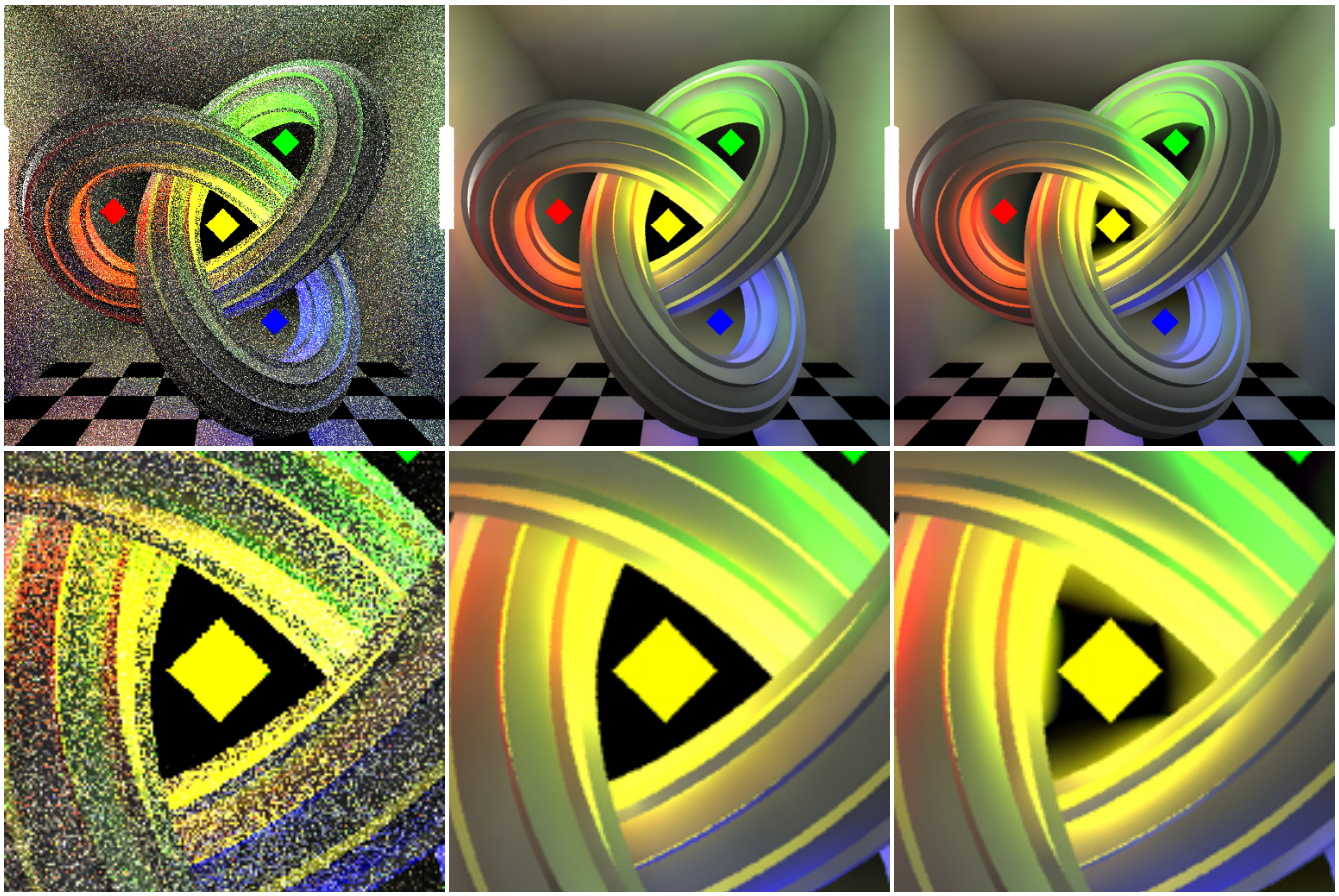
(a) Photograph **(b)** Bilateral $\sigma_s = 16, \sigma_r = 0.1$ **(c)** Bilateral $\sigma_s = 8, \sigma_r = 0.25$ **(d)** Bilateral $\sigma_s = 16, \sigma_r = 0.5$

(e) Ours $\sigma_s = 16, \sigma_r = 0.1$ **(f)** Ours $\sigma_s = 8, \sigma_r = 0.25$ **(g)** Ours $\sigma_s = 16, \sigma_r = 0.5$

(h) Inset **(i)** Bilateral $\sigma_s = 16, \sigma_r = 0.1$ **(j)** Bilateral $\sigma_s = 8, \sigma_r = 0.25$ **(k)** Bilateral $\sigma_s = 16, \sigma_r = 0.5$

(l) Ours $\sigma_s = 16, \sigma_r = 0.1$ **(m)** Ours $\sigma_s = 8, \sigma_r = 0.25$ **(n)** Ours $\sigma_s = 16, \sigma_r = 0.5$

Figure 3: *Our filter achieves a PSNR above 40 dB against brute-force bilateral filtering, which is practically indistinguishable visual difference.*



(d) Path-traced image **(e)** Our AM filter **(f)** Guided filter

Figure 4: *Example of global illumination filtering. The path-traced image in (a) was rendered with 1 sample per pixel for direct and indirect illumination. Our filter in (b) generates a smooth shading from the noisy input by working in a 4-D space composed of 3-D geometric normal vectors and 1-D scene depth. The guided filter in (c) works on the same 4-D space, however it introduces haloing artifacts in certain image regions (indicated by the red arrows). This happens since the guided filter does not compute true Euclidean distance between pixels, as discussed in our main text.*