Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination

Christoph Schied
NVIDIA
Karlsruhe Institute of Technology

Anton Kaplanyan
Chris Wyman
Anjul Patney
NVIDIA

Chakravarty R. Alla Chaitanya
NVIDIA
University of Montreal
McGill University

John Burgess
Shiqiu Liu
NVIDIA

Carsten Dachsbacher
Karlsruhe Institute of Technology

Aaron Lefohn
Marco Salvi
NVIDIA

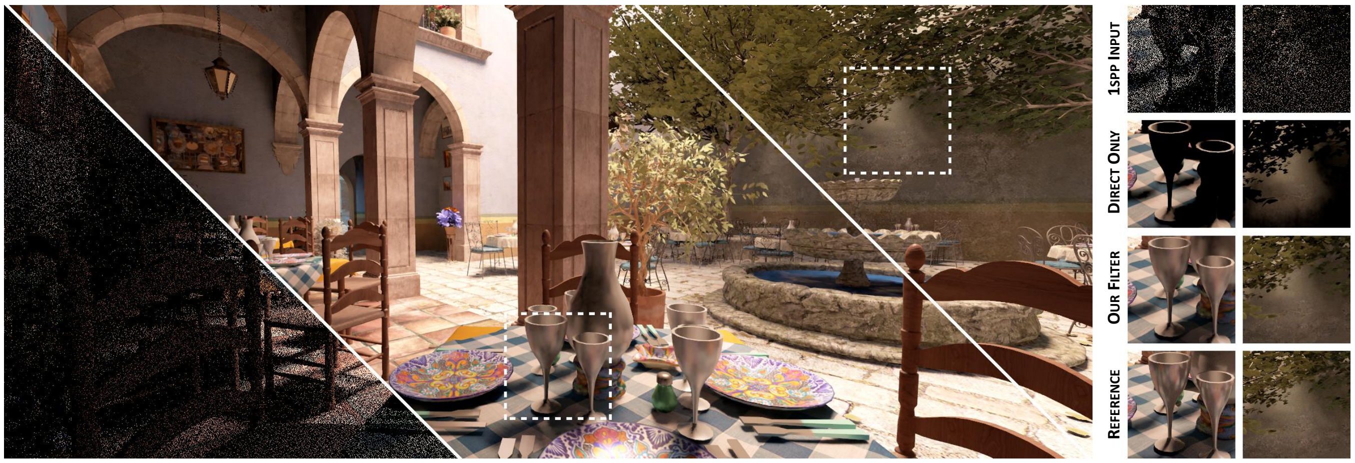


Figure 1: Our filter takes (left) 1 sample per pixel path-traced input and (center) reconstructs a temporally stable 1920×1080 image in just 10 ms. Compare to (right) a 2048 samples per pixel path-traced reference. Insets compare our input, our filtered results, and a reference on two regions, and show the impact filtered global illumination has over just direct illumination. Given the noisy input, notice the similarity to the reference for glossy reflections, global illumination, and direct soft shadows.

ABSTRACT

We introduce a reconstruction algorithm that generates a temporally stable sequence of images from one path-per-pixel global illumination. To handle such noisy input, we use temporal accumulation to increase the effective sample count and spatiotemporal luminance variance estimates to drive a hierarchical, image-space wavelet filter [Dammertz et al. 2010]. This hierarchy allows us to distinguish between noise and detail at multiple scales using local luminance variance.

Physically based light transport is a long-standing goal for real-time computer graphics. While modern games use limited forms of ray tracing, physically based Monte Carlo global illumination does not meet their 30 Hz minimal performance requirement. Looking ahead to fully dynamic real-time path tracing, we expect this to only be feasible using a small number of paths per pixel. As such, image reconstruction using low sample counts is key to bringing path tracing to real-time. When compared to prior interactive reconstruction filters, our work gives approximately 10× more temporally stable results, matches reference images 5–47% better (according to SSIM), and runs in just 10 ms (± 15%) on modern graphics hardware at 1920×1080 resolution.

CCS CONCEPTS

•**Computing methodologies** →Ray tracing;

KEYWORDS

global illumination, reconstruction, real-time rendering

ACM Reference format:
Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R. Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination. In *Proceedings of HPG '17, Los Angeles, CA, USA, July 28-30, 2017,* 12 pages.
DOI: 10.1145/3105762.3105770

1 INTRODUCTION

In recent years, path tracing [Kajiya 1986] emerged as the rendering algorithm of choice for film and visual effects [Keller et al. 2015]. This spurred development of advanced filter and reconstruction kernels that reduce Monte Carlo sampling's inevitable noise (e.g., see Zwicker et al. [2015]). These kernels allow noise-free reconstruction of images with dozens to hundreds of samples per pixel. With further algorithmic advances, we argue that a shift to path-traced global illumination also lies on the horizon for real-time graphics.

Both video games and film recently migrated from empirical models to physically-based shading [McAuley and Hill 2016], but the simplified light transport available in rasterization continues to push developers to consider ray tracing [Whitted 1980] for accurate shadows and reflections, and multi-bounce global illumination. But current ray tracing performance is limited to around 200–300 Mrays/sec [Binder and Keller 2016; Wald et al. 2014], giving just a few rays per pixel at 1920×1080 and 30 Hz. This number is even lower for production usage with dynamic acceleration structures, large scenes, and variable CPU/GPU performance. Therefore, with multiple rays per path and the trends towards higher resolutions and refresh rates, practical performance is not likely to exceed *one path per pixel* for the foreseeable future. By developing a reconstruction filter that respects this constraint, we aim to make real-time path tracing a reality much sooner.

We extend Dammertz et al.'s [2010] hierarchical, wavelet-based reconstruction filter to output temporally stable global illumination, including diffuse and glossy interreflections, and soft shadows from a stream of one sample per pixel (spp) images (c.f., Figure 1).

Reconstruction at very low sampling rate presents many challenges. High variance from poor sampling obscures high-frequency signals and, with just one sample, distinguishing between sources of noise proves difficult. For example, noise from spatially sampling a high-frequency surface texture becomes conflated with variance introduced by light transport and visibility events.

Given our real-time performance target, our filter leverages prior frames' samples to help isolate fine details and decouple sources of noise, even in the context of animated scenes. Today, at 1920× 1080, it runs in around 10 ms on current GPUs, opening the door for future real-time path-traced global illumination. Our specific contributions include:

- An efficient and temporally stable algorithm for real-time reconstruction from single path-per-pixel inputs, built using a combination of filters guided by estimated variance in spatial and temporal domains.
- A temporal pass that computes per-pixel variance estimates using information from past frames, and falls back to a spatial estimate during disocclusions and other temporal undersampling events.
- A spatial pass that builds on prior work (Dammertz et al. [Dammertz et al. 2010]) to filter input color through multiple wavelet iterations. Our spatial pass starts by using the temporal estimate of variance, but updates it during each iteration to improve its reliability.
- New, scene-agnostic geometric edge-stopping functions.

While our filter requires no scene-dependent parameters or knowledge of the underlying light transport algorithm, it assumes input of a noise-free G-Buffer [Saito and Takahashi 1990]. This means we do not support stochastically sampled visibility for depth-of-field or motion blur. However, today's games approximate these effects well using post-processing techniques.

2 RELATED WORK

Real-time global illumination has eluded researchers for many years. Current approximations often rely on precomputing or caching light transport computations [Kajiya 1986], either on surfaces or in sparsely sampled volumes. To augment cached lighting, additional techniques allow inclusion of specific desired effects, such as ambient occlusion, glossy screen-space reflections, and soft shadowing [Ritschel et al. 2012]. While often plausible and pleasing, the resulting lighting is far from realistic. To reach true realism in real-time, we believe developers must switch to physically-based Monte Carlo light transport.

Path tracers account for light transport by stochastic sampling, allowing accurate rendering of distribution effects [Cook et al. 1984] including depth of field, motion blur, caustics, soft shadows, and global illumination. Beyond increased realism, path tracing also avoids use of per-effect rendering algorithms, potentially reducing code complexity. This transition already occurred in offline rendering [Keller et al. 2015], leading to the development of numerous filters designed to remove residual stochastic noise. Zwicker et al. [2015] provides an excellent survey of these techniques.

Monte Carlo Denoising. Denoising filters reduce variance—at the expense of introducing bias—by combining multiple per-pixel Monte Carlo estimators. They aim to smooth the output while preserving any sharp image features, such as edges and surface details. Most offline denoisers spatially filter over input images with tens to hundreds of samples per pixel, with computation times measured in seconds or minutes.

Regression-based approaches [Bitterli et al. 2016; Moon et al. 2014, 2015, 2016; Rousselle et al. 2012] have been shown to yield good results at higher sample rates (≥ 128), however, these filters do not work reliably at low sample rates since they are sensitive with respect to outliers. Furthermore, they are not applicable to real-time rendering given their high computational complexity.

Munkberg et al. [2016] shows operating in texture space allows simpler filters that benefit from both spatial and temporal reuse. However, scaling this approach to large assets for real-time rendering is nontrivial. Filtering can also occur in path space [Keller et al. 2016], but this uses expensive kNN-searches and couples rendering and filtering algorithms. The black box nature of image-space filters is an attractive feature for real-time applications. While path-space metrics can improve image-space filtering [Gautron et al. 2014], these metrics remain costly and only converge progressively over many frames at extremely low sample counts.

Some of the aforementioned methods are designed to be used to to drive adaptive sampling that locally enhances quality, however, our requirements differ. To maintain stable and predictable performance we prefer to avoid adaptive sampling, fixing sampling rate and allowing bias to vary.

Interactive Monte Carlo Denoising. Seminal work by Durand et al. [2005] derives an optimal filter footprint via frequency analysis of a Monte Carlo sampled light field. Later extensions apply to soft shadows, motion and defocus blur, and indirect illumination using *linear* sheared filters (Egan et al. [2011a; 2011b; 2009]) and axis-aligned filters (Mehta et al. [2012; 2013; 2014]). Using separable sheared filters leads to higher performance [Hasselgren et al. 2015; Munkberg et al. 2014; Vaidyanathan et al. 2015; Yan et al. 2015], although frequency analysis increases total cost and typically requires at least 4–16 spp for good results. Additionally, in high-frequency regions frequency-space filters reduce filter footprints to small regions, leading to higher variance without adaptive sampling.

Instead of shrinking the filter footprint, non-linear filters adapt filter weights to preserve salient features. Pioneering work on non-linear Monte Carlo denoising uses outlier removal [Lee and Redner 1990], smooth energy redistribution [Rushmeier and Ward 1994], and anisotropic diffusion [McCool 1999]. An edge-preserving bilateral filter [Tomasi and Manduchi 1998] can be applied to Monte Carlo denoising [Xu and Pattanaik 2005]. A reformulated cross (or joint) bilateral filter [Eisemann and Durand 2004; Petschnigg et al. 2004] replaces each pixel by a weighted average of nearby pixels, using Gaussian-distributed weights that account for distance, color, and other differences to *guide* images via edge-stopping functions. Accounting for geometric information in the edge-stopping function improves the cross bilateral filter's robustness under input noise.

Weighting the edge-stopping functions' components per-pixel further improves robustness to spatially varying sampling noise. Li et al. [2012], Rousselle et at. [2013], Kalantari et al. [2013], and Bauszat et at. [2015a] create a bank of candidate filters and select or interpolate filters per pixel depending on estimated input variance or filter error. Kalantari et al. [2015] propose applying a small neural network to control a cross bilateral filter's per-pixel feature weights. These filters can apply at even relatively low sampling rates, but rely on significant preprocessing or smoothing error estimates and they currently do not run in real-time.

Several fast approximations to cross bilateral filters exist, including guided image filters [Bauszat et al. 2011; He et al. 2013], edge-aware wavelets [Dammertz et al. 2010; Fattal 2009], and adaptive/linear manifolds [Bauszat et al. 2015b; Gastal and Oliveira 2012]. Generally, these approximations introduce ringing and haloing artifacts. With coarsely sampled inputs (i.e., ≤ 4 spp) image structure may be impossible to infer, so establishing important features via filter guide images is vital to prevent over blurring and loss of detail.

Temporal Filtering. Exploiting temporal information across multiple frames helps address spatial filters' shortcomings and improve temporal stability at low sampling rates. Delbracio et al. [2014] consider ray histograms across three frames to reduce flickering and Meyer and Anderson [2006] compute a PCA over all frames, discarding insignificant bases to improve temporal stability. Zimmer et al. [Zimmer et al. 2015] use a path-space decomposition for motion estimation and apply denoising on multiple buffers. But these methods require an input set of precomputed frames rather than temporally filtering only over previous frames, as required for real-time.

Interactive filters often *reproject* samples from one frame to another based on motion vectors [Nehab et al. 2007; Walter et al. 1999]. This is similar to view interpolation [Chen and Williams 1993], which reprojects samples from a prerendered set of images to generate novel viewpoints. Recently, reprojection has gained popularity through the broad adoption of temporal anti-aliasing (TAA) [Karis 2014]. TAA draws inspiration from temporally amortized supersampling [Yang et al. 2009], but rather than discarding stale samples, they are conditioned to match the color at their reprojected locations. Patney et al. [2016] improve on this by estimating the statistical distribution of colors.

Mara et al. [2017] independently and contemporaneously developed a denoising method that operates on full paths rather than separately filtering direct and indirect light; we look forward to comparison as future work.

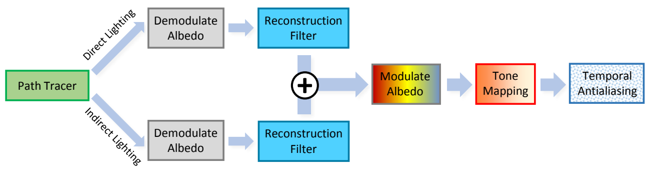


Figure 2: We reconstruct direct and indirect lighting separately. Prior to reconstruction, surface albedo is demodulated based on the rasterized G-buffer. This focuses our reconstruction on light transport, rather than denoising high frequency texture detail. After filtering, we recombine direct and indirect light with primary albedo, then apply tone mapping and temporal antialising.

3 RECONSTRUCTION PIPELINE

Below we provide a high-level overview of our reconstruction pipeline (see Figure 2), including our rasterization and path tracing inputs, how we isolate noise sources by separating components, our reconstruction filter, and our post processing steps. Section 4 covers the core reconstruction algorithm in greater detail.

Path Tracing. As input to our reconstruction filter, we use standard path tracing with next event estimation to generate 1 spp color samples. Our path tracer includes optimizations to better utilize available GPU resources, including use of the rasterizer to efficiently generate primary rays. This provides a noise-free G-buffer [Saito and Takahashi 1990] containing additional surface attributes used to steer our reconstruction filter (see Section 4).

A low-discrepancy Halton sequence [Halton and Smith 1964] is used to sample light sources and scattering directions. We loop through a small set of Halton samples (e.g., 16), as our temporal filters' exponential moving average loses contributions from earlier samples after a few frames. For non-diffuse surfaces after a path's first scattering event, we apply path space regularization [Kaplanyan and Dachsbacher 2013]. Regularization essentially increases surface roughness in secondary scattering events, allowing the path tracer to find contributions for indirect bounces by connecting to the light source, even with highly glossy materials. This increases light transport robustness and allows paths to contribute more uniformly. We also restrict the path length to one additional scattering event per path, limiting computational costs. We thus trace one ray to find indirectly visible surfaces, plus two shadow rays to connect primary and secondary hit points to a light source.

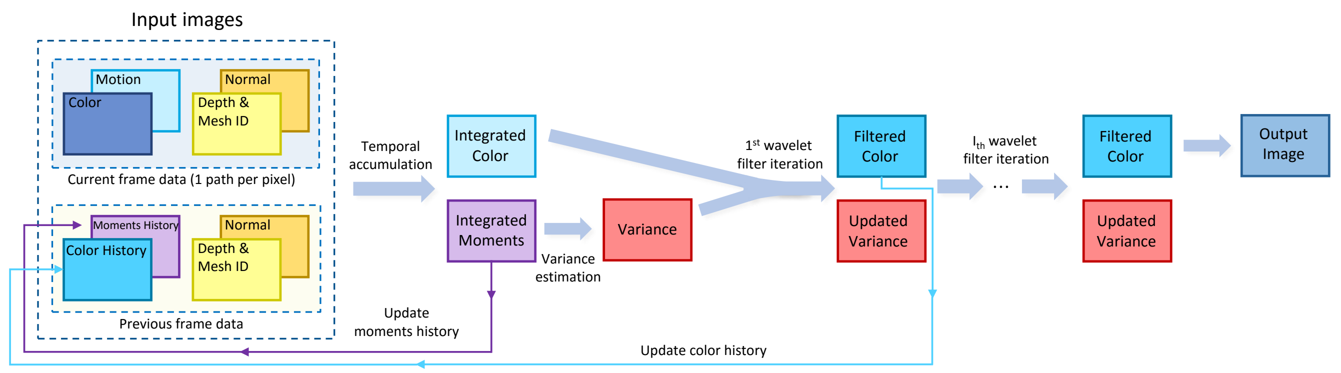


Figure 3: An overview of the core reconstruction filter (shown as blue boxes in Figure 2). We temporally filter deep frame buffers (left) to get temporally integrated color and moments. We use an estimated luminance variance to drive an edge-aware spatial wavelet filter (center). The wavelet filter's first iteration provides a color and moment history to help temporally filter future frames. The reconstructed result (right) feeds back into Figure 2, where we remodulate albedo, perform tonemapping, and run a final temporal antialiasing pass.

Our path tracer outputs direct and indirect illumination separately. This enables the filter to account for local smoothness independently in both components, and allows better reconstruction of poorly sampled shadow edges. Separation appears to double cost, but since many steps use the rasterized G-buffer, significant work is shared.

Reconstruction. We first demodulate surface albedo (including textures and spatially-varying BRDFs) of directly visible surfaces from our sample colors. This avoids our filter having to prevent overblurring of high-frequency texture details. In other words, we filter *untextured illumination components* and reapply texturing after reconstruction. Besides removing the need of preventing the filter from overly blurring texture details, this also increases the possible spatial reuse for neighboring samples. In case of multi-layer materials we add the per-layer albedos, weighted by their sampling probability.

Our reconstruction performs three main steps: temporally accumulating our 1 spp path-traced inputs to increase effective sampling rate, using these temporally augmented color samples to estimate local luminance variance, and using these variance estimates to drive a hierarchical à-trous wavelet filter. Figure 3 provides an overview, and Section 4 dives into these steps in greater detail. After reconstruction, we (re-)modulate the filter output with the surface albedo.

Post Processing. After reconstruction, we perform post processing similar to many of today's real-time renderers. Our filtered result goes through a tone mapping operator to handle a high dynamic range. Finally, we perform temporal antialiasing [Karis 2014] to increase temporal stability and filter aliasing along geometric edges that our reconstruction filter preserves.

4 SPATIOTEMPORAL FILTER

Our reconstruction filter takes a 1 spp path-traced color buffer as input, along with a rasterized G-buffer [Saito and Takahashi 1990] and *history buffers* from the prior frame's reconstruction. We output a reconstructed image and the following frames' history buffers.

Our G-buffer contains depth, object- and world-space normals, mesh ID, and screen-space motion vectors generated from a rasterization pass for primary visibility. Our history buffers include temporally integrated color and color moment data along with the prior frame's depths, normals, and mesh IDs. To increase robustness, we deliberately avoid using scene-specific information, such as light positions, shape, or other scene properties, and we do not assume any particular light transport method.

Figure 3 highlights the main steps in our filtering pipeline. Section 4.1 describes our temporal sample accumulation, Section 4.2 presents our spatiotemporal estimation of luminance variance, and Section 4.3 details our variance-guided wavelet filter. Section 4.4 provides the edge-stopping functions that control our filter weights.

4.1 Temporal filtering

With temporal antialiasing (TAA) [Karis 2014] now widely adopted for amortized supersampling in video games, it seems natural to apply it to real-time path tracing. Unfortunately, color-based temporal filtering introduces artifacts when applied to very noisy inputs. We minimize these artifacts and increase the effective sample count by instead adopting a geometry-based temporal filter inspired by ideas from Nehab et al. [2007] and Yang et al. [2009].

As in TAA, we require a 2D motion vector associated with each color sample C_i for frame i. This describes geometric motion from the prior frame, and allows us to backproject C_i to its screen space location in the prior frame. By accessing a *color history* buffer, output by our filter in the prior frame, we can continuously accumulate color samples over multiple frames. For each C_i we backproject to find sample C_{i-1} from the color history buffer, and compare the two samples' depths, object-space normals, and mesh IDs to determine if they are consistent (i.e., on the same surface). These consistency tests use empirical similarity metrics similar to the fragment merge heuristics in prior work [Jouppi and Chang 1999; Kerzner and Salvi 2014]. Consistent samples are accumulated as a new *integrated color* C'_i via an exponential moving average:

$$C'_i = \alpha \cdot C_i + (1 - \alpha) \cdot C'_{i-1},$$

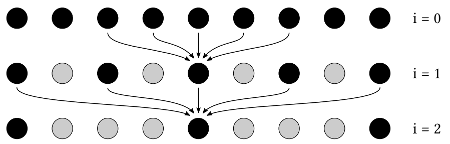


Figure 4: Illustration of three levels of the 1D à-trous wavelet transform. Arrows always point to the non-zero coefficients (black dots) of the i_{th} filter kernel, whose number remain constant. Each filter iteration increases the kernel footprint by introducing 2^{i-1} zero-elements (grey dots) between the initial entries, depicting elements accounted for by the undecimated transformation.

where α controls the temporal fade, trading temporal stability for lag. We found $\alpha = 0.2$ worked best. Our motion vectors currently handle camera and rigid object motion, though more complex transformations are possible with the addition of more data in our history buffer and slightly more involved consistency tests.

To improve image quality under motion we resample C_{i-1} by using a 2×2 tap bilinear filter. Each tap individually tests backprojected depths, normals and mesh IDs. If a tap contains inconsistent geometry, the sample is discarded and its weight is uniformly redistributed over consistent taps. If no taps remain consistent, we try a larger 3×3 filter to help find thin geometry such as foliage. If we still fail to find consistent geometry, the sample represents a disocclusion, so we discard the temporal history and use $C_i' = C_i$.

4.2 Variance estimation

To locally adjust the à-trous wavelet filter to the signal, we estimate the per-pixel variance of color luminance using temporal accumulation as an efficient proxy for detecting noise. The key idea is that our reconstruction should avoid changing samples in regions with little or no noise (e.g., fully shadowed regions) while filtering more in sparsely sampled, noisy regions. By analyzing the different samples over time, the filter detects the reliability of a specific sample. Note that spatially computed variance estimates can only provide an imperfect proxy for noise: noise increases variance, but variance can occur without noise.

We estimate per-pixel luminance variance using μ_{1_i} and μ_{2_i}, the first and second raw moments of color luminance. To collect sufficiently many samples for a meaningful estimate we temporally accumulate these moments, reusing the geometric consistency tests. We then estimate our temporal variance from the *integrated moments* μ'_{1_i} and μ'_{2_i} using the simple formula $\sigma_i'^2 = \mu'_{2_i} - \mu'^2_{1_i}$.

Camera motion, animations, and viewport boundaries all cause disocclusion events, which impact the quality of our variance estimates. Where our temporal history is limited (<4 frames after a disocclusion), we instead estimate the variance $\sigma_i'^2$ spatially, using a 7×7 bilateral filter with weights driven by depths and world-space normals. Essentially, for a few frames after a disocclusion our filter relies on a spatial estimate of variance until the temporal accumulation has collected sufficient data for a stable estimate.

4.3 Edge-avoiding à-trous wavelet transform

The à-trous wavelet transform hierarchically filters over multiple iterations, each with increasing footprint but a constant number of non-zero elements (see Figure 4). Discarding detail coefficients smooths the input while edge-stopping functions preserve sharp details by limiting filter extent at boundaries.

The rasterized G-buffer contains no stochastic noise, allowing us to define edge-stopping functions that identify common surfaces using G-buffer attributes. Our implementation follows the work by Dammertz et al. [2010] that realizes each step of an edge-aware à-trous wavelet decomposition using a 5×5 cross-bilateral filter with weight function $w(p,q)$ between pixels p, q:

$$\hat{c}_{i+1}(p) = \frac{\sum_{q \in \Omega} h(q) \cdot w(p,q) \cdot \hat{c}_i(q)}{\sum_{q \in \Omega} h(q) \cdot w(p,q)}, \tag{1}$$

where $h = \left(\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16}\right)$ is the filter kernel and Ω is the gathered filter footprint.

The weight function $w(p,q)$ typically combines geometrical and color based edge-stopping functions [Dammertz et al. 2010]. Our novel weight function instead uses depth, world-space normals, as well as the luminance of the filter input:

$$w_i(p,q) = w_z \cdot w_n \cdot w_l \,. \tag{2}$$

Before applying our wavelet filter we *tune* the luminance edge-stopping function, based on the local estimate of luminance variance (see Section 4.4). We then apply the wavelet filter to our temporally integrated color as per Equation 1, and by assuming our variance samples are uncorrelated we filter as follows:

$$\mathrm{Var}(\hat{c}_{i+1}(p)) = \frac{\sum_{q \in \Omega} h(q)^2 \cdot w(p,q)^2 \cdot \mathrm{Var}(\hat{c}_i(q))}{\left(\sum_{q \in \Omega} h(q) \cdot w(p,q)\right)^2}.$$

We use the result to steer the edge-stopping functions for the next level of the à-trous transform. Our reconstruction uses a five-level wavelet transform, giving an effective 65×65 pixel filter footprint.

As part of our wavelet transform, we output the filtered color from the first wavelet iteration as our *color history* used to temporally integrate with future frames (see Figure 3). While we could use filtered colors from other wavelet levels, we empirically found using the first wavelet iteration for temporal integration best balances improved temporal stability with bias from spatial filtering.

4.4 Edge-stopping functions

Given our real-time requirement, we chose the three edge-stopping functions in Equation 2 to maximize temporal stability and robustness, potentially in exchange for increased spatial bias. Each function's ability to reject samples is individually controlled by parameters σ_z, σ_n and σ_l . While a range of values for these parameters are effective, we found through experimentation that $\sigma_z = 1$, $\sigma_n = 128$ and $\sigma_l = 4$, work well on all scenes we tested. For this reason, we do not expose these parameters to the user.

Depth. Realistic scenes often contain large variations in geometric scale, especially in open landscapes. This makes global edge-stopping functions difficult to control. We thus assume a local linear model for the surface depths and measure deviation from its clip-space plane. We estimate the local depth model using screen-space partial derivatives of clip-space depth. This give a weight function defined as:

$$w_z = \exp\left(-\frac{|z(p) - z(q)|}{\sigma_z |\nabla z(p) \cdot (p-q)| + \varepsilon}\right), \tag{3}$$

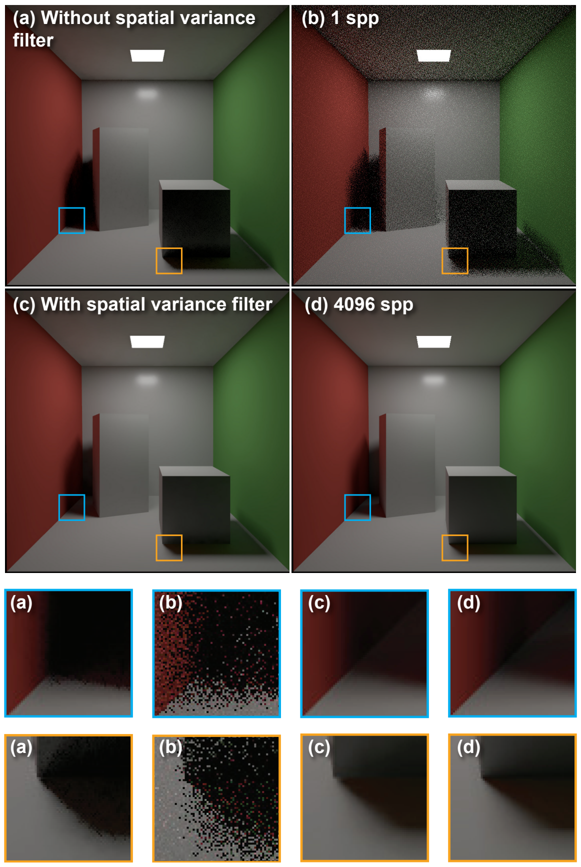

(a) Without spatial variance filter
(b) 1 spp
(c) With spatial variance filter
(d) 4096 spp
(a)
(b)
(c)
(d)
(a)
(b)
(c)
(d)

Figure 5: Low sample counts cause (a) underestimation of the variance where paths with lighting contributions occur with low probability (e.g., see (b) input samples). Filtering variance with (c) a small spatial filter improves results in these regions, giving results closer to (d) our reference.

where ∇z is the gradient of clip-space depth with respect to screen-space coordinates, and ε is a small value to avoid division by zero.

Normal. We adopt a cosine term for our edge-stopping function on world-space normals:

$$w_n = \max(0, n(p) \cdot n(q))^{\sigma_n}, \tag{4}$$

for an input normal $n(p)$ at point p on the image plane. Prior work in mesh simplification and anti-aliasing algorithms uses similar terms to control whether to merge two surfaces together.

Luminance. A key aspect of our luminance edge-stopping function is its ability to automatically adapt to all scales by re-normalizing luminance based on its local standard deviation. But operating at low sample counts introduces instabilities in our estimates of variance and standard deviation; this can introduce artifacts. To avoid these, we pre-filter our variance image using a 3×3 Gaussian kernel, which significantly improves reconstruction quality (see Figure 5). Our luminance edge-stopping function then becomes:

$$w_l = \exp\left(-\frac{|l_i(p) - l_i(q)|}{\sigma_l \sqrt{g_{3x3}(\mathrm{Var}(l_i(p))) + \varepsilon}}\right), \tag{5}$$

for a Gaussian kernel g_{3x3} and luminance $l_i(p)$ at position p. Since the luminance variance tends to reduce with subsequent filter iterations, the influence of w_l grows with each iteration, preventing overblurring.

Note that this Gaussian prefilter is only used to drive the luminance edge-stopping function, and it is not applied to the variance image propagated to the next iteration of the wavelet transform.

5 RESULTS AND DISCUSSION

We evaluate our new spatiotemporal variance-guided filter on a number of metrics, including final image quality, performance, and temporal stability. Our evaluation focuses on 1 spp images, since our design decisions all revolve around targeting this sampling rate. Our prototype uses OpenGL to generate primary visibility and to implement SVGF. Secondary and shadow rays are traced using OptiX [Parker et al. 2010]. All reported performance results use an NVIDIA TITAN X (Pascal).

5.1 Comparison With Existing Work

We compare to various denoising techniques, implemented independently in our common framework based on their original papers. For general image-space filtering, we compare to the edge-avoiding à-trous wavelet filter *(EAW)* by Dammertz et al. [2010], SURE-based filter *(SBF)* by Li et al. [2012], and the learning-based filter *(LBF)* by Kalantari et al. [2015]. For direct illumination (shadows), we also compare SVGF against the axis-aligned filter *(AAF)* by Mehta et al. [2012].

Our reconstruction filter relies on temporal filtering, thus operating at higher effective sample counts when using our temporal history buffer. Generally, this allows SVGF to achieve significantly better quality than prior published work, both visually (see Figure 6) and in terms of RMSE and SSIM (Table 1).

Additionally, Figure 10(a) shows that SVGF provides 10× less frame-to-frame variability, providing much higher temporally stability. We refer to the supplementary video for visual temporal stability comparisons.

The rest of our comparative evaluations augment prior work using our temporal accumulation and postprocess temporal antialiasing. Since temporal accumulation increases effective sample count, we feel this provides a level playing field for comparing our variance-guided reconstruction filter. While the original EAW, SBF, LBF, and AAF algorithms use no temporal accumulation, we saw significant improvement in both image quality (Table 1) and temporal stability (Figure 10(b)). For clarity, we refer to the implementations without temporal improvements as EAW_{orig}, SBF_{orig}, and LBF_{orig}.

AAF follows the description by Mehta et al. [2012]. Since our ray budget only allows for one shadow ray per pixel, we gather minimum and maximum occlusion distance in a fixed spatial window. We apply temporal filtering to the estimated occlusion distance as well as the visibility input.

Our SBF [2012] implementation does not perform any adaptive sampling. We only have one sample per pixel, so we estimate color variance using a small spatial window. We follow the authors' advice and skip the renormalization of noise-free feature distance

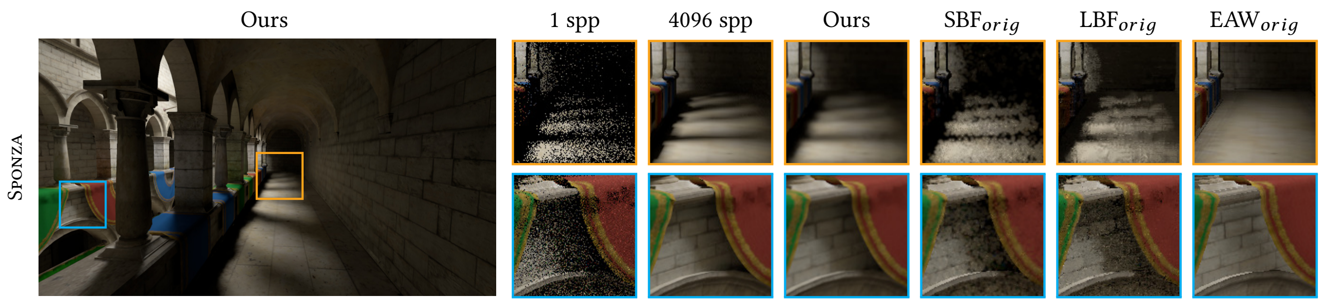


Figure 6: Frames captured from animated sequences comparing the image quality of our algorithm to reimplementations of prior work (per the original publications). SBF$_{orig}$, LBF$_{orig}$, and EAW$_{orig}$ all provide much lower image quality than our SVGF using only 1 spp. Due to increased effective sample count from temporal filtering, we modify comparative techniques in all other results to include our temporal reuse, for fairer comparison (e.g., as in Figure 7).

functions (e.g. depth, normal, albedo). We run the filter on temporally accumulated, demodulated samples containing both direct and indirect illumination.

LBF [2015] uses the pre-trained network weights provided with their sample code. We did not retrain on our scenes, and all network inputs were computed as described in their paper. We provide the filter with temporally filtered color input, containing both direct and indirect illumination. The inputs to the network include direct and indirect albedo, as well as first-bounce visibility information, therefore we did not apply demodulation to the samples.

Since our filter takes spatially filtered samples as input to our temporal accumulation (see Figure 3), our spatial filter technically does not use inputs *identical* to our comparison filters. However, all results are rendered deterministically, so all filters are run over an identical set of path traced samples at identical sample locations.

5.2 Image Quality

We evaluate image quality in multiple scenes. Sponza and GlossySponza evaluate isolated diffuse and highly glossy illumination. RedRoom, SanMiguel, and Classroom provide complex geometric and material interactions that resemble real-world environments. Animations include moving lights in RedRoom and Sponza and include moving camera in Sponza, GlossySponza, SanMiguel, and Classroom. We rendered animated sequences at 60 Hz, providing 1 spp path-traced inputs to each reconstruction technique.

Figure 7 selects frames in these animations for detailed comparisons, using insets to highlight key differences in our results. As with the rest of our results, this figure uses temporally augmented versions of prior work. Table 1 quantifies differences to a 4096 spp reference using Root-Mean Square Error (RMSE) and Structural Similarity Index (SSIM) [Wang et al. 2004] metrics. Our supplementary material provides video comparisons of final animations.

Qualitatively, our reconstructed results are more faithful to the reference than previous work. It also provides plausible results in areas with high geometric detail and many disocclusions, e.g., the foliage in SanMiguel. RMSE and SSIM support these observations, showing SVGF gives better or equal results compared to prior work. The SSIM of images reconstructed using prior work is between 2% and 25% lower than that of our algorithm.

Due to globally adjusted edge-stopping functions, the EAW filter fails to retain prominent details. This is particularly apparent in the soft shadows in Sponza and glossy reflections in GlossySponza. The EAW filter also suffers from suboptimal geometry-based edge-stopping functions, which cause halos in the reconstructed output (Figure 7, Sponza, cyan inset). LBF loses surface texture and exhibits noise in shadow penumbras. SBF captures many important features, but it exhibits strong mid-frequency noise.

Effectiveness of temporal variance estimation. In order to test the ability of SVGF to preserve details we designed a synthetic test scene with structured noise with varying sampling density. Figure 9 shows the input of this test and compares the results for our spatial fallback and spatiotemporal filters against reference. The input test pattern consists of an image where for each pixel we sample from the following distribution

$$f(x,\alpha) = \begin{cases} 0 & \text{if } x < \alpha, \\ \frac{1}{1-\alpha} & \text{otherwise,} \end{cases} \tag{6}$$

with $\alpha \in [0, 1)$ to estimate the integral $\int_0^1 f(x,\alpha)\,dx = 1$. From left to right, with varying α the variance is gradually increased.

When temporal reprojection fails, our reconstruction filter does not have any history to estimate statistics and has to rely on a spatial estimate. The temporal variance estimate allows the filter to retain sharpness in regions with lower noise whereas the spatial variance has higher amounts of bias across the whole image. Note that our filter fails to faithfully reconstruct the structure after a certain sampling probability (right side of Figure 9), however the temporal variance estimation pushes the breaking point further.

5.3 Performance

We designed SVGF for reconstruction in real-time contexts, so here we try to characterize its performance. Figure 8 shows frame execution times for our filter during a 1280×720 flythrough of the SanMiguel courtyard, which we believe includes a realistic amount of thin detail and disocclusions. Throughout the sequence, SVGF's cost varies between 4.1 and 5.8 ms. Excluding the first frame, performance variation is under 15% of the average 4.4 ms cost. This variability comes entirely from the variance estimation (Section 4.2), which reverts to a cross-bilateral filter for variance estimates when temporal reprojections fail. Frame cost then increases with number of disocclusions, e.g., frames with lots of foliage. Worst-case frames

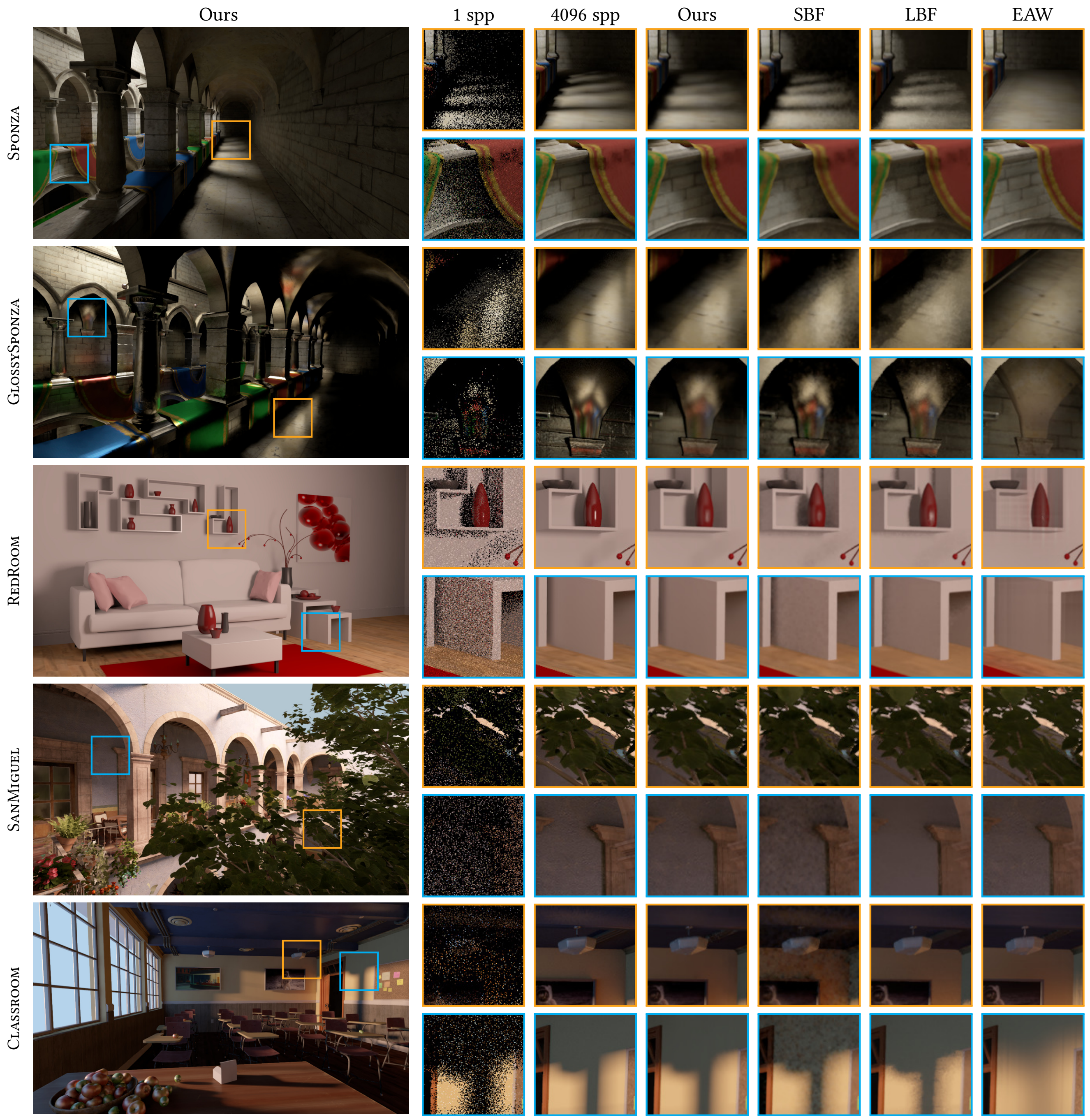


Figure 7: Frames captured from animated sequences. Our filter captures and preserves prominent structures and provides plausible, noise-free results. While SBF reconstructs most structure, it fails to fully denoise the input. LBF has difficulty in regions with low sample density, overblurring texture detail and creating a facetted flat appearance. EAW uses a globally-adjusted color edge stopping function, which fails to retain important structure.

Table 1: RMSE and SSIM difference comparing to a 4096 spp reference. Parenthesized numbers indicate metrics for filters augmented by our temporal filtering. Our algorithm performs consistently equal to or better than previous work.

Filter	Sponza		GlossySponza		RedRoom		SanMiguel		Classroom	
	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM	RMSE	SSIM
Ours	0.022	0.947	0.049	0.779	0.021	0.972	0.059	0.863	0.027	0.940
EAW	0.043 (0.040)	0.902 (0.909)	0.089 (0.087)	0.585 (0.586)	0.049 (0.048)	0.923 (0.924)	0.088 (0.070)	0.767 (0.818)	0.059 (0.052)	0.841 (0.876)
LBF	0.032 (0.022)	0.777 (0.924)	0.058 (0.052)	0.655 (0.764)	0.033 (0.031)	0.915 (0.946)	0.074 (0.058)	0.651 (0.833)	0.047 (0.035)	0.787 (0.912)
SBF	0.060 (0.031)	0.500 (0.858)	0.058 (0.048)	0.648 (0.757)	0.038 (0.027)	0.822 (0.957)	0.088 (0.061)	0.571 (0.815)	0.072 (0.035)	0.610 (0.884)

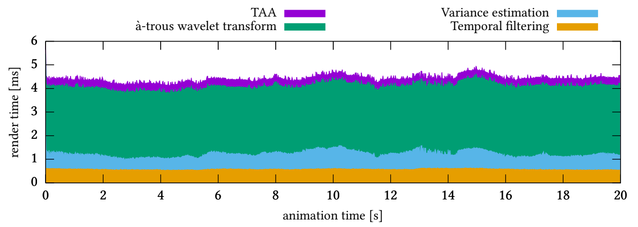


Figure 8: Runtime of our filter for an animated flythrough in the San Miguel courtyard, rendered at 1280 × 720 on a NVIDIA TITAN X (Pascal). The performance of our technique is consistent across the sequence and is also largely independent of scene properties. See Section 5.3 for details.

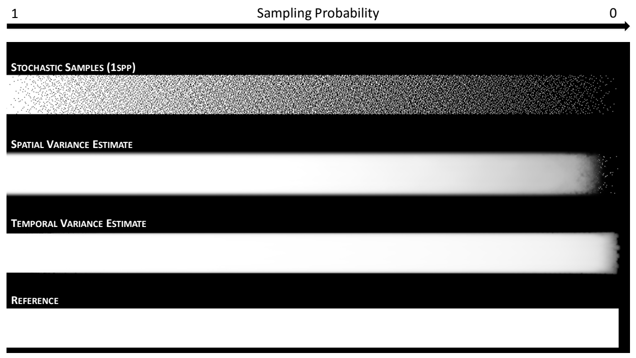

1
Sampling Probability
0
Stochastic Samples (1spp)
Spatial Variance Estimate
Temporal Variance Estimate
Reference

Figure 9: Synthetic test case for our filter. We sample a function with varying sampling probability. The spatial variance estimate case refers to reconstruction without any temporal history, i.e. as it occurs in newly disoccluded regions. It relies on a spatial estimate of variance, therefore overblurring stable features. Once the filter has acquired sufficient temporal history it switches to a purely temporal variance estimate which allows the filter to retain sharp features.

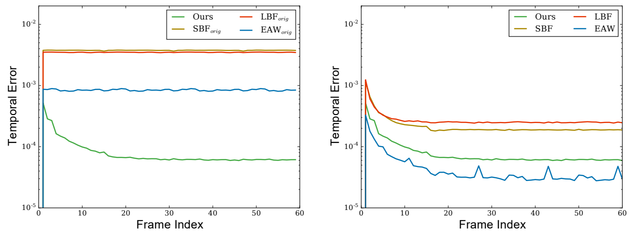
are rare, usually occurring at application startup or at discontinuities in camera paths when disocclusions cover the entire frame. In such cases (not shown in Figure 8), our implementation can be around 50% worse than the best-case performance.

SVGF performance scales linearly with resolution. Costing an average of 4.4 ms per frame at 1280×720 and 10.2 ms at 1920×1080, our algorithm processes around 200 MPixels/sec.

5.4 Temporal Stability

Figure 10(b) compares our temporal stability with prior work. For a 60 frame sequence in Sponza, using static light, camera, and scene, we quantify temporal stability by measuring the average luminance of the difference between consecutive frames, which we call *temporal error*. The shows filters' temporal convergence.

All algorithms start with high temporal error, due to lack of sufficient history, but they quickly converge until all variations arise from repetitions in the sample pattern. While EAW has the lowest temporal error, it loses significant detail during reconstruction (see Figure 7). Due to its high spatial bias (caused by insufficiently sharp edge-stopping functions), EAW combines more pixels together, achieving both higher bias and temporal stability. SVGF gives temporal error close EAW, but at much higher image quality.



(a) Without temporal reuse (b) With temporal reuse

Figure 10: Temporal stability of various algorithms compared using *temporal error*, the average luminance of the per-pixel differences, for a fixed view and lighting configuration. We compare our algorithm with both (a) reimplementations of comparative techniques directly from the literature and (b) the prior work augmented with our temporal reuse. When using temporal histories, algorithms lack sufficient history data in early frames, after which they converge over 10–20 frames. Our SVGF filter performs 10× better than prior published work. While EAW exhibits higher temporal stability than our SVGF, it has significantly higher reconstruction error (see Figure 7 and Table 1).

5.5 Applicability

As SVGF requires no special knowledge about a scene's light transport, it applies to various contexts where ray tracing may be desired. Beyond accurate reconstruction of glossy reflections (see Figure 7), Figure 12 shows SVGF can reconstruct ambient occlusion from 1 spp input. Additionally, we can reconstruct illumination from multiple lights from a single sample, a big improvement over rasterization, where shadow maps are queried for each light. In both cases, the reconstruction closely matches our 4096 spp reference.

Figure 11 compares our quality when reconstructing direct lighting. The scenes include complex shadows from bright area lights. EAW loses detail due to the non-adaptive edge-stopping functions and SBF retains significant noise. AAF also preserves many shadow details, but at the cost of losing subtle shadow shape. Overall, SVGF provides an accurate reconstruction without using any domain-specific knowledge. Figure 11 shows that prior work performes worse even without indirect lighting, emphasizing that splitting apart and separately filtering direct and indirect illumination is not solely responsible for our improved reconstruction quality.

5.6 Filter Scalability with Additional Samples

We aimed to develop a high-quality filter for real-time path tracing, which today is only viable at 1 spp or less. This target sampling rate drove many of our design decisions. These decisions need revisiting to extend our work on higher sampling rates, which is outside the scope of our current filter design.

For instance, with 1 spp traditional variance estimates are meaningless over the single sample from each pixel. To estimate variance, Section 4.2 accumulates variance temporally using an exponential moving average, as typical in temporal antialiasing. Without using an arithmetic mean, this estimate does not converge to the actual variance using additional samples. Instead, when using better sampled images, it makes sense to reevaluate whether a more traditional variance computation should be used.

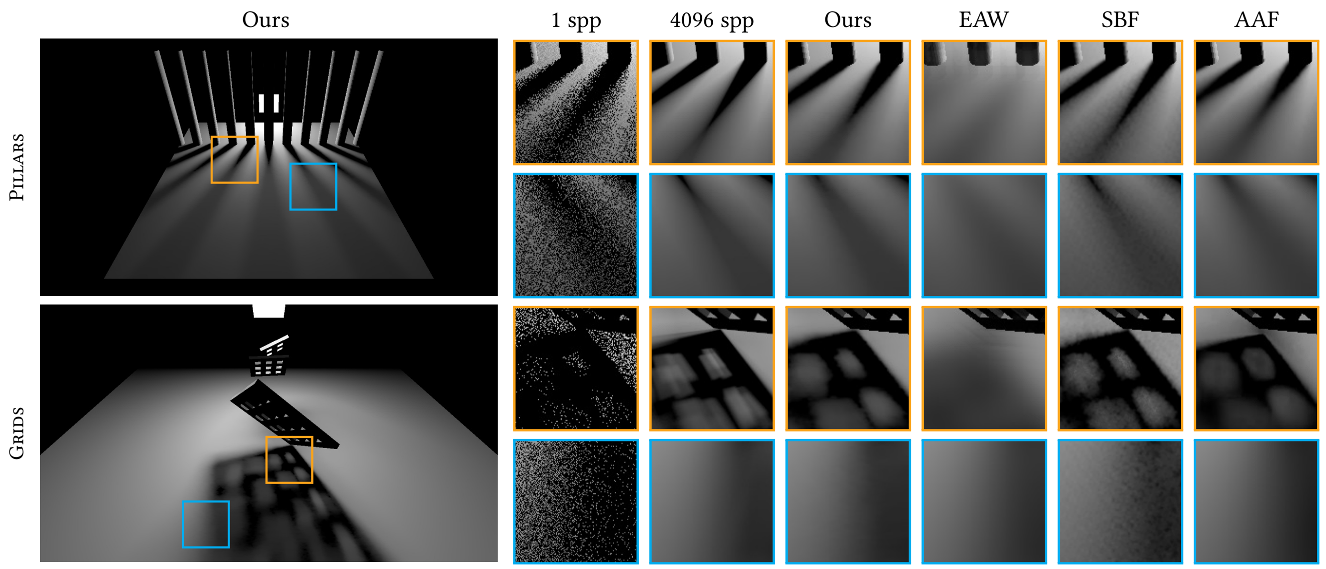


Figure 11: Quality comparison of various reconstruction filters on direct lighting. Both scenes use fixed view and light positions. The Pillar scene shows simple geometric shapes blocking an area light source. Due to the input's high dynamic range and its global edge-stopping function, EAW fails to preserve sharp features. SBF and AAF preserve the contact shadows, but SBF produces mid-frequency noise. Our filter preserves the shadow shape and the discontinuity present in the shadow (refer to the second inset) which AAF does not capture. In Grids, multiple stacked occluders result in complex visibility with very few samples. Our filter captures the most prominent features of the shadow compared to AAF, which loses shape information.

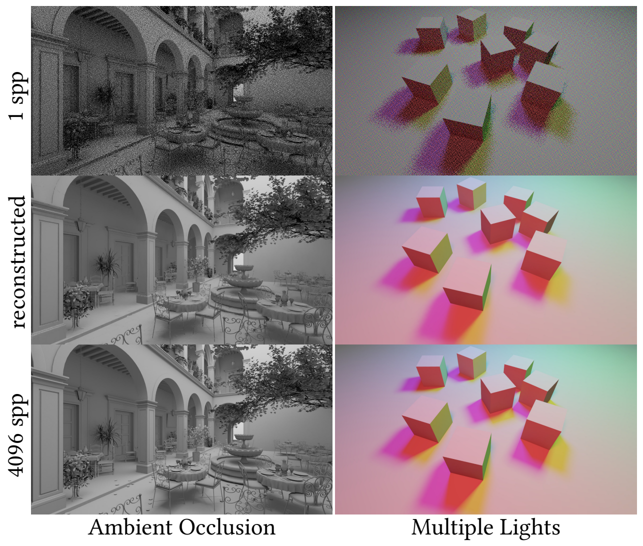


Figure 12: Beyond direct shadows and global illumination, our filter seamlessly reconstructs ambient occlusion and contributions from multiple lights with the same settings.

6 LIMITATIONS

While our filter is able to perform temporally stable real-time reconstruction of very noisy Monte Carlo images, we identify scenarios in which our algorithm can be improved.

Over-Blurred Chrominance. Because we solely compute and track luminance variance in the edge-stopping functions, our filter will sometimes over-blur chrominance. For example, see the multi-colored reflection in the bottom inset for GlossySponza in Figure 7. Note that our filter over-blurs the colored reflection compared to the reference and SBF. This issue can be addressed by estimating variance individually for each color component, however this comes at increased memory and computation costs.

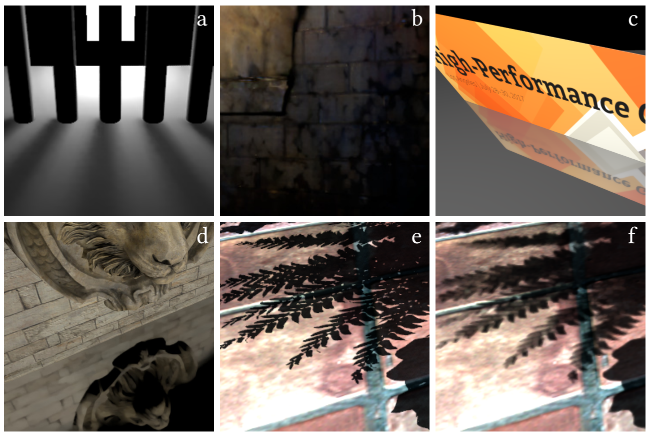


Figure 13: Limitations of our filter. (a) shows fast-moving geometry close to a light source, resulting in slightly detached shadows. (b) shows noise in a corner of the diffuse Sponza scene dimly lit by a small amount of indirect light. (c) shows a correct, sharp specular reflection processed by our filter, but (d) shows our filter erroneously blurring a reflection due to undersampled lighting. (e) shows our system correctly rendering a hard shadow when the camera is still, whereas (f) shows over-blurring of shadows during camera motion.

Detached Shadows in Motion. Fast-moving geometry close to a light source can produce detached shadows (see shadows of the pipes in Figure 13(a)). This is due to the temporal accumulation not being aware of the changing illumination due to object motion.

Temporally Unstable Noise in Extremely Low-Light. Our algorithm can produce temporally unstable noise in cases where the light transport results in very few samples, for example in regions dimly lit with indirect light. See the end of the upstairs hallway in the Diffuse Sponza video, the exposure-enhanced image from this video sequence in Figure 13(b), and the right most region of Figure 9. As the sample density approaches zero, the estimate of variance becomes increasingly unreliable and our filter is unable to recover a plausible image.

Over-Blurred Specular Reflections. Under ideal conditions our algorithm can correctly reconstruct specular reflections (see Figure 13(c)). However, if the reflection is noisy due to stochastically sampling indirect illumination, the filter will remove the noise, thus blur the reflection (see Figure 13(d)).

Over-Blurred Features Under Motion. Due to the blur introduced by back-projecting the color and moments history buffers, our filter can over-blur sharp features under motion. In addition, back-projection also introduces non-zero variance estimates, which causes further spatial bias (see Figure 13(e,f)).

Incompatible with Stochastic Primary Ray Effects. Our filter relies on measuring variance from the buffer of primary ray hits (G-buffer). We therefore require that the primary rays do not contain Monte Carlo noise, thus making our method incompatible with stochastic primary ray effects such as stochastic transparency, stochastic depth-of-field, and stochastic motion blur.

Incompatible with Highly Aliased Primary Rays. We demonstrated our filter works well on foliage and other fine-detail geometry (e.g., the bushes and tree leaves in SanMiguel). However, with extremely aliased sub-pixel geometry in the G-buffer, our edge-stopping functions will prevent the filter from finding valid samples in the image neighborhood, causing little illumination filtering.

7 CONCLUSION

In this paper we have presented a novel reconstruction algorithm suitable for real-time rendering. With the achieved performance and quality of our filter, we have significantly narrowed the performance gap limiting the use of path tracing in real-time graphics, presenting what we believe is the first practical real-time reconstruction filter operating on just a single path per pixel.

The quality improvements of SVGF stem from its reliance on our spatiotemporal variance estimate, which we use as an imperfect proxy for per-pixel variance. We use our estimate to control a spatial wavelet-based filter, which hierarchically filters color as well as our variance estimate. Therefore the reconstruction error, which is highest where the variance estimate is imprecise, is kept local, and the reliability of the reconstruction grows with each iteration.

Consequently, SVGF is able to achieve a much higher quality of reconstruction as compared to previous work. We believe our work represents a significant step in the exploration of the large design space of non-linear spatiotemporal reconstruction, and we wish to continue exploring more algorithms in this area that are faster, higher quality, and free from artifacts discussed in Section 6.

8 ACKNOWLEDGEMENTS

We thank Petrik Clarberg for his assistance on writing, Nir Benty for assisting with the Falcor framework, and Nicholas Hull for adjusting the scenes to our renderer. Additionally, thanks to the numerous anonymous reviewers for their valuable feedback in improving the paper.

REFERENCES

Pablo Bauszat, Martin Eisemann, Elmar Eisemann, and Marcus Magnor. 2015a. General and Robust Error Estimation and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* 34, 2 (2015), 597–608.

P. Bauszat, M. Eisemann, S. John, and M. Magnor. 2015b. Sample-Based Manifold Filtering for Interactive Global Illumination and Depth of Field. *Computer Graphics Forum* 34, 1 (2015), 265–276.

Pablo Bauszat, Martin Eisemann, and Marcus Magnor. 2011. Guided Image Filtering for Interactive High-quality Global Illumination. In *Eurographics Symposium on Rendering*. 1361–1368.

Nikolaus Binder and Alexander Keller. 2016. Efficient Stackless Hierarchy Traversal on GPUs with Backtracking in Constant Time. In *High Performance Graphics*. 41–50.

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. In *Computer Graphics Forum*, Vol. 35. 107–117.

Shenchang Eric Chen and Lance Williams. 1993. View Interpolation for Image Synthesis. In *Proceedings of SIGGRAPH 93*. 279–288.

Robert L Cook, Thomas Porter, and Loren Carpenter. 1984. Distributed Ray Tracing. In *Computer Graphics*, Vol. 18. 137–145.

Holger Dammertz, Daniel Sewtz, Johannes Hanika, and Hendrik Lensch. 2010. Edge-Avoiding À-Trous Wavelet Transform for fast Global Illumination Filtering. In *High Performance Graphics*. 67–75.

Mauricio Delbracio, Pablo Musé, Antoni Buades, Julien Chauvier, Nicholas Phelps, and Jean-Michel Morel. 2014. Boosting Monte Carlo Rendering by Ray Histogram Fusion. *ACM Transactions on Graphics* 33, 1 (2014), 8:1–8:15.

Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X Sillion. 2005. A Frequency Analysis of Light Transport. *ACM Transactions on Graphics* 24, 3 (2005), 1115–1126.

Kevin Egan, Frédo Durand, and Ravi Ramamoorthi. 2011a. Practical Filtering for Efficient Ray-traced Directional Occlusion. *ACM Transactions on Graphics* 30, 6 (2011), 180:1–180:10.

Kevin Egan, Florian Hecht, Frédo Durand, and Ravi Ramamoorthi. 2011b. Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders. *ACM Transactions on Graphics* 30, 2 (2011), 9:1–9:13.

Kevin Egan, Yu-Ting Tseng, Nicolas Holzschuch, Frédo Durand, and Ravi Ramamoorthi. 2009. Frequency Analysis and Sheared Reconstruction for Rendering Motion Blur. *ACM Transactions on Graphics* 28, 3 (2009), 93:1–93:13.

Elmar Eisemann and Frédo Durand. 2004. Flash Photography Enhancement via Intrinsic Relighting. *ACM Transactions on Graphics* 23, 3 (2004), 673–678.

Raanan Fattal. 2009. Edge-Avoiding Wavelets and Their Applications. *ACM Transactions on Graphics* 28, 3 (2009), 22:1–22:10.

Eduardo SL Gastal and Manuel M Oliveira. 2012. Adaptive Manifolds for Real-Time High-Dimensional Filtering. *ACM Transactions on Graphics* 31, 4 (2012), 33:1–33:13.

Pascal Gautron, Marc Droske, Carsten Wächter, Lutz Kettner, Alexander Keller, Nikolaus Binder, and Ken Dahm. 2014. Path Space Similarity Determined by Fourier Histogram Descriptors. In *ACM SIGGRAPH Talks*. 39:1–39:1.

J. H. Halton and G. B. Smith. 1964. Algorithm 247: Radical-Inverse Quasi-Random Point Sequence. *Commun. ACM* 7, 12 (1964), 701–702.

Jon Hasselgren, Jacob Munkberg, and Karthik Vaidyanathan. 2015. Practical Layered Reconstruction for Defocus and Motion Blur. *Journal of Computer Graphics Techniques* 4, 2 (2015), 45–58.

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (2013), 1397–1409.

Norman P. Jouppi and Chun-Fa Chang. 1999. Z3: An Economical Hardware Technique for High-Quality Antialiasing and Transparency. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*.

James T Kajiya. 1986. The Rendering Equation. In *Computer Graphics*, Vol. 20. 143–150.

Nima Khademi Kalantari, Steve Bako, and Pradeep Sen. 2015. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics* 34, 4 (2015), 122:1–122:12.

Nima Khademi Kalantari and Pradeep Sen. 2013. Removing the Noise in Monte Carlo Rendering with General Image Denoising Algorithms. *Computer Graphics Forum* 32, 2 (2013), 93–102.

Anton S. Kaplanyan and Carsten Dachsbacher. 2013. Path Space Regularization for Holistic and Robust Light Transport. *Computer Graphics Forum* 32, 2 (2013), 63–72.

Brian Karis. 2014. High-Quality Temporal Supersampling. In *SIGGRAPH Courses: Advances in Real-Time Rendering in Games*.

Alexander Keller, Ken Dahm, and Nikolaus Binder. 2016. Path Space Filtering. In *Monte Carlo and Quasi-Monte Carlo Methods 2014*, Ronald Cools and Dirk Nuyens (Eds.). Springer, 423–436.

A Keller, L Fascione, M Fajardo, I Georgiev, P Christensen, J Hanika, C Eisenacher, and G Nichols. 2015. The Path Tracing Revolution in the Movie Industry. In *SIGGRAPH Courses*.

E. Kerzner and M. Salvi. 2014. Streaming G-buffer Compression for Multi-sample Anti-aliasing. In *Proceedings of High Performance Graphics*. 1–7.

M. E. Lee and R. A. Redner. 1990. A Note on the Use of Nonlinear Filtering in Computer Graphics. *IEEE Computer Graphics and Applications* 10, 3 (1990), 23–29.

Tzu-Mao Li, Yu-Ting Wu, and Yung-Yu Chuang. 2012. SURE-based Optimization for Adaptive Sampling and Reconstruction. *ACM Transactions on Graphics* 31, 6 (2012), 194:1–194:9.

Michael Mara, Morgan McGuire, Benedikt Bitterli, and Wojciech Jarosz. 2017. An Efficient Denoising Algorithm for Global Illumination. In *Proceedings of High Performace Graphics*. 6.

Stephen McAuley and Stephen Hill. 2016. Physically Based Shading in Theory and Practice. In *SIGGRAPH Courses*.

Michael D McCool. 1999. Anisotropic Diffusion for Monte Carlo Noise Reduction. *ACM Transactions on Graphics* 18, 2 (1999), 171–194.

Soham Uday Mehta, Brandon Wang, and Ravi Ramamoorthi. 2012. Axis-Aligned Filtering for Interactive Sampled Soft Shadows. *ACM Transactions on Graphics* 31, 6 (2012), 163:1–163:10.

Soham Uday Mehta, Brandon Wang, Ravi Ramamoorthi, and Fredo Durand. 2013. Axis-Aligned Filtering for Interactive Physically-Based Diffuse Indirect Lighting. *ACM Transactions on Graphics* 32, 4 (2013), 96:1–96:12.

Soham Uday Mehta, JiaXian Yao, Ravi Ramamoorthi, and Fredo Durand. 2014. Factored Axis-Aligned Filtering for Rendering Multiple Distribution Effects. *ACM Transactions on Graphics* 33, 4 (2014), 57:1–57:12.

Mark Meyer and John Anderson. 2006. Statistical Acceleration for Animated Global Illumination. *ACM Transactions on Graphics* 25, 3 (2006), 1075–1080.

Bochang Moon, Nathan Carr, and Sung-Eui Yoon. 2014. Adaptive Rendering Based on Weighted Local Regression. *ACM Transactions on Graphics* 33, 5 (2014), 170:1–170:14.

Bochang Moon, Jose A Iglesias-Guitian, Sung-Eui Yoon, and Kenny Mitchell. 2015. Adaptive Rendering with Linear Predictions. *ACM Transactions on Graphics* 34, 4 (2015), 121:1–121:11.

Bochang Moon, Steven McDonagh, Kenny Mitchell, and Markus Gross. 2016. Adaptive Polynomial Rendering. *ACM Transactions on Graphics* 35, 4 (2016), 40:1–40:10.

Jacob Munkberg, Jon Hasselgren, Petrik Clarberg, Magnus Andersson, and Tomas Akenine-Möller. 2016. Texture Space Caching and Reconstruction for Ray Tracing. *ACM Transactions on Graphics* 35, 6 (2016), 249:1–249:13.

Jacob Munkberg, Karthik Vaidyanathan, Jon Hasselgren, Petrik Clarberg, and Tomas Akenine-Möller. 2014. Layered Reconstruction for Defocus and Motion Blur. *Computer Graphics Forum* 33, 4 (2014), 81–92.

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating Real-time Shading with Reverse Reprojection Caching. In *Graphics Hardware*. 25–35.

Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics* 29, 4 (2010), 66:1–66:13.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards Foveated Rendering for Gaze-Tracked Virtual Reality. *ACM Transactions on Graphics* 35, 6 (2016), 179:1–179:12.

Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital Photography with Flash and No-flash Image Pairs. *ACM Transactions on Graphics* 23, 3 (2004), 664–672.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The State of the Art in Interactive Global Illumination. *Computer Graphics Forum* 31, 1 (2012), 160–188.

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive Rendering with Non-local Means Filtering. *ACM Transactions on Graphics* 31, 6 (2012), 195:1–195:11.

Fabrice Rousselle, Marco Manzi, and Matthias Zwicker. 2013. Robust Denoising Using Feature and Color Information. *Computer Graphics Forum* 32, 7 (2013), 121–130.

Holly E Rushmeier and Gregory J Ward. 1994. Energy Preserving Non-linear Filters. In *Proceedings of SIGGRAPH 94*. 131–138.

Takafumi Saito and Tokiichiro Takahashi. 1990. Comprehensible Rendering of 3-D Shapes. *Computer Graphics* (1990), 197–206.

C. Tomasi and R. Manduchi. 1998. Bilateral Filtering for Gray and Color Images. In *IEEE International Conference on Computer Vision*. 839–846.

Karthik Vaidyanathan, Jacob Munkberg, Petrik Clarberg, and Marco Salvi. 2015. Layered Light Field Reconstruction for Defocus Blur. *ACM Transactions on Graphics* 34, 2 (2015), 23:1–23:12.

Ingo Wald, Sven Woop, Carsten Benthin, Gregory S. Johnson, and Manfred Ernst. 2014. Embree: A Kernel Framework for Efficient CPU Ray Tracing. *ACM Transactions on Graphics* 33, 4 (2014), 143:1–143:8.

Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive Rendering Using the Render Cache. In *Eurographics Workshop on Rendering*, Vol. 10. 19–30.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

Turner Whitted. 1980. An Improved Illumination Model for Shaded Display. *Commun. ACM* 23, 6 (1980), 343–349.

Ruifeng Xu and Sumanta N Pattanaik. 2005. A Novel Monte Carlo Noise Reduction Operator. *IEEE Computer Graphics and Applications* 25, 2 (2005), 31–35.

Ling-Qi Yan, Soham Uday Mehta, Ravi Ramamoorthi, and Fredo Durand. 2015. Fast 4D Sheared Filtering for Interactive Rendering of Distribution Effects. *ACM Transactions on Graphics* 35, 1 (2015), 7:1–7:13.

Lei Yang, Diego Nehab, Pedro V Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized Supersampling. *ACM Transactions on Graphics* 28, 5 (2009), 135:1–135:12.

Henning Zimmer, Fabrice Rousselle, Wenzel Jakob, Oliver Wang, David Adler, Wojciech Jarosz, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2015. Path-space Motion Estimation and Decomposition for Robust Animation Filtering. *Computer Graphics Forum* 34, 4 (2015), 131–142.

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, Bochang Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril 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.