Practical Denoising for VFX Production Using Temporal Blur

Daniel Dresser
Image Engine
danield@image-engine.com

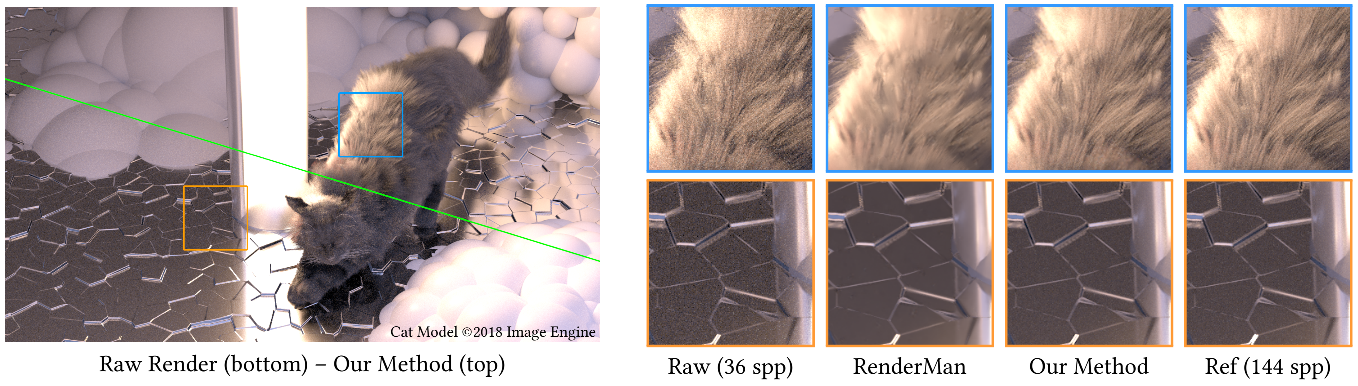
Cat Model ©2018 Image Engine
Raw Render (bottom) – Our Method (top)
Raw (36 spp)
RenderMan
Our Method
Ref (144 spp)

Figure 1: A 36 spp render with hair, motion blur, and difficult illumination. The RenderMan denoiser produces very clean results but loses detail in tricky areas such as hair. Our method preserves details and provides a moderate noise reduction, with results similar to a 144 spp reference render. Both denoising methods use data from adjacent frames.

ABSTRACT

We present a simple, efficient, and reliable approach to denoising final ray traced renders during VFX production. Rather than seeking to remove all noise, we combine several simple steps that reduce noise dramatically. Our method has performed well on a wide variety of shows in Image Engine's recent portfolio, including Game of Thrones Season 7, Lost in Space, and Thor: Ragnarok.

CCS CONCEPTS

• **Computing methodologies → Ray tracing**;

KEYWORDS

denoising, Monte Carlo rendering

ACM Reference Format:
Daniel Dresser. 2018. Practical Denoising for VFX Production Using Temporal Blur . In *Proceedings of SIGGRAPH '18 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214762

1 INTRODUCTION

Much recent denoising research has focused on removing as much noise as possible. Many papers have demonstrated dramatic results, starting with extremely noisy images rendered with very few samples per pixel (spp), and producing results that are clean, plausible, and appealing [Bitterli et al. 2016; Chaitanya et al. 2017]. Broadly, most of these methods are similar to joint non-local means filtering, blurring together all pixels that match some similarity criteria. This can achieve exceptionally good results, but it is difficult to achieve consistently good results in complex cases. There is risk of overblurring if a criteria is missed or artifacting if the similarity criteria is itself undersampled.

Feature film VFX companies have different requirements for denoising. They are prepared to spend substantial computing resources in order to reliably obtain a high-quality image. We start with ray traced images of reasonable quality — in the case of Image Engine, we render at a minimum of 36 spp for primary visibility and with enough secondary samples to clean up extreme noise from illumination. The denoiser must solve issues such as buzzing or sparkling on very thin objects, tight corners with sharp speculars, or lighting from difficult-to-sample lights. The RenderMan denoiser refers to this as removing the "long tail" of render time — stopping rendering before the point of diminishing returns [Pixar 2017]. Additionally, we do not remove all noise. In VFX, compositors add noise in order to match the characteristics of real cameras. Ideally, a clean render allows us freedom in choosing noise to add, but a small amount of leftover noise is less objectionable than other errors.

In this environment, we noticed that our compositing artists' approach to denoising was just to average together several adjacent frames using optical flow vectors. This provided surprisingly effective denoising on many production shots. Cinematic camera motion often involves minimal screen space motion of the area in focus. Aside from the obvious noise reduction from averaging more samples, noise that stays coherent over a couple of frames may be perceived as small details catching the light, rather than objectionable buzzing.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGGRAPH '18 Talks, August 12-16, 2018, Vancouver, BC, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5820-0/18/08.
https://doi.org/10.1145/3214745.3214762

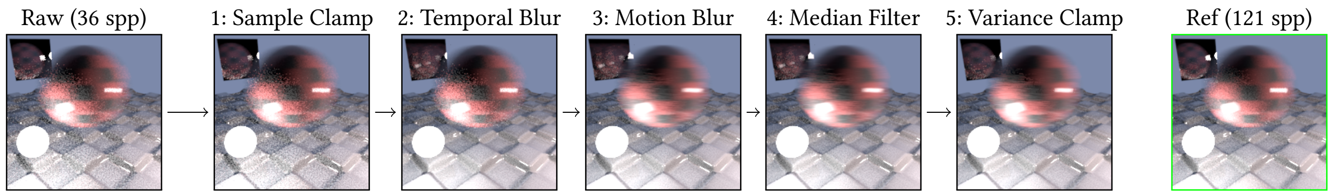

Figure 2: Our 5 steps. Note the errors introduced on the mirror during blurring, which are corrected by the Variance Clamp. For illustration, we start with extreme noise, resulting in some leftover noise and a specular caustic which gets clamped.

2 METHOD

We automated and improved the approach used by our compositors. Rather than incorporating information from adjacent frames into a more complex denoising algorithm [Goddard 2014; Pixar 2017], we blend per pixel between pure temporal blur, and some simple alternate steps with no dependency on adjacent frames.

Figure 2 demonstrates how our steps work together. Our implementation uses a custom pixel filter, followed by a series of image compositing operations run on sequences of images with adjacent frame data — we do not attempt to handle still images. All channels of both the main image and any supplemental images (arbitrary output variables, or AOVs) are filtered using the same filter kernel, which preserves additive compositing. Our supplemental material discusses a high-quality implementation of these steps.

Step 1 : Preprocess — Sample Clamp During ray tracing we remove intensity outliers that would be difficult to average out. A common approach is to clamp all camera samples to a fixed brightness threshold, but we do slightly better; we use a pixel filter that computes a global weight multiplier for each camera sample based on its total brightness. This preserves additive compositing and reduces clamping of desirable highlights.

Step 2 : Temporal Blur Our most important step is to average 5 frames: the current frame, and 2 frames before and after. We render AOVs for screen-space offsets to the previous and next frame. We sum the offsets forward and backward, and compare to the starting point, which yields an error representing how well an adjacent frame matches the current point. Thresholding this error gives us a weight for each pixel from the adjacent frames. If the sum of weights from adjacent frames is over 2, a pixel is replaced with a weighted average. This error calculation does not catch shading changes on stationary geometry — for example shadows and reflections from moving objects — but we partially address this with the final variance clamp step.

Step 3 : Motion Blur We render a motion vector pass to identify fast-moving pixels, which are not handled by temporal blur. We replace these pixels with the average of a set of samples taken along a short line segment aligned to the motion vector. Adding extra blur aligned with existing motion blur is crude, but generally not visually objectionable.

Step 4 : Median Filter At this point, almost all pixels have been denoised, but a few exceptional pixels will fall in between the categories handled previously. For pixels that were not affected by the previous two steps we use a median filter variant, which removes the worst noise. It blurs out some detail, but in practice very few pixels are affected.

Step 5 : Variance Clamp Finally, we mask the previous 3 steps to only affect pixels that are actually noisy. The loss of quality from the previous steps is generally low, but there could be some unnecessary blur or artifacts. We identify noisy areas of the image with an AOV which estimates the variance of each pixel as a whole, based on the variance of the subpixel samples. The final pixel value is the result from the previous steps, clamped to the original result of the preprocess, plus or minus this AOV. This results in blurring only where necessary to remove noise.

3 CONCLUSION

The biggest difference between our method and most recent denoising methods is that for any output pixel, the set of source pixels we consider is quite small. Many denoising techniques consider a large number of possible source pixels, narrowing down this large pool of information using similarity criteria. This allows for dramatic denoising but also for significant errors. By considering a small number of possible source pixels, we limit how much noise we can remove, but also the errors that can be introduced. The behaviors of median filters, motion blur, and frame averaging are all well-understood, so worst-case errors are usually small and predictable.

This compromise has worked well for us in production. Our method has been used without modification in projects such as Game of Thrones Season 7, Lost In Space, and Thor: Ragnarok (see supplemental video). It has minimal requirements: only 4 AOVs (variance, motion vectors, and 2 frame offsets) compared to the Renderman denoiser's 13. It does not require expensive computation: for the 1080p sequence in Figure 1, an unoptimized implementation of our denoiser takes 10 seconds per frame on an 8-core Xeon, compared to 100 seconds for the the RenderMan denoiser. Without a denoiser, our default camera sampling was 64 spp, but difficult scenes could require 144 or even 324 spp. With the denoiser, almost all scenes can be rendered acceptably at 36–64 spp.

REFERENCES

Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. 2016. Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings. *Computer Graphics Forum* 35, 4 (2016), 107–117. https://doi.org/10.1111/cgf.12954

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages. https://doi.org/10.1145/3072959.3073601

Luke Goddard. 2014. Silencing the Noise on Elysium. In *ACM SIGGRAPH 2014 Talks (SIGGRAPH '14)*. ACM, New York, NY, USA, Article 38, 1 pages. https://doi.org/10.1145/2614106.2614116

Pixar Animation Studios. 2017. RenderMan : Denoise Workflow. (2017). https://rmanwiki.pixar.com/display/REN/Denoise+Workflow