Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

Volume 34 (2015), Number 4

Path-space Motion Estimation and Decomposition for Robust Animation Filtering

SUPPLEMENTARY MATERIAL

Henning Zimmer[1] Fabrice Rousselle[1] Wenzel Jakob[2] Oliver Wang[1] David Adler[3]
Wojciech Jarosz[1] Olga Sorkine-Hornung[2] Alexander Sorkine-Hornung[1]

[1]Disney Research Zurich [2]ETH Zurich [3]Walt Disney Animation Studios

1 Introduction

This document complements our paper with additional technical details.

2 Decomposition

2.1 Classification of Light Paths

We define our decomposition using Heckbert's [Hec90] light path notation to classify light paths or families of similar light paths. We label the first vertex on the camera sensor `E` ("eye"), and denote the subsequent scattering vertices based on a classification of the underlying materials. We use the following notation:

- **(D)** diffuse,
- **(R)** specular (or glossy) reflection,
- **(T)** specular (or glossy) transmission

We classify glossy light interactions (e.g. scattering off rough metal or glass) as `R` or `T` if the roughness is below a threshold (Beckmann roughness $\alpha < 0.1$ in our experiments), otherwise we consider them to be diffuse `D`.

Families of similar light transport paths can be expressed using a regular expression syntax, e.g. `ETT.*` denotes objects seen through two transmission events, such as the interfaces of a glass window. The "`.`" is a wild card, while "$\star$" indicates a chain of arbitrary length; "`.`$\star$" is therefore an arbitrary chain of arbitrary length. Similarly, "`.+`" is an arbitrary chain with at least one element, which can be used to represent the indirect diffuse component, `ED.+` (in contrast with the direct diffuse component, `ED`).

2.2 Decomposition and Feature Extraction in Mitsuba

The decomposition and feature extraction is done in a single pass using a modified implementation of the path tracer integrator of the Mitsuba renderer that employs a finite state machine (FSM) to track the throughput of each component as the paths are traced.

This general approach using a deterministic FSM is fairly common in the movie industry and referred to as "light path expressions" in products such as RenderMan, IRay or the Open Shading Language. The key difference in our implementation is that we are interested in extracting many components at the same time: for instance, after interacting with coated diffuse material, subsequent interactions could contribute both to the `ED.`$\star$ and `ERD.`$\star$ components of the output image, which cannot be modeled using a deterministic FSM. We address this by simulating all possible states of a nondeterministic FSM at the *same time* – thus, the combined system is still deterministic. Each state also stores a color weight that is updated as paths are traced. Initially all are set to zero, except for the initial state `E` whose weight is one.

Operations such as tracing a ray and sampling a BRDF generate new symbols and importance sampling weights that are sent to all states. Suppose that the FSM has a transition from state `E` to `ER` given symbol `R`: when `R` is generated, the color weight in state `E` is multiplied by the BRDF sampling weight and moved to state `ER`. Unsupported transitions cause weights to be moved to $\perp$. In addition to a color weight, each state is also associated with an output variable that will eventually become a component of the final decomposition. When a light source is sampled at a surface position, we iterate over all FSM states and multiply the current weight with the incident illumination and store the product into this output variable.

3 Image-based Irradiance and Residual Flow

We use image-based optical flow to compute motion vectors for the irradiance and residual components.

We base our approach on the variational optical flow method of Brox et al. [BBPW04] which finds the motion

vectors $\mathbf{v}_t$ between frames at time t and $t+1$ by minimizing a continuous energy formulation

$$E(\mathbf{v}_t) = \int_\Omega D(\mathbf{v}_t) + \lambda\, S(\nabla \mathbf{v}_t)\ \mathrm{d}\mathbf{p}\,, \tag{1}$$

consisting of a data term D, a smoothness term (regularizer) S and a smoothness parameter λ. We set $\lambda = 0.75$ unless otherwise noted.

The data term models the assumption of constant color between corresponding pixels in frame I_t and I_{t+1} via

$$D(\mathbf{v}_t) = m(\mathbf{p}) \cdot \Psi\Big(\|I_{t+1}(\mathbf{p}+\mathbf{v}_t(\mathbf{p})) - I_t(\mathbf{p})\|^2 \Big)\,, \tag{2}$$

where

$$\Psi(s^2) = \sqrt{s^2 + 0.001^2} \approx |s| \tag{3}$$

is a robust penalizer function that reduces the influence of outliers compared to a quadratic penalizer $\Psi(s^2) = s^2$, and the function m is a binary mask that disables the data term at occluded pixels; see Section 4.2 and Figure 7.

To reduce artifacts at image edges, we found a normalization in the data term useful [ZBW11].

The smoothness term imposes smoothness of the resulting flow field and is defined as

$$S(\nabla \mathbf{v}_t) = \Psi\Big(\|\nabla \mathbf{v}_t(\mathbf{p})\|^2 \Big)\,, \tag{4}$$

using the same robust penalizer function Ψ as before to preserve sharp discontinuities in the flow field.

Glossy materials. Moving glossy objects can create complex caustics, as for example shown in the *Glossy Sphere* scene in our supplementary video. To best capture the motion of these caustic effects, we found it useful to reduce smoothness weight of the optical flow formulation for the indirect irradiance component that captures caustics. In the shown results we set $\lambda = 0.25$.

4 Denoising Filter and Parameters

Before we give the parameters of our denoising algorithm, we will first briefly define the equation for computing the weights of the filter. Detailed explanations for these equations can be found in the original work of Rousselle et al. [RMZ13].

The value $\hat{u}(\mathbf{p})$ of a pixel $\mathbf{p}$ is computed as a weighted average over a square neighborhood $\mathcal{N}(\mathbf{p})$ of side $2r+1$:

$$\hat{u}(\mathbf{p}) = \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} w(\mathbf{p},\mathbf{q})\, u(\mathbf{q}).$$

The weight $w(\mathbf{p},\mathbf{q})$ is computed according to the data contained in the color and feature buffers, $w(\mathbf{p},\mathbf{q}) = \min(w_c(\mathbf{p},\mathbf{q}), w_f(\mathbf{p},\mathbf{q}))$, and w_c and w_f are the weights computed using the color and features, respectively. If multiple features are available (as in our case), then w_f is the minimum of the feature weights.

The color weight, w_c, is an NL-Means [BCM05] weight, that depends on the distance $d_c^2(P(\mathbf{p}), P(\mathbf{q}))$ between square patches P of side $2f+1$ centered on $\mathbf{p}$ and $\mathbf{q}$:

$$d_c^2(P(\mathbf{p}),P(\mathbf{q})) = \frac{1}{3(2f+1)^2} \sum_{i=1}^{3} \sum_{\mathbf{n}\in P(0)} \Delta_i^2(\mathbf{p}+\mathbf{n}, \mathbf{q}+\mathbf{n}),$$

where $\Delta_i(\mathbf{p}+\mathbf{n}, \mathbf{q}+\mathbf{n})$ is the pixel-wise distance in color channel i and $\mathbf{n} \in P(0)$ is the offset to each pixel within a patch. The pixel-wise distance is computed as

$$\Delta_i^2(\mathbf{p},\mathbf{q}) = \frac{(u_i(\mathbf{p}) - u_i(\mathbf{q}))^2 - (\mathrm{Var}_i[\mathbf{p}] + \mathrm{Var}_i[\mathbf{q},\mathbf{p}])}{\varepsilon + k_c^2(\mathrm{Var}_i[\mathbf{p}] + \mathrm{Var}_i[\mathbf{q}])},$$

where $u_i(\mathbf{p})$ and $u_i(\mathbf{q})$ are noisy pixel values, with variances $\mathrm{Var}_i[\mathbf{p}]$ and $\mathrm{Var}_i[\mathbf{q}]$, $\mathrm{Var}_i[\mathbf{q},\mathbf{p}] = \min(\mathrm{Var}_i[\mathbf{q}], \mathrm{Var}_i[\mathbf{p}])$, and k_c is a parameter controlling the aggressiveness of the filter. The color weight is then computed using an exponential kernel,

$$w_c(\mathbf{p},\mathbf{q}) = \exp^{-\max(0, d_c^2(P(\mathbf{p}),P(\mathbf{q})))}.$$

The feature weight, w_f, is a bilateral [TM98] weight, based on the feature distance,

$$\Phi^2(\mathbf{p},\mathbf{q}) = \frac{(f(\mathbf{p}) - f(\mathbf{q}))^2}{\max(\tau^2, \|\nabla f[\mathbf{p}]\|^2)},$$

where k_f is a parameter controlling the aggressiveness of the filter, τ is the expected standard deviation of the feature, and $\|\nabla f[\mathbf{p}]\|$ is the norm of the gradient of the feature at pixel p. The feature weight is also computed using an exponential kernel,

$$w_f(\mathbf{p},\mathbf{q}) = \exp^{-\max(0, \Phi^2(\mathbf{p},\mathbf{q}))}.$$

Since the feature buffers are assumed to be noise free, we denoise them in a preprocessing step using the NL-Means filter as proposed by Rousselle et al.

In our application, we set the joint NL-Means filter parameters to: $r = 5$, $f = 3$, $k_c = 0.45$, $k_f = 0.6$, $\tau^2 = 0.001$, and extend the filtering window to cover the previous and next frames in the sequence. When not using spatio-temporal filtering, we set $r = 10$. The denoising of the feature buffers in our preprocessing step uses the following parameters: $r = 1$, $f = 3$, $k_c = 0.45$.

5 Denoising Results at Lower Sampling Rates

We give denoising results on our *Robot* scene across a range of sampling rates in Figure 1. While the denoised results are significantly degraded at very low sampling rates, our decomposition still allows for a reasonable preview of the ground truth (bottom right image) at 16 samples per pixel. Additionally, we provide an equal error rendering to our denoised output at 512 samples per pixel (top right image).

References

[BBPW04] BROX T., BRUHN A., PAPENBERG N., WEICKERT J.: High accuracy optical flow estimation based on a theory for warping. In *ECCV* (2004), pp. 25–36. 1

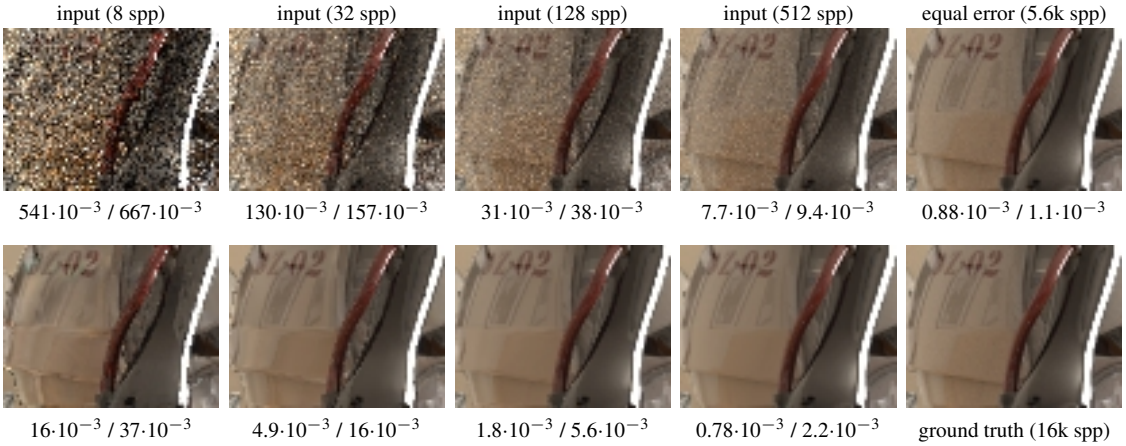

Figure 1: *Denoising results using 8 to 512 samples per pixels. We give the relative MSE of each image for the full frame (first value) and the crop shown (second value). The reconstruction quality degrades as the sampling rate decreases, however, our reconstruction offers a useful preview of the ground truth image even at 8 samples per pixel. The top right image shows a rendering at 5.6k samples per pixel with a similar error overall to our denoised reconstruction at 512 samples per pixel.*

[BCM05] BUADES A., COLL B., MOREL J.: A non-local algorithm for image denoising. In *CVPR* (2005), pp. 60–65. 2

[Hec90] HECKBERT P. S.: Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH* (1990), pp. 145–154. 1

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV* (1998), pp. 839–846. 2

[ZBW11] ZIMMER H., BRUHN A., WEICKERT J.: Optic flow in harmony. *International Journal of Computer Vision 93*, 3 (2011), 368–388. 2