DOI: 10.1111/cgf.12415
Eurographics Symposium on Rendering 2014
Wojciech Jarosz and Pieter Peers
(Guest Editors)

Volume 33 (2014), Number 4

Layered Reconstruction for Defocus and Motion Blur

Jacob Munkberg[1], Karthik Vaidyanathan[1], Jon Hasselgren[1], Petrik Clarberg[1], and Tomas Akenine-Möller[1,2]

[1] Intel Corporation, [2] Lund University

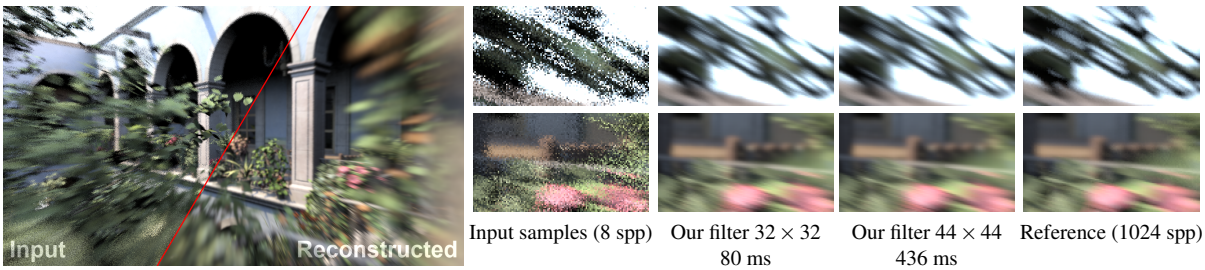


Figure 1: *The San Miguel scene with depth of field and large camera motion. The closeups show the input samples, the result using our new reconstruction algorithm on the input samples, and the reference rendered with 1024 spp. The two images for our algorithm show different filter search window configurations, where a larger window improves quality at the expense of performance. Our novel reconstruction method dramatically reduces noise, and produces images close to the reference.*

Abstract

Light field reconstruction algorithms can substantially decrease the noise in stochastically rendered images. Recent algorithms for defocus blur alone are both fast and accurate. However, motion blur is a considerably more complex type of camera effect, and as a consequence, current algorithms are either slow or too imprecise to use in high quality rendering. We extend previous work on real-time light field reconstruction for defocus blur to handle the case of simultaneous defocus and motion blur. By carefully introducing a few approximations, we derive a very efficient sheared reconstruction filter, which produces high quality images even for a low number of input samples. Our algorithm is temporally robust, and is about two orders of magnitude faster than previous work, making it suitable for both real-time rendering and as a post-processing pass for offline rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms

1. Introduction

Stochastic sampling is a powerful technique that can simulate realistic camera effects. This is achieved by evaluating a high-dimensional integral using point sampling. Motion blur is obtained by distributing samples in time over the open camera shutter, and depth of field by point sampling over the camera lens [Coo86]. However, as with most Monte Carlo techniques, a large number of samples must be drawn to reduce noise in the result. A different way to tackle this problem is to instead spend efforts on reconstructing a final image with substantially reduced noise from a sparsely sampled light field, e.g., with as few as 4 or 8 samples per pixel [LAC*11, LALD12, VMCS13].

Accurately reconstructing the four-dimensional light field for depth of field is well understood [SSD*09, LAC*11, VMCS13]. The algorithm by Vaidyanathan et al. can reconstruct images with defocus blur from a small number of samples per pixel with real-time performance. Reconstructing motion blur, and the combination of motion blur and depth of field is a much more difficult problem, as each object and the camera can have arbitrary motion (i.e., a unique 3D motion vector for each vertex in the scene). In contrast, defocus blur is purely a function of the vertex depth and a few cam-

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

era constants. In this paper, we develop an algorithm that is both accurate *and* fast for reconstructing and filtering the light field for simultaneous motion and defocus blur.

By analyzing a diffuse, moving emitter in the frequency domain using recent tools [DHS*05, BSS*13], and by cautiously introducing approximations, we develop an efficient filter that can be used in a layered composition framework [ML85, VMCS13]. The input to our algorithm is a stochastic, sparsely sampled light field with a fixed number of samples per pixel. Such light fields can be generated at interactive rates either by high-performance distributed ray tracers, such as OptiX [PBD*10] and Embree [WWB*14], or using stochastic rasterization [MESL10]. Since we do not rely on adaptive sampling, our algorithm can be implemented as a final reconstruction pass, orthogonal to the rest of the rendering system, which is an important benefit.

Our algorithm fills an important gap in the field of light field reconstruction algorithms. It competes with the quality of offline algorithms designed to work on stochastic light fields [ETH*09,LAC*11,LALD12], while performing close to the much more approximate techniques targeting real-time rendering [MHBO12]. For typical scenes, our algorithm runs in under 50 ms at 1280×720 on a discrete graphics card, making it a potential candidate for real-time use in the near future. It may also be useful as a post-processing tool for reducing noise in offline rendering.

2. Previous Work

Light Field Frequency Analysis Chai et al. [CTCS00] derive minimum sampling rates for plenoptic sampling of light fields using frequency analysis. They also show that the spectrum is mostly bounded by the minimum and maximum depth, and as a consequence, tighter bounds can be obtained by splitting the light field into depth layers. In their seminal work, Durand et al. [DHS*05] present a framework for analyzing light transport in the frequency domain. Since then, these tools have been used and extended in several research projects. Egan et al. [ETH*09] analyze motion blur in the frequency domain, and develop an adaptive sampling algorithm and a sheared reconstruction filter. Soler et al. [SSD*09] present a similar analysis for defocus blur. Similar frequency analysis has also been done for soft shadows [EHDR11, MWR12] directional occlusion [EDR11], and diffuse indirect illumination [MWRD13]. Belcour et al. [BSS*13] present a unified framework, where each transport operator, e.g., lens refraction, translation, and motion, is expressed as a five-dimensional matrix. They show that first-order motion can be described by simple changes of coordinates in five dimensions. Besides developing this useful framework, they also apply their theory to develop a method for adaptive sampling and reconstruction, which uses a filter that is an anisotropic 2D Gaussian in (x,y) and axis-aligned in the remaining dimensions (lens and time coordinates). The filters by Mehta et al. [MWR12, MWRD13] are similarly axis-aligned in the higher dimensions.

Reconstruction There is a wealth of post-processing methods, where visibility is only sampled at the center of the lens and at one unique time. The image is computed by blurring in a post pass based on auxiliary information, such as depth and motion vectors [PC81,PC83]. There are also newer, phenomenological post-processing methods targeting contemporary GPUs [MHBO12,GMN13]. Since such methods only have access to a pinhole-generated image at a particular instant in time, they can never converge to a fully correct image. However, in the context of real-time rendering, such methods are very valuable. Our ambition is different: we aim to use a few stochastically generated samples per pixel, and reconstruct an accurate image using filters applied to this sparsely sampled five-dimensional light field. Ideally, our algorithm should converge to the correct result when more samples are added.

Lee et al. [LES10] generate depth layers using depth-peeling from a pinhole camera and then ray trace through this representation to generate plausible depth of field. Max and Lerner [ML85] split up their scene in object layers, and use a single motion vector per layer to produce a blurred image and blurred opacity mask. The final image is then obtained by compositing the layers back-to-front using the over operator [PD84]. A similar approach is also taken by Potmesil and Chakravarty [PC83]. This type of layer compositing, each holding also a blurred opacity mask, forms the basis for the method by Vaidyanathan et al. [VMCS13].

In a separate line of research, image denoising filters are applied to stochastically generated samples in order to remove noise, while preserving important detail. Many of these are based on variants of the bilateral filter [TM98]. For example, Xu and Pattanaik [XP05] use a joint bilateral filter for denoising of global illumination. Sen and Darabi [SD12] improve on this by calculating the statistical dependency between the outputs and inputs of a rendering system, and use this information to guide their image-space filter. The reconstruction takes minutes, however. Other examples include algorithms based on wavelet transforms [ODR09, DSHL10], guided image filtering [BEM11], non-local means filtering [RKZ12], and ray histograms [DMB*14]. While most of these methods have been developed for global illumination or related effects, they can be successfully applied to motion and defocus blur. We believe image-space methods hold great promise for fast and high-quality reconstruction, but argue that these are best suited for adaptive rendering systems, where the possibility exists to refine difficult regions.

For non-adaptive rendering using a fixed number of input samples, domain-specific methods that operate directly in higher-dimensional space should intuitively be able to generate better results. In this case, the goal is to reconstruct the light field for depth of field and motion blur by starting with a sparse set of stochastically generated samples. Shirley

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

et al. [SAC*11] process samples front-to-back and sort a large number of samples per pixel in order to use an ad-hoc filter to produce the final image. Lehtinen et al. [LAC*11] present a higher-quality reconstruction method for high-dimensional light fields, including motion blur, depth of field, and soft shadows. They upsample a sparse light field by exploiting depth and motion vectors, and resolve occlusion by reconstructing plausible surfaces from the samples. This important work has spurred new interest in reconstruction algorithms. Lehtinen et al. [LALD12] have also presented a more accurate method, primarily developed to reconstruct indirect illumination, which they show can be used also for motion and defocus blur. However, this algorithm was reported to be $3-4\times$ slower than their previous method. Vaidyanathan et al. [VMCS13] reconstruct defocus blur by partitioning samples in screen space tiles and depth layers, and derive an anisotropic, separable reconstruction filter for each partition. In the combined case of defocus and motion blur, the light field is anisotropic in a five-dimensional space, and their reconstruction filter is no longer separable. We extend their framework with a new separable anisotropic filter, motivated by a 5D light field analysis. We reconstruct defocus and motion blur with near real-time performance.

3. Overview

Similar to previous work [LAC*11], we work with light field samples on the form: $\mathbf{x} : (x_i, y_i, u_i, v_i, t_i) \longmapsto (z_i, \mathbf{d}_i, l_i)$, e.g., the light field stores a depth z_i, a 3D motion vector $\mathbf{d}_i$, and radiance value l_i, for each 5D input coordinate $(x_i, y_i, u_i, v_i, t_i)$, where (x_i, y_i) are screen-space coordinates, (u_i, v_i) are lens positions, and t_i is the time within the frame. In contrast to most previous work, our algorithm produces a smooth reconstructed image by operating on a fixed number of light field samples per pixel, instead of relying on a feedback loop for adaptive sampling.

We first apply a light field analysis inspired by Belcour et al. [BSS*13] in Section 4 to study the anisotropy of the light field in presence of defocus and motion blur. We make a similar first order approximation of motion, where a a light field sample's motion vector is approximated by its projection on the xy-plane. Following their work, we assume a Gaussian shutter function and lens aperture.

We depart significantly from Belcour et al. in the derivation of the reconstruction filter. Our filter is anisotropic in $xyuvt$, while their filter is anisotropic only in xy. As a consequence, we achieve much higher quality at equal number of samples per pixel. The input samples are grouped into *partitions*, i.e., a set of depth layers for each screen space tile. In contrast to most previous work, where unique filters per pixel are derived, we formulate and apply a common sheared reconstruction filter kernel in 5D for all samples within a partition (Section 5). This, somewhat coarser, approximation allows for a very efficient implementation. We present a non-trivial extension of the filter framework of Vaidyanathan et al. [VMCS13], and show that the 5D filter is separable in xut and yv. We carefully design a novel warped Gaussian distribution such that even an anisotropic xut filter can be plugged into their framework.

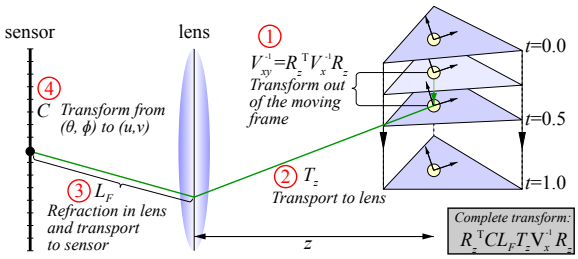


Figure 2: *The light field transformation from a moving diffuse emitter (right) at depth z to the light field at the sensor (left), following Belcour et al.'s analysis. For an efficient filter implementation, we express the motion operation in terms of rotations and a motion along the x-axis, and added a change of basis C to transform from angles to lens coordinates.*

We evaluate the filter by integrating over uvt, and convolving in screen space xy. This gives us a filtered irradiance (e) and opacity (α) for each layer. Finally, the layers are composited in front-to-back order using alpha blending based on the opacity values (Section 6). Implementation details are discussed in Section 7, and we evaluate the visual quality and performance in Section 8 and the accompanying video.

4. Light Field Analysis

In this section, we follow the light field analysis by Durand et al. [DHS*05], and include the extensions of a thin lens [SSD*09] and first-order motion [BSS*13]. We use a space-angle parametrization [DHS*05]: (x, y, θ, ϕ, t), and let corresponding frequencies be denoted Ω, e.g., $x \mapsto \Omega_x$. We transform the light field of a moving, emitter at depth z, to the light field at the sensor using a sequence of linear operators. The light field information will be used to derive a reconstruction filter in Section 5. Our frequency analysis is only valid for a diffuse, fully visible emitter, and does not take occlusions or angular BRDF variations into account. Occlusions are handled by partitioning samples into depth layers. As shown in Figure 2, the sequence of operators is:

1. Transform out of the moving coordinate system (V_{xy}^{-1}).
2. Transport to the lens (T_z).
3. Refraction in lens and transport to sensor (L_F).
4. Coordinate transform $C : (x, y, \theta, \phi, t) \rightarrow (x, y, u, v, t)$.

These four steps form a linear operator:

$$CL_F T_z V_{xy}^{-1}, \tag{1}$$

which transforms the light field from the moving coordinate system at depth z to the light field at the sensor. All

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

operators are 5×5 matrixes [BSS*13], which are included in Appendix A for reference. The differences from Belcour et al.'s analysis are that we include a coordinate transform $C : (\theta, \phi) \mapsto (u, v)$, i.e., from angles to lens coordinates at the sensor, and express the motion operator in terms of rotations (around the z-axis) and motion along x: $V_{xy} = R_z^\top V_x R_z$.

Inserting $V_{xy} = R_z^\top V_x R_z$ into Equation 1 gives:

$$C L_F T_z V_{xy}^{-1} = R_z^\top C L_F T_z V_x^{-1} R_z = R_z^\top M R_z, \qquad (2)$$

where we have used $C L_F T_z R_z^\top = R_z^\top C L_F T_z$. This equality can easily be verified by performing the matrix multiplications. With the change of variables $\mathbf{x}' = R_z \mathbf{x}$, the light field transform can be expressed in the rotated frame (where the motion vector is aligned with the x-axis) as the transform:

$$M = C L_F T_z V_x^{-1} : \underbrace{(x, y, \theta, \phi, t)}_{\text{at depth } z} \rightarrow \underbrace{(x, y, u, v, t)}_{\text{at sensor}}. \qquad (3)$$

The expression for M on matrix form can be found in Appendix A. In the rotated coordinate system, the linear operator (Equation 1) of the light field is simplified to M. As we will see in Section 5, this simplified operator enables a highly efficient reconstruction filter implementation.

In the Fourier domain, we denote frequencies at the sensor with $\hat{\Omega}$ and frequencies at the moving frame at depth z with $\hat{\Omega}^z$. In the same coordinate system as a moving *diffuse emitter*, the radiance spectrum, expressed in (rotated) frequencies $\hat{\Omega}^z = (\Omega_x^z, \Omega_y^z, \Omega_\theta^z, \Omega_\phi^z, \Omega_t^z)$, has the shape:

$$L(\Omega_x^z, \Omega_y^z) \delta(\Omega_\theta^z) \delta(\Omega_\phi^z) \delta(\Omega_t^z). \qquad (4)$$

$L(\Omega_x^z, \Omega_y^z)$ represents the spatial frequencies of the emitter. A diffuse emitter has no angular variation, which gives the Dirac, δ, in Ω_θ^z and Ω_ϕ^z. Also, in the moving frame, the emitter is static, which gives the Dirac in Ω_t^z.

Assuming that the object is moving at a constant speed, we can relate the spectrum at depth z to the spectrum at the sensor using the linear transform, M, from Equation 2. When applying a linear transform M in the primal domain, the corresponding transform in the Fourier domain is $\propto M^{-\top}$ [Hec89]. Thus, in our case: $\hat{\Omega} = M^{-\top} \hat{\Omega}^z$, which can be used to find expressions for the Dirac functions from Equation 4 in terms of sensor frequencies $\hat{\Omega} = (\Omega_x, \Omega_y, \Omega_u, \Omega_v, \Omega_t)$. Using $\hat{\Omega}^z = M^\top \hat{\Omega}$ and inserting the expression for M from Appendix A we obtain:

$$\begin{aligned} \delta(\Omega_\theta^z) &= \delta\Big(\frac{z-F}{F}\Omega_x - z\Omega_u\Big), \\ \delta(\Omega_\phi^z) &= \delta\Big(\frac{z-F}{F}\Omega_y - z\Omega_v\Big), \\ \delta(\Omega_t^z) &= \delta\Big(\frac{d}{F}\Omega_x - d\Omega_u + \Omega_t\Big), \end{aligned} \qquad (5)$$

where F is the focus plane, d is the length of the motion vector, and we assume an image plane at unit length from the lens. These equations show that the light field spectrum

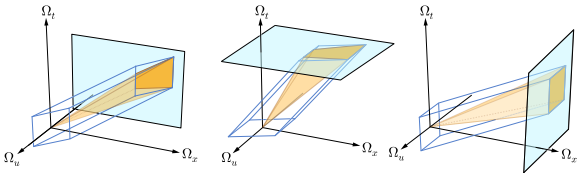


Figure 3: *Light field spectrum bandlimits in the $(\Omega_x, \Omega_u, \Omega_t)$ slice. For clarity of presentation, we only show the part of the spectrum in the positive Ω_t half-space. From Equation 7, the spectrum has energy along a set of lines through the origin (orange frustum). Depending on the motion amplitude and defocus blur, the spectrum bandlimit in either Ω_u (left), Ω_t (middle), or Ω_x (right) will clip the frustum first. We approximate this clipped frustum with a scaled and sheared Gaussian distribution (visualized as a blue parallelepiped).*

energy at the sensor is mostly restricted to a plane, Π, in 5D:

$$\Pi(\Omega_x, \Omega_y) : (\Omega_x, \Omega_y, c(z)\Omega_x, c(z)\Omega_y, -\frac{d}{z}\Omega_x), \qquad (6)$$

where we have introduced the notation $c(z) = \frac{z-F}{zF}$, which is proportional to the circle of confusion. In the next section, we will design a reconstruction filter that captures this spectrum at the sensor.

5. Filter Design

We split our samples into screen space tiles, and partition each tile into depth layers. The goal of this partitioning is to reduce the variation in depth and motion vector (direction and magnitude), thereby allowing for a better reconstruction filter in each partition. Vaidyanathan et al. [VMCS13] assumed a Gaussian aperture, and designed a filter kernel per partition. Their filter can be efficiently evaluated as an integration over the uv dimensions, followed by a convolution in screen space, xy. We extend their approach to include motion blur by assuming a Gaussian shutter function, and aiming at integrating in the uvt dimensions first, followed by screen space convolution. This is in contrast to previous work on sheared filters [ETH*09] and reprojection techniques [LAC*11], where a unique kernel is derived and evaluated for each pixel, which includes performing a costly search in a high-dimensional data structure of samples.

Fourier domain filter From Equation 6, the energy of the light field spectrum for a diffuse emitter moving at constant speed is described by a plane. For a partition of samples with similar depth values and motion vectors, assuming no interaction between the emitters, i.e., no occlusion, the spectrum will be mostly represented as a collection of parametric planes. The planes' coefficients are functions of the samples' depths and motion vectors. Since all planes go through the origin, the spectrum of a set of samples will be mostly bounded by a hyper-wedge, similar to how the shape

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

of the spectrum is a wedge for a 2D slice, e.g., (Ω_x, Ω_u) or (Ω_x, Ω_t) [CTCS00, ETH*09].

A Gaussian aperture bandlimits the spectrum at $\Omega_u = \pm\Omega_u^{\max}$, $\Omega_v = \pm\Omega_v^{\max}$, and the Gaussian shutter at $\Omega_t = \pm\Omega_t^{\max}$. Here, $\Omega_u^{\max} = \Omega_v^{\max} = (2\pi\sigma_a)^{-1}$, where σ_a is the standard deviation of the aperture in the primal domain. $\Omega_t^{\max}$ is defined similarly, using σ_{shutter}. $\Omega_x^{\max}$ stems from the finite resolution of the image, σ_{pixel}.

In our rotated coordinate system (Equation 3), the Ω_y and Ω_v components of the plane in Equation 6 are independent of Ω_t, so we can separate the filter design in the (Ω_y, Ω_v) and $(\Omega_x, \Omega_u, \Omega_t)$ subspaces. The filter in (Ω_y, Ω_v) has been previously described in detail by Vaidyanathan et al. [VMCS13], and can be efficiently implemented using oblique projections to capture the anisotropy of the light field. We instead focus on the filter in the $(\Omega_x, \Omega_u, \Omega_t)$ subspace.

Finding coordinate frame for a partition From Equation 6, in the $(\Omega_x, \Omega_u, \Omega_t)$ subspace, the spectrum from a diffuse emitter at depth z, moving with velocity d along the x-axis will have energy along the parametric line:

$$\mathbf{r}(\Omega_x) = \Omega_x\Big(1, c(z), -d/z\Big). \tag{7}$$

Within each partition of samples, there is some variation in the circle of confusion $c(z) \in [c_{\min}, c_{\max}]$ and (signed) screen space motion vectors $-\frac{d}{z} \in [d_{\min}, d_{\max}]$. For all samples in a partition, we compute the average screen space motion vector and its variance per component. If the variance is small, there is a common motion direction in the partition. The angle ψ between the x-axis and the average motion vector determines the rotation angle, i.e., the rotated frame defined by R_z we apply to the input coordinates to simplify the light field transform (Equation 2). This is shown in Figure 4. We also slightly expand the $[d_{\min}, d_{\max}]$ range as shown in Figure 4 (right) to account for variation in motion direction. The lines produced by $\mathbf{r}(\Omega_x)$ with these variations in motions and depth span a (double) frustum. Note that if the variance is too high, there is no clear motion direction in the partition, and we set $\frac{d}{z} = 0$ and $\psi = 0$. This will disable the motion blur filter.

Depending on the magnitude of motion and the circle of confusion, one of the bandlimits in Ω_u, Ω_t, or Ω_x will clip the frustum first, as shown in Figure 3. In cases where the spectrum is clipped by bandlimits along two or more axes, we simplify our filter design by selecting a single clip plane that is intersected first by the corner ray $\mathbf{r}_c(\Omega_x) = \Omega_x(1, c_{\min}, d_{\min})$. For each of the three clipping scenarios, we approximate the clipped double frustum using a scaled and sheared Gaussian distribution. Below, we discuss the case when the spectrum is clipped against $\Omega_u = \Omega_u^{\max}$.

Clipping against $\Omega_u = \Omega_u^{\max}$: To approximate the clipped spectrum (shown by the blue box in Figure 3 (left)), we transform a three-dimensional isotropic Gaussian distribution in with a scaling, (s_x, s_u, s_t), followed by a sequence of two shears:

1. A shear in Ω_x as function of Ω_u: $\Omega_x' = \Omega_x + \alpha\Omega_u$.
2. A shear in Ω_t as function of Ω_x: $\Omega_t' = \Omega_t + \beta\Omega_x$.

The concatenation of the scaling and shears is a matrix of the form:

$$S = \begin{bmatrix} 1 & \alpha & 0 \\ 0 & 1 & 0 \\ \beta & \alpha\beta & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_u & 0 \\ 0 & 0 & s_t \end{bmatrix}, \tag{8}$$

where the parameters are derived in Appendix B. Given the warp matrix, S, the Gaussian distribution in Fourier space $\hat{\Omega} = (\Omega_x, \Omega_u, \Omega_t)$ can be written as [Hec89]:

$$w(\hat{\Omega}) = |S|^{-1} e^{-\frac{1}{2}\hat{\Omega}^\top (SS^\top)^{-1}\hat{\Omega}}. \tag{9}$$

However, Vaidyanathan et al. [VMCS13] showed that for a 2D warp matrix, if the shear in Ω_x is independent of Ω_u, the filter can be efficiently evaluated. We extend this idea to 3D, and search for a warp matrix on the form:

$$S' = \begin{bmatrix} 1 & 0 & 0 \\ \eta & 1 & 0 \\ \xi & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x' & 0 & 0 \\ 0 & s_u' & 0 \\ 0 & 0 & s_t' \end{bmatrix}, \tag{10}$$

i.e., both shears are purely functions of Ω_x. To replace the transform S with S' *without* changing the Gaussian distribution, we solve the equation $SS^T = S'S'^T$ to obtain:

$$s_x' = \gamma,\ s_u' = s_x s_u \gamma^{-1},\ s_t' = s_t,\ \eta = \alpha s_u^2 \gamma^{-2}, \text{ and } \xi = \beta, \tag{11}$$

where $\gamma = \sqrt{s_x^2 + \alpha^2 s_u^2}$. If the spectrum is instead clipped against $\Omega_t = \Omega_t^{\max}$, the shear transformations are similar, but with the Ω_t and Ω_u axes swapped. The case of clipping against $\Omega_x = \Omega_x^{\max}$ is even simpler, with $S = S'$.

Primal domain filter To create the corresponding filter in the primal domain, we transform an (un-normalized) isotropic Gaussian distribution with standard deviation $\sigma = 1$: $w(\mathbf{x} = (x,u,t); \sigma = 1) = e^{-\frac{1}{2}\mathbf{x}^\top \mathbf{x}}$, using $S'^{-\top}$. The transformed Gaussian is [Hec89]:

$$w(\mathbf{x}) = |S'| e^{-\frac{1}{2}\mathbf{x}^\top (S'S'^\top)\mathbf{x}}. \tag{12}$$

This can be written as a product of three scalar Gaussians:

$$w_{\text{shear}}(x,u,t) = \underbrace{w(x + \eta u + \xi t; \sigma_x)}_{w_x} \underbrace{w(u; \sigma_u)}_{w_u} \underbrace{w(t; \sigma_t)}_{w_t}, \tag{13}$$

where $\sigma_x = (s_x')^{-1}, \sigma_u = (s_u')^{-1}$, and $\sigma_t = (s_t')^{-1}$. This filter is applied to the primal domain light field at the sensor within the current depth layer. Our filter reverts exactly to a sheared motion blur filter [ETH*09] when the lens size is zero and to Vaidyanathan's [VMCS13] sheared defocus blur filter if the motion vectors are zero. These are desirable properties.

Mehta et al. [MWR12] showed that an axis-aligned filter can be separated into an integration (in u) and convolution (in x), which is critical for performance. However,

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

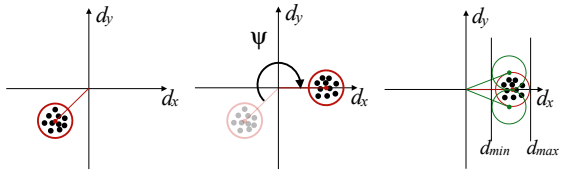


Figure 4: *Bounding and rotating motion vectors. Left: find the average motion vector and a bounding circle. Middle: rotate an angle ψ such that the average motion vector aligns with the x-axis. Right: find $v_{\min}$ and $v_{\max}$ by taking the spread in motion vectors into account.*

our signal in *xut* is highly anisotropic, which would not be captured by an axis-aligned filter. Recently, Vaidyanathan et al. [VMCS13] showed that also a sheared 2D filter (in *xu* and *yv* for defocus blur) can be formulated as an integration and convolution. We use their filter in the *yv* subspace, and as shown above, we successfully extend their technique to the 3D *xut* subspace. With a careful selection of the warp matrix (Equation 10), the primal domain filters (Equation 13) over u and t are independent of x, which allows us to pre-integrate over *ut*. The result is a very fast anisotropic filter.

6. Layer Visibility and Depth Layer Compositing

To composite the filtered layers together and take occlusion between layers into account, we closely follow Vaidyanathan et al.'s [VMCS13] approach. Samples are partitioned into depth layers, and a sample in a layer k represents a ray that has penetrated all layers $j < k$. Thus, we exploit that all samples in k contribute to the visibility approximation of layers in front of it. We track layer radiance and opacity separately per pixel, and composite layers front-to-back using a modified alpha blending. We approximate the irradiance by *pre-integrating* the radiance and opacity over the lens and shutter separately within each layer, i.e., when factoring in the radiance from layer j, we directly use the product of the *average opacities* over the lens and shutter for each layer in front of j. This is in contrast to the exact result, obtained from integrating over the opacity of layers in front of j within the lens integral and over the shutter. Please refer to Vaidyanathan et al.'s paper for a discussion of this approximation.

7. Implementation

For each screen space tile containing $N \times N$ pixels, we collect samples in a search window of size $N_s \times N_s$. We typically use a tile size of $N = 16$ pixels and a search window of $N_s = 32$ pixels, and have found this to be a good tradeoff between quality and performance. If the primal domain filter extends beyond the search window, its width is clamped in order to avoid truncation. This clamping is discussed in Appendix C.

The samples in the search window are statically partitioned in depth. We use a uniformly distributed set of depth layers, and add a set of layers symmetrically around the plane in focus, following Vaidyanathan et al. [VMCS13]. The number of active layers (i.e., layers containing samples) in our test scenes for $N_s = 44$ are listed below.

		Layers/tile(max)	Layers/tile(avg)
ARENA	4 spp	5	1.9
CITADEL	4 spp	10	3.0
SANMIGUEL1	8 spp	11	5.3
SANMIGUEL2	8 spp	8	3.2
WALL	4 spp	10	3.1
HAIRBALL	8 spp	12	5.6

We combine the defocus filter from Vaidyanathan et al. [VMCS13] in *yv* (with a shear ρ in *yv*) and the new filter from Section 5 in *xut*. The 5D reconstruction filter (in rotated coordinates) is a product of five scalar Gaussians:

$$\underbrace{w(x+\eta u+\xi t;\sigma_x)}_{w_x}\underbrace{w(y+\rho v;\sigma_y)}_{w_y}\underbrace{w(u;\sigma_u)}_{w_u}\underbrace{w(v;\sigma_v)}_{w_v}\underbrace{w(t;\sigma_t)}_{w_t}. \quad (14)$$

The integrated opacity, $\bar{\alpha}_j$, and irradiance, e_j, for each pixel in layer j are now computed by a reprojection step followed by a screen space filter.

Algorithm 1 Reprojection step for layer j

```
for all light field samples (x_i, y_i, u_i, v_i, t_i) do
    Compute p, q (Eq. 16), and w = w_u(u_i) w_v(v_i) w_t(t_i)
    if z_i in depth range of layer j then
        I_α(p,q) += w                          ▷ opacity
        I_e(p,q) += l(x_i) w                   ▷ irradiance
        I_w(p,q) += w                          ▷ weight
    else if z_i is in layer k > j then
        I_w(p,q) += w                          ▷ α = 0 in layer j
    end if
end for
```

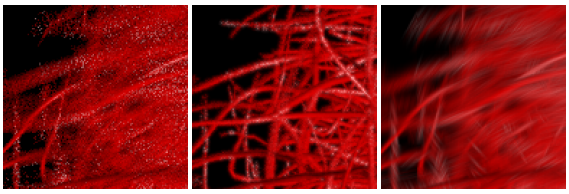
Reprojection The reprojection step is expressed in coordinate system rotated in xy, which is unique for each layer (i.e., determined by the direction of the layer's average motion vector). Let R represent the upper left 2×2 matrix of R_z from Appendix A, i.e., a standard 2D rotation matrix. For each input sample with index i, we accumulate the weight $w = w_u(u_i)w_v(v_i)w_t(t_i)$, weighted radiance wl_i, and weighted opacity $w\alpha_i$ at the rotated pixel position $R[p,q]^\top$:

$$R\begin{bmatrix} p \\ q \end{bmatrix} = R\begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} \eta & 0 \\ 0 & \rho \end{bmatrix} R\begin{bmatrix} u_i \\ v_i \end{bmatrix} + \begin{bmatrix} \xi \\ 0 \end{bmatrix} t_i. \quad (15)$$

However, in order to composite samples from different layers, we reproject and accumulate samples directly in the (unrotated) (x,y) coordinate frame, by simply multiplying Equation 15 with $R^{-1} = R^\top$ from the left:

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + R^\top \begin{bmatrix} \eta & 0 \\ 0 & \rho \end{bmatrix} R\begin{bmatrix} u_i \\ v_i \end{bmatrix} + R^\top \begin{bmatrix} \xi \\ 0 \end{bmatrix} t_i. \quad (16)$$

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.



Input (8 spp) Reprojection Full algorithm

Figure 5: *A cutout from the rotating* HAIRBALL *scene. The middle image shows our algorithm restricted to using only the reprojection step, i.e., without screen space convolution. Since the image is relatively sharp, this can be interpreted as an indication that our visibility approximation works well for complex scenes, even though we project all samples in a depth layer along an average vector, and use a simplified compositing (Section 6).*

The complete reprojection step is outlined in Algorithm 1. Note that we quantize (p,q) to pixels in our implementation. An example of only applying the reprojection step on a sparsely sampled light field is shown in Figure 5.

Screen Space Filter After the reprojection, an anisotropic Gaussian screen space filter (separable convolution in xy) is applied over the reprojected samples. This filter is outlined in Algorithm 2. Finally, as discussed in Section 6, the filtered

Algorithm 2 Screen space filter for layer j

for all pixels (x,y) in N_s **do** ▷ along motion
  for all k in [-radius, radius] **do**
    $x' = x + \cos(\psi)k,\ y' = y - \sin(\psi)k,\ w = w_x(k)$
    Get I_α, I_e, and I_w from Algorithm 1 at (x',y')
    $\bar{\alpha}_j(x,y)$+= $I_\alpha w$, $e_j(x,y)$ += $I_e w$, $\bar{w}(x,y)$+= $I_w w$
  end for
end for
for all pixels (x,y) in N_s **do** ▷ perp. to motion
  for all k in [-radius, radius] **do**
    $x' = x + \sin(\psi)k,\ y' = y + \cos(\psi)k,\ w = w_y(k)$
    $\bar{\alpha}_j(x,y)$+=$\bar{\alpha}_j(x',y')w$, $e_j(x,y)$ += $e_j(x',y')w$,
    $\bar{w}(x,y)$+= $\bar{w}(x',y')w$
  end for
end for
for all pixels (x,y) in N **do** ▷ normalization
  $\bar{\alpha}_j(x,y)$ /= $\bar{w}(x,y)$, $e_j(x,y)$ /= $\bar{w}(x,y)$
end for

layers are composited together using an approximate alpha blending [VMCS13]. The irradiance is approximated as:

$$e(x,y) \approx e_0(x,y) + \sum_{j=1}^{N-1} e_j(x,y) \prod_{k=0}^{j-1} (1 - \bar{\alpha}_k(x,y)). \quad (17)$$

	TLFR		Our $N_s = 32$		Our $N_s = 44$	
	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
ARENA	99.4	47.7	99.2	46.1	99.3	46.1
CITADEL	98.5	45.4	99.1	45.1	99.3	45.9
SANMIGUEL1	92.1	32.6	96.4	36.5	96.5	35.9
SANMIGUEL2	95.9	31.6	97.5	33.4	97.5	35.8
WALL	95.5	32.6	96.7	35.6	96.4	35.4

Table 1: *Structural similarity index (SSIM) scores in %, and peak signal to noise ratio (PSNR) in dB for all test scenes. For both metrics, a higher score indicates less error.*

8. Results

Image Quality Figure 1 and Figure 6 show reconstruction quality with our algorithm for a variety of scenes. Figure 6 also includes a comparison with Lehtinen et al.'s [LAC*11] technique, denoted TLFR, which has reconstruction times of 8-17 s on a discrete GPU. For ease of comparison, we have modified their code to use a Gaussian shutter and aperture. We obtain similar quality with reconstruction times in 28-80 ms. To evaluate image quality, we use the structural similarity index (SSIM) [WBSS04], where we have averaged over the color channels (RGB), as well as PSNR. As can be seen in Table 1, our algorithm has better or similar scores compared with TLFR. Our algorithm allows balancing quality and performance by tuning the size of the search window, N_s. We have found that $N_s = 32$ produces high quality images in most cases and consider it the best tradeoff for real-time applications. However, for high-quality rendering, larger N_s may be desirable, as this helps reduce noise in regions with very large defocus or motion blur. For our test scenes, this was only noticeable in SAN MIGUEL 1 (Figure 1). We include reconstructed images with $N_s = 44$ in the supplemental material. This size is motivated by the available on-chip memory in Direct3D 11 compute shaders.

In practice, we use a clamped filter that is smaller than the maximum motion or circle of confusion in a scene. This was done to limit the search window around each tile, and is important for parallelization of the algorithm. However, we have noted that image quality does not decrease significantly due to this. The stochastic samples locally capture the anisotropy of the motion or depth of field accurately, and the filter typically has enough samples locally to suppress noise.

Compared to a reference solution and TLFR, our filter has more noise in regions with large variations in motion. TLFR tends to expand geometry around some silhouettes, as can be seen around the leaves in the SAN MIGUEL 2 scene in Figure 6. This effect may be less visible in a recent follow-up work [LALD12], where the visibility approximation is tuned by re-tracing all rays and checking for false positives. However, that algorithm is $3-4\times$ slower than TLFR.

The WALL scene in Figure 7 has many small objects with vastly varying motion vectors. This is a difficult case for our algorithm as it stresses our original assumption that the mo-

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

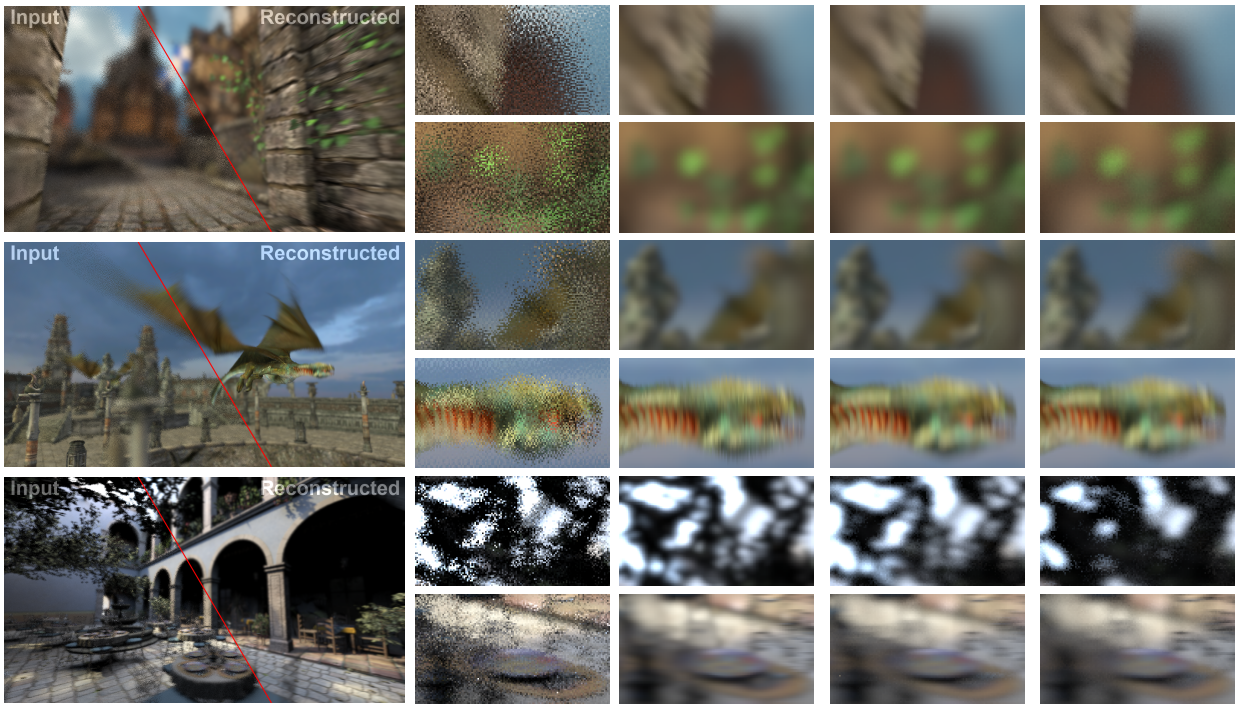


Figure 6: *From top to bottom, the* CITADEL *(4 spp),* ARENA *(4 spp), and* SAN MIGUEL 2 *(8 spp) test scenes with simultaneous motion and defocus blur. The reconstructed images are essentially noise-free, and closely match the reference images.* CITADEL *is adapted from Epic Citadel courtesy of Epic Games, Inc.* ©*2013.* SANMIGUEL *is courtesy of Guillermo M. Leal Llaguno.*

tion in a depth partition within a tile is approximately uniform. Please refer to the accompanying video for the entire animation.

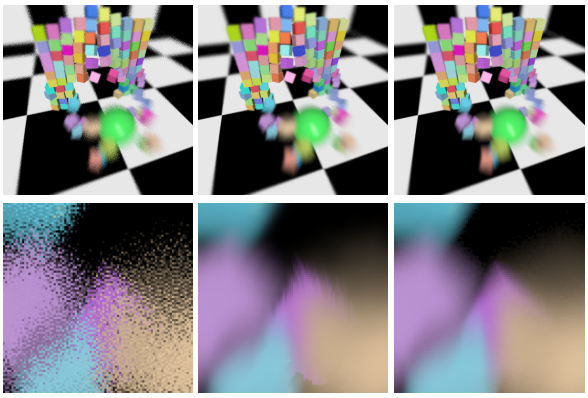
GPU Performance We have measured performance of our implementation on a Geforce GTX 780 discrete graphics card, with thermal design power (TDP) of 250 W, as well on an Intel Iris Pro 5200 integrated GPU (TDP 47 W for CPU+GPU). The image quality generated by our GPU implementation closely matches the CPU implementation, with the exception that HDR color depths are not handled as well on the GPU. Direct3D 11 does not currently support for floating point atomics in Direct3D 11, and parts of the implementation therefore use a 16-bit fixed point color format internally. Performance is summarized in Table 2. The implementation scales linearly with the number of samples per pixel as shown in Figure 8. The algorithmic complexity of reconstructing a pixel is linear in the side length of the search window and number of non-empty depth layers. Interestingly, the GTX 780 scales poorly when increasing the search window. This is likely due to the 32×32 pixel window hitting a sweet spot, where all data fits in on-chip memory. Performance on the Iris Pro scales proportionally to search window size, and we have noted similar behavior on AMD HD7900 hardware.

	TLFR GTX780, 250W	Our GTX780, 250W	Our IRIS PRO, 47W
ARENA	9726 ms	28 / 99 ms	150 / 229 ms
CITADEL	12825 ms	36 / 185 ms	220 / 434 ms
SANMIGUEL1	17111 ms	80 / 406 ms	450 / 808 ms
SANMIGUEL2	14425 ms	51 / 238 ms	308 / 536 ms
WALL	8281 ms	34 / 151 ms	198 / 363 ms

Table 2: *Reconstruction times for different scenes. For our algorithm, execution time is reported with $N_s = 32$ and $N_s = 44$ search windows, where we consider $N_s = 32$ to be a good trade-off for real-time or interactive applications.*

Comparison Against Axis-Aligned Filtering We initially evaluated axis-aligned filters as an alternative to the sheared filters that are used for reconstructing each layer. Those filters are derived by computing axis-aligned frequency bounds in the (Ω_y, Ω_v) and $(\Omega_x, \Omega_u, \Omega_t)$ subspaces using an approach similar to Mehta et al. [MWR12]. Belcour et al. [BSS*13] use an anisotropic Gaussian in the *xyuvt* domain and slice it to produce an anistropic Gaussian in *xy*. This leads to a reconstruction filter that is axis-aligned in *uvt*. Unlike their approaches with a per-pixel filter and adaptive sampling, we apply a common filter for all samples within

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.



Input (4spp) Reconstructed Reference (256 spp)

Figure 7: *A stress test for simultaneous motion blur and depth of field. For the moving objects close to the static floor, there are large variations in the motion vectors within a depth partition, and our reconstruction filter becomes very small. This shows up as noise in some tiles. Please refer to the video to see this scene in motion.*

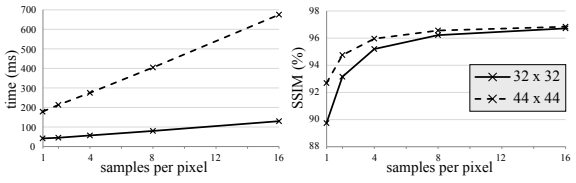


Figure 8: *Performance and SSIM scores of our GPU implementation (on GTX780), with increasing number of input samples per pixel. We used the* SAN MIGUEL 1 *scene for this test. The ceiling on image quality is due to that the reference image is generated using 1024 spp in* `pbrt` *with super-sampled shading. Also, we do not account for shadows nor non-diffuse shading.* TLFR *converges similarly in our tests.*

each depth partition in our comparison. As mentioned above, this common filter is anisotropic in xy, but axis-aligned in uvt. Figure 9 shows a comparison of reconstruction quality with axis-aligned vs. sheared filters on SAN MIGUEL 1. This is the expected quality of Belcour's and Mehta's filters if adapted to the same restrictions (fixed spp and fixed motion per tile). The reconstruction quality with a common axis-aligned filter for each depth partition in a tile is inadequate for sparsely sampled inputs, unless combined with an adaptive sampling pass.

Limitations The layered visibility approximation may introduce artifacts. The technique is fundamentally tile-based (in both screen space and depth) with a fixed search window. For a thin object flying past the camera very quickly, details can be missed. Still, with 8 spp and $N_s = 44$, we have 15k samples within the tile, so the probability of completely missing objects is low. If there is too large variance in motion within a depth partition, we revert to a small axis-aligned filter. Furthermore, the algorithm does not incorporate shadows nor indirect illumination. e.g., a shadow on a static receiver in focus is not filtered. Figure 10 highlights these cases. At low input sampling rates, some streaking artifacts are visible, e.g., in the dragon's head in Figure 6. This can be mitigated by using a slightly larger pixel filter (σ_{pixel}), but at the cost of minor decrease in sharpness.

Axis-aligned SSIM: 86.3% Sheared (Our) SSIM: 95.8% Reference

Figure 9: SAN MIGUEL 1 *at 4 spp, reconstructed using an axis-aligned filter vs. our sheared filter.*

Our Fourier analysis exploits bandlimits from the Gaussian aperture and shutter, similar to previous work. One could go "against the theory" and instead apply, e.g., a hexagonal aperture in the filtering step (as suggested by Mehta et al. [MWR12] for area lights). A $2N$-gonal aperture can be implemented as N separable filters.

We use the same approximate alpha blending as Vaidyanathan et al. [VMCS13] (Equation 17). Please refer to their paper for a thorough discussion of this approximation. The blending formula is only approximated at layer $j > 0$ and it gets coarser as j increases (but the layer contribution to the final pixel color typically decreases with increasing j). Layer 0 (the frontmost layers) is not affected by this approximation. For example, a scene with a flat moving surface sloping away from the camera (e.g., CITADEL) still produces correct opacity.

9. Conclusions

In this paper, we have used recent frequency analysis tools for light transport to derive an efficient layered reconstruction algorithm for simultaneous motion and defocus blur. We have cautiously used a set of approximations, including, Gaussian shutter and lens, analysis using a diffuse emitter only, a layered representation, a single filter per partition, and using motion in only xy. Despite these approximations and the fact that motion and defocus blur reconstruction is a much more difficult problem to tackle than defocus blur alone, we have demonstrated that using a fixed, predetermined number of stochastic samples per pixel (e.g., 4–8) results in reconstructed images of very high quality. We found that our image quality is on par with previous work,

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

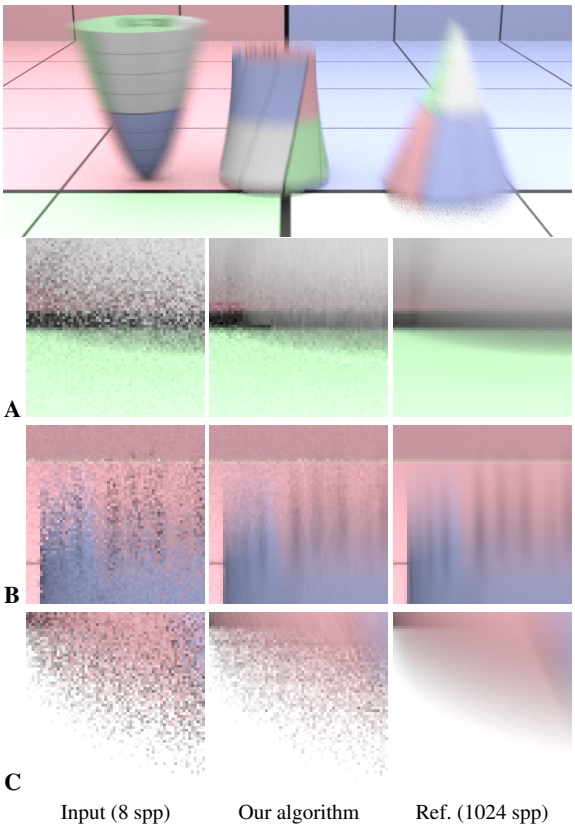


Figure 10: *The three shapes move along x, y, and z respectively. The large image shows our reconstructed result from 8 spp. If there is too much motion variation within a depth layer, we revert to a small filter. This is visible on the bottom of the moving objects (**A**). This effect diminishes with increasing number of depth layers and smaller tile sizes, but is an inherent limitation from our approximation of one motion direction per partition. At the top of the objects, there is large depth separation, and the filter works fine (**B**). Furthermore, we do not filter shadows or indirect illumination, so soft shadows below the objects remain noisy (**C**).*

or better. Our implementation on a high-end GPU with our high-quality settings (which we recommend for offline rendering) runs about $40-100\times$ faster than the previously best method for reconstruction (TLFR). Using our basic setting ($N_s = 32$), we found that our algorithm is about $200-360\times$ faster, and this is what we recommend for real-time rendering. Note that with the latter settings, the execution time of our reconstruction pass is between 30–80 ms per frame, which can be considered real time. Even on an integrated graphics processor on a laptop, our algorithm runs at interactive rates. We believe that our work is a big step forward for rapid and high-quality image reconstruction for motion and defocus blur.

Acknowledgements

We thank the anonymous reviewers and Intel's Advanced Rendering Technology (ART) team. We also thank David Blythe, Tom Piazza and Chuck Lingle for supporting this research. Tomas Akenine-Möller is a *Royal Swedish Academy of Sciences Research Fellow* supported by a grant from the Knut and Alice Wallenberg foundation. The CITADEL test scene is courtesy of Epic Games, Inc., SANMIGUEL was modeled by Guillermo M. Leal Llaguno, www.evvisual.com. HAIRBALL is courtesy of Samuli Laine and MISCQUADS is taken from the PBRT distribution. We are very grateful for all these scenes. Many thanks also to Lehtinen et al. for providing the TLFR source code.

References

[BEM11] BAUSZAT P., EISEMANN M., MAGNOR M. A.: Guided Image Filtering for Interactive High-quality Global Illumination. *Computer Graphics Forum (Proceedings of EGSR), 30*, 4 (2011), 1361–1368. 2

[BSS*13] BELCOUR L., SOLER C., SUBR K., HOLZSCHUCH N., DURAND F.: 5D Covariance Tracing for Efficient Defocus and Motion Blur. *ACM Transactions on Graphics, 32*, 3 (2013), 31:1–31:18. 2, 3, 4, 8, 11

[Coo86] COOK R. L.: Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics, 5*, 1 (1986), 51–72. 1

[CTCS00] CHAI J.-X., TONG X., CHAN S.-C., SHUM H.-Y.: Plenoptic Sampling. In *Proceedings of SIGGRAPH 2000* (2000), ACM, pp. 307–318. 2, 5

[DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A Frequency Analysis of Light Transport. *ACM Transactions on Graphics, 24*, 3 (2005), 1115–1126. 2, 3, 11

[DMB*14] DELBRACIO M., MUSÉ P., BUADES A., CHAUVIER J., PHELPS N., MOREL J.-M.: Boosting Monte Carlo Rendering by Ray Histogram Fusion. *ACM Transactions on Graphics, 33*, 1 (2014), 8:1–8:15. 2

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-Avoiding À-Trous Wavelet Transform for Fast Global Illumination Filtering. In *High Performance Graphics* (2010), pp. 67–75. 2

[EDR11] EGAN K., DURAND F., RAMAMOORTHI R.: Practical Filtering for Efficient Ray-Traced Directional Occlusion. *ACM Transactions on Graphics, 30*, 6 (2011), 180:1–180:10. 2

[EHDR11] EGAN K., HECHT F., DURAND F., RAMAMOORTHI R.: Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders. *ACM Transactions on Graphics, 30*, 2 (2011), 9:1–9:13. 2

[ETH*09] EGAN K., TSENG Y.-T., HOLZSCHUCH N., DURAND F., RAMAMOORTHI R.: Frequency Analysis and Sheared Reconstruction for Rendering Motion Blur. *ACM Transactions on Graphics, 28*, 3 (2009), 93:1–93:13. 2, 4, 5

[GMN13] GUERTIN J.-P., MCGUIRE M., NOWROUZEZAHRAI D.: *A Fast and Stable Feature-Aware Motion Blur Filter*. Tech. Rep. NVR-2013-003, NVIDIA Corporation, 2013. 2

[Hec89] HECKBERT P. S.: *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, University of California, Berkeley, 1989. 4, 5

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

[LAC*11] LEHTINEN J., AILA T., CHEN J., LAINE S., DURAND F.: Temporal Light Field Reconstruction for Rendering Distribution Effects. *ACM Transactions on Graphics, 30*, 4 (2011), 55:1–55:12. 1, 2, 3, 4, 7

[LALD12] LEHTINEN J., AILA T., LAINE S., DURAND F.: Reconstructing the Indirect Light Field for Global Illumination. *ACM Transactions on Graphics, 31*, 4 (2012), 51:1–51:10. 1, 2, 3, 7

[LES10] LEE S., EISEMANN E., SEIDEL H.-P.: Real-Time Lens Blur Effects and Focus Control. *ACM Transactions on Graphics, 29*, 4 (2010), 65:1–65:7. 2

[MESL10] MCGUIRE M., ENDERTON E., SHIRLEY P., LUEBKE D.: Real-Time Stochastic Rasterization on Conventional GPU Architectures. In *High Performance Graphics* (2010), pp. 173–182. 2

[MHBO12] MCGUIRE M., HENNESSY P., BUKOWSKI M., OSMAN B.: A Reconstruction Filter for Plausible Motion Blur. In *Symposium on Interactive 3D Graphics and Games* (2012), pp. 135–142. 2

[ML85] MAX N. L., LERNER D. M.: A Two-and-a-half-D Motion-Blur algorithm. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (1985), vol. 19, ACM, pp. 85–93. 2

[MWR12] MEHTA S., WANG B., RAMAMOORTHI R.: Axis-Aligned Filtering for Interactive Sampled Soft Shadows. *ACM Transactions on Graphics, 31*, 6 (2012), 163:1–163:10. 2, 5, 8, 9

[MWRD13] MEHTA S. U., WANG B., RAMAMOORTHI R., DURAND F.: Axis-Aligned Filtering for Interactive Physically-based Diffuse Indirect Lighting. *ACM Transactions on Graphics 32*, 4 (2013), 96:1–96:12. 2

[ODR09] OVERBECK R. S., DONNER C., RAMAMOORTHI R.: Adaptive Wavelet Rendering. *ACM Transactions on Graphics, 28*, 5 (2009), 140:1–140:12. 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics 29*, 4 (2010), 66:1–66:13. 2

[PC81] POTMESIL M., CHAKRAVARTY I.: A Lens and Aperture Camera Model for Synthetic Image Generation. In *Computer Graphics (Proceedings of SIGGRAPH 81)* (1981), vol. 15, pp. 297–305. 2

[PC83] POTMESIL M., CHAKRAVARTY I.: Modeling Motion Blur in Computer-generated Images. In *Computer Graphics (Proceedings of SIGGRAPH 83)* (1983), pp. 389–399. 2

[PD84] PORTER T., DUFF T.: Compositing Digital Images. In *Computer Graphics (Proceedings of SIGGRAPH 84)* (1984), vol. 18, ACM, pp. 253–259. 2

[RKZ12] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive Rendering with Non-Local Means Filtering. *ACM Transactions on Graphics, 31*, 6 (2012), 195:1–195:11. 2

[SAC*11] SHIRLEY P., AILA T., COHEN J., ENDERTON E., LAINE S., LUEBKE D., MCGUIRE M.: A Local Image Reconstruction Algorithm for Stochastic Rendering. In *Symposium on Interactive 3D Graphics and Games* (2011), ACM, pp. 9–14. 3

[SD12] SEN P., DARABI S.: On Filtering the Noise from the Random Parameters in Monte Carlo Rendering. *ACM Transactions on Graphics, 31*, 3 (2012), 18:1–18:15. 2

[SSD*09] SOLER C., SUBR K., DURAND F., HOLZSCHUCH N., SILLION F.: Fourier Depth of Field. *ACM Transactions on Graphics, 28*, 2 (2009), 18:1–18:12. 1, 2, 3

[TM98] TOMASI C., MANDUCHI R.: Bilateral Filtering for Gray and Color Images. In *Proceedings of ICCV* (1998), IEEE Computer Society, pp. 839–846. 2

[VMCS13] VAIDYANATHAN K., MUNKBERG J., CLARBERG P., SALVI M.: Layered Light Field Reconstruction for Defocus Blur. *referred to ACM Transactions on Graphics* (2013). http://software.intel.com/en-us/articles/layered-light-field-reconstruction-for-defocus-blur. 1, 2, 3, 4, 5, 6, 7, 9

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing, 13*, 4 (april 2004), 600–612. 7

[WWB*14] WALD I., WOOP S., BENTHIN C., JOHNSON G. S., ERNST M.: Embree – A Kernel Framework for Efficient CPU Ray Tracing. *ACM Transactions on Graphics (to appear)* (2014). 2

[XP05] XU R., PATTANAIK S. N.: A Novel Monte Carlo Noise Reduction Operator. *IEEE Computer Graphics and Applications, 25* (2005), 31–35. 2

Appendix A: Light Field Operators

In this section, we list the light transport operators [DHS*05, BSS*13] applied to the primary domain light field. For the lens operator, L_F, we set the image plane at unit distance from the lens and express it in terms of the distance to the plane in focus, F, using the thin lens formula: $\frac{1}{f} = 1 + \frac{1}{F}$, where f is the focal length. At the sensor, we transform from angular coordinates to lens coordinates: $(u,v) = (x + \tan\theta, y + \tan\phi)$, which, similar to previous work, we locally approximate with $(x+\theta, y+\phi)$. We denote this coordinate transform $C : (x,y,\theta,\phi,t) \rightarrow (x,y,u,v,t)$, and concatenate C to the lens operator. The lens and transport operators are:

$$CL_F = \begin{bmatrix} -\frac{1}{F} & 0 & -1 & 0 & 0 \\ 0 & -\frac{1}{F} & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_z = \begin{bmatrix} 1 & 0 & -z & 0 & 0 \\ 0 & 1 & 0 & -z & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We use Belcour et al.'s first order motion operator, but further approximate 3D motion with motion purely in xy, so the angular motion terms (d_θ, d_ϕ) are zero. Our motion operator is applied in a rotated coordinate system (using the rotation matrix R_z below), where the rotation angle ψ is chosen so the motion in y is zero. Let $c = \cos\psi$ and $s = \sin\psi$, then:

$$V_x = \begin{bmatrix} 1 & 0 & 0 & 0 & d \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_z = \begin{bmatrix} c & -s & 0 & 0 & 0 \\ s & c & 0 & 0 & 0 \\ 0 & 0 & c & -s & 0 \\ 0 & 0 & s & c & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The light field transform in a rotated coordinate system, where the motion vector is aligned with the x-axis, is:

$$M = CL_F T_z V_x^{-1} = \begin{bmatrix} -\frac{1}{F} & 0 & \frac{z-F}{F} & 0 & \frac{d}{F} \\ 0 & -\frac{1}{F} & 0 & \frac{z-F}{F} & 0 \\ 1 & 0 & -z & 0 & -d \\ 0 & 1 & 0 & -z & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.

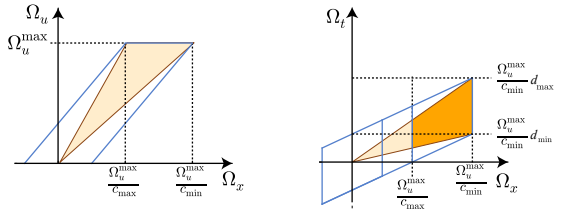


Figure 11: *Projection of the 3D frequency bounds (light orange) shown in Figure 3 for the case where the spectrum gets clipped along the Ω_u axis. The blue box represents the Gaussian distribution that approximates the frequency bounds. Left: projection in the (Ω_x, Ω_u) space. Right: projection in the (Ω_x, Ω_t) space, showing the frequency bounds and intersection of the spectrum with the $\Omega_u = \Omega_u^{\max}$ plane (orange polygon).*

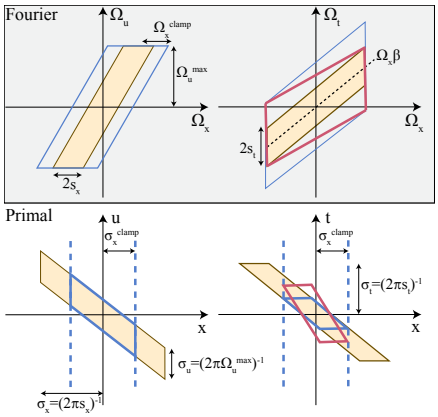


Figure 12: *Projection of the filter bounds in the (Ω_x, Ω_u) slice (left) and (Ω_x, Ω_t) (right). If the filter is too large in the primal domain, we expand the frequency domain scale factors s_x and s_t, such that the filter in the primal domain (lower row) is reduced. Optionally, as shown in red, we also slightly adjust the shear coefficient β, such that the primal filter encompasses more samples. This slightly increased image quality in our test scenes.*

Appendix B: Filter Parameters

As discussed in Section 5, we approximate the clipped frequency spectrum by applying a scale followed by two shears to an isotropic Gaussian distribution. We derive the scale and shear values separately for the three cases, where the spectrum gets clipped either at the Ω_u, Ω_t, or Ω_x axis. Given a primal domain Gaussian with standard deviation σ, we choose the bandlimit as $\Omega^{\max} = (2\pi\sigma)^{-1}$. In our implementation, we use $\sigma_{\text{pixel}} = 1/3$, $\sigma_a = 1/3$, and $\sigma_{\text{shutter}} = 2/3$.

Clipping against $\Omega_u = \Omega_u^{\max}$: Figure 11 (left) shows the frequency bounds in the (Ω_x, Ω_u) space, where the spectrum is clipped at $\Omega_u = \Omega_u^{\max}$. The transformed Gaussian, which approximates the light field spectrum, is represented by the blue parallelogram. This is derived from an isotropic unit Gaussian, by first applying a scaling in frequency space by:

$$(s_x, s_u, s_t) = 2\pi\Omega_u^{\max}\left(\left|\frac{c_{\max} - c_{\min}}{2c_{\min}c_{\max}}\right|, 1, \left|\frac{d_{\max} - d_{\min}}{2c_{\min}}\right|\right),$$

where (s_x, s_u) are equal to half of the width and height of the parallelogram. Then shear an amount α in Ω_x as function of Ω_u, where $\alpha = \frac{c_{\max}+c_{\min}}{2c_{\min}c_{\max}}$. The scaling coefficient s_t, is obtained by looking at Figure 11 (right), which shows a projection of the frequency bounds in the (Ω_x, Ω_t) space. Here, the blue box, which represents the transformed 3D Gaussian, projects to a parallelogram with height $h = \Omega_u^{\max}\left|\frac{d_{\max}-d_{\min}}{c_{\min}}\right|$. This gives $s_t = \pi h$. Finally, the second shear $\beta = \frac{d_{\max}+d_{\min}}{2}$ applied in Ω_t as function of Ω_x is the average slope of the light field spectrum in (Ω_x, Ω_t).

Appendix C: Filter Clamping

If the spatial extents of the primal domain filter extend beyond the search window, its width is clamped in order to avoid truncation. In the Fourier domain, this corresponds to an expansion of the included frequencies. Figure 12 shows an expanded Fourier domain filter for the case when the frequency bounds are clipped against $\Omega_u = \Omega_u^{\max}$ (Figure 3 left). We increase the scale factor s_x such that the filter frequency bounds grow, resulting in a more conservative filter with smaller extents in the primal domain (bottom row). The scale s_t is also increased and, optionally the shear β modified, as shown in Figure 12 (right). In our implementation, we clamp the spatial extents of the primal domain spatial filter to $\sigma_x^{\text{clamp}} = \frac{N_s - N}{4}$. This ensures that at least 95% (2σ) of the clamped Gaussian filter's integral lies within the search window, which is sufficient to avoid tile artifacts. Clamping is applied in a similar manner when the spectrum is clipped against the other clip planes.

© 2014 The Author(s)
Computer Graphics Forum © 2014 The Eurographics Association and John Wiley & Sons Ltd.