High Performance Graphics (2012)
C. Dachsbacher, J. Munkberg, and J. Pantaleoni (Editors)

Adaptive Image Space Shading for Motion and Defocus Blur

Karthik Vaidyanathan[1], Robert Toth[1], Marco Salvi[1], Solomon Boulos[2], and Aaron Lefohn[1]

[1]Intel Corporation [2]Stanford University

Figure 1: *Shading cost comparison for a complex scene rendered without motion and defocus blur (left), stochastic motion and defocus blur with decoupled sampling (center), and stochastic motion and defocus blur with our adaptive anisotropic sampling technique (right). Our approach reduces shading cost for this scene by a factor of three compared to the other two techniques.*

Abstract

We present a novel anisotropic sampling algorithm for image space shading which builds upon recent advancements in decoupled sampling for stochastic rasterization pipelines. First, we analyze the frequency content of a pixel in the presence of motion and defocus blur. We use this analysis to derive bounds for the spectrum of a surface defined over a two-dimensional and motion-aligned shading space. Second, we present a simple algorithm that uses the new frequency bounds to reduce the number of shaded quads and the size of decoupling cache respectively by 2X and 16X, while largely preserving image detail and minimizing additional aliasing.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Rendering methods based on advanced optics models have been used for decades in the off-line rendering community, although such techniques have been out of the reach for real-time graphics systems. Stochastic rasterization provides an attractive alternative to the standard pinhole camera model supported by current rasterization pipelines and it has gained traction in the real-time graphics research community. While new types of rasterization have the potential of improving image quality by incorporating realistic motion and defocus blur effects into the real-time domain, they require shading many samples per pixel, which poses severe limitations to their feasibility.

This problem can be addressed by decoupling visibility from shading while performing the latter at lower rate. Current real-time graphics APIs support a limited form of decoupling with multi-sampling anti-aliasing [Ake93] (MSAA), which shades primitives once per pixel while sampling visibility at higher rates. However, efficient shading in a stochastic rasterization pipeline requires further decoupling visibility from shading to efficiently handle blurry primitives covering large regions of the image [MCH*11]. The shading rate can be more efficiently controlled by using advanced decoupling techniques that map visibility samples to a separate shading space via a *memoization cache* [RKLC*11].

We also note that blurring an image reduces its frequency content. This implies that is possible to render an accurate image using a lower shading rate than is used for a static (i.e. not blurred) image.

DOI: 10.2312/EGGH/HPG12/013-021

To exploit this observation we improve upon previous image space decoupled sampling algorithms by using Fourier analysis to derive frequency bounds, to which the signal of a moving and defocused surface may be band-limited. We use these bounds to guide the shading rate, without having a noticeable impact on the image quality.

We also introduce the concept of *anisotropic adaptive sampling*, where we align the shading space to the direction of motion. This method, in conjunction with our newly derived frequency bounds, makes it possible to sample the scene signal along the main axis of motion at a significantly lower rate, while still resolving fine detail along the orthogonal axis.

We implement anisotropic adaptive sampling in a decoupled sampling system, and show that the shading rates dictated by our frequency analysis results in up to 50% reduction in shading and minimal impact on image quality. Furthermore, we demonstrate a 16X reduction of the size of the memoization cache size over previous work without impacting performance.

Our primary contributions are:

- Deriving lower shading rates for shading motion and defocus blurred primitives in a stochastic rasterization pipeline by analyzing which parts of a surface spectra are visible; and
- Introducing a motion-aligned shading space that allows using the aforementioned reduced shading rates.

2. Related Work

The earliest real-time GPU implementations of stochastic rasterization, most notably the implementation by McGuire et al. [MESL10], shade using MSAA. A shader is thus invoked for each pixel overlapped by a primitive. This approach is inefficient with large blurs, as shown by Munkberg et al. [MCH*11].

Ragan-Kelley et al. [RKLC*11] introduced decoupled sampling for real-time graphics pipelines using a separate shading space. Similarly to Reyes, the shading space is independent of the time and aperture distributions. The amount of defocus and motion therefore do not affect shading rates significantly, and the authors also mention adaptively shading at a rate depending on the circle of confusion. Liktor et al. [LD12] propose a new data structure called *compact geometry buffer* which allows implementing decoupled sampling techniques on current graphics hardware.

Micropolygon pipelines [CCC87] are popular for offline rendering. In such systems, geometry is tessellated into grids of pixel-sized primitives and each vertex is shaded prior to visibility sampling. The amount of motion and defocus do therefore not significantly affect the number of shaded points. Furthermore, modern micropolygon renderers have support for adaptive shading rates for defocus and motion blur [Pix09]. However, these systems can only – to the best of our knowledge – select shading rates along the parametric axis of the geometry. This significantly limits the amount of shading reduction that can be achieved without perceivably degrading image quality.

Burns et al. [BFM10] proposed decoupling the shading space from the grids in micropolygon renderers, to allow larger primitives to be rasterized while preserving most surface detail. While they do not discuss adaptive shading rates, our analysis should be applicable to the shading space used by their architecture.

Frequency analysis has lately been used for many aspects in graphics. Durand et al. [DHS*05] present a general framework for analysing light transport, and discuss complex interactions such as occlusion and surface BRDFs. Chai et al. [CTCS00] employ frequency analysis to determine required sampling rates of light-fields to reconstruct views. Soler et al. [SSD*09] analyse the frequency content, and required sampling rates, over both the image and lens for rendering depth of field. Egan et al. [ETH*09] use frequency analysis to determine suitable reconstruction filters for stochastically rendered motion blurred images.

Loviscach [Lov05] works in texture space and integrates texture footprints over time for a gaussian shutter using modified gradients for the EWA texture filter [GH86].

3. Frequency Analysis

Surfaces exhibiting motion and defocus often do not convey high frequency surface detail, due to blur. We reduce the surface shading rate without introducing significant errors, under the assumption that the shader output frequency can be bandwidth limited for these surfaces. We do so by estimating spectral bounds in shading space that constitute a significant contribution to the final image. In this section, we will first characterize the image contribution of a surface using Fourier analysis, and then derive these bounds.

We can express the output signal value $O(x,y)$ at a point x,y in the image using the equation:

$$O(x,y) = E * R \tag{1}$$

where E is the irradiance, which is convolved with R, a reconstruction filter that is chosen to reduce aliasing that might result from discretizing the signal $O(x,y)$ [MN88].

We define the irradiance E as

$$E(x,y) = \int_{\mathbb{R}^3} L(x,y,u,v,t)A(u,v)S(t)\,dudvdt, \tag{2}$$

where L is the radiance at (x,y) corresponding to the point (u,v) on the camera lens at time t. We ignore the lens form factor [KMH95], which is a fairly common assumption [CPC84]. $S(t)$ is the camera shutter response and $A(u,v)$ describes the shape of the camera aperture.

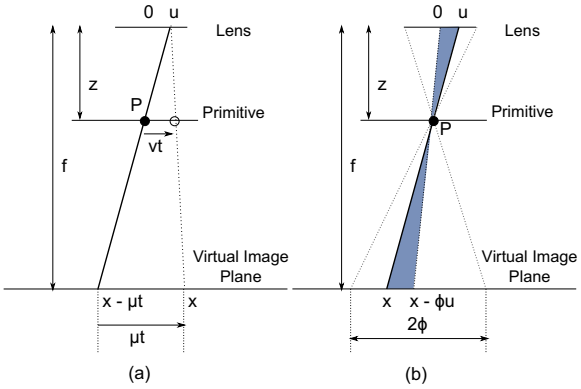

Figure 2: *(a) Knowing a point in image space at t we can determine the position at $t = 0$ based on it's image space velocity μ. (b) Knowing a point in image space at u, v we can predict the position at $u, v = 0, 0$ based on it's circle of confusion ϕ. Note that ϕ is a signed value.*

Similarly to Reyes [CCC87] and the decoupled sampling approach [RKLC*11], we assume that the radiance L corresponding to a point $\vec{p}$ on a surface is constant inside the shutter interval and across all points on the lens. We can therefore always evaluate radiance on the 2D subspace given by slicing the temporal light field at (u_0, v_0, t_0). We call this space the *shading space* and the corresponding 2D radiance function L':

$$E(x,y) = \int_{\mathbb{R}^3} L'(x_0,y_0)A(u,v)S(t)\,dudvdt \tag{3}$$

We will now derive the shading-space coordinates (x_0, y_0) on which to evaluate L'.

Referring to Figure 2, a shift in the lens position produces a proportional shift in image space. The amount of shift ϕ is governed by $\phi = k_c \frac{p_z - f}{p_z}$, where f is the focus distance and k_c is a constant scale that depends on the camera lens system. We assume $\vec{p}$ has a constant velocity in screen space. While this is not always true, it is often a reasonable approximation. With this simplification, if we know the location (x, y) of a point in image space for a given (u, v, t) we can compute the shading space position (x_0, y_0), that is, the point at $(u_0, v_0, t_0) = (0, 0, 0)$:

$$\begin{aligned} x_0 &= x - \mu_x t - \phi u \\ y_0 &= y - \mu_y t - \phi v \end{aligned} \tag{4}$$

We call this space the *shading space* and the corresponding 2D radiance function L', where $L'(x_0, y_0) = L(x, y, u, v, t)$ By substituting Equation 4 into Equation 3, we obtain:

$$E(x,y) = \int_{\mathbb{R}^3} L'(x - \mu_x t - \phi u, y - \mu_y t - \phi v)A(u,v)S(t)\,dudvdt$$

We will now apply a series of variable changes in order to express this integral as convolutions to facilitate the frequency analysis.

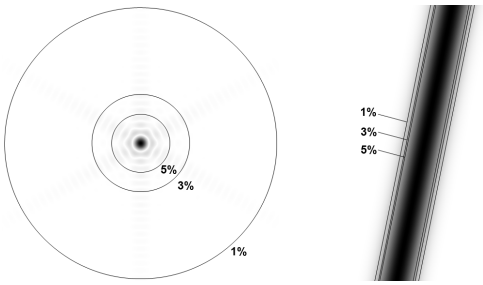

Figure 3: *Left: Spectrum of A' for a hexagonal aperture. The circles show cutoff radii $\Omega^{\max}_{A'}$ that contain all but a small fraction of the spectrum energy, as indicated by their labels. Right: Spectrum of S' for a Gaussian shutter. The labeled lines show cutoff widths $\Omega^{\max}_{S'}$.*

By introducing $A'(u,v) = \frac{1}{\phi^2} A(\frac{u}{\phi}, \frac{v}{\phi})$, we can rewrite the equation above as:

$$\begin{aligned} E(x,y) &= \int_{\mathbb{R}^3} L'(x - \mu_x t - u, y - \mu_y t - v)A'(u,v)S(t)\,dudvdt \\ &= \int_{\mathbb{R}} \left(L' * A'\right)(x - \mu_x t, y - \mu_y t)S(t)dt \end{aligned} \tag{5}$$

We can also rewrite the time integral in Equation 5 as a convolution by mapping the time domain to a line along the direction of motion in 2D space; $x' = \mu_x t$ and $y' = \mu_y t$. Therefore the shutter response S gets transformed to its spatial analog S' and we get:

$$S'(x',y') = \delta(y'\mu_x - x'\mu_y)\frac{1}{\|\vec{\mu}\|} S\left(\frac{(x',y')\cdot\vec{\mu}}{\|\vec{\mu}\|^2}\right)$$

$$\begin{aligned} E(x,y) &= \int_{\mathbb{R}^2} \left(L' * A'\right)(x - x', y - y')S'(x',y')dx'dy' \\ &= \left(L' * A' * S'\right)(x,y) \end{aligned}$$

We can now write the computed pixel values as:

$$O(x,y) = \left(L' * A' * S' * R\right)(x,y)$$

or, finally, in the Fourier domain as:

$$\mathcal{F}(O) = \mathcal{F}(L')\mathcal{F}(A')\mathcal{F}(S')\mathcal{F}(R) \tag{6}$$

3.1. Frequency Bounds at Shader Output

Now that we have expressed the spectral content of the image in *shading space*, we can draw some interesting conclusions. From Equation 6, we can see that the spectrum of O is the product of the spectrum of L', A', S' and R. It is therefore safe to bandlimit L' to the support of the spectrum of A', S' and R. By bandlimiting the shading space L', we may sample shading less densely, and thus reduce the cost of shading.

As with traditional real-time rendering, actually bandlimiting shading according to the shading sample spacing is the

Figure 4: *The required sampling frequencies are calculated using several quantities, which are shown for a frame from the ARENA scene. Left: The minimum circle of confusion radius of the primitives. Center: The minimum screen space velocity of the primitives (with constant vertex velocity approximation). Right: Span of motion directions, $\hat{\theta}$.*

responsibility of the shader author; we are only interested in safe limits to which the shader *should* bandlimit its output (by means of texture filtering or otherwise) and determine sample spacing accordingly. A', S' and R have typically infinite support in the frequency domain, but in practice a reasonable threshold can be used. As example this is illustrated for a hexagonal aperture in Figure 3.

While A' and R are often roughly radially symmetric, and thus boundable by radii $\Omega_{A'}^{\max}$ and $\Omega_R^{\max}$ in frequency space, this is not the case for S'. The spectrum of S' is compressed in the direction of motion, and extends unattenuated in the orthogonal direction. This is illustrated in Figure 3. The spectrum of S' is related to the spectrum of S as follows:

$$\begin{aligned}
\mathcal{F}(S')(\vec{\Omega}) &= \iint S'(x,y)e^{-2\pi i\vec{\Omega}\cdot(x,y)}dxdy \\
&= \int S'(\mu_x t,\mu_y t)e^{-2\pi i(\vec{\Omega}\cdot\vec{\mu})t}dt \\
&= \int \frac{1}{\|\vec{\mu}\|} S\left(\frac{(\mu_x t,\mu_y t)\cdot\vec{\mu}}{\|\vec{\mu}\|^2}\right) e^{-2\pi i(\vec{\Omega}\cdot\vec{\mu})t}dt \\
&= \frac{1}{\|\vec{\mu}\|}\int S(t)e^{-2\pi i(\vec{\Omega}\cdot\vec{\mu})t}dt \\
&= \frac{1}{\|\vec{\mu}\|}\mathcal{F}(S)(\vec{\Omega}\cdot\vec{\mu}). \qquad (7)
\end{aligned}$$

From Equation 7 we see that if the spectrum of S is bounded by the shutter constant $\Omega_S^{\max}$, then the spectrum of S' is bounded by $\Omega_{S'}^{\max} = \|\vec{\mu}\|^{-1}\Omega_S^{\max}$ in the direction of motion.

3.2. Frequency Bounds For a Primitive

Up until now, we have considered a single point moving at constant velocity. For real scenes, the motion direction and magnitude, as well as the defocus amount, vary over a primitive and during the shutter interval (see Figure 4). This would also produce variations in the frequency response of A' and S'.

We can however approximate the overall frequency bounds based on the frequency response computed at the bounding values of $\|\vec{\mu}\|$, θ and ϕ. The underlying assumption for this approximation is that a significant portion of the spectral energy lies between the extents of the variation. This is similar to the assumption used in Chai et al. [CTCS00] and Egan et al. [ETH*09].

We can estimate the cutoff frequency of A', $\Omega_{\Delta A'}^{\max}$, by identifying the smallest circle of confusion radius $\phi_{\min}$ for the primitive. Assuming linear motion in clip space, this can easily be detected as follows: first determine the depths of each vertex at the start and end of the shutter interval, and determine the minimum and maximum of these depths. If they are on opposite sides of the plane in focus, then A' cannot be bounded. Otherwise, compute $\phi_{\min}$ using the depth that is closer to the plane in focus. Finally, the cutoff radius for the primitive is $\Omega_{\Delta A'}^{\max} = \phi_{\min}^{-1}\Omega_A^{\max}$, where the lens dependent constant $\Omega_A^{\max}$ is the cutoff radius of A.

We approximate the bounds of $\mathcal{F}(S')$ using the lowest screen space velocity within the primitive. We define $\vec{\mu}_i$ to be the screen space velocity of each vertex i of the primitive. Velocity is assumed to vary linearly over a primitive in clip space, and each point of the primitive will thus have a screen space velocity that is within the convex hull of $\{\vec{\mu}_i\}$. For the common case of triangular primitives, the convex hull is just the triangle itself. In order to compute frequency bounds for S' over the entire primitive, we will first determine three quantities: the minimum speed $\|\mu_\Delta^{\min}\|$ of the primitive, and the interval $\hat{\theta}$ of velocity directions. The quantities are illustrated in Figure 5.

The minimum speed $\|\mu_\Delta^{\min}\|$ can be computed using conventional closest-point-in-convex-hull algorithms between $\{\mu_i\}$ and the origin. Computing $\hat{\theta}$ is also straightforward and will not be described here. If $\|\mu_\Delta^{\min}\| = 0$, then $\mathcal{F}(S')$ has infinite extents. Otherwise, since $\hat{\theta}$ contains the motion directions of all points on the primitive, we bound $\mathcal{F}(S')$ over the primitive by taking the union of the bounds of the spectra of S' along each point on the arc defined by $\|\mu_\Delta^{\min}\|$ and $\hat{\theta}$ as illustrated in Figure 5. The resulting shape $\Omega_{\Delta S'}^{\max}$ is an hourglass defined by $\Omega_{S'}^{\max}(\|\mu_\Delta^{\min}\|)$ and the extremes of $\hat{\theta}$, and is illustrated in Figure 6.

With $\Omega_{\Delta A'}^{\max}$, $\Omega_{\Delta S'}^{\max}$ and $\Omega_R^{\max}$ determined, we can derive a bounding box $\Omega_\Delta^{\max}$ in the frequency domain, that bounds $\mathcal{F}(A' * S' * R)$ for the entire triangle. The bounding box, as depicted in Figure 6, is aligned to the vector which points towards the center of $\hat{\theta}$, which we denote $\hat{e}_\mu$. We denote the orthogonal vector $\hat{e}_\perp$. We let $\Omega_\Delta^{\max}$ extend to $r = \min(\Omega_{\Delta A'}^{\max}, \Omega_R^{\max})$ along $\hat{e}_\perp$.

To determine the extents along $\hat{e}_\mu$, we intersect the circle

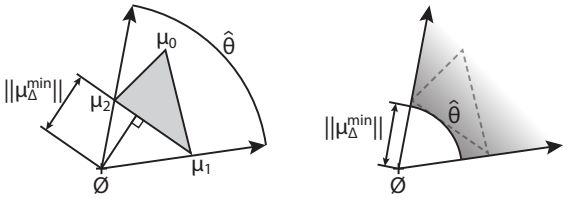

Figure 5: *Left: A triangle that represents the three vertex velocities μ_i in a space spanned by μ_x and μ_y. The velocity direction span $\hat{\theta}$ and the minimum speed $\|\mu_\Delta^{min}\|$ can be determined from this triangle. Right: An arc that represents the direction span $\hat{\theta}$ and the minimum velocity $\|\mu_\Delta^{min}\|$. We can bound the spectrum of each point in the primitive by bounding the spectrum on the arc.*

with radius r with any one of the four lines that define $\Omega_{\Delta S'}^{max}$; this gives us up to two intersection points $\vec{q}_i$. We project the two points $\vec{q}_i$ onto $\hat{e}_\mu$ to get the final extents of Ω_Δ^{max}. The bounding box dimensions are given by:

$$d_\mu = 2\left(r\cos\hat{\theta} + \sqrt{r^2 + \Omega_{S'}^{max}(\|\mu_\Delta^{min}\|)^2}\sin\hat{\theta}\right) \quad (8)$$

$$d_\perp = 2r \quad (9)$$

If $\Omega_{S'}^{max}(\|\mu_\Delta^{min}\|)$ is larger than r, the spectrum of $A' * R$ is tighter than that of S'. In this case we use a square bounding box $d_\mu = d_\perp = 2r$.

To conclude, we have shown that it is safe to bandlimit the shader output L' to include only frequencies contained in the oriented bounding box Ω_Δ^{max}.

3.3. Tight Packing in Frequency Space

In most rendering systems, the shader output L' is point sampled which produces frequency replicas that may overlap to produce aliasing artifacts. The spacing of these frequency replicas is the inverse of the sample spacing in the primal domain. Therefore to avoid visible aliasing artifacts the sample spacing must be small enough to ensure that a significant portion of the spectral energy does not overlap.

With the assumption that a significant part of the shader output spectrum is contained in the oriented bounding box Ω_Δ^{max}, we can derive a sampling grid such that the replicas of Ω_Δ^{max} do not overlap. Moreover in order to sample L' efficiently, we also have to ensure that the replicas are tightly packed. Figure 7 shows two different sampling strategies and the corresponding frequency replicas. It can be seen that the tightest packing of replicas can be achieved with an anisotropic sampling grid oriented along $\hat{e}_\mu$.

The sample spacing along $\hat{e}_\mu$ and $\hat{e}_\perp$ is given by the inverse of the bounding box dimensions d_μ and $d_\perp$ derived in Equations 8 and 9.

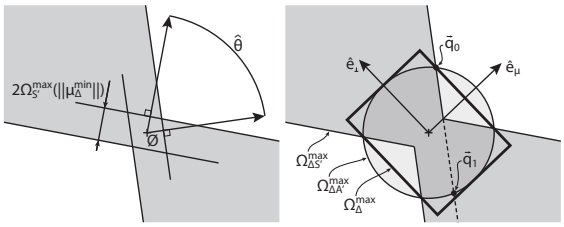

Figure 6: *Derivation of frequency bounds for a primitive. Left: Each point on the arc shown in Figure 5 produces a band in the frequency domain (Figure 3). The width of the band is $\Omega_{S'}^{max}(\|\mu_\Delta^{min}\|)$ and depends on the minimum velocity. Tracing such bands for all points on the arc produces an hourglass shape $\Omega_{\Delta S'}^{max}$. Right: The desired frequency bounds can be determined as the intersection of $\Omega_{\Delta S'}^{max}$ and $\Omega_{A'}^{max}$. We can easily bound this intersection with an oriented bounding box.*

3.4. Anisotropic Mapping Function

Ragan-Kelley et al. [RKLC*11] show that 5D samples can be mapped to shading space using a 2D projective mapping function M_p. To account for the grid orientation and increased sample spacing we introduce an additional transform M_g. Therefore the overall mapping function is $M = M_g M_p$ where M_g is given by:

$$M_g = \begin{bmatrix}\hat{e}_\mu & \hat{e}_\perp\end{bmatrix}^T \begin{bmatrix} d_\mu & 0 \\ 0 & d_\perp \end{bmatrix}$$

M_g applies a rotation and scaling such that the anisotropic sampling grid gets transformed to a unit pixel grid. Therefore after transformation by M_g, derivative computations using finite differences and texture filtering can be performed as in a conventional graphics pipeline. With the modified mapping function, input textures are automatically bandlimited for the anisotropic sampling grid.

We also note that to avoid artifacts from extrapolation of shader attributes, it is important to constrain the shading points to always lie inside primitive boundaries. If the center of a pixel in shading space is found to lie outside the primitive, the shading point has to be clamped to the primitive boundaries [RKLC*11], We address this problem by analytically determining a point on the primitive that is closest to the center of the shading pixel [Eri05].

3.5. Cost And Quality vs. Complexity Balance

Although vertices move linearly in clip space, their screen space velocities are not generally constant within a frame. To conservatively bound $\mathcal{F}(S')$, the velocity space convex hull used to determine $\|\mu_\Delta^{min}\|$ and $\hat{\theta}$ should include the velocities both at the start and end of the shutter interval. In practice, average velocities can be used instead, reducing the cost of the closest-point computation.

With this simplification, the number of operations re-

Parameter	ADD	MUL	MISC
$\Omega^{max}_{\Delta A'}$	15	1	1
$\Omega^{max}_{S'}(\|\|\mu^{min}_{\Delta}\|\|), \hat{\theta}$	26	35	17
$d_{\mu}, d_{\perp}$	3	4	4
Total	44	40	22

Table 1: *Estimated cost of evaluating parameters required to compute the bounding box Ω^{max}_{Δ}. Costs are listed separately for additions/subtractions, multiplications/divisions and other miscellaneous operations such as reciprocals, trigonometric functions and square roots.*

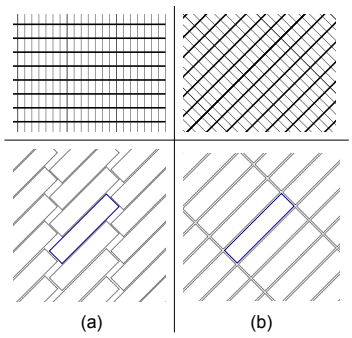

Figure 7: *Sampling grids in shading space (top row) and the corresponding frequency domain replicas of Ω^{max}_{Δ} (bottom row): (a) Packing along y followed by packing along x (b) Sampling grid oriented along $\hat{e}_{\mu}$. The oriented sampling grid gives the best packing of frequency replicas.*

quired to compute the bounding box Ω^{max}_{Δ} for each primitive is listed in Table 1.

For real-time applications to which computational efficiency is more important than correctness, there are opportunities for further reducing the cost of the computations. For example, the velocity parameters could be computed at a reduced precision. Computation of $\Omega^{max}_{\Delta A'}$ could also be simplified by calculating the circle of confusion at the center of the shutter interval instead of computing it at the start and the end of the shutter time.

4. Results

For evaluation purposes we have implemented both our algorithm (AAS) and the decoupled sampling method (DS) introduced by Ragan-Kelley et al. [RKLC*11] as extensions to a software simulator of the D3D11 rendering pipeline, modified to support stochastic rasterization. Our framework substitutes the standard 2D rasterizer with a 5D hierarchical stochastic rasterizer based on recent work by Munkberg et al. [MAM12]. The rasterizer uses a 3-level hierarchy, from a top level tile of 8x8 pixels down to a leaf level tile of 2x2 pixels.

While Ragan-Kelley et al. [RKLC*11] experiment with reducing shading rates for defocus blur, they do not provide any relation between the reduction factor and image quality. We therefore do not apply any adaptive approach for DS in scenes with defocus blur.

As shown in Figure 10, we test DS and AAS under three different scenarios. ARENA presents a complex scenario with a combination of camera motion, character animation and large camera defocus. This represents a sequence typical of an in-game cut scene. SUBD, a scene from the D3D11 SDK, displays a character animation with large variations in motion but no defocus effects. Finally CITADEL is a level from Epic Games' Unreal SDK and includes rapid movements of the player camera combined with moderate defocus. The magnitude of motion is highest for the CITADEL scene. The CITADEL scene includes a post-process pass where stochastic rasterization is disabled. We therefore do not include the shader executions for this post-process pass. All scenes are rendered at a resolution of 1280x720 pixels with 16 samples per pixel and a with a 16 tap anisotropic texture filter. We use these scenes in unmodified form and do not incorporate any additional bandlimiting in the shaders.

For the ARENA scene we use two different lens models. A *sharp* lens model with a truncated circular aperture and a *smooth* lens model with a slow falloff. The smooth lens has a reduced spectral support as compared to the sharp lens and therefore makes it possible to sample more efficiently (i.e. further lowering the shading cost) without significant compromise on image quality. The smooth lens function is derived by applying a smoothstep around the edge of the lens $[0.9r, 1.1r]$, where r is the lens radius.

4.1. Performance

To measure shading performance and the required cache sizes with the two sampling techniques, we chose one representative frame from each of the three test sequences. We pick a frame that has large blur which presents a more challenging scenario for shading reuse. These frames are shown in Figure 10. Figure 8 shows the shading cost (number of shaded quads) with DS and AAS under different cache size constraints.

For the ARENA scene, it can be seen that DS requires a cache size of 1K entries to achieve close to its optimal shading cost. With a smooth lens model, AAS can lower this shading cost by more than 53% with a cache size which is 16 times smaller (64 entries). With this cache size, the shading cost with DS is around nine times higher than AAS. With a sharp lens, AAS can lower the shading cost by more than 40% with a cache size of 256 entries.

Similarly, in the CITADEL scene DS requires a cache size of at least 1K entries to achieve close to its lowest shading cost, while AAS achieves a 75% reduction in this cost with a cache size of just 64 entries. The SUBD scene has a lower magnitude of blur as compared to the other scenes and therefore both DS and AAS require smaller caches in

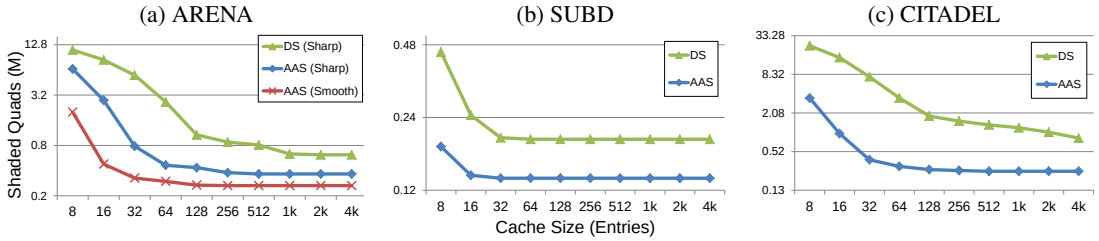

Figure 8: *A comparison of the shading cost in terms of the number of shading quads with Decoupled Sampling (DS) and Adaptive Anisotropic Sampling (AAS) for different cache sizes. The shading cost is presented on a logarithmic scale. DS requires a cache size of 1K entries to achieve close to its lowest shading cost across the three test scenarios. With a soft lens model, AAS can achieve a 30% to 50% reduction in shading cost with a cache size that is 16 times smaller (64 entries).*

this scenario. In spite of the relatively small blur magnitude, AAS can achieve close to 31% reduction in shading costs as compared to DS.

We also measure shading costs across multiple frames for the ARENA scene as shown in Figure 9. This sequence has a combination of motion and defocus blur with reduced motion blur towards both the ends of the sequence. Because of the large spectral support of the hard lens model, the savings in shading cost is largely derived from motion blur. Therefore the shading cost is lowest at the center of the sequence where the savings is close to 50%. At the ends of the sequence the savings are much lower at close to 8%. With the soft lens model however, the shading cost is consistently low with savings between 50% to 60% across all frames.

4.2. Quality

Examples of the visual quality obtained from adaptive anisotropic sampling are shown in Figure 10.

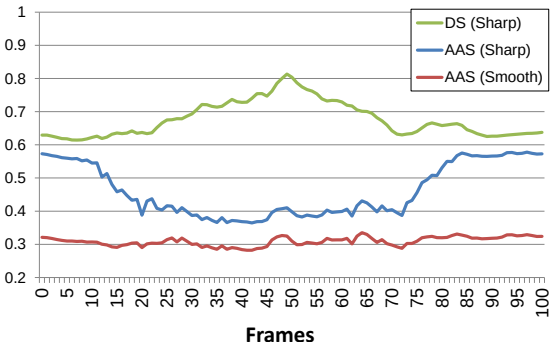

Figure 9: *Shading costs (millions of shaded quads per frame) with the DS and AAS techniques for individual frames in the ARENA animation sequence. With a sharp lens model, AAS derives a large portion of the savings in shading cost from motion blur. Therefore depending on the amount of motion blur in each frame, the shading cost varies significantly with savings between 8% to 50%. With the smooth lens model, the savings are consistent (50% to 60%).*

The most noticeable difference in the images produced by Decoupled Sampling (DS) and Adaptive Anisotropic Sampling (AAS) is reduced noise as a result of improved texture filtering. In scenes with large motion and defocus blur, 16 samples per pixel is usually not adequate for producing noise free images. By modifying the shader to use blur-adaptive texture filtering methods such as Loviscach [Lov05] this noise can be effectively reduced in regions of the image that are fully covered by a primitive. With our method, blur-adaptive texture filtering is automatically provided by the anisotropic sampling grids. Noise can be further reduced with aperture and shutter functions that have sharper falloffs in the frequency domain. For instance Egan et al. [ETH*09] assume a Gaussian function as the shutter function.

With large motion or defocus blur, adaptive texture filtering can produce large texture footprints. This can lead to increased filtering across texture seams and may produce artifacts. In such cases there is a visible improvement in image quality when shading points are clamped to primitive boundaries, as can be seen in Figure 11. In order to completely avoid sampling across texture seams, techniques like seamless texture atlases [PCK04] can be used.

With AAS, it is also important to bandlimit specular light-

Figure 11: *Filtering across texture seams. Left: without clamping and Right: with clamping. There is a visible improvement in quality with clamping as texture footprints are centered inside the primitive.*

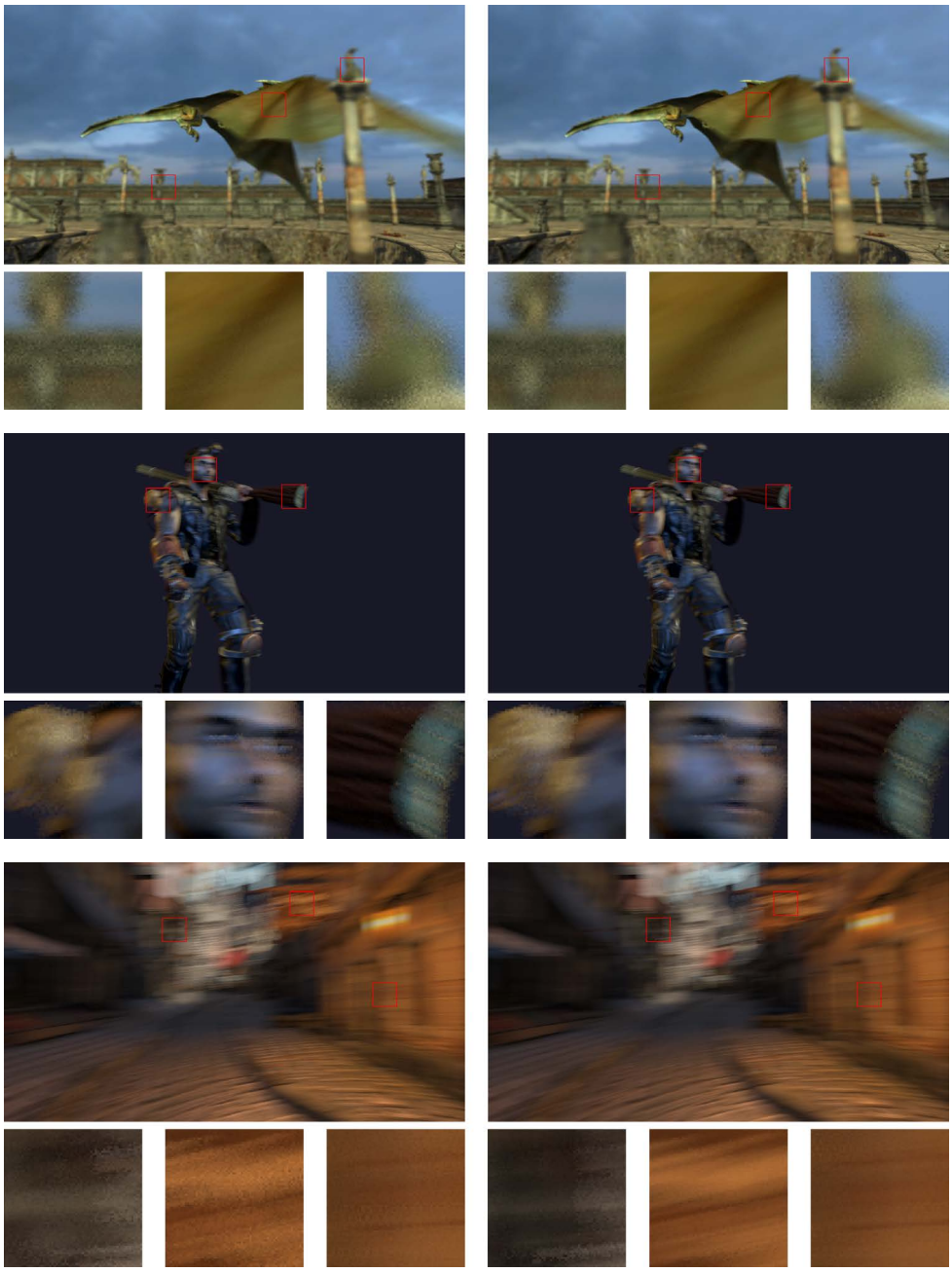

Figure 10: *Quality comparison between Decoupled Sampling (DS, left) and Adaptive Anisotropic Sampling (AAS, right). Top: ARENA scene. The foreground blur on pillar ornament is accurately reproduced. The far wall has a high frequency bump map which is reproduced to a lesser degree of accuracy due to inadequate bandlimiting in the shader. Motion on dragon wings is reproduced very well. Middle: SUBD scene. This is a challenging scene due to a large number of specular objects. With AAS smoother regions such as the face are accurately reproduced while sharp specular regions including the backpack and the gun have minor noise artifacts. Bottom: CITADEL scene. This scene has large motion blur which results in noisy images with only 16 samples per pixel. However AAS produces less noise as a result of improved texture filtering. The anisotropic features on the signboard (middle inset) are well preserved with a 16 tap anisotropic filter. There are small differences in the background region which can be caused by filtering across texture seams.*

ing, bump maps and sharp shadows as they can produce artifacts as seen in Figure 10. Inadequate bandlimiting can also produce visible temporal artifacts. These issues can be mitigated by adopting methods that can filter these shading terms in real-time such as [OB10].

5. Conclusion

We introduce a shading system for a stochastic rasterization pipeline that dynamically sets anisotropic shading rates based on the amount of motion and defocus blur. We derive these shading rates from the estimated output frequency of the shaders on the blurry surfaces, assuming that shaders are properly band limited and constant from the beginning to end of the frame. The result is that we can render images that are similar in quality to previously described decoupled shading pipelines, but shade two to three times fewer points and require up to sixteen times less storage for the decoupled shading cache.

The assumptions we make to support our derivation are based on approximations used previously in rendering systems (notably in the original Reyes pipeline). We demonstrate results that show the assumptions hold for a number of cases, and the errors that result when they do not hold are often not objectional. However, future work includes designing a pipeline that allows users to compute some shading terms, such as shadows, at higher sampling rates (e.g., once per pixel), while leaving the majority of the shading computation at the reduced rates we derive in this paper.

6. Acknowledgements

The authors thank Charles Lingle, Aaron Coday, and Tom Piazza at Intel for supporting this research. We thank Jacob Munkberg, Petrik Clarberg, Nir Benty and Uzi Sarel at Intel for contributing to our rasterization and simulation infrastructure. We also thank Jon Hasselgren and Magnus Andersson for helping prepare the test scenes. Finally we thank Epic Games for the CITADEL scene.

References

[Ake93] AKELEY K.: RealityEngine Graphics. In *Proceedings of SIGGRAPH 93* (1993), ACM, pp. 109–116. 1

[BFM10] BURNS C. A., FATAHALIAN K., MARK W. R.: A Lazy Object-Space Shading Architecture with Decoupled Sampling. In *Proceedings of High-Performance Graphics 2010* (2010), pp. 19–28. 2

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The Reyes Image Rendering Architecture. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (1987), vol. 21, ACM, pp. 95–102. 2, 3

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed Ray Tracing. In *Computer Graphics (Proceedings of SIGGRAPH 84)* (1984), vol. 18, ACM, pp. 137–145. 2

[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, 4

[DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A frequency analysis of light transport. *ACM Transactions on Graphics 24* (2005), 1115–1126. 2

[Eri05] ERICSON C.: *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology)*. Morgan Kaufmann, 2005. 5

[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* (2009), 93:1–93:13. 2, 4, 7

[GH86] GREENE N., HECKBERT P.: Creating raster omnimax images from multiple perspective views using the elliptical weighted average filter. *Computer Graphics and Applications, IEEE 6*, 6 (june 1986), 21 –27. doi:10.1109/MCG.1986.276738. 2

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A Realistic Camera Model for Computer Graphics. In *Proceedings of SIGGRAPH 1995* (1995), ACM, pp. 317–324. 2

[LD12] LIKTOR G., DACHSBACHER C.: Decoupled deferred shading for hardware rasterization. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2012), I3D '12, ACM, pp. 143–150. 2

[Lov05] LOVISCACH J.: Motion Blur for Textures by Means of Anisotropic Filtering. In *Rendering Techniques 2005* (2005), pp. 105–110. 2, 7

[MAM12] MUNKBERG J., AKENINE-MÖLLER T.: Hyperplane Culling for Stochastic Rasterization. Unpublished draft. Submitted to Eurographics 2012, 2012. 6

[MCH*11] MUNKBERG J., CLARBERG P., HASSELGREN J., TOTH R., SUGIHARA M., AKENINE-MÖLLER T.: Hierarchical Stochastic Motion Blur Rasterization. In *Proceedings of High-Performance Graphics 2011* (2011), ACM, pp. 107–118. 1, 2

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

[MN88] MITCHELL D. P., NETRAVALI A. N.: Reconstruction filters in computer-graphics. *SIGGRAPH Comput. Graph. 22* (June 1988), 221–228. URL: http://doi.acm.org/10.1145/378456.378514, doi:http://doi.acm.org/10.1145/378456.378514. 2

[OB10] OLANO M., BAKER D.: Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 181–188. URL: http://doi.acm.org/10.1145/1730804.1730834, doi:10.1145/1730804.1730834. 9

[PCK04] PURNOMO B., COHEN J. D., KUMAR S.: Seamless texture atlases. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), SGP '04, ACM, pp. 65–74. URL: http://doi.acm.org/10.1145/1057432.1057441, doi:10.1145/1057432.1057441. 7

[Pix09] PIXAR: RenderMan Studio 2.0 Documentation, 2009. URL: http://penguin.ewu.edu/RenderMan/RMS_2.0/. 2

[RKLC*11] RAGAN-KELLEY J., LEHTINEN J., CHEN J., DOGGETT M., DURAND F.: Decoupled Sampling for Graphics Pipelines. *ACM Transactions on Graphics, 30*, 3 (2011). 1, 2, 3, 5, 6

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