DOI: 10.1111/j.1467-8659.2010.01840.x

COMPUTER GRAPHICS *forum*
Volume 30 (2011), number 1 pp. 3–26

Motion Blur Rendering: State of the Art

Fernando Navarro[1], Francisco J. Serón[2] and Diego Gutierrez[2]

[1]Lionhead Studios, Microsoft Games Studios
fernandn@microsoft.com
[2]Universidad de Zaragoza, Instituto de Investigación en Ingeniería de Aragón (I3A), Spain
{seron, diegog}@unizar.es

Abstract

Motion blur is a fundamental cue in the perception of objects in motion. This phenomenon manifests as a visible trail along the trajectory of the object and is the result of the combination of relative motion and light integration taking place in film and electronic cameras. In this work, we analyse the mechanisms that produce motion blur in recording devices and the methods that can simulate it in computer generated images. Light integration over time is one of the most expensive processes to simulate in high-quality renders, as such, we make an in-depth review of the existing algorithms and we categorize them in the context of a formal model that highlights their differences, strengths and limitations. We finalize this report proposing a number of alternative classifications that will help the reader identify the best technique for a particular scenario.

Keywords: motion blur, temporal antialiasing, sampling and reconstruction, rendering, shading, visibility, analytic methods, geometric substitution, Monte Carlo sampling, postprocessing, hybrid methods

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Antialiasing

1. Introduction

Motion blur is an effect that manifests as a visible streak generated by the movement of an object in front of a recording device. It is the result of combining apparent motion in the scene and an imaging media that integrates light during a finite exposure time. This relative motion can be produced from object movement and camera movement and can be observed both in still pictures and image sequences. In general, sequences containing a moderate amount of motion blur are perceived as natural, whereas its total absence produces jerky and strobing movement.

This phenomenon is an integral effect of photography and film recording. It needs to be accounted for and in some cases compensated with the use of specific techniques [Ada80]. Moreover, it has become part of the toolkit used by filmmakers and photographers [Pet08] and is commonly used to enhance the perception of motion in still images (Figure 1).

Motion blur for synthetic images has been an active area of research from the early 1980s. Unlike recorded footage that automatically integrates motion blur, rendering algorithms need to explicitly simulate it. In these cases, an accurate temporal integration is needed to avoid aliasing artefacts that can be easily noticed by an untrained eye. This simulation is one of the most expensive processes in the production of high-quality renders. This cost becomes more relevant knowing that the result is a blurred image whose high spatial frequencies have been removed.

In this work, we describe the origin of the phenomenon and make a detailed discussion of the algorithmic solutions that have been found to simulate it. In Section 2, we start with a description of the physical phenomena that generate motion blur in a recording device. In Section 3, it is mathematically formalized based on the models by Meredith [MJ00] and Sung *et al*. [SPW02]. In Section 4, we review the techniques that can produce synthetic motion blur and classify them according to this formalization. Their similarities and

Figure 1: *Motion blur is frequently used to increase the perceived motion in photographic snapshots. As seen in the images, carefully controlling shutter timing, camera motion, illumination, lens and filter configurations can produce dramatic effects. Images reproduced with permission of their respective authors: © Tony Margiocchi, flickr users Noodle Snacks and E01, Carl Rosendahl, Peter Heacox and Enrique Mandujano.*

differences are analysed in Section 5. Section 6 briefly discusses current research trends and future directions.

2. Origin of the Phenomenon

The amount of light entering a camera, independently of whether it uses photosensitive film or an electronic sensor, is limited by the diaphragm and shutter [Ada80]. The diaphragm defines the size and shape of the aperture where the light enters the body of the camera, and it statically influences the amount of exposure of the film, depth of field, optical aberrations, vignetting and field of view. Shutters are mechanical or electronic devices designed to limit the exposition to a finite range of time. Once opened, and because the media integrates all light hitting its surface, several views of a moving object may be projected to different areas of the image plane. The resulting interactions between light, diaphragm, shutter, exposed media and object motion produce motion blur.

This exposure process can be formalized using Equation (1). During scene capture, $I(\omega)$ represents the contents of the image plane when the scene is seen in the direction ω. Captured light is the result of the integration of the incoming radiance L during the exposition time when the shutter is open ΔT. $f(\omega, t)$ models the influence of optics, shutter, aperture and film.

$$I(\omega) = \int_{\Delta T} f(\omega, t)\, L(\omega, t)\, \mathrm{d}t. \qquad (1)$$

The previous equation already gives clues about the characteristics of the resulting images. If each picture in a sequence is generated as an instantaneous snapshot, quickly moving objects will miss parts of their trajectories and the continuity of the sequence will be lost. Even if the shutter range is long enough, different exposures may be separated by gaps which may result in double images and ghosting.

For computer-generated images, this effect needs to be explicitly simulated. Rendered image sequences usually look more natural when they integrate motion blur and three basic reasons are commonly accepted for this [Gla99]. First, images recorded using real cameras automatically incorporate motion blur. Even natural images as seen by our eyes contain a similar effect, motion smear [Bur80]. As such, motion blur is part of what we expect to find in any moving image. Secondly, sequences of images lacking motion blur can contain strobing artefacts. This makes the predictions of the human visual system (HVS) difficult and object trajectories may not be understood as continuous paths. Finally, rendering algorithms discretize continuous signals producing different forms of aliasing as a side effect. Reducing temporal aliasing by avoiding time undersampling is the main target of motion blur rendering techniques. An overview of these will be given in Section 4.

This work focuses on the problems associated with the generation of motion blur, but there are many complementary fields. Interested readers can find useful references for the algorithms used to simulate realistic camera lenses [BHK*03], depth of field [BK08], film emulsion grain and exposition [GM97] or the dynamic range of recording devices [RD06]. The inverse problem, image deblurring and restoration has also been covered elsewhere [KH96].

3. Formalization of the Motion Blur Problem

Different motion blur rendering algorithms have been proposed based on several formal models [Sha64, PC83, ML85, Mit91, Shi93]. Probably, the most widely accepted describes a motion blurred image as a spatio-temporal integral with terms representing the geometric and shading functions. Equation (2) mathematically states this idea. This expression follows the formulation of Sung *et al*. [SPW02], but similar descriptions can be found in [MJ00].

$$I_{xy} = \sum_l \int_\Omega \int_{\Delta T} r(\omega, t) g_l(\omega, t) L_l(\omega, t) \, \mathrm{d}t \, \mathrm{d}\omega. \quad (2)$$

In this equation, I_{xy} represents the contents of the image pixel with coordinates (x, y) and Ω is its corresponding subtended solid angle. Independently of their geometrical representation, the overall contribution of all primitives in the scene is considered by iterating over each individual object l. $g_l(\omega, t)$ is a geometrical term that accounts for occlusions between objects. Its value is 1 if object l is directly visible in the direction ω, 0 otherwise. As we have seen, shutter shape and efficiency, lens aberrations and film influence the final image. The reconstruction filter $r(\omega, t)$ accounts for their overall effect. In general, this term cannot be split into pure spatial and temporal components. However, an approximation that is widely accepted is to replace it with the product of two filters $r(\omega, t) = r_s(\omega)\, r_t(t)$, where each term exclusively depends on one of the dimensions. $L_l(\omega, t)$ represents the radiance of object l without explicitly establishing the method by which is calculated.

To account for the complex spatio-temporal relationships taking place in an animated scene, all terms are evaluated at an instantaneous time t over the aperture time ΔT, and over the solid angle Ω. In some cases and depending on the desired filter footprint, Ω can represent narrower or wider solid angles than the one defined by the pixel.

Although $g_l(\omega, t)$ is a binary function that assumes that a single object is visible at ω at a specific instant, this should not be considered a limitation of the model because the radiance term $L_l(\omega, t)$ already accounts for cases where images of several objects can be seen simultaneously. Kajiya's rendering equation [Kaj86] is an example of formulation for light interaction that can be used as a method to resolve complex transparent, translucent, reflective, refractive and shadowed phenomena.

Different geometrical descriptions can be included in Equation (2), even when those are based on non-traditional representations. The term L_l simply accounts for the final value of the radiance of object l independently of the nature and complexity of the phenomena. Something similar occurs with the geometrical term g_l. With this in mind, the original formulation can be used to render scenes where, for example, surfaces based on polygonal, analytical or implicit descriptions are mixed with objects built from liquids and gases.

Because of reasons we will discuss, the previous formulation is not always practical to calculate. In the following sections we will see how it can be adapted so it can be implemented as an algorithm suitable to be run in a computer.

4. Motion Blur Rendering Algorithms

Methods that gather subpixel information to produce spatial anti-aliasing will usually fail to find the subframe information on which temporal anti-aliasing relies. Also, methods that are not aware of the time dimension, when applied to each frame of a sequence, may produce images that lack any temporal coherence, and display a myriad of different artefacts.

Motion blur rendering algorithms are designed to handle the degree of complexity associated with the addition of the time dimension. However, even if they can produce temporally correct images, their computational complexity may be unacceptable. As observed in the photographic snapshots of Figure 1, the amount of fine detail is drastically reduced when the images integrate motion blur. When considered from the perspective of the Fourier theory [ETH*09], common visual phenomena suffer similar transformations under the presence of motion blur and their spectra is confined to a specific region of the domain. This supports the initial intuition that motion blur produces a reduction of the spatial complexity and frequency contents with respect to an equivalent instantaneous unblurred image.

Although the mathematical framework has been already described, different algorithms use alternative methods to evaluate the expressions associated. Knowledge of the physical phenomena may also be exploited. In all cases, it is desirable to reduce the impact of any simplifications, so it can be applied in a broader set of situations while being accurate. In the next sections, we will describe and group the existing methods based on an analysis of their assumptions, limitations and associated visual artefacts.

4.1. Overview

From the description in Equation (2), two different problems must be addressed. First, a good approximation for the evolution of the geometry needs to be found; and secondly, the shading of the objects needs to be accurately simulated. By considering the approaches and assumptions used to solve each of these problems, the following categories can be established. They are also summarized in Table 1.

- Analytic methods, although frequently relying on heavy assumptions, use a closed form solution that can be exactly evaluated. Section 4.2 covers them in detail.
- In Section 4.3, geometric substitution methods use alternative primitives that represent the original geometry and

Table 1: *Motion blur rendering methods classified according to the categories explained in Section 4.*

Category		Methods
Analytic		[KB83, Gra85]
Geometric substitution	Generic	[Ree83, Cat84, Gla88]
	Motion hints	[WZ96, MMI*98, TBI03, Gre03, GM04, JK05, SSBG10]
Texture clamping		[NRS82, Lov05]
Monte Carlo	Distributed ray tracing	[CPC84, HJW*08, ETH*09, ODR09]
	Evolutions	[VG97, CJ02, HDMS03, WABG06, NSL*07, RKLC*10]
	Accumulation buffer	[KB83, HA90]
	Frameless rendering	[BFMZ94, SZ97, DWL02, DWWL05]
	Specific to a primitive type	[Pre92, KK07, AMMH07]
Post-processing	Generic	[Sha64, PC83, ML85, CW93]
	Motion fields	[Shi93, BE01, SSC03, Zhe06, Ros07, Sou08, Vla08]
Hybrid		[CCC87, SPW02]
Mechanical and optical		[Sha64, Gla99, LC06, TSY*07, PLR09]

its evolution along time. These methods are ideal to be implemented in real time frameworks or when approximate motion hints need to be produced.

- Monte Carlo methods have received a great deal of attention due to the fact that a wide range of phenomena can be modelled inside a flexible and technically simple framework. However, stochastic point sampling methods are not deterministic and tend to produce artefacts due to insufficient sampling. These techniques and its associated methods are described in Section 4.5.
- Post-processing methods extend the information of an image snapshot to the whole exposure range. Temporal information is extracted directly from the render engine or by processing different frames of the sequence. A detailed description can be found in Section 4.6.
- A number of models have been combined together to solve specific aspects of the general problem. Section 4.7 contains a description of hybrid algorithms which have proved to be the best options when an open and unconstrained problem needs to be solved. This is probably the main reason for their wide acceptation in the visual effects and film industry.
- Finally, Section 4.8 contains recent techniques that include physically accurate models of the imaging device. The availability of CMOS/CCD chips in digital video cameras and the need for better integration of rendered images into real footage have made these solutions increasingly relevant.

4.2. Analytic methods

Analytic methods are among the pioneering solutions to render motion blur. They replace Equation (2) with closed form expressions that efficiently provide an exact value of the pixel's radiance. In general, lighting equations are highly non-linear functions that may not have analytical equivalents so this family of methods are only valid under strong compromises.

The first example of this approach is the work of Korein and Badler [KB83]. Their algorithm finds a continuous function that represents the time intervals when the projection of an object covers a given image plane pixel. Once those visibility ranges τ_l are found, the problem is reduced to determining a single shading sample in one of the directions of the pixel ω at a time t inside this temporal range. Equation (3) models this approach:

$$I_{xy} = \sum_l \int_{\Delta T} r(\omega,t) g_l(\omega,t) L_l(\omega,t)\, \mathrm{d}t = \sum_l \int_{\tau_l} r(\omega,t) L_l(\omega,t)\, \mathrm{d}t = \sum_l \tau_l L_l(\omega,t). \quad (3)$$

Because their implementation is limited to objects represented as spheres, polygons and polyhedra, and their trajectories are approximated using piecewise linear interpolation, the corresponding projections in motion are determined by simple primitives (Figure 2). Occluded objects are stripped out from the calculations using depth sorting and a single object is considered for each time range and pixel. This method focuses on solving geometric temporal aliasing and, because it uses a single sample L_l, changes in shading are beyond the capabilities of the algorithm.

An alternative method, adapted for polygonal primitives, is described by Grant [Gra85]. Dynamic three-dimensional (3D) polygons are converted to static 4D polyhedra that describe the volumes swept by the geometry. Visibility is determined by scan-converting those continuous 4D (*x,y,z,t*) primitives into continuous visible 3D (*x,y,t*) geometry using

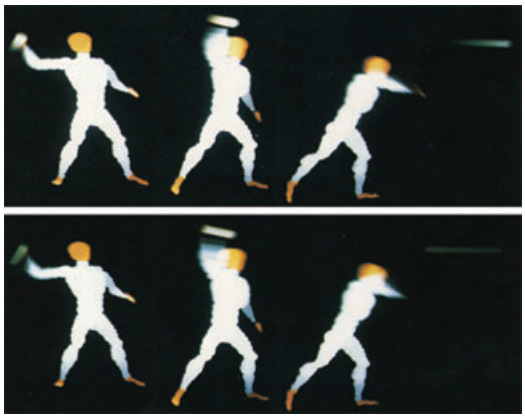

Figure 2: *Method of Korein and Badler [KB83]. Top row images use a temporal box filter, whereas the bottom row show the same motion using a filter that emphasizes later movement.*

a temporal extension of Catmull's visible surface algorithm [Cat84]. Non-occluded geometry is rendered using an extension of Feibush *et al*. analytical filtering algorithm [FLC80]. Intersecting and non-intersecting objects under image space translational and scaling motion can be represented with this approach.

Grant's polyhedral algorithm can also be included in the category of methods that use geometry substitution. Those methods are explained in the following section.

4.3. Geometric substitution

Methods that use geometric substitution are all based on the idea that, to render motion blur, each moving object can be replaced by static geometry built from the original primitives and their motion. Time-sampling is no longer needed, as the new geometry contains an implicit description of the temporal evolution of the shape, shading and trajectory. In general, these methods can be described by adapting the formulation from Equation (2) to

$$I_{xy} = \sum_{l} \int_{\Omega} \int_{\Delta T} r(\omega, t) g_l(\omega, t) L_l(\omega, t) \, \mathrm{d}t \, \mathrm{d}\omega$$
$$\approx \sum_{l'} \int_{\Omega} r_s(\omega) g'_{l'}(\omega) L'_{l'}(\omega) \, \mathrm{d}\omega. \qquad (4)$$

In Equation (4), each geometric object l is replaced by an alternative primitive l'. Functions r, g_l and L_l are also swapped with r_s, $g'_{l'}$ and $L'_{l'}$, their time independent counterparts. This formulation is just an approximation that drops the integration over time while keeping the corresponding image space evaluation over the domain Ω. Different algorithms make different interpretations of this expression and, as we will see, each approach is associated with a specific set of implications and varying flexibility.

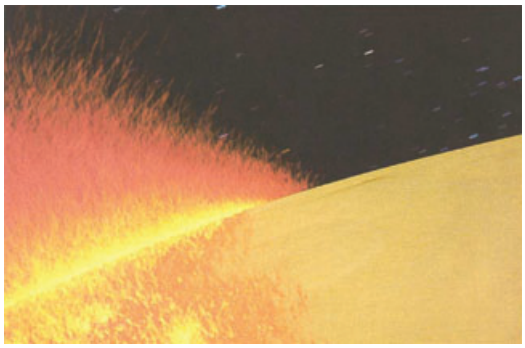

Figure 3: *Particle systems using the motion blur technique described in [Ree83].*

One of the earliest proposals is made by Reeves. He establishes the fundamentals of particle systems [Ree83]. Motion blur is achieved by replacing each moving particle by an anti-aliased line segment describing its trajectory during the exposure time. Their colour is based on a single-shading sample. Because they are rendered as point light sources, all particles projected to a given pixel can be simply added together without accounting for visibility, that is, $g'_{l'}$ is always one. The original algorithm is not applicable to particles that use different shapes or textured billboards and layer composition is needed to solve occlusion with other types of primitives. An image rendered with the original technique can be seen in Figure 3.

Catmull's visibility algorithm [Cat84] focuses on finding that polygons are visible from a given pixel. Each polygon is decomposed into a set of micro-polygons whose shading is weighted with a circularly symmetric filter [FLC80] and composited using depth sorting. Spatio-temporal filtering is achieved by using the original filter on geometry that has been compressed according to the speed of each of its vertices. Primitives whose vertices are moving at different rates, following different paths or non-linear screen space speeds can be accurately handled.

In the space–time raytracer [Gla88], scene primitives are represented as static entities in a 4D space. Because the 4D representations implicitly contain the time dimension, the scene only needs to be evaluated during the algorithm's initialization where the evolution of each object can be fully determined. Intersection tests are based on tracing rays in the sampled space–time direction. Because the method focuses on determining the $g'_{l'}$ term, the evolution of the shading function needs to be independently handled. As in the original ray-tracing algorithm, motion blur is calculated by shooting rays at different times.

4.3.1. *Motion hints*

A subset of the substitution methods, although not inspired by physically accurate models, are designed to produce visually appealing motion blur. They assume that, at high frame rates, observers cannot distinguish between correct blur and rough approximations.

As they simplify geometry, shading and visibility, they can be efficiently executed. Integrating these methods into existing pipelines is also simplified by the fact that original and replacement geometries use similar representations. These are the main reasons why these methods have been widely implemented inside real time rendering frameworks.

Wloka and Zeleznik [WZ96] use a representation of the volume described by the original moving geometry. Each object is divided into a leading and trailing polygons and a contour is built with the edges connecting them. An in-between surface is generated by sweeping this contour along a piecewise linear approximation of the trajectory of the object. Leading polygons are drawn fully opaque, while trailing and connecting parts use transparency determined by their motion vectors. This method cannot properly handle general rotation, object scaling or independent vertex deformation.

An extension to rendering deforming polygonal meshes and handling arbitrary rotation is presented by Jones [JK05]. A shell is built based on the vertices of the motion silhouette at different time snapshots. Age decaying opacity is used to render it. Because this method is based on extruding the silhouette, it cannot handle textured geometry or complex shading. Visibility computation is replaced by depth tests in the raster engine which may result in backward facing polygons being rendered and self-intersections between object and trail.

Programmable motion effects [SSBG10] build bilinear patches defined by the trajectory of a set of seed points placed on the surface of the object. Based on this 4D representation, the concept of surface shader is extended to a flexible post-process that can produce speed lines, stroboscopic effects, temporal offsetting as well as stylized and realistic motion blurring. Figure 4 shows a range of effects achieved with this technique.

The method of Green [Greb] uses multiple OpenGL passes. An initial pass renders a sharp image of the scene and a per pixel velocity field. These velocities are used to expand and render the original objects. The resulting image is blurred using multiple texture samples that are later applied to the replacement geometry.

A related approach is followed in [TBI03]. Moving geometry, in this case polygonal spheres, are replaced by capsules whose lengths are dependent on its image-space speed, with their orientations matching the direction of motion. As

Figure 4: *Different motion trail styles achieved with the technique of Schmid et al. [SSBG10].*

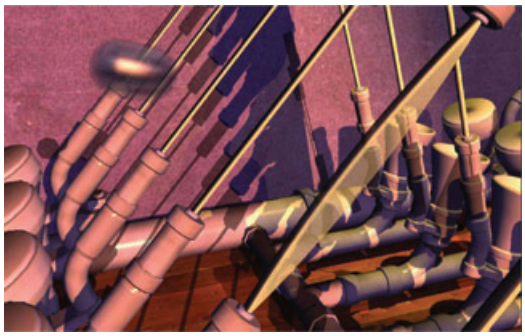

Figure 5: *Real time motion blur using the technique of Tatarchuk et al. [TBI03]. Specular highlights and environment reflections are spread according to the speed of ball. The original spherical shape is also elongated along the direction of motion.*

seen in Figure 5, the original shading is modified so that the energy of specular highlights is spread among the pixels they would move across during the exposure time. Object transparency and environment mipmaps are modulated based on the speed of the object. Each pixel is independently shaded with a single sample that accounts for the whole shutter range, with occlusion being handled by the hardware rasterizer.

Point-based rendering can also integrate motion blur by using the method of Guan and Mueller [GM04]. Instead of motion blurring all the voxels of the volume [MMI*98], temporal anti-aliasing is performed on an isosurface extracted using a simplified version of the EWA algorithm [ZPvBG01]. During render, a Gaussian temporal filter is assumed and an ellipsoid represents the projection of the original 2D round splat and its linear motion trail. Figure 6 shows different effects that can be achieved by modifying the sizes, colours and number of ellipsoids generated per point. This approach cannot resolve varying shading and situations where geometry is partially hidden, intersect or change its relative positions.

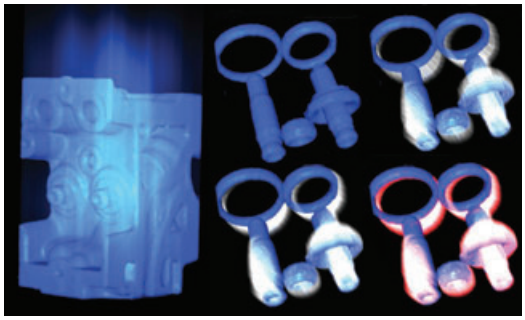

Figure 6: *Motion blur applied to point-based rendering [GM04]. Left image shows a volumetric model of an engine moving vertically with its corresponding motion blur. Images to the right show a different model rendered without motion blur (left top), motion as streak lines (right top) and two different artistic effects applied to the motion trails (bottom).*

Recently, the solution for time-varying point samples has been described [HWK*10]. Heinzle's *et al*. algorithm also assumes piecewise linear trajectories, but a continuous reconstruction of the spatio temporal functions is handled using 3D Gaussian kernels. Although Guan's method renders a static image that is composited with the corresponding motion trails, the new method incorporates both the render of the points and its trails into the sampling of the kernels.

4.4. Texture clamping

Aliasing artefacts can be thought of as a mismatch between the frequency contents of the signal and the rate selected to sample it. Lighting functions, occlussions and their evolution in space and time produce sharp changes and discontinuities, which boost the frequency contents of the original shading functions. Techniques like super-sampling handle these high frequencies by using extra samples (Section 4.5). A more efficient alternative consists in reducing the bandwidth of the original signal so less samples are needed for a reconstruction that is free of aliasing.

Textured objects can be anti-aliased with the method of Norton et al. [NRS82]. Assuming perspective projections are linear transforms, the shape of an image-space box filter can be approximated by an object-space parallelogram. With this in mind, a texture can be convolved, in the Fourier and spatial domains, with a low-pass filter whose support is defined by the texels covered by the projection of each image pixel. This formulation extends to the temporal domain by taking into account the evolution of this projection. More recently, Loviscach uses anisotropic texture filtering to perform a similar procedure [Lov05]. When a polygon in motion is rasterized, each of the resulting pixels define a parallelogram in texture space. By carefully selecting the size and orientation of the filter, texture reads can return values that incorporate motion blur.

It should be noted that none of these methods generate motion blur *per se*, but focus on alleviating the problems associated with temporal and spatial aliasing. Even if they can efficiently solve problems for textured geometry, shading or lighting functions can themselves produce aliasing. Filtering after shading may reduce these problems, however directionally dependent effects may need to be frequently re-evaluated to accommodate for scene updates. Also, occlusions are not considered, so situations where moving polygons overlap need to be solved with a complementary algorithm.

4.5. Monte Carlo methods

In those cases where no analytical expression or an alternative geometrical description can be used, Equation (2) can be approximated using numerical methods. Given the multidimensional nature and unpredictability of the integrands, standard methods relying on the smoothness of the functions cannot always be used. However, Monte Carlo methods are designed to handle high rates of change and discontinuities. Instead of regular artefacts that are usually easily noticeable, errors due to low levels of sampling are shown as random patterns and noise.

In the following section, we will describe different stochastic algorithms grouped under the generic name of Monte Carlo methods. Because our focus is the production of motion blurred images, we assume a basic knowledge of the fundamentals of numerical methods in computer graphics. Interested readers are referred to the excellent descriptions that can be found, among others, in [Laf96, Vea98] or [WJAD*03].

4.5.1. *Distributed ray tracing*

As seen in Section 3, a visual phenomenon is parametrized in a highly multidimensional space. Traditional raytracers [Whi80] focus on producing image space anti-aliased pictures. For phenomena such as soft shadows, translucency, glossy reflections and so on; any additional dimensions need to be explicitly and independently sampled [CT82, Whi80, Wil78]. This is cumbersome and frequently results in an excessive number of samples being calculated. With Cook *et al*. distribution ray tracing [CPC84], each ray is stochastically allocated so all dimensions are simultaneously sampled. Motion blur is solved by sampling the time domain and as seen in Figure 7, different effects can be modelled: glossy reflections, translucency, penumbras and depth of field are rendered by evaluating the specular distribution function, the directions of the transmitted rays, the solid angle of light sources or the camera lens area. Because of the elegance and simplicity of the method, distributed ray tracing has received extensive

Figure 7: *Image rendered using distribution ray tracing [CPC84]. This technique can simulate high-quality motion blur including shadows, penumbras and specular reflections where the movement is non-linear during the exposure time. © 1984 Thomas Porter, Pixar.*

attention and it has become one of the most popular approaches in industry and academia.

$$I_{xy} \approx \frac{1}{N_j} \sum_{j}^{N_j} i_d(\omega_j, t_j), \tag{5}$$

$$i_d(\omega_j, t_j) = \sum_{l} r(\omega_j, t_j) g_l(\omega_j, t_j) L_l(\omega_j, t_j). \tag{6}$$

In general, distribution ray tracing approximates Equation (2) as in Equation (5). Each pixel is calculated as a sum of N_j discrete point samples. For simplicity, we will consider that motion blur is based on sampling locations in the spatio-temporal domain only. A sample $i_d(\omega_j, t_j)$ accounts for the contribution of each object l as seen in the direction ω_j at an instantaneous time t_j. This value can be calculated by different means, but in general it will be the result of evaluating the visibility g_l and radiance L_l functions for each of them. These contributions are weighted with the value determined by the filter $r(\omega_j, t_j)$. The resulting samples can be simply added together and averaged, or as we will see, more sophisticated weighting methods can be applied.

Increased dimensionality accentuates the problems of point sampling. Because distribution ray tracing is not tied to a specific sampling technique, the problem of how and where samples are generated has received great attention. Simple methods such as uniform and regular adaptative sampling [Cro81, Whi80] while successfully used by standard ray tracing produce poor results with distributed ray tracing. Stochastic methods like minimum distance Poisson or jittered

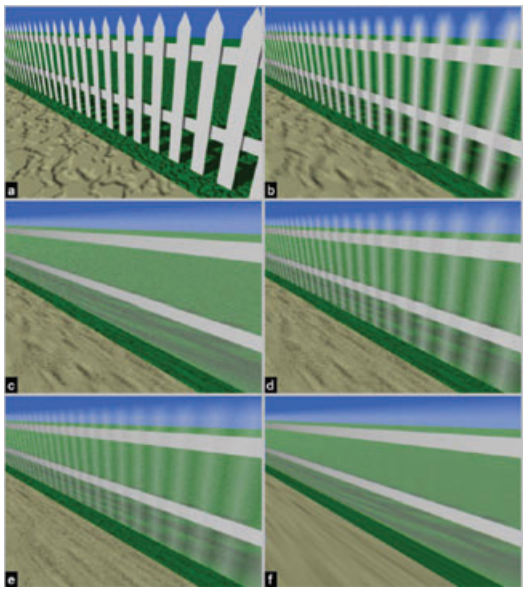

Figure 8: *Scene rendered with a static camera (a). Images (b)–(e) use a camera that moves progressively faster. A box reconstruction filter creates aliasing patterns in images (d) and (e). The patterns are corrected in image (f) by using a Gaussian filter. Reproduced from [DK00].*

sampling [Coo86, DW85], pre-computed sampling patterns [Coo86], adaptative sampling [Kaj86, LRU85, Mit87], importance sampling [DBB06] or stratified sampling [Mit96] are designed to reduce those problems. As seen in Figure 8, alternative reconstruction filters can combine a given sample set resulting in varying qualities [MN88, DK00]. The topic of sampling and reconstruction has been discussed in detail in [PH04, DBB06].

Methods which produce estimates as seen in image space may produce noisy reconstructions and miss image features. Alternative methods, such as Metropolis light transport [VG97] or the recent Hachisuka's *et al*. [HJW*08] multidimensional adaptive sampling, place samples based on the changes of the functions in the original multidimensional space. The latter evaluates the contrast of the rendering equations to determine the sampling level and analytically reconstructs the original function in all but the image dimensions. A Riemann sum is used to integrate the illumination of each pixel.

Several studies have considered the rendering problem from the perspective of the frequency domain [DW85, CTCS00, DHS*05]. Recent results [ETH*09] have shown new methods to design sampling and reconstruction filters. In essence, for common effects such as object and texture motion, rotations of both the BRDF and lighting and non-static shadows, the influence of motion blur can be approximated

by a shear filter. The frequency contents and the sampling level required can be estimated for each image pixel based on their respective image space velocities. An improved reconstruction can also be obtained by tuning the orientation and extent of the filter based on this data.

In contrast to previous methods that exhaustively sample the integral to reconstruct a smooth image, adaptive wavelet rendering [ODR09] directly estimates a smooth function in the wavelet domain. With this multiscale representation, coarse-scale wavelets represent large smooth regions whereas finer-scale ones represent edges and small details. Adaptive sampling can efficiently detect sharp changes in image-space as well as smooth image regions with high variance in the non-image-space dimensions. The algorithm reconstructs smooth images even in regions of high variance, scales well for high-dimensional problems but it may show ringing and over smoothing artefacts.

4.5.2. *Evolutions of naive distribution ray tracing*

The original distribution ray-tracing method has been modified to produce results more efficiently with increased accuracy. In this section, we will focus on exploring those techniques.

Metropolis light transport [VG97] is an evolution of the original bidirectional path tracing [LW93, VG94]. Paths are built from the light sources to the eye, and different mutation strategies are used to generate variations of these. This method naturally extends to calculate motion blur when the mutations include the temporal dimension.

Caching techniques, such as Jensen's photon mapping, increase performance by storing and reusing lighting results [Jen96]. However, they produce estimates that are specific to a given value in the time domain so they cannot represent objects with changing shading, motion or shape. An extension of this technique [CJ02] handles motion blur by assigning a random time to each photon and its descendants in the light path. The final reconstruction pass relies on a filter that averages the photons of the spatio-temporal neighbourhood of the pixel being calculated. In Figure 9, motion blur is incorporated to a set of shadowing, reflective, refractive and caustic effects.

The original Lightcuts algorithm [WFA*05] has also been expanded to support temporal sampling [WABG06]. The method builds on the original discretization of the lighting integrals and graph partitioning. However, the temporal domain is represented with a fixed set of time instants and any light interactions are limited to those subintervals. As seen in Figure 10, this method can provide high-quality results while drastically reducing the cost and the noise produced.

Other methods exploit spatio-temporal coherence [HDMS03]. Because a given area of an object can usually

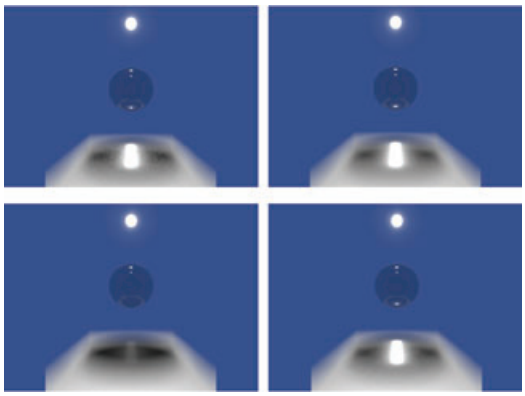

Figure 9: *A transparent sphere projects a caustic on a diffuse cube that moves downwards. Left-top: Noisy image rendered using path tracing with 10 000 random paths per pixel. Right-top: Accumulation buffer averaging 20 images placed at equal intervals. Bottom-left: Photon mapping using standard estimation that underestimates the final value of the radiance. Bottom-right: Photon mapping using the time dependent radiance estimation from [CJ02].*

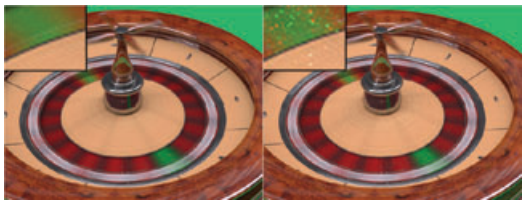

Figure 10: *Comparison of two renders performed using multidimensional lightcuts (left) and traditional Metropolis light transport (right). Lightcuts' image is less noisy and can be rendered faster (15×). Reproduced from [WABG06].*

be seen within several consecutive frames, novel views can be built by reprojecting existing samples. The algorithm of Cabral and Leedom can be used to account for the image space projection [CL93]. Motion blur is computed by determining the trajectory of each sample and accumulating their energy contribution over the motion path. Like similar methods [BE01], it uses linear constant speed motions, but in contrast, precise trajectories are computed from object space and not image space data. Although reprojection clamps the shading functions, the technique is only available for non-deformable objects, cannot directly model view-dependent effects and due to the simplified nature of the paths, may produce occlusion artefacts.

Using a similar approach, real-time reprojection caching [NSL*07] allows supersampling to be applied at high frame rates. In this case, cached values are reused using hardware texture filtering. Noise is reduced while keeping storage

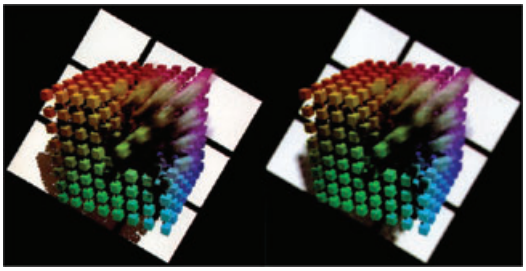

Figure 11: *Images rendered with the original accumulation buffer method [HA90]. Left image contains a motion blurred image composed from 23 different passes. Right image uses 66 individual images to simultaneously render motion blur, depth of field and soft shadows.*

Figure 12: *Images generated using frameless rendering. Left, naive frameless rendering where each sample is displayed as soon as is calculated. Right, the same image using adaptive frameless rendering [DWWL05].*

requirements low by using a recursive low-pass filter that averages new samples with the existing ones.

Decoupled shading [RKLC*10] leverages the intrinsic coherence of depth and motion blurred scenes by using a shading cache. The technique densely evaluates the visibility function and reuses any previously calculated shading samples. The defining parameters of each visibility location are hashed so they deterministically map to a shading value. On a cache miss, the value is calculated on demand and stored so they can be later retrieved. This method is built on the assumption that shading does not change rapidly and is view independent, so a single value is representative of the whole shutter range. Shading rates are reduced respect to other methods that shade before visibility or those in which visibility and shading samples have a one-to-one relation.

It should be noted that exploiting spatio-temporal coherence is not only applied to calculating motion blur. Other methods have relied on similar approaches for the rendering of sequences of images where the interest is obtaining high frame rates and not temporal supersampling [WDP99, TPWG02, SKAB03, SS00].

4.5.3. *The accumulation buffer*

A number of methods have been designed to super-sample multidimensional spaces using accumulation buffers. These hardware framebuffers are suitable to efficiently weight and combine samples in image space using high precision computations.

Korein and Badler [KB83] include the description of an algorithm that can be considered a precursor to the accumulation buffer method. It calculates temporal anti-aliasing by supersampling the temporal domain using several independent renders performed at different times. Full images are stored externally and filtered down to produce the resulting image.

Haeberli and Akeley [HA90] weight and add several passes computed using a single sample per pixel. By updating different parameters between passes, the resulting image is spatially anti-aliased, integrates depth of field, soft shadows, anisotropic reflection or filtered texture maps. As seen in Figure 11, motion blurred images are the result of combining scene renders at time instants are that equally placed in the shutter range. A final image is the result of N subframe passes or, in the case of repeated integration [Hec86], every two passes.

Until the appearance of flexible programable GPUs, this was the preferred method to leverage the parallelism available in hardware frameworks. Because the solution comes by iteratively applying the original algorithm, the implementation is simple and avoids the overhead of elaborated control logics [SI05]. However, this method shows poor results with scenes with high temporal correlation and does not scale well with scene complexity.

4.5.4. *Frameless rendering*

Frameless rendering [BFMZ94] is a paradigm designed to reduce the latency of interactive display systems. Unlike traditional approaches that expose frames after they have been fully computed, frameless rendering derives each pixel from the most recent instant in time and immediately shows it. As a consequence, high motion scenes show an effect that resembles motion blur, whereas low motion ones converge to the same solution as a traditional renderer. Figure 12 shows an example of the output generated by this technique.

The original technique randomly selects the samples to be rendered. Further evolutions use a probability distribution function based on the age of the pixels [DWL02] or even closed loop adaptative sampling to steer computation to regions undergoing significant changes [DWWL05]. Related techniques use one temporal sample per scanline [OCMB95], dynamically modify frame rate [WLWD03] or adapt frame rates for each individual object [RP94]. Perceptually based heuristics have also been envisaged [Zag96, SZ97]. Image plane pixels can be reconstructed from a sin-

gle sample [BFMZ94], a temporally weighted set of samples corresponding to a slice of time or even from a 3D volume in which the filter is adapted to local sampling density and colour gradients [DWWL05].

Because computation speed is bounded, these methods trade temporal supersampling with spatial undersampling. Scenes with high spatio-temporal coherence are well suited for this schema. Otherwise quick changes are shown as a dissolve filter between scene snapshots. As a side effect, traditional strobing effects associated with temporal low sampling, are replaced by less perceptually disturbing noise.

4.5.5. *Methods designed for specific types of primitives*

Distribution ray tracing can handle a wide range of primitive types. This section covers the methods that have focused on using specific geometric representations to obtain optimizations.

Preston implements motion blurred distribution tracing for NURBS surfaces [Pre92]. His algorithm expands the original parametric representation so it represents the surface and its temporal evolution. The surface's control points are stored as slices representing different moments in time and any surface configuration is constructed as an in-between of two of them. Motion blur is achieved by sampling the resulting surface at different time lapses.

Fluids, based on non-polygonal surfaces, can be rendered with Eulerian motion blur [KK07]. Because Newtonian inertia is not a valid assumption, the status of the system cannot be inferred using interpolation between known states. Kim's approach is based on defining intermediate states based on sparse level-sets, density data and the fluid's semi-Lagrangian advection. From that point, existing Monte Carlo methods can be used to render the fluid [FSJ01, NFJ02]. Figure 13 shows the result of the algorithm.

Rendering motion blur for voxel sets, rigid and time-varying point clouds has already been described in Section 4.3.1 Guan and Mueller briefly introduce an alternative approach [GM04] using optical flow or mpeg-like motion estimation to determine motion flows inside the evolving voxel grids. However this method has not yet been implemented neither described in detail.

Akenine-Möller *et al*. implement real time rasterization of triangles [AMMH07] whose motion is described as a continuous function of their vertices at two moments in time. Using this representation, any surface attribute can be interpolated using GPU tiled rasterization, accumulation buffers and piecewise linear approximation of the trajectories. Motion blur, soft shadows and depth of field can be rendered using this framework. Inspired by the shadow mapping algorithm [Wil78, RSC87], this method can render motion blurred shadows without relying on ray tracing. Unlike the original algorithms or deep shadow maps [LV00], see Figure 14, time-dependent textures are not limited to static scenes or static shadow receivers.

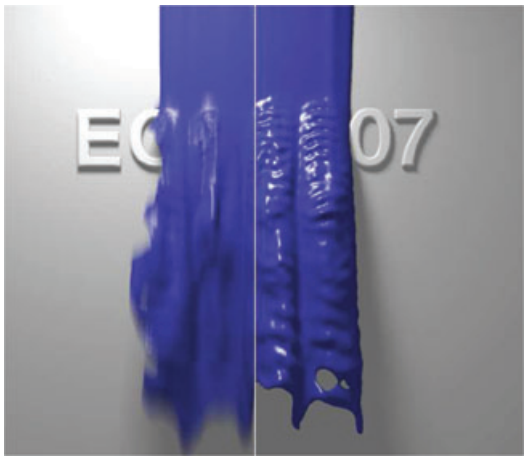

Figure 13: *Images comparing an Eulerian system rendered with and without motion blur. Left image has been rendered using the method of Kim and Ko [KK07].*

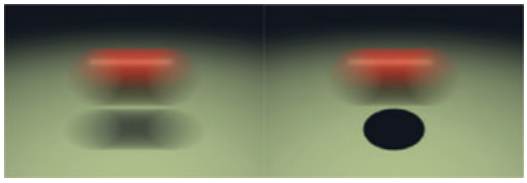

Figure 14: *Left image shows a motion blurred shadow rendered using deep shadow maps [LV00]. This technique can only be used when the object receiving the shadow is stationary respect to the light emitter. Right image shows the same scene with static shadows. © Pixar.*

4.6. Post-processing

Post-processing methods use one or several pre-rendered snapshots of the scene and blur them using motion information built from the images themselves or the objects' animation data. With this approach, motion blurring and rendering methods are fully decoupled. As it operates in image space, it is also independent of scene complexity. Although this improves the efficiency, it is also associated with quality compromises.

In general, these methods can be modelled as in Equation (7). A motion blurred image is calculated by convolving each pixel $I(x, y, t_i)$ of an instantaneous image at time t_i, with a function $\Phi(t_0, t_\infty, t_i)$. This function is built based on the information available in the scene or its rendered images. Theoretically, it uses all image data available between t_0 and t_∞, but most methods will reduce the scope of the analysis.

In general, the details of the algorithm used to calculate the unblurred image are not needed, but in some cases it may provide guidance to setting up the convolution kernel.

$$I_{xy} \approx I(x, y, t_i) \otimes \Phi(t_0, t_\infty, t_i). \quad (7)$$

The first of the methods is described by Potmesil and Chakravarty [PC83] and extends their previous formulation of an aperture camera model [PC81]. In their work, Whitted's pinhole raytracer [Whi80] is used to render static images of the world. The image formation process is modelled as a series of image degradation transformations represented by point spread functions (PSF). PSFs can be analytically calculated for shutters of different types [Sha64] and are specific of each object's relative motion. In general, post-processing do not properly handle effects that are baked into the image, but this method can blur reflections and refractions by explicitly convolving each object's rendering samples with the PSF. Figure 15 shows an image rendered with this technique.

A similar model is proposed by Max and Lerner [ML85]. Objects having similar motion are skewed in the direction of maximum movement according to the intensity of the motion. Incremental summation is used to add blur. Blurred images are unskewed using the inverse of the original transformation before they are alpha composited. In this method, Potmesil's PSF is replaced by an equivalent blurring process, with the rest of the algorithm remaining similar. As a result, both methods have the same limitations.

In a similar way to the re-projection methods explained in Section 4.5.2, morphing methods exploit frame to frame coherence [CW93]. Sample motion is described using precomputed morph maps that are constructed from the image pixels, their depth information and the camera transform. Although the main focus of the technique is efficiently generating scene walkthroughs, images can incorporate motion blur by generating several temporal samples per rendered pixel. This method is not specific to CG imagery and can also be applied to any images given range data is available [Che95].

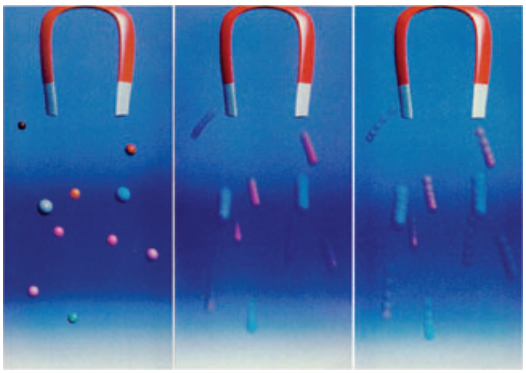

Figure 15: *Method of Potmesil [PC83]. Left to right: Instantaneous snapshot, using extended exposure and simulating several exposures of finite length.*

Figure 16: *Motion blur using the technique of Browstow and Essa [BE01]. Images at the top are two instantaneous pictures taken from the original scene. Bottom image shows the resulting motion blur applied to the propeller.*

4.6.1. *Methods based on motion fields*

An alternative approach is fundamented on the synthesis vectorial fields containing the direction and speed of each image pixel. A motion field represents a snapshot of the dynamic status of the system and it is used as a replacement for the real movement during exposure time. Although object motion can be intricate, these methods are a compromise. Some of them are capable of efficiently rendering production quality motion blur.

Shinya [Shi93] calculates spatial and temporal antialiasing by applying an image transformation built on the motion flow of a sequence of images. The pixel-tracing filter is adapted to gather information from sequences of different lengths, with linear computation times. Samples from each image are weighted and combined together so a trail of the object is produced.

The method proposed by Brostow and Essa [BE01] can be used when no information from the original scene is available. This is the case for stop motion animated sequences where each image is a photograph. Initially, foreground and background pixels are separated using a motion detection

procedure. The moving foreground is then segmented into contiguous pixel blobs that are matched between consecutive frames using an exhaustive search. An initial estimate of the velocity field is locally refined using hierarchical optical flow [BA96] and the resulting velocity is used to smear each pixel assuming constant speed and linear trajectories during the simulated exposure. The results can be observed in Figure 16.

Previous methods provide smooth velocity fields that will generate incorrect results when, for example, overlapping objects with different speeds are projected to the same screen area. The results are also greatly determined by the quality of the input. Bad lighting, camera shaking, the existence of shadows or the capture process itself can compromise the results.

Zheng *et al*. propose a hybrid approach that combines motion from the scene with pure image-based optical flow [Zhe06]. An incomplete motion map can be easily determined at render time from the scene database. These motion vectors are independent of the illumination and can be incorporated as constraints or landmarks for the minimization step of the optical flow computation [HS80]. In Figure 17, we can see the resulting field, that smoothly blends exact values and image-based estimations where needed. Because of the fundamental nature of the algorithm, it does not completely solve the problems associated with occlusions and pattern blurring in areas containing shadows and reflections.

Shimizu *et al*. present a method that produces motion blurred polygonal geometry in real time [SSC03]. It efficiently computes an approximation to the optic flow of the scene by using the motion vectors of the model's vertices. As seen in Figure 18, they are used as displacement offsets to produce lagging and leading trails of the original geometry that are rendered using different degrees of transparency. This method implicitly implements a line integral convolution [CL93].

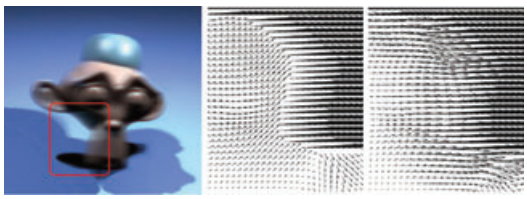

Figure 17: *Motion blur calculated with the method of Zheng et al. [Zhe06]. Centre image shows a motion field calculated using optical flow computation and, where available, information from the renderer has replaced the original field. Right image shows a better field that results from using renderer data as landmarks for the optical flow computation.*

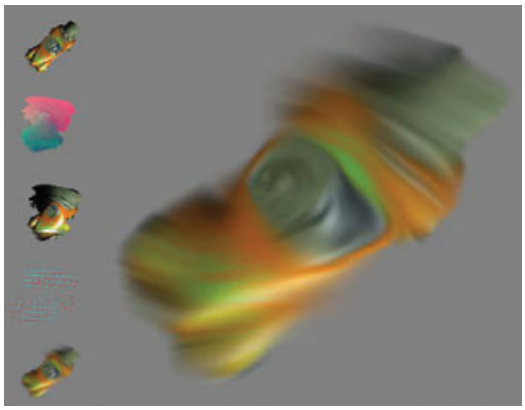

Figure 18: *Real time motion blurred car using the method of Shimizu et al. [SSC03]. Insets show, from top to bottom, a render of the original geometry, motion vectors, warped geometry, optic flow vector field and final rendered geometry.*

Game engines have also benefited from post-processing algorithms. A recent example is described by Sousa [Sou08]. His method uses a velocity field that is calculated in an independent pass using the current and previous camera transforms. The original images are progressively blurred, while keeping near objects sharp, using two alternating ping-pong buffers. Deformable geometry can be handled by taking into account the motion of the skeleton bones. Although it avoids leaking artefacts by using an object-ID buffer, transparency and alpha compositing cannot be properly handled. Similar techniques, limited to camera motion and rigid objects [Ros07] or exclusively camera rotation and translation [Vla08] are also used in other next-gen games.

4.7. Hybrid methods

Hybrid methods combine the strengths of individual algorithms to resolve specific aspects of the general motion blur problem. This approach results in algorithms that, relying on mild assumptions, can efficiently be executed in a broad set of scenarios. Their flexibility and the high quality of results have made them part of the selective group that have been accepted as part of the graphic pipelines of the most demanding production environments. Because of their importance, we will dedicate the following section to explain them in detail.

4.7.1. *The REYES architecture*

The first hybrid technique is implemented as part of the REYES rendering architecture [CCC87]. This framework is designed to render geometrically heavy scenes and efficiently implements stochastic point sampling for complex shading functions. In this environment, shading functions

are evaluated before visibility is considered, and once non-occluded samples are determined, their values are filtered down according to their spatial relationship with respect to the camera.

Primitives, independently of their type, are subdivided into micropolygons. Considering their screen space dimensions, shading samples $L_l(\omega_{uv}, t_j)$ are calculated and stored in the object's *uv* space. In this expression, ω_{uv} represents the direction that corresponds to each *uv* location. The time dimension is discretized using constant steps, and each of the previous samples are also evaluated at those t_i locations.

Stochastic sampling is used to reconstruct the radiance of each image pixel. As such, Equation (5) is evaluated at a number of ω_j and t_j spatio-temporal locations. The value of each $i_d(\omega_j, t_j)$ estimate can be determined using Equation (8).

$$i_d(\omega_j, t_j) = \sum_l r(\omega_j, t_j) g_l(\omega_j, t_j) L'_l(\omega_j, t_j). \quad (8)$$

Shading samples L'_l are generated using a chain of linear interpolations on cached values. Each ω_j is mapped to the corresponding (u, v) position in parametric space and its neighbourhood is used to interpolate new values at arbitrary locations. An estimate for time t_j can be interpolated from the corresponding values at the endpoints of the discretized interval $[t_i, t_{i+1}]$. The reconstruction filter $r(\omega_j, t_j)$ accounts for partial occlusion due to transparent and overlapping surfaces and reduced coverage due to the motion of the objects.

Note that even if the shading function is spatially and temporally clamped, visibility changes will result in high-frequency contents and discontinuities. In cases with extreme variations, the image may suffer from noise that can be alleviated by different means [EMP*03, AG00].

As seen in Figure 19, the shading functions can integrate complex phenomena which can be described using shade trees, procedural texturing, image-based texture mapping, analytical models or Monte Carlo ray tracing [AG00].

Figure 19: *Image rendered using the REYES architecture [CCC87]. This architecture can render high-quality motion blur. In this image temporal anti-aliasing has been applied to geometry, shadows, reflections and lighting. © 1989 Thomas Porter, Pixar.*

A variation of this method may be used in those situations where a full solution cannot be applied. A non-motion blurred render of the scene is calculated using high-quality shading and lighting. A final motion blurred pass is calculated after replacing the original object materials by camera projecting the initial image. This is clearly a compromise that limits the shading information to a snapshot of the areas of the object that are visible in the initial image, but has revealed itself as a powerful tool when the evolution of both shading and geometry is bounded.

4.7.2. *Method by Sung, Pearce and Wang*

The visibility function can be fully determined in the early steps of the render by analysing the geometry and motion of the scene. On the other hand, the evolution of the shading functions is difficult to predict and cannot be fully solved in a pre-processing step. Based on those premises, the method of Sung *et al*. [SPW02] uses an efficient analytical solution for visibility and a more flexible stochastic method for subsequent shading computations.

Visibility is determined with an approach inspired by the methods of Catmull [Cat78] and Korein and Badler [KB83]. These algorithms are used, respectively, to find the areas of the pixel and the ranges of time where visibility remains constant. With the first, pixels and polygons are subdivided until a single micro-polygon can be seen through each ω_j image space subregion. A modified version of the second algorithm generates contiguous time segments τ_l^j in which the visibility of an object remains unchanged. Adaptative supersampling using Mitchell's contrast [Mit87] may be triggered to spatially refine each subpixel estimate.

Shading computation relies on stochastic sampling to approximate the radiance of each pixel. The radiance of a polygon, $L_l(\omega_j, t_l^j)$, is assumed to remain unchanged for the duration of each of its constant-visibility time ranges. Adaptative supersampling can further refine the estimates of the shading function and those regions that are subdivided will reuse the visibility of neighbouring areas. According to the authors, this is a safe assumption as visibility supersampling has already reduced the differences in the estimations between close samples.

The final pixel value is determined, as in Equation (9), by adding the contribution of the N_j sampling areas that result from both refinement steps. Each shading sample $L_l(\omega_j, t_l^j)$ is weighted with a factor $F(\tau_l^j)$ that accounts for both the stochastic probability function and the integrated value of the temporal filter along the whole τ_l^j interval. The contribution of any visible objects in a given sampling area is also

weighted by $H(\omega_j)$, a value that represents the integrated spatial filter.

$$I_{xy} \approx \sum_{j}^{N_j} H(\omega_j) \sum_{l} \sum_{\tau_l^j} F(\tau_l^j) L_l(\omega_j, t_l^j). \qquad (9)$$

Because of the adaptative nature of the algorithm this method is specially suited for large on-screen motion trails and sharp changes in illumination. The analytical results of visibility computation are used to more efficiently direct the stochastic steps which contributes to a reduction in the amount of noise from traditional Monte Carlo methods.

4.8. Mechanical, optical and sensory inspired models

How the optics of the camera, shutter geometry and motion, film and sensor influence the final image? How the images captured by a real camera diverge from ones produced with the previous models? In the following section, we will address these points and describe different approaches capable of simulating the internals, limitations and deficiencies of the device.

The geometrical and optical characteristics of cameras can be modelled using different approaches [BHK*03]. In most of the cases, pin-hole or thick lens models provide enough accuracy. However, they are an oversimplification when more realistic depth of field, zoom, flares due to internal reflections or optical aberrations are required.

As described in Section 2, the aperture controls the amount and structure of the light arriving at the image plane and determines the amount of static defocus. Among the existing methods [BK08], distributed ray tracing [CPC84] and post-production filters [Bri99] are the most widely used to produce such effects.

If we exclusively account for motion blur, the camera shutter is the main element of interest. The vast majority of the algorithms in Section 4 are based on a shutter that moves infinitely fast. As a result, the reconstruction filter $r(\omega, t)$ is constant and the motion blur is homogeneous. In general, a mechanical shutter transitions in a finite amount of time, producing uneven exposure. In the context of computer graphics, the influence of shutter shape and its evolution has received little attention [Sha64, Gla99]. To our knowledge, only optical design software has implemented accurate models.

Interestingly, shutter simulation has recently received extra attention with the widespread popularity of video cameras and other digital recording devices. Even if professional systems may include a mechanical shutter, consumer devices simulate or replace it with electronic components. This produces a different family of artefacts. As an example, CCD cameras provide global shutters that expose the whole image simultaneously and the resulting motion blur is similar to the one produced by a uniform shutter. On the other hand, CMOS sensors process images using a rolling shutter with an exposure time that is directly proportional to the frame rate. Motion blur is coherent for the pixels of each scanline but shows temporal discontinuities between consecutive lines [MGS05]. Skew, wobble and partial exposure artefacts are typical of those systems [Grea]. Fields such as robotics or artificial vision have developed models to compensate and simulate those effects [PLR09].

Another aspect that is frequently overlooked is how the incoming light affects the imaging media. Radiometry clearly states that the relationship between the calculated scene radiance L and the irradiance at the image plane E is non-linear [FP03]. Using box or Gaussian kernels to downsample radiance is an approach that deviates from the real phenomena. Because film and sensor response is highly non-linear and produces noise [TZ69, Kod05], recorded image intensity cannot be assumed to be proportional to the radiance in the image plane. Different algorithms try to mimic more accurate behaviours using high-dynamic range imaging and tonemapping techniques [RD06]. Just a few algorithms have tried to produce photometrically correct motion blur. Lin and Chang describe a method that models the response of the camera based on the capacitor charging process [LC06]. The algorithm is tuned with a calibration step and its response can be adapted according to the F-stop of the camera. The usual uniform point spread function is replaced with a more realistic filter that enhances accuracy. Figure 20 shows the differences between traditional motion blur and the new technique.

Motion blur may also be seen as an effect that needs to be corrected. Pictures taken with long exposure times may

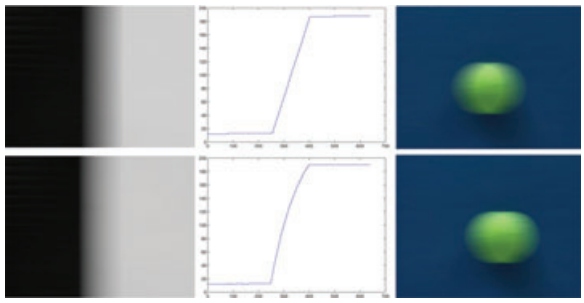

Figure 20: *Photo-consistent motion blur. First and second columns show the image of an ideal edge in motion and the corresponding simulated intensity profile. Third column shows a tennis ball rendered using two different methods. All images in the first row assume an uniform PSF. Second row uses the model of [LC06]. Note the differences from the traditional method and the non linear asymmetric shape of the function.*

contain unexpected motions that can compromise the spatial resolution of the captured image. Artificial motion blur can be recreated by integrating motion compensated snapshots [TSY*07].

5. Summary and Discussion

In previous sections, we have provided a detailed description of most of the algorithms that can integrate motion blur in computer generated images. In this section we compare them, identify their computational requirements and focus on identifying which methods are applicable to each specific situation.

From an operational point of view, an important element to consider is the type of inputs that the algorithm can process. Some methods are specific to certain geometrical representations whereas others are capable of accepting a wider set of primitives. In the first category, the most popular representation is polygonal geometry [Gra85, Cat84, CCC87, SPW02], that may be processed using hardware acceleration both for primitives [WZ96, TBI03, Gre03, JK05] and textures [AMMH07, Lov05]. Other methods can handle NURBS [Pre92], spheres [KB83], particles [Ree83] or voxels [GM04]. Non-Eulerian fields, such as liquids and gases, can be converted to polygonal representations but they have also been considered without previous transformation [KK07]. Monte Carlo methods, Section 4.5, can accept any geometrical description given that it can be point sampled. Post-processing algorithms are fairly independent from the type and complexity geometry of the scene. Motion information can be extracted directly from the images using *ad hoc* heuristics [Shi93] and optical flow methods [BE01]. In some cases, motion data from the scene is used [SSC03], whereas in other situations rendered velocity, object-id or depth passes [Zhe06, Sou08] may be used. An in-between approach is used by re-projection algorithms [CW93], where the original geometry of the scene is needed, but only as a mean to finding novel positions for pre-calculated shading samples.

An important criteria that helps determining the respective merits of each method is the degree of quality of the results. Do we want to produce motion blur that is physically based, photorrealistic or just a cue to the evolution of the scene? In one extreme, geometric replacement use gross shading and geometrical approximations but is capable of producing high frame rates. Common simplifications are based on roughly approximating the original motion [WZ96, Gre03, JK05], non-physically based energy distribution [Ree83, TBI03], artistically driven motion trails [GM04, SSBG10] or even alpha transparency as a replacement for the light integration process [WZ96, JK05]. A number of methods avoid the computation of all but local lighting events [NRS82, KB83, Gra85, WZ96]. With those methods motion blurred shadows, reflections, transparency or refraction are difficult to achieve.

A special case is represented by post-processing and re-projection methods. They both base their computations on pre-rendered samples whose radiance is the result of completely simulating a given lighting model. View-dependent phenomena have to be explicitly handled by, for example, storing intermediate stages of the render.

A small group of algorithms accurately model dynamic geometry and shading. They are frequently based on intensive point sampling which makes them the most computationally heavy approaches [CPC84, SPW02, CJ02, VG97, WABG06, HJW*08]. Recent developments have improved this situation [ETH*09, ODR09, RKLC*10]. Extended descriptions and a discussion of their respective sampling approaches have already been exposed in Sections 4.5.1, 4.5.2 and 4.7. These can be combined with the methods described in Section 4.8 so that the influence of the recording device is accurately modelled.

If we consider the complexity of the motion, most algorithms approximate intra-frame object motion with piecewise linear paths. These paths may be defined at object or vertex level which allows rigid object motion [ML85, WZ96, GM04] and deforming polygonal meshes [JK05], respectively. Most methods inter-polate motion using a single straight segment built from one or two temporal samples. Some algorithms assume the original motion has to be linear in world space [Cat84], while other will use alternative coordinate systems to enforce more restrictive motions [KB83, MMI*98, BE01]. To our knowledge there is no published algorithm capable of using the original path or equivalent non linear approximations. The nature of the motions and the shutter times involved in the computation (a maximum span of 33 ms at 30 frames/s) makes assumptions like path linearity and constant velocities safe for an ample range of situations. The most flexible extreme is represented by distributed ray tracing [CPC84] and its Monte Carlo based descendants that can generally evaluate both true shading and intricate motion at any point of the problem domain.

Different methods use alternative assumptions on the number of samples that are representative of the whole aperture range. Some of them, post-production and geometric replacement methods, simply evaluate the lighting functions once, usually at the start or middle of the time slot. Image samples can also be based on several time instants that are randomly selected [DWL02], from a pre-defined pattern [HA90, AMMH07] or stochastically determined [CCC87]. Although the differences can be subtle, a properly selected sampling method can alleviate aliasing artefacts that are perceptually objectionable [DK00].

In certain situations, the availability of computational resources determines which algorithms are applicable to a given scenario. Geometrical replacement methods, Section 4.3, are usually good candidates to be included in those hardware frameworks where polygonal geometry, texturing

and depth culling can be efficiently evaluated. Real time anisotropic filtering allows the use of techniques previously reserved to offline renderers [Lov05]. Gaming frameworks can also provide real time rates by using post-processing [Ros07, Sou08, Vla08]. However, they are frequently constrained to certain types of motion. Given the existence of GPU-based optical flow algorithms [MT07, PVH08], motion field based methods may also be implemented in real time. Monte Carlo methods, due to computational complexity, are usually reserved for batch rendering environments. Only a few methods based on accumulation buffers [HA90] and frameless rendering [BFMZ94, DWL02, DWWL05] have implemented subsets of distribution ray tracing at interactive frame rates. Recent works based on the REYES rendering architecture may also open the door to efficient hybrid methods [ZHR*09, FLB*09, BLF*10, BFH10, HQL*10].

Most Monte Carlo and hybrid algorithms are also known for being memory hungry algorithms. They are usually associated with intensive ray traversals which results in incoherent memory accesses and the need to store the whole scene in memory. Acceleration structures such as Kd-trees and BSPs can improve efficiency [WMG*07] at the cost of extra memory usage. The need to keep in memory both light paths [VG97, WABG06] and samples until the image is reconstructed [HJW*08, ETH*09, ODR09] also contribute to this extra cost.

In summary and from a strictly practical point of view, there is a tradeoff between the models that can efficiently produce approximate results, and a computationally heavy implementation that accurately calculates the light integration process. For those cases where rendering time and computational resources are not the limiting factors, both hybrid methods and those based on distributed ray tracing are the best candidates. They are capable of producing the most accurate solutions as well as modelling the most complex phenomena. Among all of them, the REYES architecture is probably the one with lower requirements even if, in some cases, it needs to rely on external raytracers to implement global lighting models [CFLB06]. Together with distribution ray tracing, their extensions for photon mapping, lightcuts and metropolis transport and the model of Sung *et al*. are probably the most flexible.

In those cases where images must be produced at interactive frame rates, geometrical replacement and post-processing are the primary categories to consider. Every algorithm in these categories are tailored to solve specific subsets of the problem. Real time post-processing methods are constrained to specific types of camera and object deformation. In any case, there is a clear preference for them in current game architectures [Ros07, Sou08, Vla08].

In those cases where a full flagged method is not an option and there is no need for real time solutions, post-processing methods such as the one of Browstow and Essa [BE01] and Zheng et al. [Zhe06] are capable of producing adequate results that in some cases can be comparable to other more expensive methods.

6. Future Directions

Recent research trends seem to be focused on two diverging directions: finding improved sampling schemes to evaluate the rendering equation; and implementing more efficient algorithms based on hardware acceleration. Whether the solution comes from focusing computation in the most important areas of the image or redesigning algorithms to make use of parallelism, the main challenge is the efficient use of limited computational resources.

An unexplored approach is based on the complex relationships between the HVS and how objects in motion are perceived. We know the HVS is an incredible biological design even though it is not free of limitations in its spatial [CG65] and temporal resolutions [Kel74, SB06]. Its response is non-linear with respect to light wavelength, chromatic and achromatic light [WS82]. Perceived images are not fully focused [ASE79] and in some circumstances may be noisy [Wil83] or contain aliasing [Yel82]. Also, temporal and spatial integration seem to be an important factor in the perception of images in motion [KM71, Bur80, SEC84].

Knowledge of perceptual mechanisms have been successfully exploited in fields like image quality assessment [Dal92, RPG99], tonemapping [ČWNA08] and geometric modelling, rendering and simulation [DDM03, Mys02, OHM*04, ASGC06]. There is good foundation to think a similar approach can be applied to rendering motion blur.

Computational optimizations can exploit the fact that certain stimuli are not perceived or generate such a low response that can be ignored. Also, existing methods are tuned to produce anti-aliased images at the image plane sampling rate. Even if this is a reasonable assumption for static images, the limited attentional bandwidth of the HVS makes us think this rate can be relaxed. Also, given the existence of a temporal window of integration, each frame can be rendered taking into account a context that includes the images immediately before and after. This gives opportunities for further improvement.

In previous sections, we also have implicitly accepted that all temporal samples are equally important. In light of HVs' non-linearities this may not be a correct assumption. Determining which integration mechanisms take part, whether they are biased or not and how we can simulate them are important questions that remain unanswered. Algorithms that are aware of the differences between what is perceptually acceptable and what is physically and numerically accurate can also be used to produce motion cues as a substitute for ideal motion blur.

7. Conclusion

This report provides an overview of the state-of-the-art in motion blur rendering techniques. We have started our discussion with a description of the physical phenomena and mechanisms involved in the generation of motion blur in film and recording devices.

We have also described a formalization that, in the context of computer generated images, mathematically models the light integration process. It explains the complex interactions that take place in an animated scene based on the objects' geometrical relations and their shading functions. Existing motion blur rendering methods have been categorized according to the approach followed to evaluate this expression. An organization based on analytical, geometrical substitution, texture clamping, Monte Carlo, post-production, hybrid and physically inspired methods is the result of this effort.

Simulation of motion blur is known to be one of the areas of the rendering pipeline that is heavier in computational resources. We have classified the methods based on their processing requirements and areas of application. Finally, we have briefly introduced a possible area of improvement that, by exploiting the limits of our perceptual system, can provide performance gains. This family of temporally aware methods is an exciting direction that is yet to be explored.

Acknowledgments

The authors would like to express their gratitude to Matthew Jaques, Adonis Stevenson and the anonymous reviewers for their careful revisions and helpful suggestions. The images in Figure 1 have been kindly made available by Tony Margiocchi, flickr users Noodle Snacks and E01, Carl Rosendahl, Peter Heacox and Enrique Mandujano. We also want to acknowledge the help and prompt responses of the authors of the original papers. Images have been reproduced with permission of, in order of inclusion: Norman Badler, Natalya Tatarchuk, Bob Sumner, Thomas Porter, Henrik Wann Jensen, Bruce Walter, Paul Haeberli, Doyub Kim, Tom Locovic, Gabriel J. Brostow, Yuanfang Zheng, Clement Shimizu and Huei-Yung Lin. This research has been funded by a Marie Curie grant from the Seventh Framework Programme (grant agreement no.: 251415), the Spanish Ministry of Science and Technology (TIN2007-63025, TIN2010-21543) and the Gobierno de Aragón (projects OTRI 2009/0411 and CTPP05/09).

References

[Ada80] ADAMS A.: *The Camera. The Ansel Adams Photography Series I*. Little, Brown and Company, 1980.

[AG00] APODACA A., GRITZ L.: *Advanced RenderMan: Creating CGI for Motion Pictures*. Morgan Kaufmann Publishers, 2000.

[AMMH07] AKENINE-MÖLLER T., MUNKBERG J., HASSELGREN J.: Stochastic rasterization using time-continuous triangles. In *GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (San Diego, CA, USA, 2007), Eurographics Association, pp. 7–16.

[ASE79] ATCHISON D. A., SMITH G., EFRON N.: The effect of pupil size on visual acuity in uncorrected and corrected myopia. *American Journal of Optometry and Physiological Optics 56* (May 1979), 315–323.

[ASGC06] ANSON O., SUNDSTEDT V., GUTIERREZ D., CHALMERS A.: Efficient selective rendering of participating media. In *APGV 2006: Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization* (New York, NY, USA, 2006), ACM, pp. 135–142.

[BA96] BLACK M. J., ANANDAN P.: The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding 63*, 1 (1996), 75–104.

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 561–566.

[BFH10] BRUNHAVER J. S., FATAHALIAN K., HANRAHAN P.: Hardware implementation of micropolygon rasterization with motion and defocus blur. In *HPG'10: Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), Eurographics Association, pp. 1–9.

[BFMZ94] BISHOP G., FUCHS H., MCMILLAN L., ZAGIER E. J. S.: Frameless rendering: Double buffering considered harmful. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM, pp. 175–176.

[BHK*03] BARSKY B. A., HORN D. R., KLEIN S. A., PANG J. A., YU M.: Camera models and optical systems used in computer graphics: Part I: Object-based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications: Part III* Springer-Verlag, pp. 246–255.

[BK08] BARSKY B. A., KOSLOFF T. J.: Algorithms for rendering depth of field effects in computer graphics. In *ICCOMP'08: Proceedings of the 12th WSEAS International Conference on Computers* (Stevens Point, Wisconsin, USA, 2008), World Scientific and Engineering Academy and Society (WSEAS), pp. 999–1010.

[BLF*10] BOULOS S., LUONG E., FATAHALIAN K., MORETON H., HANRAHAN P.: Space-time hierarchical occlusion culling

for micropolygon rendering with motion blur. In *HPG '10: Proceedings of the Conference on High Performance Graphics* (Aire-la-Ville, Switzerland, 2010), Eurographics Association, pp. 11–18.

[Bri99] BRINKMANN R.: *The Art and Science of Digital Compositing*. Morgan Kaufmann Publishers, 1999.

[Bur80] BURR D.: Motion smear. *Nature 284* (Mar 1980), 164–165.

[Cat78] CATMULL E.: A hidden-surface algorithm with antialiasing. In *SIGGRAPH '78: Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1978), ACM, pp. 6–11.

[Cat84] CATMULL E.: An analytic visible surface algorithm for independent pixel processing. *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer graphics and Interactive Techniques* (New York, NY, USA, 1984), ACM, pp. 109–115.

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The reyes image rendering architecture. In *SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), ACM, pp. 95–102.

[CFLB06] CHRISTENSEN P., FONG J., LAUR D., BATALI D.: Ray tracing for the movie 'cars'. *Symposium on Interactive Ray Tracing 0* (2006), 1–6.

[CG65] CAMPBELL F. W., GREEN D. G.: Optical and retinal factors affecting visual resolution. *Journal of Physiology (London) 181* (Dec 1965), 576–593.

[Che95] CHEN S. E.: Quicktime vr: An image-based approach to virtual environment navigation. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), ACM, pp. 29–38.

[CJ02] CAMMARANO M., JENSEN H. W.: Time dependent photon mapping. In *EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering* (Aire-la-Ville, Switzerland, 2002), Eurographics Association, pp. 135–144.

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 263–270.

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Transactions on Graph. 5*, 1 (1986), 51–72.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), ACM, pp. 137–145.

[Cro81] CROW F. C.: A comparison of antialiasing techniques. *IEEE Comput. Graph. Appl. 1*, 1 (1981), 40–48.

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (1982), 7–24.

[CTCS00] CHAI J.-X., TONG X., CHAN S.-C., SHUM H.-Y.: Plenoptic sampling. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 307–318.

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 279–288.

[ČWNA08] ČADÍK M., WIMMER M., NEUMANN L., ARTUSI A.: Evaluation of hdr tone mapping methods using essential perceptual attributes. *Computers & Graphics 32*, 3 (June 2008), 330–349.

[Dal92] DALY S. J.: Visible differences predictor: an algorithm for the assessment of image fidelity. vol. 1666, SPIE, pp. 2–15.

[DBB06] DUTRE P., BALA K., BEKAERT P.: *Advanced Global Illumination*. A. K. Peters, Ltd., 2006.

[DDM03] DAMEZ C., DMITRIEV K., MYSZKOWSKI K.: State of the art in global illumination for interactive applications and high-quality animations. *Computer Graphic Forum 22*, 1 (2003), 55–77.

[DHS*05] DURAND F., HOLZSCHUCH N., SOLER C., CHAN E., SILLION F. X.: A frequency analysis of light transport. *ACM Trans. Graph. 24*, 3 (2005), 1115–1126.

[DK00] DACHILLE F., KAUFMAN A.: High-degree temporal antialiasing. In *CA '00: Proceedings of the Computer Animation* (Washington, DC, USA, 2000), IEEE Computer Society, pp. 49–54.

[DW85] DIPPÉ M. A. Z., WOLD E. H.: Antialiasing through stochastic sampling. *SIGGRAPH Computer Graphics 19*, 3 (1985), 69–78.

[DWL02] DAYAL A., WATSON B., LUEBKE D.: Improving frameless rendering by focusing on change. In *SIGGRAPH '02: ACM SIGGRAPH 2002 Conference Abstracts and Applications* (New York, NY, USA, 2002), ACM, pp. 201–201.

[DWWL05] DAYAL A., WOOLLEY C., WATSON B., LUEBKE D.: Adaptive frameless rendering. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses* (New York, NY, USA, 2005), ACM.

[EMP*03] EBERT D., MUSGRAVE F., PEACHEY D., PERLIN K., WORLEY S.: *Texturing and Modeling: A Procedural Approach* (3rd edition). Morgan Kaufmann Publishers, 2003.

[ETH*09] EGAN K., TSENG Y.-T., HOLZSCHUCH N., DURAND F., RAMAMOORTHI R.: Frequency analysis and sheared reconstruction for rendering motion blur. In *Proceedings of the SIGGRAPH (ACM Transactions on Graphics)* (2009).

[FLB*09] FATAHALIAN K., LUONG E., BOULOS S., AKELEY K., MARK W. R., HANRAHAN P.: Data-parallel rasterization of micropolygons with defocus and motion blur. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009* (New York, NY, USA, 2009), ACM, pp. 59–68.

[FLC80] FEIBUSH E. A., LEVOY M., COOK R. L.: Synthetic texturing using digital filters. In *SIGGRAPH '80: Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1980), ACM, pp. 294–301.

[FP03] FORSYTH D., PONCE J.: *Computer Vision: A Modern Approach*. Prentice-Hall, 2003.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 15–22.

[Gla88] GLASSNER A. S.: Spacetime ray tracing for animation. *IEEE Computer Graphics and Applications 8*, 2 (1988), 60–70.

[Gla99] GLASSNER A. S.: An open and shut case. *IEEE Computer Graphics and Applications 19*, 3 (1999), 82–92.

[GM97] GEIGEL J., MUSGRAVE F. K.: A model for simulating the photographic development process on digital images. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 135–142.

[GM04] GUAN X., MUELLER K.: Point based surface rendering with motion blur. In *SPBG '04: Symposium on Point Based Graphics* (2004).

[Gra85] GRANT C. W.: Integrated analytic spatial and temporal anti-aliasing for polyhedra in 4-space. *SIGGRAPH Computer Graphics 19*, 3 (1985), 79–84.

[Grea] GREEN B.: Sensor artifacts and CMOS rolling shutter. http://www.dvxuser.com/jason/CMOS-CCD/, Accessed on December 2010.

[Greb] GREEN S.: Stupid opengl shader tricks. http://developer.nvidia.com/page/documentation.html. Accessed December 2010.

[HA90] HAEBERLI P., AKELEY K.: The accumulation buffer: hardware support for high-quality rendering. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), ACM, pp. 309–318.

[HDMS03] HAVRAN V., DAMEZ C., MYSZKOWSKI K., SEIDEL H.-P.: An efficient spatio-temporal architecture for animation rendering. In *EGRW '03: Proceedings of the 14th Eurographics Workshop on Rendering* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 106–117.

[Hec86] HECKBERT P. S.: Survey of texture mapping. *IEEE Computer Graphics and Applications 6* (1986), 56–67.

[HJW*08] HACHISUKA T., JAROSZ W., WEISTROFFER R. P., DALE K., HUMPHREYS G., ZWICKER M., JENSEN H. W.: Multidimensional adaptive sampling and reconstruction for ray tracing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 Papers* (New York, NY, USA, 2008), ACM, pp. 33:1–33:10.

[HQL*10] HOU Q., QIN H., LI W., GUO B., ZHOU K.: Micropolygon ray tracing with defocus and motion blur. In *SIGGRAPH '10: ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), ACM, pp. 64:1–64:10.

[HS80] HORN B. K., SCHUNCK B. G.: *Determining Optical Flow*. Tech. rep., Massachusetts Institute of Technology, 1980.

[HWK*10] HEINZLE S., WOLF J., KANAMORI Y., WEYRICH T., NISHITA T., GROSS M.: Motion blur for ewa surface splatting. In *Computer Graphics Forum (Proceedings Eurographics '10)* (2010).

[Jen96] JENSEN H. W.: Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (London, UK, 1996), Springer-Verlag, pp. 21–30.

[JK05] JONES N., KEYSER J.: Real-time geometric motion blur for a deforming polygonal mesh. In *Proceedings of the Computer Graphics International 2005* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 14–18.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), ACM, pp. 143–150.

[KB83] KOREIN J., BADLER N.: Temporal anti-aliasing in computer generated animation. *SIGGRAPH '83: Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1983), pp. 377–388.

[Kel74] KELLEY D. H.: Spatio-temporal frequency characteristics of color-vision mechanisms. *Journal of the Optical Society of America 64* (July 1974), 983–990.

[KH96] KUNDUR D., HATZINAKOS D.: Blind image deconvolution. *Signal Processing Magazine, IEEE 13*, 3 (May 1996), 43–64.

[KK07] KIM D., KO H.-S.: Eulerian motion blur. In *EGWNP '07: Eurographics Workshop on Natural Phenomena* (Prague, Czech Republic, 2007), ACM, pp. 39–46.

[KM71] KRAUSKOPF J., MOLLON J. D.: The independence of the temporal integration properties of individual chromatic mechanisms in the human eye. *Journal of Physiology (London) 219* (Dec 1971), 611–623.

[Kod05] KODAK: *CCD Image Sensor Noise Sources*. Application note, Kodak, January 2005.

[Laf96] LAFORTUNE E.: *Mathematical models and Monte Carlo algorithms for physically based rendering*. Ph.D. thesis, Katholieke Universiteit Leuven, February 1996.

[LC06] LIN H., CHANG C.: Photo-consistent motion blur modeling for realistic image synthesis. In *Pacific-Rim Symposium on Image and Video Technology* (2006), pp. 1273–1282.

[Lov05] LOVISCACH J.: Motion blur for textures by means of anisotropic filtering. In *Eurographics Symposium on Rendering* (2005), pp. 7–14.

[LRU85] LEE M. E., REDNER R. A., USELTON S. P.: Statistically optimized sampling for distributed ray tracing. *SIGGRAPH Comput. Graph. 19*, 3 (1985), 61–68.

[LV00] LOKOVIC T., VEACH E.: Deep shadow maps. In *SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 385–392.

[LW93] LAFORTUNE E. P., WILLEMS Y. D.: Bi-directional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (COMPUGRAPHICS '93)* (1993), pp. 145–153.

[MGS05] MEINGAST M., GEYER C., SASTRY S.: Geometric models of rolling-shutter cameras. *Omnidirectional Vision, Camera Networks and Non-classical Cameras* (2005).

[Mit87] MITCHELL D. P.: Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), ACM, pp. 65–72.

[Mit91] MITCHELL D. P.: Spectrally optimal sampling for distribution ray tracing. *SIGGRAPH Computer Graphics 25*, 4 (1991), 157–164.

[Mit96] MITCHELL D. P.: Consequences of stratified sampling in graphics. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), ACM, pp. 277–280.

[MJ00] MEREDITH-JONES R.: *Point Sampling Algorithms for Simulating Motion Blur*. Master's thesis, University of Toronto, 2000.

[ML85] MAX N. L., LERNER D. M.: A two-and-a-half-d motion-blur algorithm. *SIGGRAPH Computer Graphics 19*, 3 (1985), 85–93.

[MMI*98] MUELLER K., MÜLLER T., II J. E. S., CRAWFIS R., SHAREEF N., YAGEL R.: Splatting errors and antialiasing. *IEEE Transactions on Visualization and Computer Graphics 4*, 2 (1998), 178–191.

[MN88] MITCHELL D. P., NETRAVALI A. N.: Reconstruction filters in computer-graphics. *SIGGRAPH Computer Graphics 22*, 4 (1988), 221–228.

[MT07] MIZUKAMI Y., TADAMURA K.: Optical flow computation on compute unified device architecture. In *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 179–184.

[Mys02] MYSZKOWSKI K.: Perception-based global illumination, rendering, and animation techniques. In *SCCG '02: Proceedings of the 18th Spring Conference on Computer Graphics* (New York, NY, USA, 2002), ACM, pp. 13–24.

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically based modeling and animation of fire. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), ACM, pp. 721–728.

[NRS82] NORTON A., ROCKWOOD A. P., SKOLMOSKI P. T.: Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. *SIGGRAPH Computer Graphics 16*, 3 (1982), 1–8.

[NSL*07] NEHAB D., SANDER P. V., LAWRENCE J., TATARCHUK N., ISIDORO J. R.: Accelerating real-time shading with

reverse reprojection caching. In *GH '07: Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (Aire-la-Ville, Switzerland, 2007), Eurographics Association, pp. 25–35.

[OCMB95] OLANO M., COHEN J., MINE M., BISHOP G.: Combatting rendering latency. In *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1995), ACM, pp. 19-ff.

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

[OHM*04] O'SULLIVAN C., HOWLETT S., MORVAN Y., MCDONNELL R., O'Conor K.: Perceptually Adaptive Graphics. In *STAR-Proceedings of Eurographics 2004* (2004), State of the Art Reports, INRIA and the Eurographics Association, pp. 141–164.

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *SIGGRAPH '81: Proceedings of the 8th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1981), ACM, pp. 297–305.

[PC83] POTMESIL M., CHAKRAVARTY I.: Modeling motion blur in computer-generated images. In *In SIGGRAPH '83: Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1983), ACM, pp. 389–399.

[Pet08] PETERSON B.: *Understanding Shutter Speed: Creative Action and Low-Light Photography Beyond 1/125 Second* (1st edition). Amphoto Books, 2008.

[PH04] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[PLR09] PACHUR D., LAUE T., RÖFER T.: *Real-Time Simulation of Motion-Based Camera Disturbances, RoboCup 2008: Robot Soccer World Cup XII*. Springer-Verlag, Berlin, Heidelberg, 2009.

[Pre92] PRESTON M.: *Temporal Issues in the Rendering of Non-Uniform Rational B-Spline Surfaces*. M.Sc. thesis, University of Manchester, 1992.

[PVH08] PAUWELS K., VAN HULLE M.: Realtime phase-based optical flow on the gpu. In *Proceedings of the CVGPU08* (2008), pp. 1–8.

[RD06] REINHARD E. W. G. P. S., DEBEVEC P.: *High Dynamic Range Imaging — Acquisition, Display and Image-based Lighting*. Morgan Kaufman Publishers, 2006.

[Ree83] REEVES W. T.: Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics 2*, 2 (1983), 91–108.

[RKLC*10] RAGAN-KELLEY J., LEHTINEN J., CHEN J., DOGGETT M., DURAND F.: *Decoupled Sampling for Real-Time Graphics Pipelines*. Tech. Rep., Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory Technical Report Series, 2010.

[Ros07] ROSADO G.: *GPU Gems 3 — Motion Blur as a Post-Processing Effect*. Addison-Wesley, 2007, ch. 27, pp. 575–583.

[RP94] REGAN M., POSE R.: Priority rendering with a virtual reality address recalculation pipeline. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), ACM, pp. 155–162.

[RPG99] RAMASUBRAMANIAN M., PATTANAIK S. N., GREENBERG D. P.: A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 73–82.

[RSC87] REEVES W. T., SALESIN D. H., COOK R. L.: Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), ACM, pp. 283–291.

[SB06] SEKULER R., BLAKE R.: *Perception* (5th edition). McGraw-Hill, Boston, 2006.

[SEC84] SHAPLEY R., ENROTH-CUGELL C.: Visual adaptation and retinal gain controls. *Progress in Retinal Research 3* (January 1984), 263–346.

[Sha64] SHACK R. V.: The influence of image motion and shutter operation on the photographic transfer function. *Applied Optics 3*, 10 (1964), 1171–1181.

[Shi93] SHINYA M.: Spatial anti-aliasing for animation sequences with spatio-temporal filtering. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), ACM, pp. 289–296.

[SI05] SANDER P., ISIDORO J.: Computation culling with explicit early-z and dynamic flow control. In *SIGGRAPH '05: ACM SIGGRAPH 2005 GPU Shading and Rendering Course* (2005), ACM.

[SKAB03] SZIRMAY-KALOS L., ANTAL G., BENEDEK B.: Global illumination animation with random radiance

representation. In *EGRW '03: Proceedings of the 14th Eurographics Workshop on Rendering* (Aire-la-Ville, Switzerland, 2003), Eurographics Association, pp. 64–73.

[Sou08] SOUSA T.: Crysis next-gen effects. In *Proceedings of the Game Developers Conference 2008* (2008).

[SPW02] SUNG K., PEARCE A., WANG C.: Spatial-temporal antialiasing. *IEEE Transactions on Visualization and Computer Graphics 8*, 2 (2002), 144–153.

[SS00] SIMMONS M., SÉQUIN C. H.: Tapestry: A dynamic mesh-based display representation for interactive rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, 2000), Springer-Verlag, pp. 329–340.

[SSBG10] SCHMID J., SUMNER R., BOWLES H., GROSS M.: Programmable motion effects. *ACM Transactions on Graphics (Proc. SIGGRAPH) 29*, 3 (2010), 57:1–57:9.

[SSC03] SHIMIZU C., SHESH A., CHEN B.: *Hardware-Accelerated-Motion-Blur-Generation*. Tech. Rep., Computer Science Department, University of Minnesota, 2003.

[SZ97] SCHER ZAGIER E. J.: A human's eye view: Motion blur and frameless rendering. *Crossroads 3*, 4 (1997), 8–12.

[TBI03] TATARCHUK N., BRENNAN C., ISIDORO J. R.: *Motion Blur Using Geometry and Shading Distortion In ShaderX2: Shader Programming Tips and Tricks with DirectX 9.0*. Wordware, 2003.

[TPWG02] TOLE P., PELLACINI F., WALTER B., GREENBERG D. P.: Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics 21*, 3 (2002), 537–546.

[TSY*07] TELLEEN J., SULLIVAN A., YEE J., WANG O., GUNAWARDANE P., COLLINS I., DAVIS J.: Synthetic shutter speed imaging. *Computer Graphics Forum 26*, 3 (2007), 591–598.

[TZ69] TODD H. N., ZAKIA R. D.: *Photographic Sensitometry; The Study of Tone Reproduction* (1st edition). Morgan and Morgan Hastings-on-Hudson, NY, 1969.

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, 1998. Adviser-Guibas, Leonidas J.

[VG94] VEACH E., GUIBAS L.: Bidirectional estimators for light transport. In *Eurographics Rendering Workshop* (New York, NY, USA, June 1994), Springer-Verlag, pp. 147–162.

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 65–76.

[Vla08] VLACHOS A.: Post-processing in the orange box. In *Proceedings of the Game Developer's Conference 2008* (2008).

[WABG06] WALTER B., ARBREE A., BALA K., GREENBERG D. P.: Multidimensional lightcuts. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 1081–1088.

[WDP99] WALTER B., DRETTAKIS G., PARKER S.: Interactive rendering using the render cache. In *Rendering Techniques (Proceedings of the Eurographics Workshop on Rendering)* (Jun 1999), vol. 10, Springer-Verlag/Wien, pp. 235–246.

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: a scalable approach to illumination. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1098–1107.

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Communications of ACM 23*, 6 (1980), 343–349.

[Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. *SIGGRAPH Computer Graphics 12*, 3 (1978), 270–274.

[Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH Computer Graphics 17*, 3 (1983), 1–11.

[WJAD*03] WANN JENSEN H., ARVO J., DUTRÉ P., KELLER A., OVEN A., PHARR M., SHIRLEY P.: Monte Carlo ray tracing. In *Proceedings of SIGGRAPH 2003 Course Notes #44* (2003).

[WLWD03] WOOLLEY C., LUEBKE D., WATSON B., DAYAL A.: Interruptible rendering. In *I3D '03: Proceedings of the 2003 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2003), ACM, pp. 143–151.

[WMG*07] WALD I., MARK W. R., GÜNTHER J., BOULOS S., IZE T., HUNT W., PARKER S. G., SHIRLEY P.: State of the art in ray tracing animated scenes. In *Eurographics 2007 State of the Art Reports* (2007).

[WS82] WYSZECKI G., STILES W.: *Color Science: Concepts and methods, Quantitative data and Formulae* (2nd edition). John Wiley and Sons, Inc., New York, 1982.

[WZ96] WLOKA M. M., ZELEZNIK R. C.: Interactive real-time motion blur. *The Visual Computer 12* (1996), 273–295.

[Yel82] YELLOTT J. I.: Spectral analysis of spatial sampling by photoreceptors: Topological disorder prevents aliasing. *Vision Research 22* (1982), 1205–1210.

[Zag96] ZAGIER E. J. S.: *Defining and Refining Frameless Rendering*. Tech. Rep., University of North Carolina, Chapel Hill, NC, 1996.

[Zhe06] ZHENG Y., KÖSTLER H., THÜREY N., RÜDE U.: Enhanced motion blur calculation with optical flow. In *Proceedings of Vision, Modeling and Visualization* (2006), pp. 253–260.

[ZHR*09] ZHOU K., HOU Q., REN Z., GONG M., SUN X., GUO B.: Renderants: interactive reyes rendering on gpus. In *Proceedings of SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), ACM, pp. 1–11.

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), ACM, pp. 371–378.