Spatial-Temporal Antialiasing

Kelvin Sung, Andrew Pearce, *Member*, *IEEE*, and Changyaw Wang

Abstract—A framework for discussing the motion blur image generation process is formulated. Previous work is studied in the context of the framework. Due to the implicit assumptions on low temporal frequencies in most motion blur algorithms, issues involved in large screen space movements and fast illumination changes in time have not been adequately addressed so far. A new approach that does not make these assumptions is introduced to solve the spatial-temporal geometric and shading aliasing problems separately. Based on newly developed adaptive algorithms in the spatial-temporal domain, an implementation of the new approach is developed to efficiently deliver high quality motion blurred images in general computer graphics production environments.

Index Terms—Rendering, antialiasing, temporal aliasing.

1 Introduction

Simulating motion blur artifacts in computer-generated images diminishes the disturbing jerkiness in animations. Algorithms for motion blur often make implicit assumptions about the scene database and solve a subset of the general spatial-temporal antialiasing problem. When developing the Maya Rendering System [14], we attempted to eliminate these implicit assumptions while preserving efficiency. This paper summarizes our experience with the motion blur image generation process, from formalizing the problem to understanding the limitations of the previous approaches to formulating a new approach to implementing our solution.

In our discussion, the luminance function (L) is sometimes referred to as the *shading function* and artifacts resulting from insufficient sampling of the shading function are referred to as *shading aliasing*. *Geometric aliasing* is used to refer to artifacts resulting from insufficient sampling of the visibility (g or *geometric*) function. For example, jagged triangle edges are geometric aliasing and broken or noisy Phong highlight is shading aliasing. Images generated *without* exposure time consideration are referred to as *instantaneous images* and images generated with exposure time consideration are referred to as *motion blurred images*.

Our contributions to the problem domain of motion blur are:

1. We introduce a framework to describe the motion blur image generation process based on formulating the artifacts as the results of weighting visibility and shading functions in the spatial-temporal domain. Studying previous approaches with our formulation, we identified a main area that is not well-addressed: scenes that contain relatively high temporal frequencies (e.g., fast moving objects or fast shading changes in time).
2. We propose solving the spatial-temporal visibility and shading functions separately with different strategies. This is based on the observation that the visibility function is well-defined by the moving geometry, while the shading function is arbitrary in general. With this observation, we propose approximating the visibility function with a more analytical approach and using the results to limit the range for stochastic sampling process when approximating the shading function and, thus, increasing the convergence rate.
3. We implement the proposed approach based on newly developed adaptive supersampling algorithms in the spatial-temporal domain. In this way, our system adapts well to spatial-temporal frequency variations in both visibility and shading functions. We also introduce a heuristic to allow the sharing and reusing of visibility results during supersampling of the shading function.

This paper is organized as follows: The next section presents our formulation for the motion blur image generation process. Section 3 examines the existing approaches based on the formulation. Section 4 uses a simple example to illustrate the limitations of motion blur algorithms in the presence of large motion and fast illumination changes in time. Our implementation is presented in Section 5. Section 6 presents the results of our adaptive algorithms. Results from a production rendered with our system [14] are included to demonstrate that the new approach is suitable for productions with sophisticated object movements and complicated shadings.

- *K. Sung is with Computing and Software Systems, University of Washington Bothell, Bothell, WA 98021. E-mail: ksung@bothell.washington.edu.*
- *A. Pearce and C. Wang are with the Rendering Team, Alias|Wavefront, 614 Chapala St., Santa Barbara, CA 93101. E-mail: {pearce, wang}@aw.sgi.com.*

Manuscript received 26 June 2000; revised 19 Jan. 2001; accepted 30 July 2001.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 112341.

2 Problem Formulation

When considering the exposure time, T, one way to describe the image generated at pixel location (x, y) would be:

$$i(x, y, t) = i(\omega, t) = \int_{\Omega} \int_{T} r(\omega, t) L(\omega, t) dt d\omega. \quad (1)$$

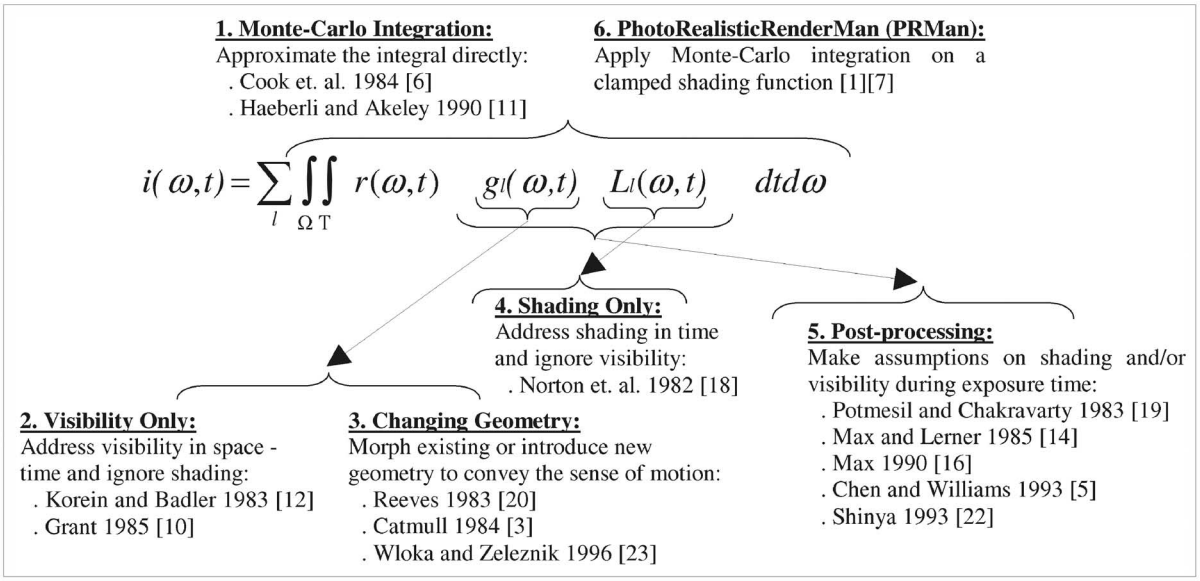


Fig. 1. Summary of previous work.

In (1), the total solid angle from the environment towards a pixel (x, y) is defined by Ω and $L(\omega, t)$ is the incoming luminance from the solid angle ω at time t. The $r(\omega, t)$ term describes a reconstruction filter, it may model the physical movement of the camera shutter closing during the exposure time. In general, this movement may affect different parts of the film in different ways at different instances during the exposure time [21]. In addition, depending on the kernel support required during spatial reconstruction, Ω may include regions outside of the current pixel.

$$i(\omega,t)=\sum_l \int_\Omega \int_T r(\omega,t)g_l(\omega,t)L_l(\omega,t)dtd\omega. \tag{2}$$

Equation (2) is an alternative representation for (1). This equation describes iterating through all objects, l, in the scene database to determine the visible object in the ω direction at time t and computing the luminance of the visible object, $L_l(\omega, t)$. The $g_l(\omega, t)$ term describes the visibility function, where it is 1 only for the visible geometry and 0 otherwise. With a perfect camera, the movement of the mechanical shutter would be instantaneous. In this case, the shutter would open instantaneously, exposing the film to the incoming light energy for the exposure time interval and the shutter would close instantaneously. In such a situation, the mechanical movement of the shutter would not interfere with the film's optical transfer function, thus, the only cause of motion blur artifacts would be the incoming light energy during the exposure interval. All of the existing approaches to modeling motion blur make this assumption and simplify the reconstruction filter $r(\omega, t)$ in (2) by separating it into spatial, $h(\omega)$, and temporal, $f(t)$, reconstruction filters:

$$i(\omega,t)=\sum_l \int_\Omega h(\omega)\int_T f(t)g_l(\omega,t)L_l(\omega,t)dtd\omega. \tag{3}$$

In the next section, the previous work done in motion blur image generation is examined based on the above formulation.

3 Previous Work

As summarized in Fig. 1, one way to understand the previous approaches to motion blur image generation work is to categorize them according to their approaches to solving (2). Except for the Monte Carlo integration work, other approaches approximate the equation by making assumptions about the visibility, g, and the luminance, L, functions. This section discusses each of the six categories in Fig. 1.

3.1 Monte Carlo Integration

Cook et al. [6] formulate the general image generation process as a multidimensional integral equation and propose Distributed Ray Tracing to approximate the solution based on Monte Carlo integration. In terms of solving the image generation problem for the spatial-temporal domain, they approximate (2) by:

$$i(\omega,t)\approx\frac{1}{N_j}\frac{1}{N_k}\sum_j\sum_k i_d(\omega_j,t_k), \tag{4}$$

where

$$i_d(\omega_j,t_k)=\sum_l r(\omega_j,t_k)g_l(\omega_j,t_k)L_l(\omega_j,t_k). \tag{5}$$

In (5), $g_l(\omega_j, t_k)$ and $L_l(\omega_j, t_k)$ are the visible geometry and luminance values computed from a ray (point sampled) based on ω_j at an instant in time t_k. Because of the generality of the underlying Monte Carlo approach, this is the only existing approach we are aware of that is capable of approximating $L_l(\omega, t)$ in general. However, as in all Monte Carlo methods, a large number of samples are usually

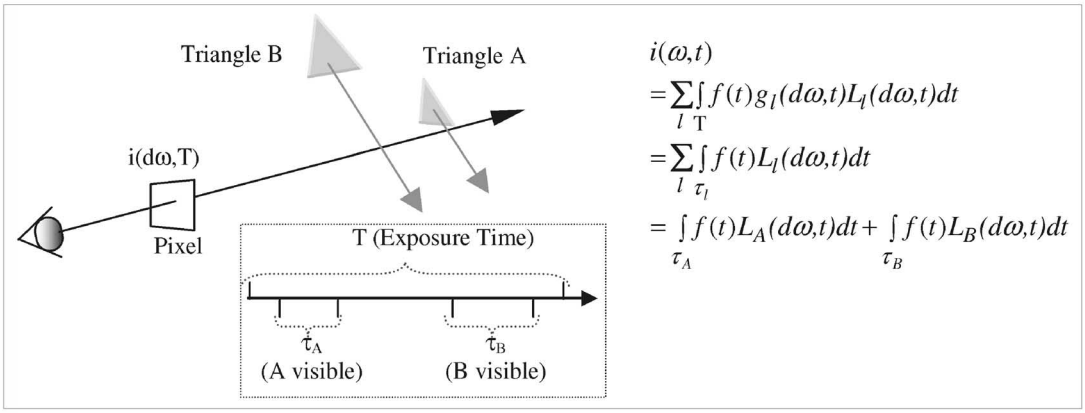


Fig. 2. Korein and Badler's Temporal Visibility.

required to generate images without excessive noise and, thus, can be time-consuming.

Haeberli and Akeley [11] introduce new hardware to implement (4) with multipass z-buffer scan-conversions. The capability of the underlying hardware is the limiting factor for this approach. For example, it is difficult for the hardware to generate images with custom defined illumination models. In addition, the space-time correlation of the sampling position often shows up as banding artifacts in the resulting images.

3.2 Temporal Visibility

Korein and Badler [12] simplify (3) by taking one sample at the center of each pixel with a box filter:

$$i(\omega,t)=\sum_l \int_T f(t)g_l(dw,t)L_l(dw,t)dt. \tag{6}$$

They iterate through all geometry[1] to compute the exact visible interval, τ_l, of each visible geometry. In this way, (6) becomes:

$$i(\omega,t)=\sum_l \int_{\tau_l} f(t)L_l(d\omega,t)dt. \tag{7}$$

Fig. 2 uses an example to help explain (7). In this case, triangles A and B passed through the sampled pixel during the exposure time, covering τ_A and τ_B intervals. Since only these two triangles were visible, only $g_A(\omega,t)$ and $g_B(\omega,t)$ were nonzero and, thus, the summation became the addition of two integrals over τ_A and τ_B intervals, respectively. In their implementation, the temporal filter and L were both assumed to be constants, so (7) was simplified to:

$$i(\omega,t)=\sum_l \tau_l L_l. \tag{8}$$

Grant [10] describes an approximated 4D representation for 3D polyhedra in linear motion and performs scan-conversion in 4D space to compute continuous visible polyhedra in 3D, (x, y, T). These visible polyhedra in the 3D image plane and the time domains are simply the solutions to $g_l(\omega,t)$. With the assumptions of one sample per pixel, box filtering in time, and constant L, Grant also approximates motion blurred images with (8). This approach suffers from accuracy problems because, in general, linear movements of polygonal vertices do not sweep out planar surfaces and thus cannot be accurately represented by 4D polyhedra.

The solutions to (8) solve the temporal geometric aliasing problem for a spatial sample point, but do not address the temporal shading or spatial domain issues. As pointed out by the authors of both articles, in general, spatial-temporal antialiasing problems (for both geometric and shading) cannot be solved separately. That is, in general, it is not possible to separate $g_l(\omega,t)$ into $g_l(\omega)g_l(t)$ (or $L_l(\omega,t)$ into $L_l(\omega)L_l(t)$) and solve the visibility (or shading) problem in the spatial and temporal domains separately.

3.3 Geometric Morphing

It has long been recognized that the appearance of motion in an image can be conveyed by deforming or introducing new geometry strategically placed with respect to the actual movements [12]. This idea can be described as:

$$\begin{aligned} i(\omega,t) &= \sum_l \int_\Omega \int_T r(\omega,t)g_l(\omega,t)L_l(\omega,t)dtd\omega \\ &\approx \sum_{l'} \int_\Omega r'(\omega)g_{l'}(\omega)L'_{l'}(w)dw. \end{aligned} \tag{9}$$

Reeves [20] renders the arealess particle points as antialiased line segments along their trajectories. Catmull [3] describes a transformation that converts geometry's temporal contribution to spatial contribution. Wolka and Zeleznik [23] simulate motion blurring of constant-colored-non-occluded opaque objects by mapping temporal coverage to transparency.[2] Wolka and Zeleznik are interested in conveying the sense of motion in an interactive setting and are not attempting to address image quality issues. All these

1. They described the mathematics for convex polygons and circular discs.

2. This is predicted in (8). Images of objects with constant white color generated from (8) are simply traditional *mask* channel. This suggests that constant-colored-opaque objects' temporal coverage can be simulated by transparency.

approaches approximate motion effects by transforming $g_l(\omega,t)L_l(\omega,t)$ to $g_{l'}(\omega)L'_{l'}(\omega)$. Since the visible geometry and/or the luminance function must be altered, it is difficult for these approaches to adequately address issues involved in complicated motions and/or shadings.

3.4 Temporal Shading

Norton et al. [18] describe an approach to clamp the frequency of texture (or shading) functions to the pixel-sampling rate. This work extends naturally to the time domain. If visibility is constant, the cut-off frequency would be defined by the required number of images to be displayed per second in the final animation. This is, in fact, clamping the frequency of $L(\omega,t)$. The authors do not address issues involved in visibility determination in either space or time. One of the shortcomings of this work is that it requires prior knowledge of the shading functions. This information is generally not available for arbitrary texture or illumination models.

3.5 Postprocessing

Postmesil and Chakravarty [19] model motion blur as a postprocess that blurs a single object based on the point spread function, $g(x,y)$, derived from the movement of the object. This is equivalent to approximating (1) by the following convolution:

$$i(x,y,t) \approx i(x,y,t_1) \otimes g(x,y). \tag{10}$$

Max and Lerner [15] present an efficient way of implementing this model in the spatial domain. Max further extends the implementation to support blurring of polygonal geometry with different vertex velocities [16].

The pixel-tracing filter [22] performs postprocessing based on an entire animation and blurs images with spatial and temporal filtering:

$$i(x,y,t) = \sum_{x'}\sum_{y'}\sum_{\tau}^{N_\tau} h(\tau)i(x',y',\tau)\varpi(x,y,x',y',\overrightarrow{\chi}(x,y,\tau)). \tag{11}$$

In (11), $h(\tau)$ is a temporal filter, $i(x,y,\tau)$ are instantaneous images from the generated animation, N_τ specifies the number of frames to be referenced (in time), $\overrightarrow{\chi}(x,y,\tau)$ is a velocity field computed from the scene animation information, and ϖ is a spatial variant filter at time τ. With a given sequence of images generated from camera motion, Chen and Williams [5] compute the morph-map, a per-pixel offset (*or velocity*) vector, to support their image-based rendering algorithm. The morph-map can be considered as $\overrightarrow{\chi}(x,y,\tau)$ in (11) for the postprocessing motion blur approach.

Approaches based on the postprocessing model are capable of generating very high quality motion blurred images efficiently. However, the implicit assumption is that visibility (g) and luminance (L) functions could be sufficiently approximated with samples from a single instance of time during the exposure interval. Max and Lerner [15] and Shinya [22] alleviate the constant visibility restriction by working with one object at a time and compositing the results in depth ordering. This solves the 2.5D problem, but does not address the situation where motion objects cannot be separated into nonoverlapping layers in depth. In general, solutions to postprocessing approaches cannot adapt to local properties of the image.

3.6 PhotorealisticRenderMan (PRMan)

PRMan [1] clamps the frequency of the shading function in the spatial-temporal domain by precomputing shaded results on the geometry at a fixed spatial resolution and temporal intervals. In their implementation [7], the spatial cut-off frequency is defined by the projected pixel size, while temporal cut-off frequency is assumed a constant.[3] With the unique pipeline of computing shading before visibility, they approximate the shading function by a discrete set of $L(\omega_{uv},t_i)$ for each object before visibility is computed. During rendering, PRMan stochastically samples space and time to solve (4) and, in this case:

$$i_d(\omega_j,t_k) = \sum_l r(\omega_j,t_k)g_l(\omega_j,t_k)L'_l(\omega_j,t_k),$$

where:

$$L'_l(\omega_j,t_k) = \xi_t(L^s_l(\omega_j,t_i), L^s_l(\omega_j,t_{i+1}))\ where\ \ t_i \le t_k \le t_{i+1} \tag{12}$$

and

$$\begin{aligned} &L^s_l(\omega_j,t_i) = \xi_s(L_l(\omega_{uv1},t_i), L_l(\omega_{uv2},t_i)) \\ &where\ \omega_{uv1} \le \omega_j \le \omega_{uv2}. \end{aligned} \tag{13}$$

ξ_s and ξ_t are linear interpolation functions in space and time, respectively,[4] and $L_l(\omega_{uv1},t_i)$ and $L_l(\omega_{uv2},t_i)$ are precomputed shading values at the predefined time instance t_i. With the given stochastically chosen time, t_k, (12) computes L'_l by interpolating the interpolated results, L^s_l, from the two predefined time instances, t_i and t_{i+1}, that bound t_k. Given a predefined time instance, t_i, (13) computes L^s_l at ω_j by interpolating the precomputed shaded results from the predefined locations ω_{uv1} and ω_{uv2}, that bound ω_j. In this way, they stochastically sample the clamped shading function.

PRMan has generated some of the best motion blurred images in numerous high profile productions. However, notice that (12) and (13) achieve similar effects as postprocessing approaches where instantaneous shaded results are *smeared* in time. In the situation where the temporal shading frequency is much higher than the assumed constant, PRMan will generate overblurring artifacts. This has been alleviated in PRMan3.8 by supporting up to six predefined temporal samples. However, since shading is computed before visibility, there is a significant associated cost in both memory and speed for increasing the temporal samples. Another potential shortcoming is that, since there are no obvious ways of clamping visibility frequency independently from shading, the stochastic visibility solution may be noisy for relatively large screen space motions.

3. By default, PRMan supersamples the temporal domain three times. With PRMan 3.8, this number can be set to six.

4. In practice, the spatial interpolation (ξ_s) is performed on the four bounding corners in the parametric (uv) space. For convenience of discussion, this is simplified in (13).

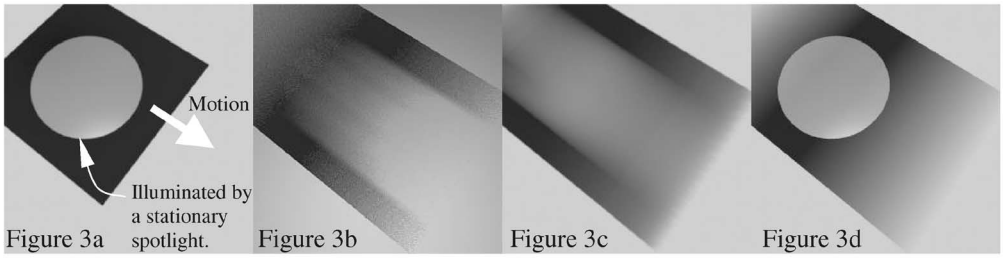


Fig. 3. Potential artifacts.

TABLE 1
Separation of Problem Domains

	Space [Pixel Area, Ω]	**Time** [Exposure Time, T]
Visibility [g(ω,t)]	g(ω)	g(t)
Shading [L(ω,t)]	L(ω)	L(t)

4 Potential Artifacts from Existing Approaches

While all of the previous approaches contributed significantly to the general area of motion blur image generation, postprocessing approaches and PRMan are the most suitable for general computer graphics productions. From the discussions in the previous section, it is clear that high temporal frequency is a potentially problematic area. The example in Fig. 3 further examines this limitation. The image of Fig. 3a shows a plane moving rapidly during the exposure time. A stationary spotlight illuminates the bright circle on the plane. As the plane passes through this region, shading on the plane changes drastically.

The image in Fig. 3b is generated with PRMan [1] with 16 samples per pixel. It is difficult to identify the spot light illuminated region. In this case, the aggressive clamping of temporal shading frequency results in excessive blurring of the illuminated region along the motion path.[5] The noisy visibility solution reflects the convergence rate of Monte Carlo integration in the space-time domain, in the presence of high temporal frequency.

The image in Fig. 3c is generated with the postprocessing motion-blur option of the Maya Rendering System [14]. In this system, a screen space motion vector is computed for each pixel and the postprocess algorithm achieves the blurring effect by scan converting and performing Gaussian filtering along the vectors. Much like PRMan, the constant luminance assumption results in incorrect blurring of the illuminated region along the motion path.

In the next section, we will present the implementation to our approach that generated the image in Fig. 3d.

5 Antialiasing in Space and Time

As described by (1), in general, solving the motion blur image generation problem can be considered as computing the product of two functions, visibility and shading, where each function is defined in the two domains, space and time. When separating the problem domains, the information shown in Table 1 holds.

Most of the existing approaches approximate the solutions by making assumptions on the above table. For example, [12], [10] assume L is constant with an infinitely small area and solve for $g(t)$, while [19], [15], [16] assume time is constant and solve for the *space* column, $g(\omega)L(\omega)$. From our discussion, it is obvious that, in general, the two columns in the table cannot be solved separately. However, the functions defined by the two rows can be solved separately. We have seen Monte Carlo style approaches approximating the product of these two functions directly and resulting in excessive noise in the presence of high frequency information. Examining the two functions closely, we realize that, in a general production environment, the shading function is usually arbitrary, while the visibility function is well-defined by the geometry and the associated motion. This observation suggests that an interesting approach would be to solve the two very different types of functions in different ways. We propose to solve the well-defined visibility function with a more analytic approach and use the results to approximate the arbitrary shading function stochastically. In this way, we limit the range for Monte Carlo approximation and thus increase the convergence rate.

5.1 A Solution Framework

In the spatial domain, when exposure time is not considered, (2) becomes:

$$i(\omega) = \sum_l \int_\Omega h(\omega) g_l(\omega) L_l(\omega) d\omega. \tag{14}$$

When trying to approximate (14), Catmull [4] clips polygons against each other and against each pixel to compute the analytical coverage of each polygon in a pixel. In this way, he divides a pixel into subregions, ω_l, where only one polygon is visible through each subregion. Since only one polygon in each ω_l region has $g_l(\omega) = 1$, (14) becomes:

5. We suspect the streaking artifacts along the motion path may also be a result of the aggressive clamping of the shading frequency.

$$i(\omega) = \sum_{l} \int_{\omega_l} h(\omega) L_l(\omega) d\omega. \quad (15)$$

By combining the results from space and time ((15) and (7)), then computing visibility in the spatial-temporal domain would mean computing spatial subregions, ω_l, and temporal subintervals, τ_l, where only one polygon (object) is visible in each (ω_l, τ_l) 3D volume:

$$i(\omega, t) = \sum_{l} \int_{\omega_l} \int_{\tau_l} r(\omega, t) L_l(\omega, t) dt d\omega. \quad (16)$$

The visible volume in the image-time domain is an area in the object-time domain, $A(\omega_l, \tau_l)$, defined by projecting the subpixel region, ω_l, onto the moving object during the τ_l subinterval. Given this area, motion blur image generation becomes the process of integrating the shading function defined in this area.

The pioneering high quality spatial antialiasing work (e.g., [9], [2]) implement solutions to (15) by first approximating visibility to compute the subregions, ω_l, and then taking multiple point samples on $A(\omega_l)$ to approximate the shadings in each subregion. We propose a similar approach in solving (16). The rest of this section describes our implementation in approximating the solutions to (16). In our discussion, we will refer to the per-object visible volume in the image-time domain, (ω_l, τ_l), as the *image-time visible volume* and the corresponding projected area on the object over time, $A(\omega_l, \tau_l)$, as the *object-time visible area*.

5.2 Solving for Spatial-Temporal Visibility

General analytical solutions of the image-time visible volume (ω_l, τ_l) is a research area. On the other extreme, Monte Carlo approaches simultaneously approximate visibility and shading with point sampling. Catmull assumes instantaneous time to compute the visible area (ω_l, τ_1) analytically. Korein and Badler assume infinitely small pixels, $d\omega$, to compute $(d\omega, \tau_l)$ analytically (as will be explained, this is actually *a line* in the image-time visible volume). We would like to avoid the potentially noisy solutions from general Monte Carlo style approaches [6] and yet maintain the generality of making no assumptions on the shading and visibility functions. We approximate (16) by:

$$i(\omega, t) \approx \sum_{j}^{N_j} H(\omega_j) \sum_{l} \int_{\tau_l^j} f(t) L_l(\omega_j, t) dt, \quad (17)$$

where $H(\omega_j)$ is the integrated result of the spatial reconstruction filter $h(\omega)$. Notice that the inner summation in (17) is (7) and we solve for τ_l based on Korein and Badler's approach. We have extended their algorithm in a subtle and yet important way: Our τ_l is computed for *constant visible geometric objects* instead of *constant visible polygons*. As will be discussed, this subtle extension allows us to decouple the actual number of temporal shading samples taken from the number of visible polygons. This dramatically reduces the rendering cost in the case of fine tessellation or fast motion.

In our implementation, for each pixel, the results of the nth point sample in space, V^n, are stored as a list of continuous temporal coverage, τ_l, grouped according to the corresponding objects, O_l. O_l represents the visible list for object-l, where it contains a list of visible polygons with per polygon temporal coverage information. To facilitate the building of the per object visible list, temporal visibility computation is carried out in a per object manner. In this way, polygons from the same object are processed together and the visible ones are kept in the object's visible list. There is a separate visible list for each visible object in each continuous interval. Visible lists from the same sampling position, V^n, are clipped against each other in depth to ensure correct depth ordering.[6]

The outer summation of (17) describes point sampling in the spatial domain, with N_j defining the number of spatial samples to take. Although the computation for τ_l is efficient, it is desirable to minimize N_j. Since (17) is based on point sampling in the spatial domain, it becomes possible to borrow the results from previous adaptive supersampling approaches (e.g., [17]). The important difference is that previous work was typically based on measurements of how fast color changes in space, whereas we must extend this to measure how fast the visible geometric object changes in the spatial-temporal domain. In our implementation, the changes in τ_l from neighboring results of V^n approximate the visibility changes of object-l in the spatial-temporal domain. According to (17), τ_l weighs (scales) the shaded results and affects the eventual luminance of pixel colors directly. This means we can borrow Mitchell's contrast criteria [17] as a measurement of our spatial-temporal coverage changes. In our implementation, after the initial sampling, we group max and min τ_l from neighboring pixels to compute a *per object visibility contrast*. The second round of visibility sampling will be triggered if any of the objects' visibility contrast is greater than the predefined threshold. Since the contrast computation is based on the coverage values from regions outside of the current pixel, from our experience, further adaptive computation is typically not required.[7]

So far, we have only addressed the issues involved in solving for visibility. At this point, our results are V^n, which are lists of accurate object temporal coverage (τ_l) at different positions (w_j) in the pixel. As illustrated in Fig. 4a, in effect, we have approximated the image-time visible volume (ω_l, τ_l) of an object with a set of line segments, (ω_j, τ_l^j). These line segments are orthogonal to the spatial domain (point sampling in the spatial domain) and cover part (or all) of the exposure time interval. The spatial-temporal adaptive visibility approach is our attempt to ensure that the *lengths* of these line segments for each object do not differ much in any given spatial region. We do not pay attention to the *location* of these line segments in the time domain (starting and ending position in time). In the spatial

6. In general, there may be more than one O_l per V^j and, to support general motion, these visible lists must be clipped against one another. To avoid arbitrary smoothing of geometric surfaces, polygons in each O_l must be at least G^1 continuous.

7. It is possible to increase the number of initial samples and/or the number of secondary samples to overcome the cases where, in the presence of thin-fasting-moving objects, geometric aliasing may still be a problem after the adaptive process.

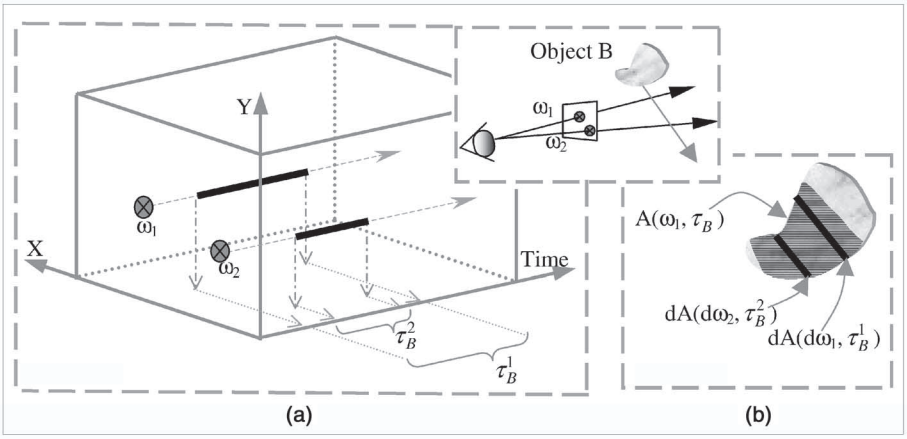


Fig. 4. The image-time visible volume and object-time visible area.

domain, this is equivalent to paying attention to the area coverage of an object but not the location of the area. This is an acceptable approximation only for relatively small sampling regions (e.g., a pixel or the exposure time).

5.3 Solving for Spatial-Temporal Shading

Fig. 4b shows that the line segments in the image-time visible volume are projected *object-time visible curves*, $dA(\omega_j, \tau_l^j)$, on the object.[8] Computing spatial-temporal shading is solving the integration of the shading function over $A(\omega_j, \tau_l)$. We approximate this function directly from (17):

$$i(\omega, t) \approx \sum_{j}^{N_j} H(\omega_j) \sum_{l} \sum_{k_l}^{N_{k_l^\tau}^j} F(t_{k_l}) L_l(\omega_j, t_{k_l}) \frac{\tau_l^j}{\tau_{k_l}^j}. \qquad (18)$$

In (18), $F(t_{k_l})$ is the integrated result of the temporal reconstruction filter $f(t)$ over $\tau_{k_l}^j$. The samples in time are stratified stochastically and chosen according to the probability density function $\frac{\tau_l^j}{\tau_{k_l}^j}$. In other words, $N_{k_l^\tau}^j$ is proportional to the time coverage $\tau_{k_l}^j$. Since $N_{k_l^\tau}^j$ is a per-object attribute, this approximation allows localized control for tuning the image quality.

Notice that spending extra resources in approximating $L(\omega_j, t_{k_l})$ in (18) results in more accurate integration over the $dA(\omega_j, \tau_l^j)$ object-time visible curve and does not contribute effectively to solving the $A(\omega_j, \tau_l)$ area integration. We need an approach to incrementally sample the shading function in the $A(\omega_j, \tau_l)$ area. Referring to (18), in this case, changes in the per-object $F(t_{k_l}) L_l(\omega_j, t_{k_l}) \frac{\tau_{k_l}^j}{\tau_l^j}$ (shaded results weighted by temporal coverage and reconstruction) in the neighboring pixels approximate the spatial-temporal shading variation. Again, we see that the results of $L(\omega, t)$ affect the eventual pixel colors directly, and thus we could borrow Mitchell's contrast criteria [17]. With the initial shaded results from the neighboring pixels, per-object contrast values are computed and extra spatial-temporal shading computation will be triggered when the contrast threshold is exceeded. As in the case of adaptive visibility computation, adaptive shading computation is performed only once.

In order to perform *extra spatial-temporal shading computations*, we need to have the corresponding *spatial-temporal visibility results* (V^n). Referring to (17), we need to compute new τ_l^n (from V^n) for extra ω_n. The straightforward approach is to solve (17) at new/extra pixel positions. However, recall that the adaptive spatial-temporal visibility computation ensures that each object's τ_l^j does not vary much in any given region. This says we should be able to reuse the adaptive visibility results for extra shading computations. Intuitively, given the assumption that the existing visibility approximation is satisfactory, for any new spatial sampling position, ω_n, we would expect its visibility results to be similar to that of the neighboring ω_j's. Based on this observation, one way to reuse the visibility results would be to compute a weighted average from the existing τ_l^j for the new τ_l^n. In practice, this would mean sharing O_l^j for the new sampling position ω_n. The major problem with this approach is that the actual visible polygons cannot be weighted and the visible polygons in O_l^j may not be visible from ω_n. In such cases, the sampling position will be projected to outside of the "visible" polygon. When this happens, we can simply *pull* the projected position to the closest edge of the polygon.

As an initial prototype implementation of the above idea, we simply choose to reuse the visibility results from the ω_j that is closest to the new sampling position (in pixel distance). Somewhat surprisingly, this simple heuristic is capable of delivering high quality images at moderate sampling settings. As such, this is the implementation in the final delivered product. Intuitively, in a given region, if visibility remains similar, then reusing the small number of V^j should be acceptable. On the other extreme, if visibility is changing rapidly in a region, then the adaptive results will deliver a richer set of V^j. Since we shade all the visibility

8. We emphasize that object-time visible area, $A(\omega, \tau)$, is not the traditional visible area *on the object*, $A(\omega)$.

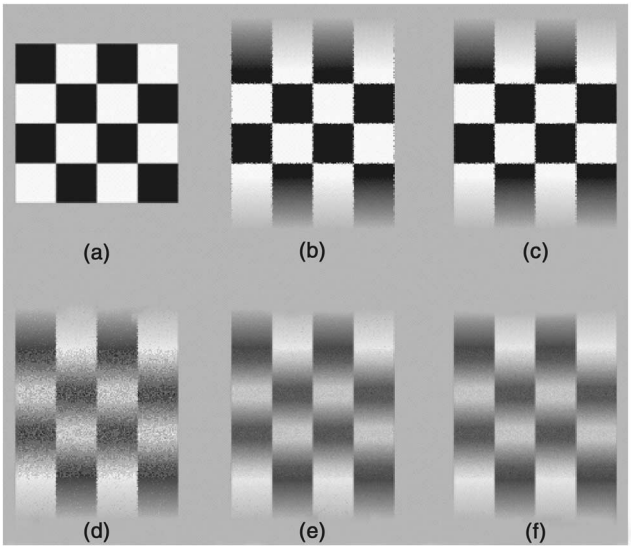


Fig. 5. Adaptive visibility and shading.

results during the first pass, the probability of a contrast violation is lower when visibility is changing rapidly. Even when adaptive shading is necessary, there are a larger number of visibility results to choose from, thus decreasing the chances of large errors. The heuristic may not predict well in a situation where visible polygons from the same object are much smaller than the pixel size. In this case, the maximum spatial shading sample settings should be clamped to that of the maximum visibility sample.

6 Results

The approaches described in Section 5.2 are implemented in the Maya rendering system [14]. This rendering system is used to generate all the images in this section. Images in Fig. 5 demonstrate the results of our adaptive approaches in the spatial-temporal domain for visibility and shading computations. All images in Fig. 5 are generated based on the same checkered plane moving vertically downward, with Fig. 5a being the instantaneous image. Notice that the movement is parallel to the vertical borders for both the plane (geometric) and the checker texture (shading). In this way, the visibility and shading functions along these borders are orthogonal in the space and time domain. This allows us to isolate the problem domains to demonstrate the results of our work incrementally. The image in Fig. 5b is generated with no adaptive computations and with one shading sample ($N^j_{k^\tau_l}$ in (18) is 1) in the time domain. Shading-wise, we can roughly attribute the noise around the horizontal checker boundary to insufficient temporal shading samples and the noise around the vertical checker boundary to insufficient spatial shading samples. Visibility-wise, the noise along the two vertical edges is a result of insufficient spatial samples. Since our solution for temporal visibility is analytical for each sampling position, notice the smooth visibility results along the horizontal boundaries of the plane. The image in Fig. 5c is generated with adaptive visibility, where one initial sample and up to eight adaptive samples may be taken. Notice that the artifacts resulting from insufficient spatial sampling along the plane edges is smoothed out and yet the quality of the checker boundaries remained the same. As expected, increasing visibility sampling does not address noise resulting from shading problems. The image in Fig. 5d is generated with the same settings as that of Fig. 5c, except six temporal shading samples are taken instead of one. In this case, notice that the horizontal checker boundaries start to appear noisy. The image in Fig. 5e is generated with adaptive shading, where up to 16 spatial sampling positions may be taken. Note that, in this case, adaptive shading for the interior checker boundaries are sharing the same visibility results.[9] For the images in Fig. 5b, Fig. 5c, Fig. 5d, and Fig. 5e, the average of 1.87 visibility samples are taken for the nonbackground pixels and, for the image in Fig. 5e, an average of 19.96 spatial-temporal shading samples are taken per nonbackground pixel. The image in Fig. 5f is generated with six temporal samples and a constant 16 spatial samples per pixel.

Fig. 6. (a) Cowboy kicking the saloon door; overall view.

Images in Fig. 6 are scenes from the *Ruby's Saloon* [8], a major *Alias|Wavefront* in-house product demonstration production. Notice the extreme visibility variation during the exposure time where the saloon doors are kicked opened. In addition, note the relatively large motion of the saloon door and the fact that the illumination on the doors changed drastically during the exposure time from the outside of the saloon, where there are only very weak light sources, to the inside the saloon, where it is very bright. This is similar to the situation in Fig. 3, where, during the exposure time, the illumination on the visible object changed drastically. Ruby's Saloon serves as a demonstration that the described algorithms work well in general productions. Fig. 6a, Fig. 6c, Fig. 6e, and Fig. 6g are images from the same production with Fig. 6b, Fig. 6d, Fig. 6f, and Fig. 6h magnifying regions with large motions. In Fig. 6f, notice that, as the leg goes through extreme motion, the handle of the pistol starts to become visible.

Images in Fig. 7 compare the results from our algorithm to that of the Monte Carlo approach. The image in Fig. 7a is

9. Remember the adaptive visibility setting is 1/8, the interior pixels of the plane have no visibility changes, so only one visibility sample results.

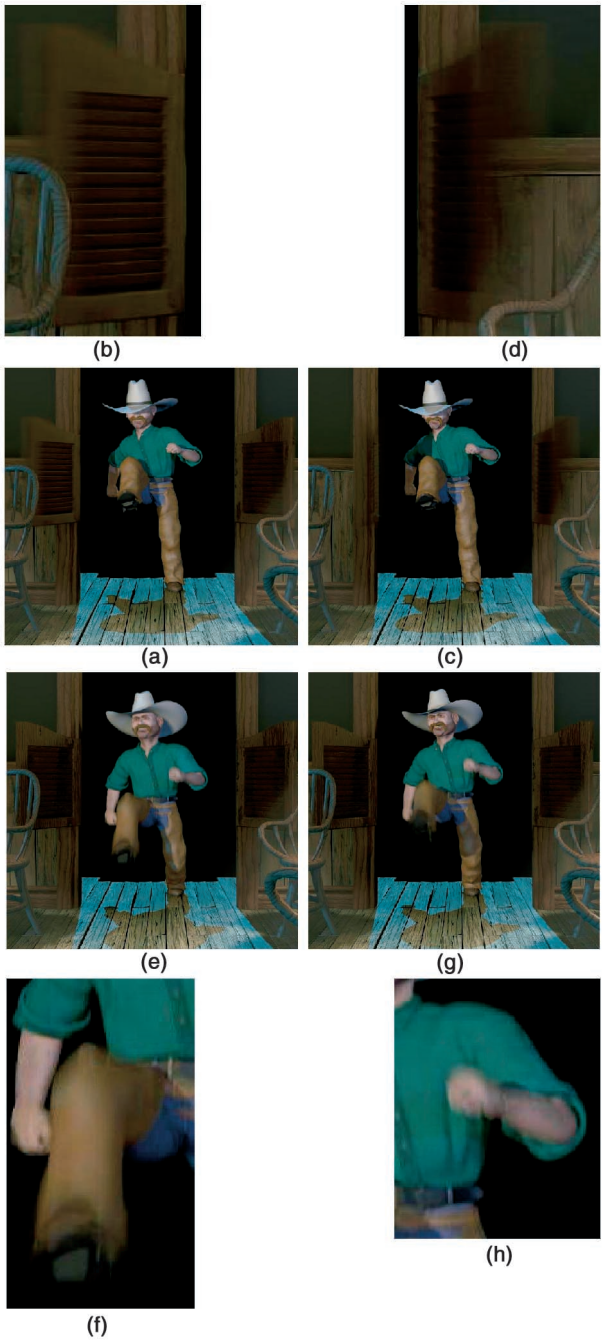


Fig. 6 (continued). Further breakdowns of the the cowboy kicking the saloon door.

generated from Alias | Wavefront PowerAnimator Version 5. In this version of our software, motion blur is simulated with the Monte Carlo Integration approach. The slightly different camera angles and difference in the ground plane between Fig. 7a and Fig. 7c are results of camera model improvements and other changes over different versions of the software. These differences point to the important fact that the comparison is based on two highly complex and independently optimized implementations. In addition, the two algorithms have different strategies in distributing the samples in the space-time domain. Given these constraints, we approach the comparison by fixing the image generation time and comparing the resulting image quality. In this

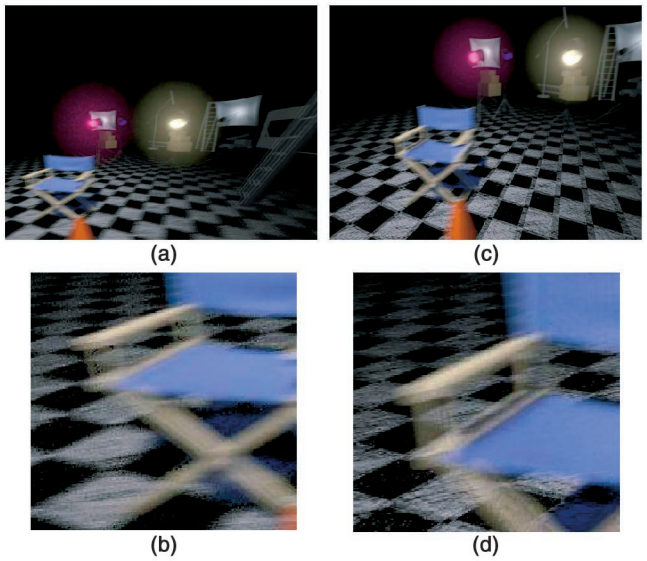


Fig. 7. Motion blue simulation with the Monte Carlo Integration approach.

case, the image in Fig. 7a (Monte Carlo Algorithm) took about 50 minutes to generate, while the image in Fig. 7c (our algorithm) took about 40 minutes. Images in Fig. 7b and Fig. 7d show the details of the centerpiece of this composition. Notice the noisy edges around the director's chair resulting from stochastic sampling in the image of Fig. 7b; the image in Fig. 7d does not have this problem.

7 Conclusion

When trying to describe previous motion blur algorithms with the existing rendering equations (e.g., [13]), we found the equations to be too general to explain the mostly screen-space-based approaches. This motivated us to simplify and develop an alternative formulation based on describing energy arriving at the screen image plane. Studying existing approaches with our formulation, we identified a main area of the motion blur image generation problem that is not well addressed: scenes that contain large motion in screen space and/or fast changing of illumination in time.

We approach the motion blur image generation problem by solving the visibility and shading functions separately. Our solution to the visibility function is analytic in the temporal domain and stochastic in the spatial domain. This approach allows us to develop an adaptive supersampling strategy for visibility computation in the spatial-temporal domain. We stochastically sample the shading function in the spatial-temporal domain based on the visibility solutions. By solving visibility before shading, it becomes possible for us to approximate and reuse visibility results during adaptive supersampling of the shading function. Because of the analytical temporal solution, this approach works especially well in the presence of large movement. The per-object contrast computation architecture allows us to localize the cause of the variance and concentrate the adaptive computation effort. For this reason, our implementation adapts well to the frequency variation of both of the functions in the spatial-temporal domain. Our algorithm

is implemented in the Maya Rendering system [14]. It has been tested in many high quality general productions.

Our implementation is based on [12]. Currently, we only support motion blurring of polygonal or tessellated primitives and we assumed piecewise linear motion in screen space. Since our algorithm is based on casting rays in the spatial domain, it would be straightforward to extend our implementation to support general ray tracing effects (e.g., refraction, reflection). One area that should be interesting for further study is the heuristic for visibility reuse. Because of the general high cost associated with visibility computation, a well-understood approximated solution is highly desirable.

ACKNOWLEDGMENTS

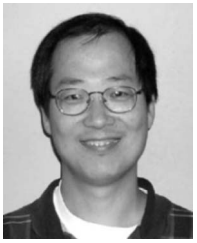
Andrew Woo was the manager of the Maya Rendering Team. The other Maya 1.0 Rendering team members were: Sanjay Bakshi, Silviu Borac, Josh Cameron, Jim Craighead, Renaud Dumeur, Antoine Galbrun, Philippe Limantour, Ryan Meredith, Chris Patmore, Joe Spampinator, Chris Thorne, Mamoudou Traore, Greg Veres, and Gianluca Vezzadini. Eugene Fiume provided us with invaluable technical guidance throughout this work. The Ruby's Saloon was designed, built, and rendered by The Ruby's Saloon Team: Corban Gossett, Kevin "Bubba" Lombadi, Jason Schleifer, and Chris Ford. Last and very importantly, we would like to thank Leslie Olsen of the University of Washington Bothell Writing Center for final proof reading of this paper.

REFERENCES

[1] T. Apodaca and L. Gritz, "Advanced RenderMan: Beyond the Companion," *SIGGRAPH '98 Course Notes 11,* 1998.
[2] L. Carpenter, "The A-Buffer, an Antialiased Hidden Surface Method," *Computer Graphics,* vol. 18, no. 3, pp. 103-108, July 1984.
[3] E. Catmull, "An Analytic Visible Surface Algorithm for Independent Pixel Processing," *Computer Graphics,* vol. 18, no. 3, pp. 109-115, July 1984.
[4] E. Catmull, "A Hidden-Surface Algorithm with Antialiasing," *Computer Graphics,* vol. 12, no. 3, pp. 6-11, Aug. 1978.
[5] S.E. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proc. SIGGRAPH 1993,* pp. 279-288, 1993.
[6] R.L. Cook, T. Porter, and L. Carpenter, "Distributed Ray Tracing," *Computer Graphics,* vol. 18, no. 3, pp. 137-145, July 1984.
[7] R.L. Cook, L. Carpenter, and E. Catmull, "The Reyes Image Rendering Architecture," *Computer Graphics,* vol. 21, no. 4, pp. 95-102, July 1987.
[8] C. Gossett, J. Schleifer, K. Lombadi, and C. Ford, "Ruby's Saloon Production," Alias|Wavefront, Toronto, Canada, http://www.aliaswavefront.com, 2002.
[9] E. Fiume, A. Fournier, and L. Rudolph, "A Parallel Scan Conversion Algorithm with Antialiasing for a General-Purpose Ultracomputer," *Computer Graphics,* vol. 17, no. 3, pp. 141-150, July 1983.
[10] C.W. Grant, "Integrated Analytic Spatial and Temporal Antialiasing for Polyhedra in 4-Space," *Computer Graphics,* vol. 19, no. 3, pp. 79-84, July 1985.
[11] P. Haeberli and K. Akeley, "The Accumulation Buffer: Hardware Support for High-Quality Rendering," *Computer Graphics,* vol. 24, no. 4, pp. 309-318, Aug. 1990.
[12] J. Korein and N. Badler, "Temporal Antialiasing in Computer Generated Animation," *Computer Graphics,* vol. 17, no. 3, pp. 377-388, July 1983.
[13] J.T. Kajiya, "The Rendering Equation," *Computer Graphics,* vol. 20, no. 4, pp.143-150, Aug. 1986.
[14] Maya Unlimited, Alias|Wavefront, Toronto, Canada, http://www.aliaswavefront.com, 2002.
[15] N.L. Max and D.M. Lerner, "A Two-and-a-Half-D Motion-Blur Algorithm," *Computer Graphics,* vol. 19, no. 3, pp. 85-93, July 1985.
[16] N.L. Max, "Polygon-Based Post-Process Motion Blur," *The Visual Computer,* no. 6, pp. 308-314, 1990.
[17] D.P. Mitchell, "Generating Antialiased Images at Low Sampling Densities," *Computer Graphics,* vol. 21, no. 4, pp. 65-72, July 1987.
[18] A. Norton, A.P. Rockwood, and P.T. Skolmoski, "Clamping: A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space," *Computer Graphics,* vol. 16, no. 3, pp. 1-8, July 1982.
[19] M. Potmesil and I. Chakravarty, "Modeling Motion Blur in Computer-Generated Images," *Computer Graphics,* vol. 17, no. 3, pp. 389-399, July 1983.
[20] W.T. Reeves, "Particle Systems: A Technique for Modeling a class of Fuzzy Objects," *Computer Graphics,* vol. 17, no. 3, pp. 359-376, July 1983.
[21] R.V. Shack, "The Influence of Image Motion and Shutter Operation on the Photographic Transfer Function," *Applied Optics,* vol. 3, no. 10, pp. 1171-1181, Oct. 1964.
[22] M. Shinya, "Spatial Anti-Aliasing for Animation Sequences with Spatio-Temporal Filtering," *Proc. SIGGRAPH 1993,* pp. 289-296, 1993.
[23] M.M. Wloka and R.C. Zeleznik, "Interactive Real-Time Motion Blur," *The Visual Computer,* no. 12, pp. 283-295, 1996.

Kelvin Sung received the PhD degree from the University of Illinois at Urbana-Champaign in 1992. His dissertation was in the areas of parallel and efficient computation for high quality image generation. He is an associate professor with the Computing and Software Systems Department at the University of Washington Bothell (UWB). His current research interests are in studying the challenges of creating new approaches to delivering and presenting real-time video and audio in 3D. Before joining UWB, he was a software architect with the Rendering Group at Alias|Wavefront, where he was one of the chief designers of the Maya Renderer and coinvented a patented motion blur algorithm. Before joining Alias|Wavefront, he was a tenure-track assistant professor with the Department of Information Systems and Computer Science, National University of Singapore.

Andrew Pearce has been with Alias|Wavefront for more than 14 years during which time he has written the Alias RayTracer, patch tessellation, depth mapped shadows, motion blur, multi-processor renderers, SDL, and led the Alias rendering development team. He holds two US patents for a motion blur technique and has published numerous technical papers. He has been a senior software architect for rendering, team leader for Maya Paint Effects, engineering lead for Maya 2.5, and led the Maya Paint Effects Screen Saver team. He is a member of the IEEE, ACM, SIGGRAPH, and Writers Guild of Canada. He is currently director of Maya Technologies and will be releasing Maya for Mac OS X in September.



Changyaw Wang received the PhD degree from Indiana University. His research was mostly involved with Monte Carlo integration and sampling. He has worked for Alias|Wavefront for the past eight years and has been involved in designing the Maya renderer since the beginning. He has worked in the areas of rendering, dynamics, and animation.