Instant Convolution Shadows for Volumetric Detail Mapping

DANIEL PATEL
Christian Michelsen Research and University of Bergen
VERONIKA ŠOLTÉSZOVÁ, JAN MARTIN NORDBOTTEN, and STEFAN BRUCKNER
University of Bergen

In this article, we present a method for rendering dynamic scenes featuring translucent procedural volumetric detail with all-frequency soft shadows being cast from objects residing inside the view frustum. Our approach is based on an approximation of physically correct shadows from distant Gaussian area light sources positioned behind the view plane, using iterative convolution. We present a theoretical and empirical analysis of this model and propose an efficient class of convolution kernels which provide high quality at interactive frame rates. Our GPU-based implementation supports arbitrary volumetric detail maps, requires no precomputation, and therefore allows for real-time modification of all rendering parameters.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Color, shading, shadowing, and texture*; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*Viewing algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Shadows, volumetric effects, procedural texturing, filtering

ACM Reference Format:

Patel, D., Šoltészová, V., Nordbotten, J. M., and Bruckner, S. 2013. Instant convolution shadows for volumetric detail mapping. ACM Trans. Graph. 32, 5, Article 154 (September 2013), 18 pages.
DOI: http://dx.doi.org/10.1145/2492684

This project has been carried out within the Geoillustrator research project, which is funded by Statoil and the PETROMAKS programme of the Research Council of Norway.

Authors' addresses: D. Patel (corresponding author), Christian Michelsen Research, Bergen, Norway and Department of Informatics, University of Bergen, Norway; email: daniel@cmr.no; V. Šoltészová, Department of Informatics, University of Bergen, Norway; J. M. Nordbotten, Department of Mathematics, University of Bergen, Norway; S. Bruckner, Department of Informatics, University of Bergen, Norway.

DOI: http://dx.doi.org/10.1145/2492684

1. INTRODUCTION

A detailed representation of model geometry is crucial for achieving subtle shading effects commonly found in the real world. Thus, many techniques have been presented to enrich the appearance of models by adding spatially varying attributes such as color and surface normals [Blinn 1978]. While very powerful and popular, these techniques cannot handle effects such as self-occlusions, self-shadowing, or changes in silhouettes arising from fine-scale details. Although these effects can be achieved by changing the actual geometry, this comes at the cost of rendering a larger number of extra polygons. Techniques such as relief mapping [Oliveira et al. 2000; Policarpo and Oliveira 2006] present a solution to this problem by using image-based detail representations mapped to the surface geometry. By computing ray intersections per fragment, convincing illumination effects can be generated. A further extension of this concept which enables the depiction of arbitrary complex details as well as correct handling of transparency is to employ volume rendering. Slicing or ray marching can be used to sample a volumetric texture or procedural function which represents the interior of the model [Meyer and Neyret 1998] or variations within a shell around the solid object [Peng et al. 2004; Wang et al. 2004; Porumbescu et al. 2005]. However, the computational expense of integrating advanced illumination effects into volume rendering approaches severely limits their applicability in real-time rendering. While precomputation can be used to reduce the runtime impact, this comes at the expense of not being able to handle interactive modification of illumination properties or dynamically changing textures [Chen et al. 2004].

In this article, we derive and study a new approach for dynamically generating approximate but realistic illumination effects for complex scenes containing objects ranging from translucent volumes to opaque meshes. Our technique features occlusion, soft shadows, self-shadowing, translucency, and the ability to handle multiple area light sources.

We approximate shadows in a 3D scene by sweeping over the scene with parallel and equally spaced blocker planes starting from the view plane. Each blocker plane represents the opacity of all intersecting features. Intersection with geometry is calculated using a Layered Depth Image (LDI) [Shade et al. 1998] while intersections with volumetric details are calculated procedurally or looked up from a 3D grid. During propagation, each slice acts both as occluder and shadow receiver. We propagate shadows along the slices by repeated convolution of a 2D shadow buffer containing the accumulated shadow, and by adding the occlusion of the blocker planes to this buffer as the scene is traversed. From a detailed analysis of our model, we are able to devise an efficient class of convolution kernels as a function of the light source shape, which enables rendering at interactive frame rates.

During the slice traversal, all effects are calculated on-the-fly. Our method uses no complex data structures, requires no data storage except for the LDI, and is efficiently parallelized on the GPU.

Therefore all parameters can be interactively modified. Due to our screen space approach, we are able to produce high-frequency shadows from translucent and dynamically changing procedural volumetric detail maps at no additional cost.

Using constant convolution kernels on parallel slices makes our area light sources distant and they are Gaussian shaped due to the Central Limit Theorem. With our fast convolution-based approach, the shape of penumbras and the fusion of shadows from different objects will look plausible but is only approximate. The approximation requires a user-specified parameter which defines an optimal shadow casting distance. Only shadows cast on receivers from objects with this distance in between will be correct, others will be approximate. We analyze this inaccuracy and compare with reference renderings. The lighting is view dependent as the area light sources are positioned behind the view plane. Rendering time increases linearly with the number of light sources since separate convolution must be maintained for each light source. Our fast geometry inside-outside test based on the LDI requires that objects are represented by closed triangle meshes. Our method is well-suited for visualization and investigation of objects containing volumetric details together with simpler opaque mesh-based objects providing global high-quality all-frequency shadow interplay.

The remainder of this article is structured as follows. In Section 2 we discuss related work. Section 3 derives the theoretical basis for our approach. In Section 4 we describe our rendering algorithm which exploits iterative convolution shadows for interactive volumetric detail mapping. Section 5 presents results achieved with our novel technique. Our approximation is analytically quantified in Section 6 and limitations are discussed in Section 7. The article is concluded in Section 8.

2. RELATED WORK

Shadows are important in the generation of realistic images and have been extensively studied in computer graphics. Fundamental shadow methods are reviewed by Woo et al. [1990] and Eisemann et al. [2009]. An overview of general hard shadow methods can be found in the work of Scherzer et al. [2011] and an overview of general soft shadow methods is given in the survey by Hasenfratz et al. [2003].

A straightforward method for creating soft shadows from area light sources is to adapt a hard shadow method by computing binary occlusion maps from several positions on the area light source and averaging them together. However, for quality shadows, many positions must be used, which increases the processing time substantially [Heckbert and Herf 1997; Agrawala et al. 2000]. Ritschel et al. [2009] present a method for screen-space directional occlusion which uses depth peeling for storing several depth values per pixel. The depth values are sampled on a hemisphere around each pixel while recording which unblocked directions are pointing towards a light source. In contrast to our approach, the method does not directly support nonopaque blockers and propagation of shadows through volumes. Deep shadow maps [Lokovic and Veach 2000] allow for volumetric shadows as each pixel in the shadow map stores an array of the fractional visibility of the light source for all depths. This visibility function must be precomputed and compressed to reduce memory requirements. Fourier opacity mapping [Jansen and Bavoil 2010] represents visibility only for low-frequency volumes by precomputing Fourier coefficients. Using few coefficients allows for fast integration of the light absorption but can only approximate low-frequency signals and exhibits ringing around abrupt changes. In contrast, we calculate the visibility while traversing the scene which eliminates the need for precomputation. Baran et al. [2010] present a real-time method with volumetric shadows for recreating the effect of light beams from a point light. The spatial positioning of light beams through a volume is defined by objects blocking the light source. To quickly identify these positions in parallel on the GPU, an epipolar rectification of a shadow map is performed so that light rays are orthogonally aligned with the viewing rays. In later work [Chen et al. 2011], speed is further improved using a min-max structure of the depth image from the camera for finding the lit segments of each viewing ray. Whereas they only model homogeneous translucent material, we allow for freely varying material opacities.

Convolution methods. Our method creates soft shadows using convolution in an iterative manner. Early use of convolution for soft shadows was presented by Max [1991] where a texture representing skylight is convolved to create ground shadows. Soler and Sillion [1998] create soft shadows by projecting the opacities of occluders in a scene to a plane in the center of the occluder geometry. The plane is then convolved with the shape of the light source scaled with the distance to the shadow receiver. This results in an increasing penumbra with increasing distance. Extensions of this approach by Donnelly and Lauritzen [2006] and Annen et al. [2007] additionally employ depth maps. However, these methods only create penumbras of constant width. This limitation was addressed by Annen et al. [2008]. Their technique approximates in real time realistic shadows from multiple area light sources. These works [Annen et al. 2007, 2008] require the calculation of Fourier bases to represent a binary visibility function which can only represent opaque shadow casters. The approach by Eisenmann and Decoret [2008] approximates soft shadows from opaque geometry by slicing the scene to create planar occluder slices. These slices are convolved with a box filter with a size corresponding to their distance to the light source. The approach creates convincing shadows using a low number of slices which allows for prefiltering. Accommodating their method for sampling high-frequency translucent objects would result in high memory consumption for storing all the prefiltered slices. Even assuming that filtering of slices would take no time with their approach, the time complexity of their algorithm will be of a higher order than ours. For each point, they must add up the shadow from all slices between the point and the light, that is, when a point is between slice k and slice $k+1$, they must add up the shadow from all the k slices behind it. For a scene with n slices, this results in $O(n^2)$ operations. In contrast, for slice k, we only add up the shadow from slice $k-1$ into our accumulated light buffer. Our method therefore has complexity $O(n)$. Hence, our approach is an efficient approximation specifically designed to enable the depiction of high-frequency volumetric details, while the method by Eisemann and Decoret provides correct shadows for opaque geometry.

Volume rendering methods. The model presented by Kniss et al. [2003] captures volumetric light attenuation effects including volumetric shadows. This is done by modeling directional scattering, which is calculated using iterative slice-based convolution. They also support procedural volumetric details. Zhang and Crawfis [2003] create shadows by iteratively convolving volumetric slices with a top-hat filter and rendering the results, including geometry using CPU-based splatting. Schott et al. [2009] used iterative convolution to achieve effects similar to ambient occlusion. The technique presented by Solteszova et al. [2010] extended this work with the ability to control the direction of scattering. The main differences from all these works [Kniss et al. 2003; Zhang and Crawfis 2003; Schott et al. 2009; Šoltészová et al. 2010] are that we derive an efficient class of convolution kernels that perform the convolution

faster and that we can model multiple, Gaussian-shaped and colored light sources acting on polygonal models with volumetric pertubations. This achievement arises from a theoretical analysis of iterative convolution in relation to physically correct shadows.

Global illumination methods. Photon mapping methods [Jensen and Christensen 1998] can create high-quality renderings with effects including soft shadows. Real-time photon mapping for simulating refractions was investigated by Sun et al. [2008] and Ihrke et al. [2007]. Sun et al. [2008] send photons from point light sources along refractive paths into a volumetric object followed by a ray casting pass. They perform voxelization into an octree structure which limits their approach to low-frequency effects and sparse scenes. Furthermore, their approach focuses on refraction effects. Refraction is also the focus of Ihrke et al. [2007]. In their approach, irradiance is precomputed using a novel wavefront-based propagation scheme. Wavefronts originating at point light sources are propagated, refracted, and attenuated according to the material properties. The light is gathered by ray casting. Their method requires storage of a complete volume containing refractive values.

Kaplanyan and Dachsbacher [2010] presented a method for realistic real-time indirect illumination. They voxelize the scene into a volume and capture blocker geometry by depth peeling similar to our approach. Radiance from direct light is inserted into the volume by rendering the scene from each point light source, and then injecting a Virtual Point Light (VPL) into the volume for each visible fragment. Area light sources and environment maps are directly injected as VPLs. To achieve interactive frame rates, volume resolution is reduced further away from the camera using a nested volume hierarchy. Light is propagated by stepwise exchange between immediate volumetric neighbors only along the three main axes and VPLs store light direction distributions with spherical harmonics. This results in a low-frequency light approximation. High-frequency surface details are achieved using bump mapping and screen-space ambient occlusion. In contrast, we achieve all-frequency surface and subsurface volumetric details with low space consumption, no precomputation, and no complex data structures, although we focus on shadows.

Crassin et al. [2011] support indirect illumination with specular highlights. Scene geometry is rasterized three times along the three main axes from which a sparse octree is created with large nodes in areas of empty space. Each octree node describes one normal distribution function for the direction of incoming light and one for the surface normal as well as a scalar occlusion value. The parts of the scene receiving direct light are injected into the octree leaf nodes. Parent octree nodes are updated to contain the averaged values of their children. Irradiance is gathered for the final rendering by summing up the contributions from cones covering a hemisphere around each surface fragment. For each cone, a ray is sent out collecting irradiance from the octree. As opposed to our method, they do not directly support translucent volumes. To be able to support this, their gathering of irradiance would require sampling over a volume instead of over a surface and would therfore yield a higher algorithmic complexity. Also, their method requires preprocessing for recreating the octree when the scene changes. Their method uses a more advanced and realistic light model than ours for surface rendering whereas our approach can work with higher resolution in the context of volumetric procedural content.

In contrast to these global illumination methods which produce a rich set of effects, we focus on convincing all-frequency soft shadows from area light sources. We do not require an intermediate discretization of the illumination information and are therefore able to handle high-frequency volumetric details. Our method is fast in execution, and conceptually simple to implement. The main contributions of our work can be summarized as follows.

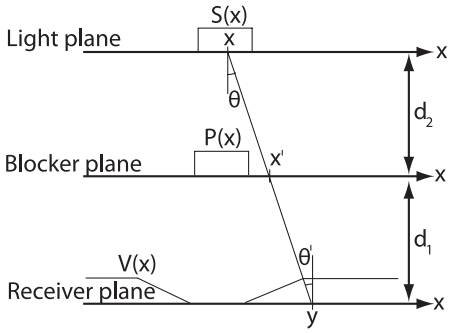


Fig. 1. Soler and Sillions' [1998] shadow model consisting of a light source with exitance function $S(x)$, a blocking geometry with extinction function $P(x)$, and a resulting shadow function $V(x)$ for the receiving surface.

—We present a model for iterative convolution shadows and provide a theoretical and experimental analysis of how it relates to physically correct shadows.

—Based on our analysis, we derive an efficient class of convolution kernels which enable high quality at only a fraction of the computational costs of other approaches.

—We present a GPU-based algorithm for interactive rendering of scenes with dynamically changing volumetric detail maps which supports self-shadowing and translucency.

3. ITERATIVE CONVOLUTION SHADOWS

In our method, we want to render a polygonal model as a volumetric object with an arbitrary detail function applied to specify the appearance of its interior. In order to generate convincing visual results, we need to be able to efficiently compute the effects of self-shadowing due to the volumetric perturbations which have been applied. In this section, we derive an approximation of correct shadows using iterative convolution and investigate the properties of this approach.

3.1 Shadows By Convolution

Soler and Sillion [1998] used convolution to express the shadow cast on a surface from a light source with a blocking geometry in between, as shown in Figure 1.

They express the irradiance at a point y on the receiver as

$$H(y) = E \underbrace{\int_S \frac{cos(\theta)cos(\theta')}{\pi d(x,y)^2} dx}_{F(y)} \underbrace{\int_S S(x)P(x')dx}_{V(y)}, \quad x' = \frac{d_1 x + d_2 y}{d_1 + d_2}. \tag{1}$$

The first integral $F(y)$ is the unoccluded point-to-polygon form factor from y to the light source which represents local surface shading. The second integral $V(y)$ is the visible area of the light source as seen from y and describes the degree of shadowing. Both integrate over the area of the light source S. E is the exitance [Sillion and Puech 1994] of the light source, $d(x, y)$ is the distance between a point x on the light source and a point y on the receiver. $S(x)$ is the degree of light on the light plane and $P(x')$ is the degree of blockage on the blocker plane. The product $S(x)P(x')$ is therefore the amount of light received at y from x.

The form factor F can be computed analytically based on the light function and local surface information. Calculating V is more involved due to the nonlocal problem of considering all potentially blocking geometry between the receiver and the light source. Soler and Sillion [1998] show how $V(y)$ can be expressed as a convolution. They model near-field lighting since their light plane can be arbitrarily close to the receiver plane. We constrain the model for diffuse and distant lighting where the light is situated behind the viewer. This has certain mathematical advantages as will be shown later. With distant lighting from behind the viewer, the light can be described by an environment map of a hemisphere.

We will now change Eq. (1) from modeling near-field lighting to model distant lighting. The distant lighting can be considered as coming from a plane infinitely far away. Therefore the light source will look identical when observed from all positions in the scene. Expressing this in the model of Soler and Sillion [1998], we need to represent the distant light on the local light plane shown in Figure 1. Since the local light plane now represents a nonlocal light, we will refer to it as the virtual light plane. The distant light source will be represented on the virtual light plane by projecting it onto the virtual light plane relative to an observation point y on the receiver plane. This projection will, for an observer moving along the receiver plane, translate with movement. This is analogous to observing a distant light source, such as the moon, through a window (the virtual light plane). When moving the viewpoint sideways, the position of the moon inside the window frame will translate with the same offset as the viewer. To express this translation, we modify the $S(x)$ term to become $S(x-y)$. For distant lighting, the distance between the virtual light plane and the receiver is an artificial construct, therefore we fix it by setting $d_1 + d_2 = 1$, resulting in a new expression for x' as shown in Eq. (2).

In this model, the form factor is reduced to Lambertian shading. Since the light is diffuse, the term $\cos(\theta)$ falls away. The distance d is constant due to distant lighting. What remains is the integration of $\cos(\theta')$ in an interval around the angle θ'. This integral evaluates to $c\cos(\theta')$ for a constant c in the size of the interval and represents Lambertian shading. Collecting constants into K, we get

$$H(y) = K \underbrace{cos(\theta')}_{F(y)} \underbrace{\int_S S(x-y)P(x')dx}_{V(y)}, \quad x' = d_1 x - d_1 y + y. \tag{2}$$

$V(y)$ can be expressed as a convolution (see Appendix A.1 for the derivation).

$$V(y) = \frac{1}{d_1} S\left(-\frac{1}{d_1} y\right) * P(y) \tag{3}$$

After having derived Eq. (3) from Soler and Sillions work [1998] using their notation, we rename the light function $S(x)$ to $l(x)$, the occluder function $P(x)$ to $o(x)$, and the shadow $V(x)$ at distance d_1 to $s_{d_1}(x)$. We change the order of the convolution arguments and we let the integral of $l(x)$ be 1. This results in the expression

$$s_{d_1}(x) = o(x) * \frac{1}{d_1} l\left(-\frac{1}{d_1} x\right) \tag{4}$$

which describes the shadow s on a plane at distance d_1 from the occluder plane as a convolution of the occlusion object $o(x)$ and a kernel derived from the light function $l(x)$.

This relates to one blocker and one receiver only. In the next section, we present an approximation for multiple blockers and receivers, for the purpose of rendering a 3D scene, which will be employed throughout the article.

3.2 Iterative Convolution

A 3D scene can be approximated by several parallel and equally spaced blocker planes where each blocker plane represents the opacity of all intersecting geometry. To render such a scene with shadows, we let all slices act both as occluders and shadow receivers. Any object in the scene will cast shadows on any other object behind it. We obtain the shadow cast from slice n to slice $n+1$ by convolving the opacity of slice n with a kernel $\kappa(x)$ which we will design later. The shadow is inserted into the scene by blending it with the opacity of slice $n+1$. This then becomes the altered opacity of slice $n+1$. We perform the same operation from slice $n+1$ to slice $n+2$ until the end of the scene is reached.

The correct shadow s_{d_1} on a receiver plane at distance d_1 from an occluder plane for a distant light source model was expressed in Eq. (4). Our iterative approach propagates the shadow from slice to slice by convolving with a kernel κ. We define κ so that after iterating it a specific number of times, we have calculated the shadow at the distance d_1 from an occluder. For the purpose of finding κ, we let r_n represent the iterated shadow behind an occluder plane $o(x)$ after n iterations. Then $r_0(x) = o(x)$ and $r_{n+1} = r_n * \kappa$. Since convolution is associative, the iterated shadow after n iterations is

$$r_n(x) = o(x) * \kappa^{*n}(x), \tag{5}$$

where κ^{*n} denotes kernel κ convolved n times with itself.

Let μ_κ denote the mean, and σ_κ^2 the variance of κ. The Central Limit Theorem states that repeated convolution will tend towards a Gaussian function. Throughout the article we write $g_{a,b}$ for the Gaussian function with mean a and variance b. Since n convolutions of κ, denoted as κ^{*n}, have a mean of $n\mu_\kappa$ and variance of $n\sigma_\kappa^2$, we can approximate κ^{*n} with (see Appendix A.2, Eq. (13) for the derivation)

$$\kappa^{*n} \approx g_{n\mu_\kappa, n\sigma_\kappa^2}. \tag{6}$$

Based on this fact, any kernel κ with mean μ_κ and variance σ_κ^2, as specified in Eq. (7), will approximate the Gaussian light $l(x) = g_{\mu_l,\sigma_l^2}(x)$ for any occluder at a plane of distance d_1 from the receiving plane with an iterative slicing distance e (see Appendix A.2, Eq. (14) for the derivation)

$$\mu_\kappa = -e\mu_l \quad \sigma_\kappa^2 = d_1 e \sigma_l^2. \tag{7}$$

Thus, to simulate a light $l(x)$ with our iterative method, we iteratively convolve with a kernel based on the mean and variance of the light. The more different from a Gaussian intensity distribution the light is, the more incorrect the shadow approximation will be. The shadows will still be plausible, but will instead represent the shadows from the closest matching Gaussian light. Also, an optimal distance d_1 for shadows must be selected, that is, for all pairs of shadow casting and shadow receiving planes at this distance from each other, the shadow on the receiver will be correct. When inserting these parameters, in addition to the slice distance e, into Eq. (7), the output will be a mean and variance that the iterative convolution kernel must have. A scene is then rendered using a convolution kernel with this mean and variance. However, an unlimited number of kernels with the same mean and variance exist. A discussion on their difference is presented in Section 3.3.

Since we are modeling a distant light source, we can define the light function over a hemisphere (Figure 2). Thus, an alternative way of specifying μ_l is to define the angle of incoming light from the hemisphere, ω, which is then translated to the mean $\mu_l = \tan(\omega)$ on the virtual light plane. Since μ_l tends to infinity as the angle ω goes towards ± 90 degrees, lower angles must be used. The extent

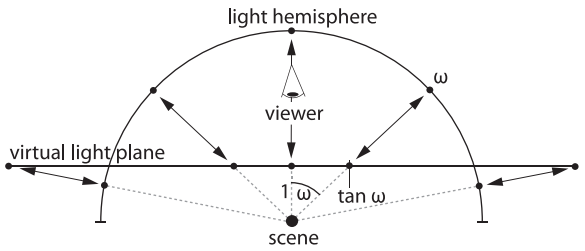


Fig. 2. Relative positions and sizes of a light hemisphere, a light plane, the viewer, and a scene. Arrows show the mapping between the virtual light plane we use in our approach and a light hemisphere.

of the light source is specified by the variance of the light on the virtual light plane.

A light function with more than one intensity peak cannot be modelled well by a Gaussian function. We therefore approximate an n-peaked or n-light-source setup by working with n separate one-peaked Gaussian light sources, each represented by separate kernels and each maintaining a separate shadow buffer when iterating the scene. The combined shadow for each intermediate slice will then be the sum of the n separately calculated shadows. Thus, we can approximate a hemispherical environment map by projecting it onto the virtual light plane to define $l(x)$, and perform Gaussian fitting on $l(x)$ so that the sum of a user-defined number of Gaussians approximate $l(x)$. Then these light sources can be used to simulate the environment map. Finding an optimal Gaussian fitting might not be trivial, however, a greedy method similar to the one suggested by Annen et al. [2008] could be applied.

3.3 1D Convolution Kernels

Due to the Central Limit Theorem, we achieve similar shadows when convolving with any kernel with the same mean and variance as κ. A specific mean and variance therefore defines an equivalence class of kernels. For a sequence of convolutions, any kernel from this class can be used, asymptotically yielding the same result. The Central Limit Theorem holds as long as the Lyapunov condition [Ash and Doleans-Dade 1999] for the kernels is satisfied. This condition allows, for a sequence of convolutions, to use *different* kernels from an equivalence class while still achieving identical shadow. Kernels in this equivalence class do, however, differ in their calculation time and their speed of convergence towards the Gaussian. In Table I we compare three kernels in this equivalence class, all with mean 0 and variance σ^2. They are the top-hat kernel, the Dirac kernel using two samples at equal distances from the center, and the stochastic Dirac kernel. The Dirac kernel has a sample in position $-\sigma$ and in σ to attain a variance of σ^2. The stochastic Dirac kernel randomly perturbs the two sampling positions from the Dirac kernel while in average having mean 0 and variance σ^2. The results of these three kernels convolved with a Heaviside function are depicted in Figure 3. The Dirac kernel, only requiring two samples irrespective of its variance, is attractive in terms of performance. However, one can expect the kernel to behave badly with its coarse sampling pattern which can be seen in the coarse approximation in the middle image. Using the stochastic Dirac kernel reduces the negative effects of the static Dirac kernel and its results can be seen in the rightmost column of Figure 3.

3.4 2D Convolution Kernels

To operate on 2D slices in 3D scenes, the 1D kernels described earlier must be extended to 2D. Since the Central Limit Theorem extends to higher dimensions, the 2D iterated kernel κ^{*n} will converge towards a 2D Gaussian. The 2D Gaussian can be described by a 2D covariance matrix or alternatively by its eigenvalues σ_x^2 and σ_y^2 and its rotation angle α representing the angle of the smallest eigenvector. A 2D light source can thus be approximated using a kernel with a corresponding mean defined by $\mu = (\mu_x, \mu_y)$ and variance defined by the rotation α and $\sigma^2 = (\sigma_x^2, \sigma_y^2)$. This means that when extending our model to 3D, we can simulate light with a rotated ellipsoid shape. For practical reasons we specify the light angle by an x-tilt angle and a y-tilt angle which then defines (μ_x, μ_y).

A natural choice for the convolution kernel would be a 2D Gaussian function. The problem with this approach is that the number of samples quickly becomes prohibitively expensive for interactive applications. A common alternative is to instead place sparse samples on a 2D Poisson disk. Additionally, in order to avoid artifacts caused by a regular sampling pattern, the disk can be rotated by a random offset [Isidoro 2006].

However, based on our observations with the 1D Dirac kernel, we propose a 2D extension of this kernel which leads to excellent visual results with considerably less computational costs. We use an elliptical kernel shape where equidistant samples are only located on the circumference of an ellipse as illustrated in Figure 4. Similar to the stochastic 1D Dirac kernel and analogously to the rotating Poisson disk, we add a random rotational offset to avoid artifacts caused by subsequent iterations using exactly the same locations.

Identifying the major and minor radius (denoted by a and b, respectively, in Figure 4) so that the kernel has the specific variance (σ_x^2, σ_y^2) is slightly more involved than what could be expected due to the lack of a closed-form expression. Therefore, we use two precalculated lookup tables. The tables are calculated for a unit ellipse parameterized by $t \in [0, 1]$ with major and minor radius $a_{unit} = 1 - t$ and $b_{unit} = t$. The first table $ratioToab()$ maps from

Table I. Equivalent Kernels

Kernel	Mean	Variance	Width	Samples
Top-hat	0	σ^2	$\sqrt{12}\sigma$	$\sqrt{12}\sigma$
Dirac	0	σ^2	2σ	2
Stochastic Dirac	0	σ^2	$2\sigma \pm 2$	2

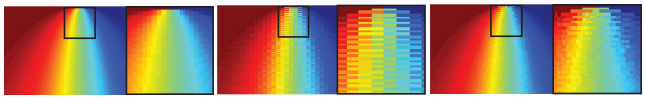


Fig. 3. Iterative shadow evolution for 100 steps using different convolution kernels. Every second image is a zoom-in of the black rectangle. Left to right: Top-hat kernel of width 19 using 19 samples, Dirac kernel of width 11 using 2 samples, and stochastic Dirac kernel of width 11 ± 2 using 2 samples.

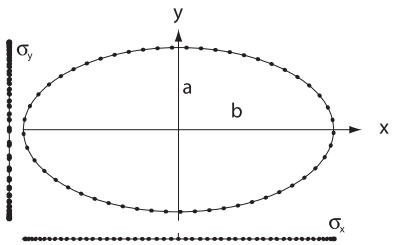


Fig. 4. Equidistant points on an ellipse with major axis a and minor axis b projected to the x-axis and y-axis. The point distribution on the x-axis has variance σ_x^2 and the y-axis has variance σ_y^2.

the ratio $rat = \sigma_y^2/\sigma_x^2$ to a t defining an unit ellipse having an identical variance ratio. The second table $abToVarx()$ maps from t to the x-variance σ_{xunit}^2 of the unit ellipse.

$$rat = \sigma_y^2/\sigma_x^2 \quad t = ratioToab(rat), \quad \sigma_{xunit}^2 = abToVarx(t)$$

$$a_{unit} = 1 - t, \quad b_{unit} = t$$

Now, using the scale difference between the target variance σ_x^2 and the unit ellipse variance σ_{xunit}^2, we must scale a_{unit} and b_{unit} to get the major and minor axis of the target ellipse. Since $cVar(X) = Var(\sqrt{c}X)$ for a stochastic variable X, we scale with $\sqrt{sc}$.

$$sc = \sigma_x^2/\sigma_{xunit}^2 \quad a = a_{unit} * \sqrt{sc}, \quad b = b_{unit} * \sqrt{sc}$$

The derivation of the tables $ratioToab()$ and $abToVarx()$ is found in Appendix A.7.

4. INTERACTIVE VOLUMETRIC DETAIL MAPPING

Our aim is to achieve convincing illumination of models enhanced with complex volumetric details which are either represented by 3D textures or procedurally generated. Furthermore, we want to enable dynamic changes of these details. Such a scenario is not handled well by previous approaches: methods based on precomputed radiance transfer, for instance, need to store illumination information. For high-frequency procedural detail, this is prohibitively expensive in terms of memory consumption and, consequently, performance. Iterative convolution shadows, however, can generate convincing soft shadowing effects without requiring any precomputation. Our approach treats a polygonal model as the boundary of a volumetric object. The interior of the object is defined by a volumetric detail function which defines its material properties, which include the opacity, at each location within the object. We assume a scalar-valued continuous volumetric function $f : \mathbb{R}^3 \to \mathbb{R}$. This volumetric detail function can be a 3D texture, procedurally defined, or a combination of both.

Our algorithm consists of two basic steps which are performed for every frame. First, we generate a Layered Depth Image (LDI) of the polygonal model. The LDI stores, for each depth layer, the depth of the corresponding surface point as seen from the current camera position as well as additional attributes used during rendering. This allows us to distinguish between interior and exterior of the object during the subsequent rendering pass. In the second step, we traverse the bounding box of the model in view-aligned slabs. For each point on a slab, the LDI gives us information on whether the corresponding viewing ray enters or leaves the object within its bounds. For inside regions, we use the detail function to determine the color and opacity accumulated within the slab. This information is used to perform alpha blending into an intermediate image. Illumination is performed using the iterative convolution approach detailed in the previous section.

4.1 Layered Depth Image Generation

An LDI is a well-known data structure for representing models as a 2D array of layered depth pixels [Shade et al. 1998]. A layered depth pixel is a set of depth fragments along one line of sight typically stored in front-to-back order. Initially proposed in the context of image-based rendering, LDIs have found a wide variety of applications in computer graphics. In our approach, we employ an LDI as a means of quickly identifying interior and exterior regions of the model using a volumetric parity test as proposed by Trapp and Döllner [2008]. In addition to depth information, our LDI representation also stores the surface normal and an optional material identifier for each depth layer. Typically, LDIs are generated using the depth peeling technique. However, recent advances in graphics hardware also enable a single-pass approach which is particularly beneficial for models with a large number of primitives. As random access to GPU memory and atomic operations can now be performed in the shader, it is possible to directly generate a per-pixel linked list of depth fragments [Gruen and Thibieroz 2010]. For each pixel, a counter for the number of depth layers and an index into a shared pool of list entries is stored in global device memory. Each of these entries, in turn, stores the fragment data together with the index of the next list entry. The model is then rendered once and for each fragment, a global counter is incremented using atomic operations to allocate a new list entry. After the linked list for each pixel has been generated in this way, an additional sorting pass is performed to sort the entries for each pixel in front-to-back order. In our case, every list entry in our LDI representation stores the depth, eye-space normal direction, and material identifier of the corresponding fragment packed into a single rgbα tuple.

This approach is very fast. For a model with 871414 triangles at a viewport size of 1024×768, LDI generation and sorting can be performed at 130 frames/second on an NVidia GeForce GTX 480. The entire LDI requires approximately 21MB of GPU memory. In cases of high depth complexity and image resolution, memory consumption could potentially become a bottleneck. In such cases, a hybrid approach which incrementally constructs the LDI only for a certain depth range could be employed to limit memory requirements.

4.2 Convolution Shadow Rendering

After the LDI has been generated, we now can employ the volumetric information it represents during rendering. The bounding box of the model is divided into slabs which are processed in front-to-back order. Each slab reads and updates the following buffers.

Light buffers. Two light buffers are required for each slab. $I_0(x, y)$ represents the lighting intensity leaving the previous slab. For each slab pixel, the convolution kernel is applied and the outgoing lighting intensity is written to $I_1(x, y)$. The two buffers are switched after every slab.

Layer buffer. This buffer stores the current parity bit $P(x, y)$, the layer index $L(x, y)$, normal $N(x, y)$, and material identifier $M(x, y)$ of the current depth layer, as well as the depth $Z(x, y)$ of the next depth layer.

Color buffer. The color buffer stores the intermediate color $C(x, y)$ and opacity $A(x, y)$ for each image pixel. Front-to-back alpha blending is used to, for each pixel, combine the contributions of the current slab with the previous values stored in the color buffer.

The layer buffer is initialized with the corresponding values of the first depth layer in the LDI, the color buffer is filled with transparent black, and each light buffer is set according to the color of the corresponding light source. Convolution is performed in a ping-pong manner. Hence, for N light sources, a total of $2N$ light buffers are required. Every slab which intersects the model's bounding box is rendered as a view-aligned quad and for each pixel (x, y) of a slab, the following steps are performed (see Figure 5).

(1) *Convolution*. We first apply the convolution kernel to the outgoing lighting intensity $I_0(x, y)$ of the previous slab. This operation has to be performed for every slab, irrespective of whether it has a visible contribution to the image. Hence, it is important that this is a fast operation. Our implementation supports Gaussian, Poisson disk, and elliptical kernels.

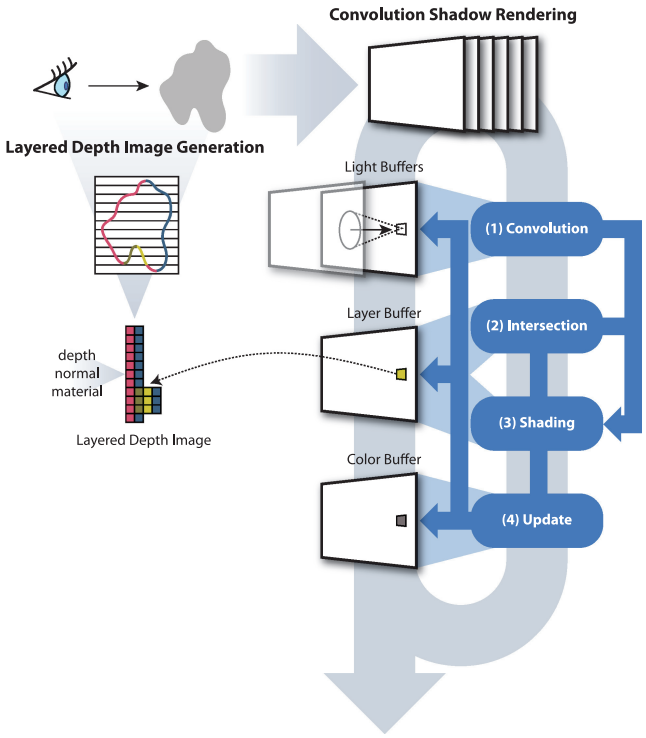


Fig. 5. Conceptual overview of our rendering algorithm.

(2) *Intersection*. We then retrieve the current layer buffer, that is, the parity bit $p = P(x, y)$, layer index $l = L(x, y)$, normal $n = N(x, y)$, and material identifier $m = M(x, y)$ of the current depth layer, as well as the depth $z = Z(x, y)$ of the next depth layer. This information is used to test whether the next depth layer intersects the current slab by comparing z to the depth at the front face and the back face of the slab. If an intersection occurs, the current parity bit p is inverted and the layer index l is incremented. Furthermore, the new values of n, m, and z are read from the layered depth image using the updated layer index l. This is repeated until the next layer's depth value z is larger than the slab's back face to avoid artifacts when multiple intersections occur within the slab.

(3) *Shading*. If the current parity bit is one, that is, the slab is located inside of the model, or an intersection has occurred, the volumetric detail function is evaluated. We assume that the volume remains constant within each slab. Hence, we evaluate the function only at a single point within the slab. If an intersection has occurred, we evaluate the function at the intersection point. Otherwise, we use a per-pixel jittered offset along the viewing ray's path through the slab. The value of the volumetric detail function together with the current material index is then used to determine the material properties of the current sample. We use a set of 2D textures which store the material properties for each function value and material identifier. Next, shading is performed using the incoming illumination intensity for each light source computed in step (1). In our approach, we apply purely diffuse illumination except when a surface intersection has occurred. In this case, we additionally add a specular component given by a microfacet BRDF [Ashikmin et al. 2000].

(4) *Update*. The final color of the sample is then blended into the color buffer $C(x, y)$. The outgoing illumination intensity is determined by the transport color of the material [Kniss et al. 2003] and written into $I_1(x, y)$. If an intersection has been detected in step (2), the updated values of p, l, n, and z are written to the layer buffer.

After all slabs have been processed, the color buffer contains the final image for display. As this approach does not employ any precomputation and is completely independent of model geometry and volumetric detail function, all parameters can be modified interactively. In our implementation, we use a simple gradient widget for the specification of material colors and opacities to achieve a wide variety of effects. Furthermore, we provide multiple procedural volumetric detail functions (fractional Brownian motion, ridged multifractal, improved Perlin noise [Perlin 2002], etc.), which can also be time dependent, and allow the user to alter settings like octaves, lacunarity, and gain. The presented method was implemented in C++ and OpenGL, but all relevant parts of the algorithm are executed on the GPU as GLSL shaders.

5. RESULTS

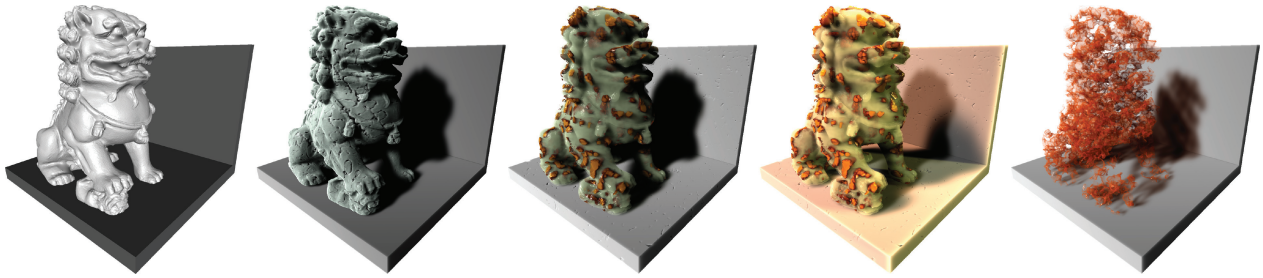
In this section, we first demonstrate that our approach is able to generate high-quality visual results for arbitrary volumetric detail functions applied to common models at interactive frame rates. High-frequency dynamic details as well as large-scale modifications are equally possible. All frame rates were measured on a system equipped with an Intel Core i7 3.20 GHz CPU and an NVidia GeForce GTX 480 GPU using a viewport size of 768×768. They include layered depth image generation, evaluation of the volumetric detail function, and rendering. A shader-based implementation of a procedural 4D noise function was used as the basis for volumetric detail mapping.

In Figure 6, our method is used to apply translucency and coloring effects to alter the appearance of the original model (rendered using standard Phong shading in Figure 6(a)). Without shadows, as shown in Figure 6(b), the result is not very dramatic, while the self-shadowing effects of the volumetric detail function in Figure 6(c) lead to a much more realistic depiction. The frame rate was 16.47 frames/second. Figure 7 shows several different variations of how our approach can be used for drastically altering a model's appearance. In Figure 7(a), the original model is shown. Figure 7(b) shows how a simple procedural detail function can be applied to achieve an aged look. In Figure 7(c), the lookup table has been modified to give the appearance of opaque pieces of material embedded in a translucent medium and a different detail map has been applied to the floor and wall. Figure 7(d) uses the same settings, but an additional colored light source has been added. Finally, Figure 7(e) shows the effects of significant topological changes, translucency, and chromatic attenuation. The frame rate was 10.47 frames/second for one light source and 7.85 frames/second for two light sources. The example in Figure 8 shows the effect of different light sources. The original model is shown in Figure 8(a), while Figure 8(b) and (c) show volumetric detail mapping rendered using two different colored light sources. The combined effect of both light sources is shown in Figure 8(d). The frame rate was 18.06 frames/second for one light source and 13.24 frames/second for two light sources. Figure 9 demonstrates large-scale procedural modifications combined with the application of a 3D brick texture. Two light sources were used and the frame rate was 7.48 frames/second. Figure 10 shows how varying volumetric texturing effects can be assigned to different materials using a detailed model of the human heart. The frame rate was 10.39 frames/second.

All these results were generated with a 2-sample ellipse kernel which has shown to provide virtually indistinguishable and

(a) (b) (c)

Fig. 6. Effect of shadows. Model rendered using (a) Phong shading; (b) volumetric detail mapping without shadows; (c) volumetric detail mapping with shadows at 16.47 frames/second. The *Raptor* model was provided courtesy of 3D Systems geomagic Solutions by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/).



(a) (b) (c) (d) (e)

Fig. 7. Detail mapping parameters. Model rendered using (a) Phong shading; (b)–(e) varying volumetric detail mapping settings. The frames rate was 10.47 frames/second for all images except (d) where the addition of a second light source resulted in a reduction to 7.85 frames/second. The *Chinese Lion* model was provided courtesy of Inria by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/).

sometimes better results compared to kernels with significantly higher computational costs. A comparison of different kernels is shown in Figure 11. The figure shows, from left to right, a 2-sample ellipse kernel, a 3-sample ellipse kernel, Poisson disk sampling with 8 and 16 samples, and a 19×19 Gaussian kernel for low (top row) and high (bottom row) light source variance. For comparison, the rightmost column shows reference renderings generated using PBRT [Pharr and Humphreys 2010]. It can be seen that even the 2-sample ellipse kernel produces results with no significant differences to the Gaussian kernel, at only a small fraction of the cost. The 8-sample Poisson disk, on the other hand, shows a considerable amount of noise for the high-variance case. The frame rates were 22.58 frames/second (2-sample ellipse), 20.34 frames/second (3-sample ellipse), 11.83 frames/second (8-sample Poisson disk), 6.71 frames/second (16-sample Poisson disk), and 0.48 frames/second (19×19 Gaussian).

To further show that our approach is able to closely approximate realistic volumetric illumination effects, we performed a comparison with Exposure Render [Kroes et al. 2012], an open-source CUDA-based implementation of Monte Carlo volume ray tracing. As Exposure Render only supports volume data, we used a medical computed tomography scan with a resolution of $512 \times 512 \times 460$ voxels as input data. The results are shown in Figure 12. While there are subtle differences in the images mainly due to differences in transfer function handling, our approach is able to accurately mimic the overall appearance of opaque and translucent structures and even captures small-scale details. Our algorithm performed at 9.43 frames/second while Exposure Render required several seconds for convergence (to ensure full convergence the presented images were generated using 1000 iterations which took approximately 40 seconds).

To demonstrate the advantages of our approach compared to other methods which employ voxelization, we compare a grid representation of a volumetric detail function to procedurally generated effects in Figure 13. Even though a fairly high grid resolution of $412 \times 412 \times 247$ was used, the magnification clearly shows that fine details cannot be accurately captured. We conclude that while voxel-based representations can be advantageous in sparse scenes, our method can handle dense volumetric details as it operates in image space and does not require an intermediate discretization.

In order to verify our theoretical light model, we performed several additional comparisons using PBRT as a reference. In Figure 14, the scene consists of a thin disk above a parallel plane. Three light sources are used: a green light with low uniform variance ($\sigma_x^2 = \sigma_y^2 = 0.8$), a red light with higher uniform variance ($\sigma_x^2 = \sigma_y^2 = 3.2$), and a blue light with zero σ_x^2 and high σ_y^2 rotated 70 degrees clockwise. This scene setup was rendered with our method in Figure 14(a). In Figure 14(b), PBRT was used by projecting the light settings from our light plane into an environment map as shown in Figure 2. We use the variance calculated from Eq. (7)

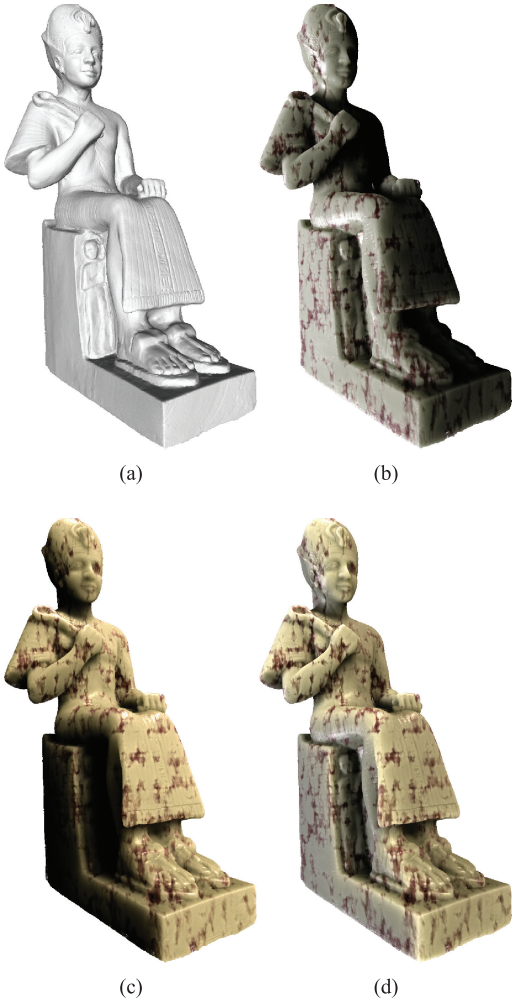


Fig. 8. Multiple light sources. Model rendered with (a) Phong shading; (b) and (c) our algorithm using one light source at 18.06 frames/second; and (d) our algorithm using two light sources at 13.24 frames/second. The *Ramesses* model was provided courtesy of Inria by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/).

to get the correct shadow for the disk-plane distance. The similarity of the renderings demonstrates that we can accurately simulate shadows from light sources of different angles and shapes when the occluder and receiver are on parallel planes at a known distance.

Figure 15 depicts increasing uniform light source variance from left to right (using the scene setup shown in Figure 18(a) but with narrow rectangular occluders) with light coming from the left. These results illustrate that even though our model is an approximation, it results in visually plausible soft shadows.

We have shown that our method is able to render interactive scenes with dynamically changing and translucent content under multiple light sources of arbitrary Gaussian shape and rotation. While our model is an approximation of correct shadows, we have analyzed the behavior of this approach and shown that it allows us to generate visually convincing results. In particular, as our elliptical kernel is able to simulate large light source variances with a low number of samples, our approach is well-suited for representing natural directional illumination from distant lights. In the two subsequent sections, we will analyze and discuss the limitations of our method.

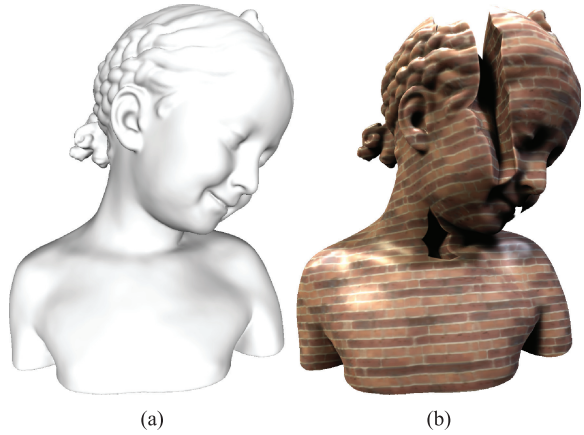


Fig. 9. Combination of different volumetric detail functions. Model rendered with (a) Phong shading; (b) our algorithm using a 3D brick texture to specify color and a procedural function to specify opacity at 7.48 frames/second. The *Bimba Con Nastrino* model was provided courtesy of IMATI-GE/CNR by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/).

6. COMPARISON OF CORRECT AND ITERATIVE CONVOLUTION SHADOWS

We will now analyze the difference between a correct shadow and our iterative convolution shadows. We have defined a kernel κ which, for a Gaussian light source, generates the correct shadow for any occluder plane $o(x)$ after iterating ahead to a plane at a distance d_1. Next we discuss how the shadow looks at distances where $d \neq d_1$ and compare it with how correct shadows behave at these distances.

For the iterative approach, to reach a distance d, given slice distance e, one must perform d/e convolutions and the convolution becomes

$$n = \frac{d}{e}, \quad \kappa^{*n} \overset{\text{Eq. 6}}{\approx} g_{n\mu_\kappa, n\sigma_\kappa^2} \overset{\text{Eq. 7}}{=} g_{-d\mu_l, dd_1\sigma_l^2}.$$

If we let z_d represent the iterated shadow at distance d behind an occluder plane o, we get

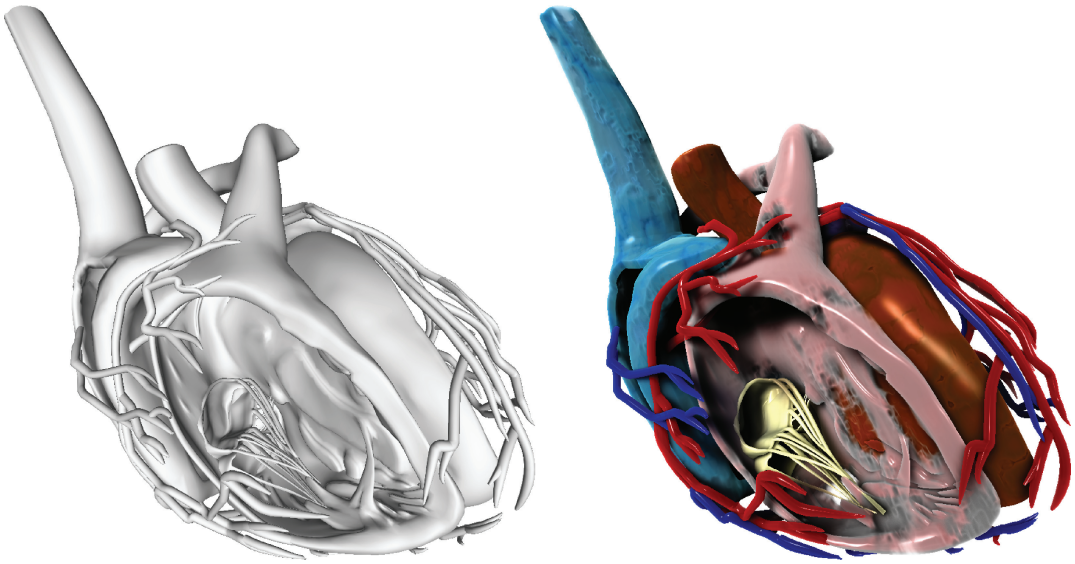
$$z_d(x) \approx o(x) * g_{\mu_d, \sigma_d^2}, \quad \mu_d = -d\mu_l,\ \sigma_d^2 = dd_1\sigma_l^2. \tag{8}$$

Now we have an expression of the shadow z_d for the iterative approach and the shadow s_d (Eq. (4)) for the correct approach and we can compare them. We compare the shadows behind the simplest occluder possible: a sharp edge occluder represented by the Heaviside function H. Therefore we set $o(x) = H(x)$.

Isovalues of correct shadows behind an edge are linear. Behind a Heaviside occluder and for any light function, the regions of shadow of equal intensity form lines (see Appendix A.3 for the full derivation). By solving $s_d(x) = s$ for x, we get the x-position of shadows with strength s at distance d from a Heaviside occluder.

$$x(d) = d \cdot C_s, \quad \text{where } C_s = -L^{-1}(1-s) \tag{9}$$

In Eq. (9), $L(x)$ denotes the indefinite integral of $l(x)$ and C_s denotes a constant defined by the isoshadow strength s. The equation



(a) (b)

Fig. 10. Multi-material support. Model of the human heart rendered with (a) Phong shading; (b) our algorithm using different detail mapping settings for individual structures at 10.39 frames/second. The *Anatomium 3D Anatomy* model was provided courtesy of CFLietzau/3DSpecial (http://www.anatomium.com/).

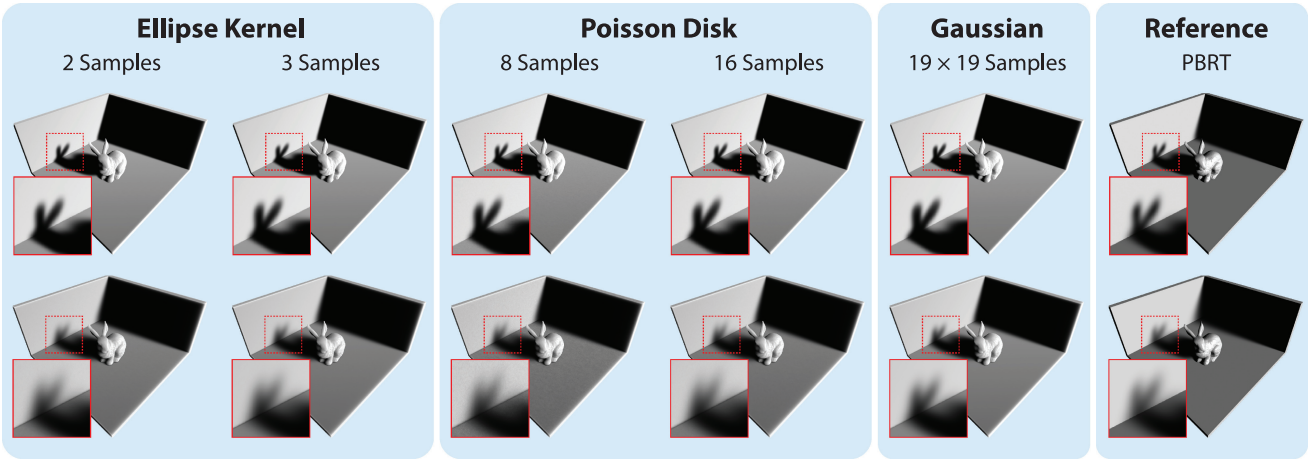


Fig. 11. Comparison of different convolution kernels using a low-variance light source (top row) and high-variance light source (bottom row). The frame rates were 22.58 frames/second (2-sample ellipse), 20.34 frames/second (3-sample ellipse), 11.83 frames/second (8-sample Poisson disk), 6.71 frames/second (16-sample Poisson disk), and 0.48 frames/second (19 × 19 Gaussian). The *Stanford Bunny* model was provided courtesy of the Stanford University Computer Graphics Laboratory by the Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep/).

shows the linearity between the distance of the occluder d to the shadow and the position x of the isoshadow. Intuitively, this can be explained as observers on all points along an isoline will see the same unblocked part of the virtual light plane. A simulation of this behavior for a sharp edge occluder and a light with mean zero is shown in the left part of Figure 16.

Isovalues of iterative shadows behind an edge are nonlinear. Behind a Heaviside occluder and for any light function, the regions of shadow of equal intensity are nonlinear and follow a square root trajectory (see Appendix A.4 for the full derivation).

$$x(d) = -d\mu_l + c\sigma_l\sqrt{d}$$

The right part of Figure 16 shows isolines for a light function with $\mu_l = 0$. When $\mu_l = 0$, the first term disappears and the isolines follow a square root trajectory in d. For noncentered lights where $\mu_l \neq 0$, the result will be a sheared version of the right part of Figure 16; the straight white isoline for shadow intensity $\frac{1}{2}$ will then be angled. In the next paragraph, we show that for any light, the white $\frac{1}{2}$-isoline for the iterative approach is correct.

Equal characteristics for both approaches. As shown, behind a Heaviside occluder, the correct shadow is linear whereas the iterative shadow is nonlinear. However, for shadow intensity $\frac{1}{2}$, that is, in half shadow, the iterative shadow is linear and equal to the correct shadow and is (see Appendix A.5 for the full derivation)

Fig. 12. Monte-Carlo volume ray tracing of a medical computed tomography scan (left) compared to our iterative convolution-based approach (right) with light from different directions. Our approach performed at 9.43 frames/second while the Monto-Carlo volume ray tracer required several seconds to converge. The *MANIX* dataset was provided courtesy of the OsiriX Foundation (http://www.osirix-viewer.com/).

as follows.

$$x(d) = -dL^{-1}\left(\frac{1}{2}\right) \tag{10}$$

This particular isoline can be considered important for shadows as it represents the centerline of the penumbra and thus conveys in which general direction the shadow is cast. One can therefore say that the iterative method casts shadows in the correct direction behind a Heaviside occluder. For light sources having zero variance, the nonlinear part falls away and the iterative model is correct everywhere. Such light sources are point light sources creating hard shadows due to a spike at μ_l.

The proofs presented for a 1D light function creating shadows on a line can be straightforwardly extended to 2D light functions casting shadows on planes. We describe this in Section A.6 in the Appendix.

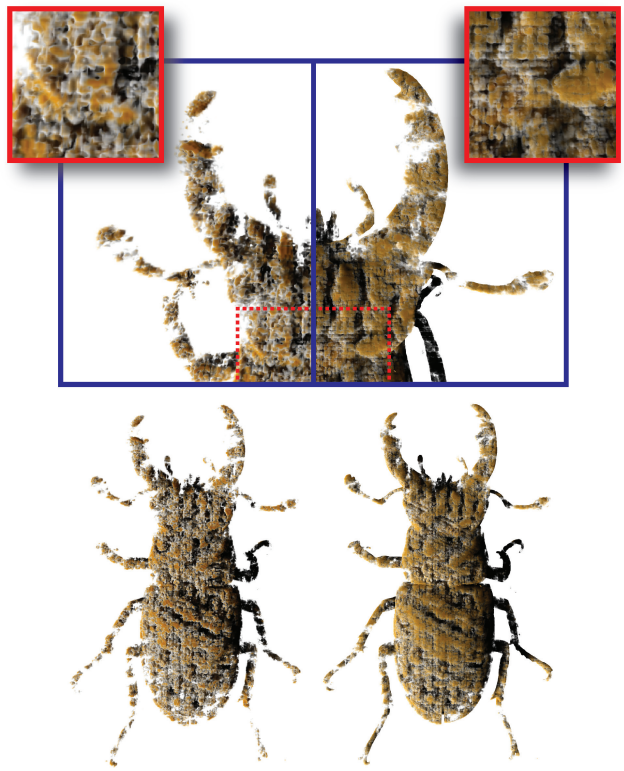
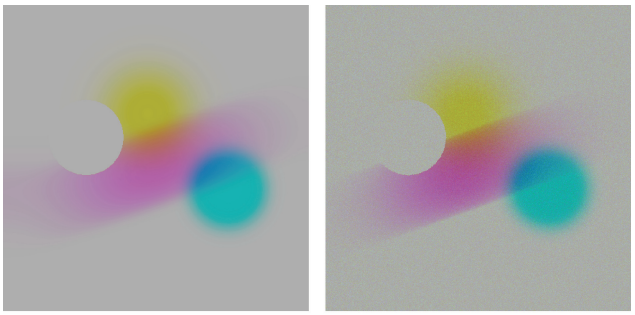


Fig. 13. Comparison of a sampled volumetric detail function (left) and procedural details (right). The *Stag Beetle* dataset was provided courtesy of the Institute of Computer Graphics and Algorithms, Vienna University of Technology (http://www.cg.tuwien.ac.at/).

We have now related the iterative shadow model to correct shadows. We showed that shadows will be cast in the right direction (regardless of the targeted optimal distance d_1) and that iterative shadows have a square root falloff. Figure 17 illustrates the deviation from a correct shadow behind a Heaviside occluder at distances different from d_1. Shadow penumbras grow with the distance from the shadow caster in both approaches. However, the optimal distance d_1 specifies at what point the penumbra for the iterative method matches the correct penumbra. In Figure 18 we demonstrate the effect of changing the optimal distance factor d_1 on a scene consisting of two Heaviside occluders at different distances from a parallel wall. The scene is shown, rendered with our method, in Figure 18(a). The lower half of the back wall is angled at 45 degrees so that the evolution of the shadow at increasing distance from each shadow caster can be monitored. Figure 18(b) shows the reference image rendered in PBRT. Figure 18(c) shows a rendering using a convolution kernel designed for optimal shadows for the distance of the right box, and Figure 18(d) for the left box. The blue and red lines in Figure 18(b), (c), and (d) are correct and approximate isoshadow lines, respectively. For each image, the outer isolines around the left and right object have the same isovalue and the step in shadow strength between isolines is equal. The linearity of the isolines at increasing distance can be seen on the angled surface in the reference rendering of Figure 18(b). The square root shape of the corresponding isolines for our method can be seen in Figure 18(c) and (d). These observations correspond with our analysis and are illustrated in Figure 16 and Figure 17.



(a) Our algorithm (b) PBRT reference rendering

(c) Virtual light plane (d) PBRT environment map

Fig. 14. Comparison between (a) our method; (b) the physically based PBRT renderer for three light sources with different characteristics. The virtual light plane shown in (c) and (d) shows the corresponding PBRT environment map projected onto a hemisphere.

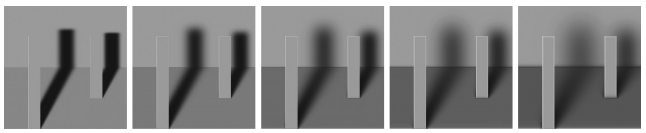


Fig. 15. Demonstration of increasing uniform variance of light source starting with zero from left to right. Ground plane is angled 45 degrees.

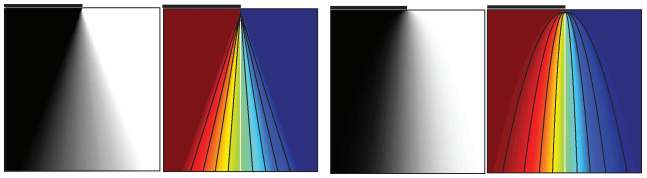


Fig. 16. Shadow behind a Heaviside occluder for light with mean of zero. Left: Correct shadow. Right: Iterative convolution shadow. First and third image show the shadows. In second and fourth image, a rainbow color map is applied with isolines from 0.1 to 0.9 with a 0.1 increment. Straight white line is isoline for the shadow intensity of $\frac{1}{2}$.

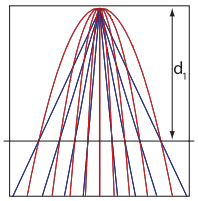
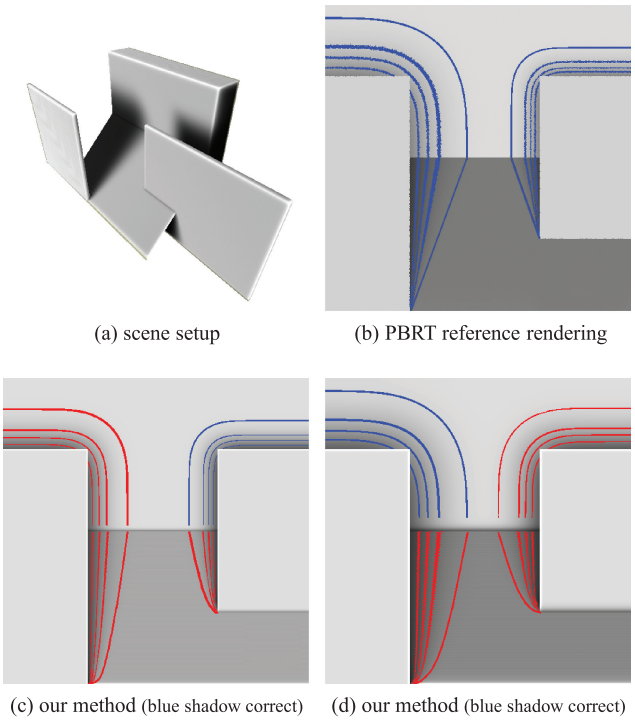


Fig. 17. Isoshadow strength behind a Heaviside occluder for Gaussian light. Correct shadow isolines in blue, and approximate iterative shadow isolines in red. At the target distance d_1 as indicated, the iterated shadow matches the correct shadow.



(a) scene setup (b) PBRT reference rendering

(c) our method (blue shadow correct) (d) our method (blue shadow correct)

Fig. 18. Comparison between a physically based renderer (b) and our method in (c) and (d) and for different distances.

7. DISCUSSION AND LIMITATIONS

The number of slabs used during rendering needs to be sufficiently high to ensure appropriate sampling of the volumetric detail function. For sampled function representations we use the dimensions of the 3D texture as a basis for automatically setting the number of slabs while for procedural details we use a constant value. In both cases, the chosen frequency scale is taken into account and an additional modulation factor can be specified by the user. For all images in this article, between 400 and 800 slabs were used (the actual slab count varies based on the viewing direction and the dimensions of the model's bounding box).

To reduce the number of slabs while maintaining high quality, preintegration can be employed [Engel et al. 2001]. Figure 19 shows a comparison of nonpreintegrated and preintegrated compositing for different slab counts. Artifacts appear as the number of slabs decreases, but preintegration substantially improves the appearance. We note that these artifacts are mostly due to undersampling of the volumetric detail function; shadow appearance does not

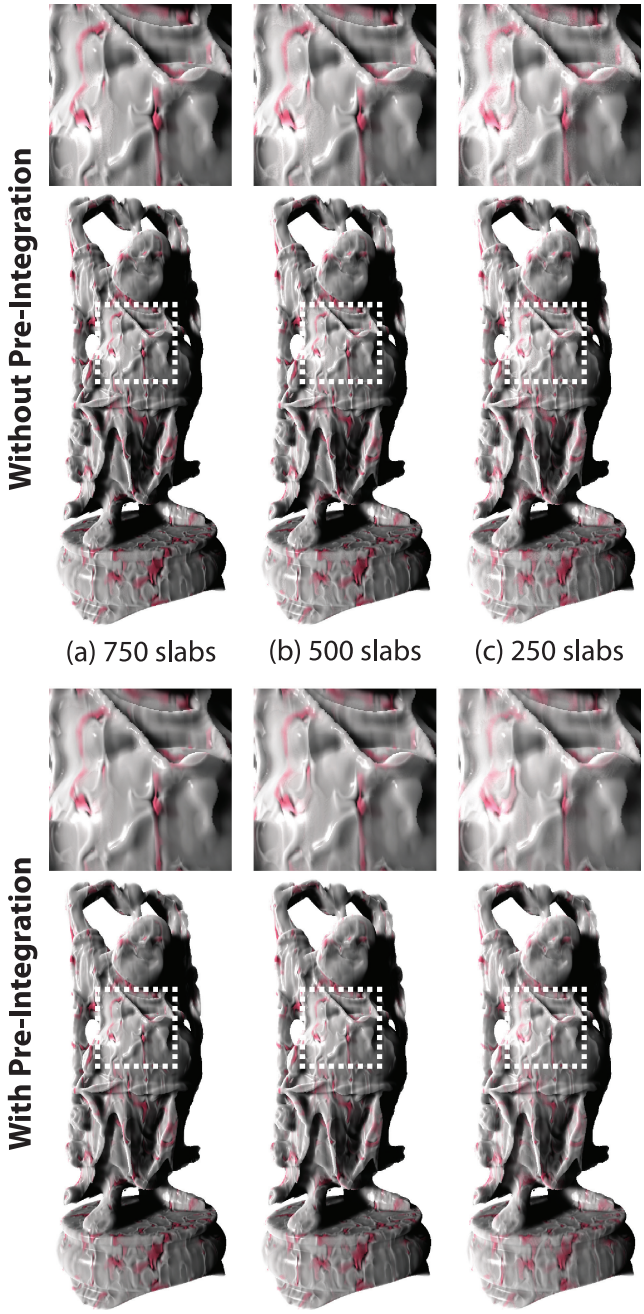


Fig. 19. Comparison of nonpreintegrated and preintegrated compositing for different slab counts. The frame rates were (a) 12.3 frames/second; (b) 18.5 frames/second; (c) 34.4 frames/second. The *Happy Buddha* model was provided courtesy of the Stanford University Computer Graphics Laboratory by the Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep/).

significantly change even though less convolutions are performed. Using the results of Bergner et al. [2006], the number of slabs could be further optimized by incorporating adaptive sampling.

Our model assumes a lighting environment which is not fixed on a sphere around the scene, but moves on a hemisphere behind the viewer. While this is generally not suitable when it is desired to anchor light sources within the scene, it is well-suited for viewing individual models (akin to common camera headlights). In scientific illustrations, for example, lighting commonly follows certain conventions (e.g., coming from a top-left direction relative to the viewer [O'Shea et al. 2008]). Indeed, one promising application we envision for our method is the high-quality visualization of individual objects such as archeological artifacts. Another application scenario is the visualization of medical or biological imaging data where it is frequently desired to combine volume rendering with polygonal meshes derived from segmentation masks. Our approach enables such a hybrid visualization with interactive high-quality lighting effects.

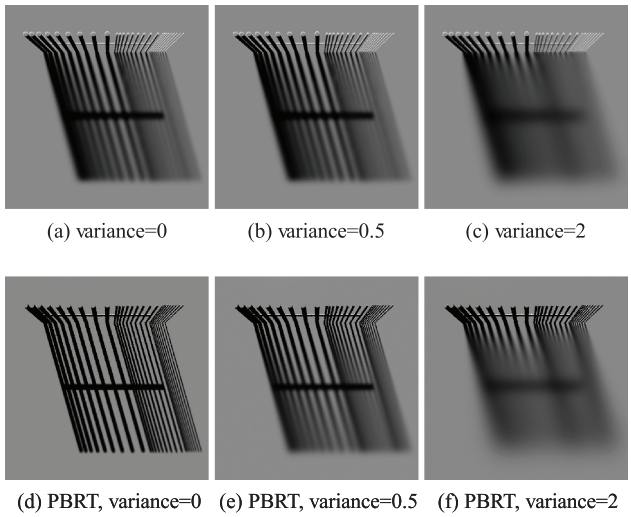


Fig. 20. A fence scene rendered with our method in top row and a PBRT reference in bottom row. Increasing uniform variance from left to right of 0, 0.5, and 2.

One current limitation of our implementation is that shadows will only be cast by objects that are rendered. Objects outside the view frustum will not contribute to the shadow in the visible scene. In order to remedy this, the viewport size during rendering would have to be increased accordingly. A further limitation of our approach is that light sources with very low variances tend to receive too much blur. This is due to the use of discrete arithmetics on the GPU. Filters with narrow support, such as a Dirac peak, representing a point light source will at most light angles be positioned between sample points. Interpolation will then smear the Dirac peak over more samples and increase its variance. This is illustrated in Figure 20 which shows renderings of a fence with varying vertical pole sizes and distances. A horizontal pole extends through about three quarters of the fence length. The top row shows our renderings with one light source of increasing uniform variance from left to right of 0, 0.5, and 2. The bottom row shows reference renderings with a corresponding light source in PBRT. Higher variances are simulated well as the correspondence with PBRT for the middle and right column demonstrates, but the zero-variance light source is not accurately represented. Figure 20 also demonstrates how our method works for objects casting shadows from several slices. Since the fence is pointing towards the viewer, it is being sliced through front-to-back and the shadow on the ground is accumulated from all slice depths. Therefore the penumbras are not identical, although close to the reference rendering.

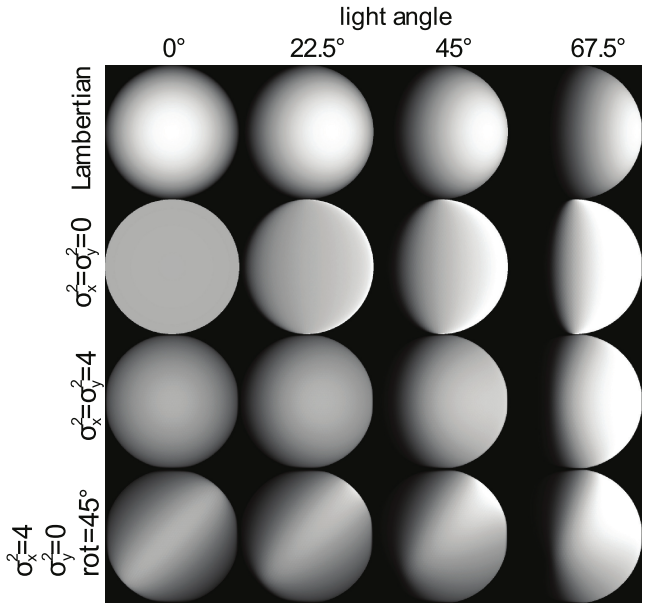


Fig. 21. Rendering of a sphere with different lighting. From left to right: Incoming light at angles 0, 22.5, 45, and 67.5 degrees. From top to bottom: Lambertian shading, scattering with $\sigma_x^2 = \sigma_y^2 = 0$, with $\sigma_x^2 = \sigma_y^2 = 4$ and $\sigma_x^2 = 4$, $\sigma_y^2 = 0$ rotated 45 degrees.

The form factor $F(y)$ from Eq. (1) for producing surface shading is not explicitly calculated in our method. Iterative slicing implicitly generates surface shading since any angled surface will be sliced and consequently cast shadows on itself. This shadow is comparable but not equal to Lambertian surface shading. In Figure 21, we show renderings of spheres under different light conditions and compared to Lambertian shading. Notice that when the variance is uniformly zero with direct light at angle 0 degrees, the sphere is uniformly white. This special case arises since no shadows are cast on consecutive slices. Notice also that the variance affects the shading. Comparing the second and third rows, it can be seen that row three has darker shades due to convolution with a larger kernel. For an angled surface, increasing the variance of the light will thus darken the shading of the surface. This darkening effect can be seen by comparing the tilted lower part of the wall in Figure 18(c) and (d), and from left to right on the tilted wall in Figure 15.

Our method does not calculate the directional distribution of incoming light for each voxel, but rather a scalar value describing the general degree of occlusion towards the light source. We are disregarding the 4D nature of light and cannot model phase functions as they depend on the angular distribution of incoming light relative to the viewpoint. However, as demonstrated in this article, we can generate plausible illumination effects. Yu et al. [2009] show that for indirect illumination, the degree of accuracy is not crucial and sometimes not even differentiable for human perception. Kozlowski and Kautz [2007] show the same on accurate occlusions for glossy reflections.

Complex volumetric details are difficult to handle using other methods whereas our approach is easy to implement and only requires three rendering parameters: slab distance, optimal distance, and the number of samples in the kernel. As demonstrated in Figure 11, the number of samples in our elliptical kernel does not affect quality noticeably and can therefore be set to the minimum of 2. This leaves only two parameters where slab distance can be considered an intuitive parameter balancing speed versus sampling quality.

For future work, it may be interesting to investigate the combination of our method with other approaches. As an example, this can be useful for complementing other shadow methods with the volumetric high-resolution procedural shadows we produce. Other methods that have in common with our method the step of slice or layer-based scene voxelization (such as Sun et al. [2008] and Crassin et al. [2011] that are slice based or Kaplanyan and Dachsbacher [2010] that use depth peeling)) are particularly easy to combine. To avoid shadow conflicts with methods that already calculate shadows, our shadow effects can be confined to local shadows by gradually fading out the light buffers. The method by Sun et al. [2008] could be used to integrate refraction and caustics effects into our approach. Kaplanyan and Dachsbacher [2010] already complement their indirect illumination method with ambient occlusion to produce high-frequency surface details. Here, our method could be a more expressive alternative.

8. CONCLUSIONS

In this article, we have interpreted iterative convolution in context of the light model by Soler and Sillion [1998] and presented a theoretical analysis of its properties. Based on the theory, we identified an equivalence class of kernels and developed a fast convolution kernel requiring a minimal number of samples. The theory also showed that iterative convolution can reproduce the effect of arbitrarily rotated Gaussian light sources. This analysis was verified by performing comparisons with reference renderings. We showed that these findings can be translated into an efficient rendering algorithm for the interactive visualization of models enhanced with dynamically changing procedural volumetric details, soft shadowing, and translucency. We demonstrated that our method works well both for sparse scenes with large shadow casters as well as for high-frequency shadows from volumetric details.

APPENDIX

A. PROOFS AND DERIVATIONS

A.1 Correct Shadow with Convolution

We perform the u-substitution.

$$u = d_1x - d_1y + y \Rightarrow x = \frac{u + d_1y - y}{d_1},\ \frac{du}{dx} = d_1 \Rightarrow dx = \frac{1}{d_1}du$$

Inserting u, x, and dx into $V(y)$ of Eq. (2) we get

$$\begin{aligned} V(y) &= \int_S S(x-y)P(d_1x - d_1y + y)dx \\ &= \int_S S\left(\frac{u + d_1y - y}{d_1} - y\right)P(u)\frac{1}{d_1}du \\ &= \frac{1}{d_1}\int_S S\left(\frac{1}{d_1}(u-y)\right)P(u)du, \end{aligned} \tag{11}$$

which can be written as the convolution

$$V(y) = \frac{1}{d_1}S\left(-\frac{1}{d_1}y\right) * P(y). \tag{12}$$

A.2 Approximate Shadow with Iterative Convolution

Since the light function $l(x)$ is defined to be nonnegative and has an integral of 1, it can be considered a statistical probability density

Table II. Notations and Rules

Normal	Statistical
1: Function $p(x)$ with integral 1	Random variable $\mathbb{P}$ with pdf $p(x)$, cdf $P(x)$ mean $\mu_p = E(\mathbb{P})$ and variance $\sigma_p^2 = Var(\mathbb{P})$
2: $p(x) * q(x)$	$\mathbb{P} + \mathbb{Q}$
3: $H(x) * p(x)$	$P(x)$ (H:Heaviside)
4: $P^{-1}(\frac{1}{2})$	$E(\mathbb{P})$
5: $f(x) = p(-x)$	$\mu_f = -\mu_p,\ \sigma_f^2 = \sigma_p^2$
6: $f(x) = cp(cx)$	$\mu_f = \mu_p/c,\ \sigma_f^2 = \sigma_p^2/c^2$ (c:constant)
7:	$E(a\mathbb{P} + b\mathbb{Q}) = aE(\mathbb{P}) + bE(\mathbb{Q})$
8:	$Var(-\mathbb{P}) = Var(\mathbb{P})$
9:	$Var(\mathbb{P} + \mathbb{Q}) \overset{\mathbb{P},\mathbb{Q} \text{ independent}}{=} Var(\mathbb{P}) + Var(\mathbb{Q})$

function (pdf). Table II summarizes statistical facts found in any standard statistical textbook, employed in the following proofs. Functions p and q represent general distribution functions. For any function p that is a pdf, we denote its random variable with double stroke letter $\mathbb{P}$ and its cumulative distribution function (cdf) with a capital letter P (Table II.1). By definition, the convolution of two pdf's is equivalent to the addition of their respective random variables (Table II.2). A function convolved with the Heaviside function becomes the indefinite integral, that is, the cdf (Table II.3). The expectation value defines the position (P^{-1}) of the center (0.5) of gravity of the pdf (Table II.4). Mirroring a pdf around the y-axis maintains its variance but negates its mean (Table II.5). Scaling the function and the parameter of a pdf equally changes the mean and variance as defined in Table II.6. Table II.7, and Table II.8. Table II.9 summarizes standard statistical calculation rules on random variables.

Expression for iterative convolution shadow. The Central Limit Theorem (CLT) [Ash and Doleans-Dade 1999] states that the sum of n equal independent random variables, each with finite expectation and nonzero variance, tends towards a Gaussian distribution as n increases. κ^{*n} can be considered as the sum of n equal independent random variables (Table II.2), each having the same distribution as κ. Therefore the result of the convolutions κ^{*n} will tend towards the Gaussian distribution. To find the function that κ^{*n} converges to, it is sufficient to find the mean and the variance of κ^{*n}.

$$\kappa^{*n} = \mathbb{K} + \mathbb{K} + ..n \text{ times} \quad \text{(Tbl.II.2)} \tag{13}$$

$$E(\kappa^{*n}) = E(\mathbb{K} + \mathbb{K} + ..n \text{ times}) = nE(\mathbb{K}) = n\mu_\kappa \quad \text{(Tbl.II.7)}$$

$$Var(\kappa^{*n}) = Var(\mathbb{K}+\mathbb{K}+..n \text{ times}) = nVar(\mathbb{K}) = n\sigma_\kappa^2 \quad \text{(Tbl.II.9)}$$

$$\text{Therefore} \quad \kappa^{*n} \approx g_{n\mu_\kappa, n\sigma_\kappa^2}$$

Correct shadow for a specific distance. To find out how an iterative kernel must look so that after n iterations it achieves the shadow that the correct approach achieves for distance d_1, we solve the following equality.

$$\underbrace{o(x) * \frac{1}{d_1} l\left(-\frac{1}{d_1}x\right)}_{\text{correct (Eq.4)}} = \underbrace{o(x) * \kappa^{*n}}_{\text{iterative (Eq.5)}}, \quad \kappa^{*n} \overset{\text{Eq.13}}{\approx} g_{n\mu_\kappa, n\sigma_\kappa^2}(x) \tag{14}$$

$$\frac{1}{d_1} l\left(-\frac{1}{d_1}x\right) = g_{n\mu_\kappa, n\sigma_\kappa^2}(x) \quad \Rightarrow \quad l(x) = d_1 g_{n\mu_\kappa, n\sigma_\kappa^2}(-d_1 x)$$

$$l(x) = g_{\mu_l, \sigma_l^2}(x) \text{ where } \mu_l = -\frac{n\mu_\kappa}{d_1} \quad \sigma_l^2 = \frac{n\sigma_\kappa^2}{d_1^2} \quad \text{(Tbl.II.5,II.6)}$$

$$\text{Solving for } \mu_\kappa \text{ and } \sigma_\kappa^2\text{:} \quad \mu_\kappa = -\frac{\mu_l d_1}{n} \quad \sigma_\kappa^2 = \frac{\sigma_l^2 d_1^2}{n}$$

Instead of specifying the number n of iterations (slices) we want to cover the distance d_1. Thus, we specify a slice distance e and the mean and variance become

$$n = \frac{d}{e} \quad \Rightarrow \quad \mu_\kappa = -e\mu_l \quad \sigma_\kappa^2 = d_1 e \sigma_l^2. \tag{15}$$

That is, simulating a Gaussian light $l(x)$ with mean μ_l and variance σ_l^2, casting correct shadows between planes distance d_1 apart, using the slice iteration distance e, can be achieved by using the kernel that has mean μ_κ and variance σ_κ^2 as specified in Eq. (15).

A.3 Linearity of Correct Shadows

The linearity of shadows behind a Heaviside occluder $H(x)$ with transparency T will now be proven (for simplicity, in the article the Heaviside occluder has been set to opaque with $T = 1$).

$$\begin{aligned}
\text{Eq.4:} \quad s_d(x) &= o(x) * \frac{1}{d} l\left(-\frac{1}{d}x\right), \quad o(x) = T \cdot H(x) \Rightarrow \\
s_d(x) &= T \cdot H(x) * \frac{1}{d} l\left(-\frac{1}{d}x\right) \\
&\overset{\text{Tbl.II.3}}{=} T \cdot \frac{1}{d} \int_{-\infty}^{x} l\left(-\frac{1}{d}t\right) dt \\
&=_{(\text{substitution } u=-\frac{1}{d}t \Rightarrow dt=-ddu, x\to-\frac{1}{d}x, -\infty\to\infty)}
\end{aligned}$$

$$\begin{aligned}
T \cdot \frac{1}{d} \int_{\infty}^{-\frac{1}{d}x} \cdot l(u) \cdot -ddu &= -T \cdot \int_{\infty}^{-\frac{1}{d}x} l(u)du = T \cdot \int_{-\frac{1}{d}x}^{\infty} \\
l(u)du &= T \cdot \left(\lim_{x\to\infty}(L(x)) - L\left(-\frac{1}{d}x\right)\right) \\
&= T \cdot \left(1 - L\left(-\frac{1}{d}x\right)\right) \\
s_d(x) &= T \cdot \left(1 - L\left(-\frac{1}{d}x\right)\right)
\end{aligned} \tag{16}$$

Here, $L(x)$ denotes the indefinite integral of $l(x)$. Since $l(x)$ has area 1, the limit expression of $L(x)$ as x goes to infinity is 1. We can now find isoregions in the cast shadow by setting expression (16) to a constant shadow value s in the interval (0, 1) of the penumbra region. The x position of shadows with strength s at distance d from a Heaviside occluder is

$$s_d(x) = s,\ s \in (0,1) \quad \overset{\text{Eq.16}}{\Rightarrow} \quad T \cdot \left(1 - L\left(-\frac{1}{d}x\right)\right) = s \overset{\text{solve for } x}{\Rightarrow}$$

$$\begin{aligned}
x &= d\left(-L^{-1}\left(1 - \frac{s}{T}\right)\right), \\
\text{Let } C_s &= -L^{-1}\left(1 - \frac{s}{T}\right) \text{ then } x = d \cdot C_s.
\end{aligned} \tag{17}$$

A.4 Nonlinearity of Iterative Shadows

We show that iterative shadows behind a Heaviside occluder are nonlinear by investigating isolines of the shadow expression $z_d(x)$.

$$o(x) = H(x) \Rightarrow z_d(x) \overset{\text{Eq.8}}{\approx} H(x) * g_{\mu_d, \sigma_d^2}(x) \Rightarrow$$

$$z_d(x) \overset{\text{Tbl.II.3}}{\approx} G_{\mu_d,\sigma_d^2}(x) = G_{0,1}\left(\frac{x-\mu_d}{\sigma_d}\right)$$

Convolving a probability density function with the Heaviside function results in the corresponding Gaussian cumulative distribution function (cdf) G. This cdf consists of the error function which is not algebraically defined. Therefore, when solving for x, a closed-form expression does not exist. However, we use the standard equivalence expressed by the right equality to define our isolines without solving for x. Setting x to $x = \mu_d + c_1\sigma_d$ where c_1 is some constant results in the same function argument and thus the same shadow strength irrespective of μ_d and σ_d. Since c_1 is a constant defined by a function of the isoshadow strength, the isocurves for a light l are given by

$$x = \mu_d + c_1\sigma_d \overset{\text{Eq.8}}{=} -d\mu_l + c_1\sqrt{d \cdot d_1\sigma_l^2} = -d\mu_l + c\sigma_l\sqrt{d},$$

where $c = c_1\sqrt{d_1}$ is a constant.

A.5 Equal Characteristics for Both Approaches

We calculate an expression for the shadow of strength $\frac{1}{2}$ behind an opaque Heaviside function for each approach and show that they are equal. We set $s = \frac{1}{2}$ in Eq. (17) to find the isoregions of half shadow in the *correct* approach.

$$s = \frac{1}{2} \Rightarrow x = d \cdot -L^{-1}\left(1-\frac{1}{2}\right) \Rightarrow x = -dL^{-1}\left(\frac{1}{2}\right)$$

Here $L^{-1}(\frac{1}{2})$ is the x value of the center of gravity of the light function. If the light function is symmetric, then $L^{-1}(\frac{1}{2})$ is simply the center of the nonzero interval of the light function.

We now find the expression of the regions with a shadow intensity of $\frac{1}{2}$ for the *approximate* approach for distance d and slice distance e. Let $t(x) = \kappa(x)^{*\frac{d}{e}}$

$$\text{Eq.5:} \quad z_d(x) = H(x) * t(x) \overset{\text{Tbl.II.3}}{=} T(x)$$

$$\begin{aligned} z_d(x) \quad &= \frac{1}{2} \Rightarrow T(x) = \frac{1}{2} \Rightarrow x = T^{-1}\left(\frac{1}{2}\right) \overset{\text{Tbl.II.4}}{=} E(\mathbb{T}) \\ &= E\left(\kappa(x)^{*\frac{d}{e}}\right) \end{aligned}$$

$$x \overset{\text{Eq.13}}{=} \frac{d}{e}E(\kappa) \overset{\text{Eq.15}}{=} \frac{d}{e}(-eE(\mathbb{L})) = -dL^{-1}\left(\frac{1}{2}\right).$$

This proof does not build on the Central Limit Theorem. It only assumes that $\mu_\kappa = -e\mu_l$ (see Eq. (15)) and is therefore true for any variance and any (also non-Gaussian) light source $l(x)$. We have shown that from a hard edge, the contour of an iterated shadow traced along the $\frac{1}{2}$ isoline is correct. This also shows that iterative shadows are cast in the physically correct direction behind a sharp (Heaviside) edge.

A.6 Extension to 2D Light Functions

In Eq. (2), the variables x and y are extended to 2D and the outer multiplication with $\frac{1}{d_1}$ is replaced by $\frac{1}{d_1^2}$ due to the double integral (see also Eq. (11) and the similar, approach in Soler and Sillions' work [Soler and Sillion 1998]). The Heaviside function is extended into 2D by extruding it along the y-axis: $H_x(x, y) = H(x)$. The convolution in Eq. (16) will now be a double integral. By performing u-substitution for both integrals, the $\frac{1}{d_1^2}$ factor will disappear. Then the inner integral can simply be regarded as the 1D pdf of the 2D light function collapsed onto the x-axis. This pdf takes the place of $l(u)$ in the following proofs. The linearity proofs of Eq. (10) (Eq. (17) in this Appendix) follow directly.

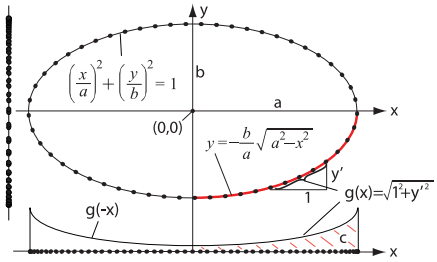


Fig. 22. Finding the variance of an elliptical kernel.

A.7 Elliptic Sampling for a Specified Variance

In this section we find the shape of an ellipse so that the projection of equidistant points on the ellipse onto the x- and y-axis has a given variance.

The equation for an ellipse with axes a and b is shown top left in Figure 22. We rewrite the ellipse formula for y to get the expression of the ellipse segment for the lower right quadrant, shown in red. Now the density of points at x for the red curve is proportional to the curve length $g(x) = \sqrt{1+y'^2}$. To simplify the calculation we fix a to $a = 1$. Then the curve length is

$$g(x) = \sqrt{1+y'^2} = \sqrt{1+b^2\frac{x^2}{1-x^2}} \quad \text{since} \quad y' = b\frac{x}{\sqrt{1-x^2}}.$$

We define the area of $g(x)$ to be c: $c = \int_0^a g(x)dx$, $a = 1$. This integral is the complete elliptic integral of second kind and has no closed-form solution. Due to symmetries, the function $g(x)/2c$ for positive x, and its reflection around the y-axis, $g(-x)/2c$, for negative x, has area 1 and is the pdf of the point density on the x-axis. We label this function $h(x)$ and for $a = 1$, its variance can be expressed as

$$\begin{aligned} Var(h(x)) \;&=\; \int_{-a}^{a} x^2h(x)dx \overset{\text{even}}{=} 2\int_0^a x^2\frac{g(x)}{2c}dx \\ &=\; \frac{1}{c}\int_0^a x^2g(x)dx. \end{aligned}$$

This expression also has no closed-form solution. Since the value of $h(x)$ depends only on b, its variance is a function of b which we call $var_x(b)$. Thus $var_x(b)$ is the variance of the point density projected on the x-axis of equally distributed points on an ellipse with x-radius of 1 and y-radius of b. Using a symbolic solver, we get

$$Var(h(x)) = var_x(b) = \frac{2b^2 - 1 - b^2\frac{EllipticK(\sqrt{1-b^2})}{EllipticE(\sqrt{1-b^2})}}{-3+3\,b^2},$$

where $EllipticK$ is the complete elliptic integral of the first kind and $EllipticE$ is the elliptic integral of the second kind. The function $var_x(b)$ is plotted in the leftmost column of Figure 23.

We let V_x and V_y respectively be the x and y variances of a unit ellipse with radii $a = 1-t$ and $b = t$. We now derive an expression for V_x and V_y based on var_x. From the unit ellipse radius t, we find the x variance of the ellipse having $a = 1$ but with the same

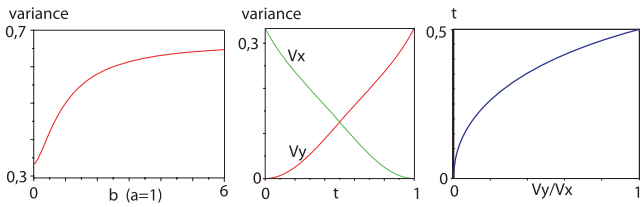


Fig. 23. Left: $Var(h(x))$ for b increasing and $a = 1$. Middle: x and y variance for ellipse parameterized by t. Right: Mapping from variance ratio V_y/V_x to unit ellipse radius having this variance ratio.

radii ratio as the unit ellipse: $var_x(\frac{t}{1-t})$. This ellipse is scaled back so that $a = 1 - t$, by scaling the variance with its square (since $Var(aX) = a^2 Var(X)$). Therefore $V_x(t) = (1-t)^2 \cdot var_x(\frac{t}{1-t})$, and due to symmetry we have $V_y = V_x(1-t)$. V_x is shown in green and V_y in red in the middle plot. From V_x and V_y, we make a function mapping t to the ratio V_y/V_x. We then invert this function numerically so we have a function mapping from V_y/V_x to t shown right in Figure 23. Now the functions V_x and V_y/V_x are used as described in the main article to find the major and minor axis of any ellipse satisfying a specific x and y variance. There they are called $abToVarx()$ and $ratioToab()$, respectively.

REFERENCES

AGRAWALA, M., RAMAMOORTHI, R., HEIRICH, A., AND MOLL, L. 2000. Efficient image-based methods for rendering soft shadows. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*.

ANNEN, T., DONG, Z., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2008. Real-time, all-frequency shadows in dynamic scenes. *ACM Trans. Graph. 27*, 3, 34:1–34:8.

ANNEN, T., MERTENS, T., BEKAERT, P., SEIDEL, H.-P., AND KAUTZ, J. 2007. Convolution shadow maps. In *Proceedings of the Eurographics Symposium on Rendering*. 51–60.

ASH, R. B. AND DOLEANS-DADE, C. A. 1999. *Probability and Measure Theory,* 2nd ed. Academic Press.

ASHIKMIN, M., PREMOZE, S., AND SHIRLEY, P. 2000. A microfacet-based brdf generator. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 65–74.

BARAN, I., CHEN, J., RAGAN-KELLEY, J., DURAND, F., AND LEHTINEN, J. 2010. A hierarchical volumetric shadow algorithm for single scattering. *ACM Trans. Graph. 29*, 6, 178:1–178:10.

BERGNER, S., MOLLER, T., WEISKOPF, D., AND MURAKI, D. J. 2006. A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Trans. Vis. Comput. Graph. 12*, 5, 1353–1360.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 286–292.

CHEN, J., BARAN, I., DURAND, F., AND JAROSZ, W. 2011. Real–time volumetric shadows using 1D min-max mipmaps. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 39–46.

CHEN, Y., TONG, X., WANG, J., LIN, S., GUO, B., AND SHUM, H.-Y. 2004. Shell texture functions. *ACM Trans. Graph. 23*, 3, 343–353.

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive indirect illumination using voxel cone tracing. *Comput. Graph. Forum 30*, 7, 1921–1930.

DONNELLY, W. AND LAURITZEN, A. 2006. Variance shadow maps. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 161–165.

EISEMANN, E., ASSARSSON, U., SCHWARZ, M., AND WIMMER, M. 2009. Casting shadows in real time. In *ACM SIGGRAPH Asia Course Notes*.

EISEMANN, E. AND DECORET, X. 2008. Occlusion textures for plausible soft shadows. *Comput. Graph. Forum 27*, 1, 13–23.

ENGEL, K., KRAUS, M., AND ERTL, T. 2001. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the Workshop on Graphics Hardware*. 9–16.

GRUEN, H., AND THIBIEROZ, N. 2010. OIT and indirect illumination using dx11 linked lists. Presentation at *Game Developers Conference*.

HASENFRATZ, J.-M., LAPIERRE, M., HOLZSCHUCH, N., AND SILLION, F. 2003. A survey of real-time soft shadows algorithms. *Comput. Graph. Forum 22*, 4, 753–774.

HECKBERT, P. S. AND HERF, M. 1997. Simulating soft shadows with graphics hardware. Tech. rep. CMU-CS-97-104, Carnegie Mellon University.

IHRKE, I., ZIEGLER, G., TEVS, A., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2007. Eikonal rendering: Efficient light transport in refractive objects. *ACM Trans. Graph. 26*, 3, 59:1–59:9.

ISIDORO, J. R. 2006. Shadow mapping: GPU-based tips and techniques. Presentation at *Game Developers Conference*.

JANSEN J. AND BAVOIL, L. 2010. Fourier opacity mapping. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 165–172.

JENSEN, H. W. AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scenes with participating media using photon maps. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 311–320.

KAPLANYAN, A. AND DACHSBACHER, C. 2010. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 99–107.

KNISS, J., PREMOZE, S., HANSEN, C., SHIRLEY, P., AND MCPHERSON, A. 2003. A model for volume lighting and modeling. *IEEE Trans. Vis. Comput. Graph. 9*, 2, 150–162.

KOZLOWSKI, O. AND KAUTZ, J. 2007. Is accurate occlusion of glossy reflections necessary? In *Proceedings of the Symposium on Applied Perception in Graphics and Visualization*. 91–98.

KROES, T., POST, F. H., AND BOTHA, C. P. 2012. Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE 7*, 7.

LOKOVIC, T. AND VEACH, E. 2000. Deep shadow maps. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 385–392.

MAX, N. 1991. Unified sun and sky illumination for shadows under trees. *Graph. Models Image Process. 53*, 3, 223–230.

MEYER, A. AND NEYRET, F. 1998. Interactive volumetric textures. In *Proceedings of the Eurographics Workshop on Rendering*. 157–168.

OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. 2000. Relief texture mapping. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 359–368.

O'SHEA, J. P., BANKS, M. S., AND AGRAWALA, M. 2008. The assumed light direction for perceiving shape from shading. In *Proceedings of the 5th Symposium on Applied Perception in Graphics and Visualization*. 135–142.

PENG, J., KRISTJANSSON, D., AND ZORIN, D. 2004. Interactive modeling of topologically complex geometric detail. *ACM Trans. Graph. 23*, 3, 635–643.

PERLIN, K. 2002. Improving noise. *ACM Trans. Graph. 21*, 3, 681–682.

PHARR, M. AND HUMPHREYS, G. 2010. *Physically Based Rendering: From Theory to Implementation*, 2nd ed. Morgan Kaufmann, San Fransisco.

POLICARPO, F. AND OLIVEIRA, M. M. 2006. Relief mapping of non-height-field surface details. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 55–62.

PORUMBESCU, S. D., BUDGE, B., FENG, L., AND JOY, K. I. 2005. Shell maps. *ACM Trans. Graph. 24*, 3, 626–633.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*. 75–82.

SCHERZER, D., WIMMER, M., AND PURGATHOFER, W. 2011. A survey of real-time hard shadow mapping methods. *Comput. Graph. Forum 30*, 1, 169–186.

SCHOTT, M., PEGORARO, V., HANSEN, C., BOULANGER, K., AND BOUATOUCH, K. 2009. A directional occlusion shading model for interactive direct volume rendering. *Comput. Graph. Forum 28*, 3, 855–862.

SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 231–242.

SILLION, F. X. AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann, San Fransisco.

SOLER, C. AND SILLION, F. X. 1998. Fast calculation of soft shadow textures using convolution. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 321–332.

SUN, X., ZHOU, K., STOLLNITZ, E., SHI, J., AND GUO, B. 2008. Interactive relighting of dynamic refractive objects. *ACM Trans. Graph. 27*, 3, 35:1–35:9.

TRAPP, M., AND DOLLNER, J. 2008. Real-time volumetric tests using layered depth images. In *Proceedings of the Eurographics Short Papers*. 235–238.

SOLTESZOVA, V., PATEL, D., BRUCKNER, S., AND VIOLA, I. 2010. A multidirectional occlusion shading model for direct volume rendering. *Comput. Graph. Forum 29*, 3, 883–891.

WANG, X., TONG, X., LIN, S., HU, S., GUO, B., AND SHUM, H.-Y. 2004. Generalized displacement maps. In *Proceedings of the Eurographics Symposium on Rendering*. 227–234.

WOO, A., POULIN, P., AND FOURNIER, A. 1990. A survey of shadow algorithms. *IEEE Comput. Graph. Appl. 10*, 6, 13–32.

YU, I., COX, A., KIM, M. H., RITSCHEL, T., GROSCH, T., DACHSBACHER, C., AND KAUTZ, J. 2009. Perceptual influence of approximate visibility in indirect illumination. *ACM Trans. Appl. Percept. 6*, 4, 24:1–24:14.

ZHANG, C. AND CRAWFIS, R. 2003. Shadows and soft shadows with participating media using splatting. *IEEE Trans. Vis. Comput. Graph. 9*, 2, 139–149.

Received August 2011; revised December 2012; accepted May 2013